Классический труд
Новое издание

Искусство программирования

ТОМ 4, А

Комбинаторные алгоритмы
Часть 1

Дональд Э. Кнут

ИСКУССТВО ПРОГРАММИРОВАНИЯ

Volume 4A / Combinatorial Algorithms, Part 1

THE ART OF COMPUTER PROGRAMMING

DONALD E. KNUTH *Stanford University*

ADDISON–WESLEY
An Imprint of Addison Wesley Longman, Inc.
Upper Saddle River, NJ · Boston · Indianapolis · San Francisco
New York · Toronto · Montréal · London · Munich · Paris · Madrid
Capetown · Sydney · Tokyo · Singapore · Mexico City

Том 4, А / Комбинаторные алгоритмы, часть 1

ИСКУССТВО ПРОГРАММИРОВАНИЯ

ДОНАЛЬД Э. КНУТ *Станфордский университет*

ВИЛЬЯМС

Москва · Санкт-Петербург
2018

ББК 32.973.26-018.2.75
К53
УДК 681.3.07

Издательский дом “Вильямс”

По общим вопросам обращайтесь в Издательский дом “Вильямс” по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Кнут, Дональд Эрвин.

К53 Искусство программирования, том 4А. Комбинаторные алгоритмы , часть 1. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2018. — 960 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1980-9 (рус.)

ББК 32.973.26-018.2.75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства Addison-Wesley Publishing Company, Inc.

Rights to this book were obtained by arrangement with Pearson Education, Inc.

Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2018.

Научно-популярное издание

Дональд Эрвин Кнут

Искусство программирования, том 4А
Комбинаторные алгоритмы, часть 1

ООО “И Д Вильямс”, 127055, г Москва ул Лесная д. 43, стр. 1

ISBN 978-5-8459-1980-9 (рус.)
ISBN 978-0-201-03804-0 (англ.)

© Издательский дом “Вильямс”, 2018
© by Pearson Education, Inc., 2011

ОГЛАВЛЕНИЕ

ГЛАВА 7. КОМБИНАТОРНЫЙ ПОИСК 19

7.1. НУЛИ И ЕДИНИЦЫ 71
7.1.1. Основы булевой алгебры 71
7.1.2. Булевы вычисления 124
7.1.3. Битовые трюки и технологии 165
7.1.4. Бинарные диаграммы решений 242
7.2. ГЕНЕРАЦИЯ ВСЕХ ВОЗМОЖНЫХ ОБЪЕКТОВ 329
7.2.1. Генерация основных комбинаторных объектов 329
7.2.1.1. Генерация всех n-кортежей 329
7.2.1.2. Генерация всех перестановок 369
7.2.1.3. Генерация всех сочетаний 408
7.2.1.4. Генерация всех разбиений 444
7.2.1.5. Генерация всех разбиений множеств 471
7.2.1.6. Генерация всех деревьев 498
7.2.1.7. Исторические и иные сведения 547

ОТВЕТЫ К УПРАЖНЕНИЯМ 577

ПРИЛОЖЕНИЕ А. ТАБЛИЦЫ ЗНАЧЕНИЙ НЕКОТОРЫХ КОНСТАНТ . 904

ПРИЛОЖЕНИЕ Б. ОСНОВНЫЕ ОБОЗНАЧЕНИЯ 908

ПРИЛОЖЕНИЕ В. СПИСОК АЛГОРИТМОВ И ТЕОРЕМ 914

ПРИЛОЖЕНИЕ Г. УКАЗАТЕЛЬ КОМБИНАТОРНЫХ ЗАДАЧ 916

ПРЕДМЕТНО-ИМЕННОЙ УКАЗАТЕЛЬ 920

ПРЕДИСЛОВИЕ

Поместить все сто́ящее в одну книгу совершенно невозможно, и даже попытки сколь-нибудь полно осветить хотя бы некоторые аспекты рассматриваемого предмета, скорее всего, приведут к стремительному увеличению книги.

— ДЖЕРАЛЬД Б. ФОЛЛАНД (GERALD B. FOLLAND), *Editor's Corner* (2005)

Том 4 называется *Комбинаторные алгоритмы*. Когда я предлагал это название, я очень хотел добавить к нему подзаголовок *Мой любимый раздел программирования*. Редакторы уговорили меня не делать этого, но факт остается фактом — программы с комбинаторным привкусом всегда были моими любимицами.

С другой стороны, я часто с удивлением обнаруживал, что в сознании многих людей слово "комбинаторный" ассоциируется с вычислительной сложностью. И действительно, Сэмюэль Джонсон (Samuel Johnson) в своем знаменитом Словаре английского языка (1755) писал, что соответствующее существительное "в настоящее время обычно используется в неверном смысле". От коллег мне приходится слышать горестные рассказы о том, как "комбинаторика нас победила". Так почему же мне комбинаторика доставляет удовольствие, а у других вызывает панику?

Да, это правда, что комбинаторные задачи часто связаны с чрезвычайно большими числами. Статья в словаре Джонсона включает цитату Эфраима Чамберса (Ephraim Chambers), который указал, что общее количество слов длиной не более 24 букв при 24-буквенном алфавите равно 1 391 724 288 887 252 999 425 128 493 402 200. Соответствующее число для 10-буквенного алфавита равно 11 111 111 110; когда же количество букв алфавита равно всего лишь 5, общее число слов сокращается до 3905. Таким образом, здесь мы, определенно, наблюдаем комбинаторный взрыв с ростом размера задачи от 5 до 10 и далее до 24 и более.

На протяжении моей жизни мощность вычислительной техники выросла неимоверно. Сейчас, когда я набираю эти слова, я знаю, что мой ноутбук обрабатывает их со скоростью, более чем в сто тысяч раз большей, чем старый добрый IBM Type 650, которому я посвящал свои книги. Количество памяти моего нынешнего компьютера также примерно в сто тысяч раз больше. Компьютеры завтрашнего дня будут еще более быстрыми и с еще бо́льшим объемом памяти. Но эти удивительные достижения не уменьшили тягу людей к получению ответов на комбинаторные вопросы — скорее даже наоборот. Наша совершенно невообразимая в прошлом способность к быстрым вычислениям повысила наши ожидания и еще больше разожгла аппетит к дальнейшим исследованиям, потому что размер комбинаторной задачи может увеличиться более чем в сто тысяч раз, когда n увеличится всего лишь на единицу.

Неформально комбинаторные алгоритмы можно определить как методы высокоскоростной работы с такими комбинаторными объектами, как перестановки

или графы. Обычно мы пытаемся найти шаблоны или схемы, которые наилучшим образом удовлетворяют определенным ограничениям. Количество задач такого рода огромно, а искусство написания соответствующих программ особенно важно, поскольку одна хорошая идея способна сэкономить годы и даже века машинного времени.

В действительности тот факт, что хорошие алгоритмы для комбинаторных задач могут давать потрясающий выигрыш, привел к огромному подъему современного состояния данной науки. Многие ранее считавшиеся неразрешимыми задачи теперь могут быть с легкостью решены, а многие алгоритмы, считавшиеся ранее очень хорошими, стали гораздо лучше. Начиная примерно с 1970 года кибернетики стали сталкиваться с явлением, которое окрестили леммой Флойда: задачи, которые представляются требующими для решения n^3 операций, на самом деле могут быть решены за $O(n^2)$ операций; задачи, которые, как кажется, требуют n^2 операций, могут быть решены за $O(n \log n)$; задачи со временем решения $n \log n$ зачастую сводятся к задачам со временем решения $O(n)$. Для более сложных задач удается добиться сокращения времени выполнения с $O(2^n)$ до $O(1.5^n)$ или $O(1.3^n)$, и т. д. Прочие задачи остаются сложными в общем случае, но могут быть найдены важные частные случаи, которые решаются гораздо проще. Многие комбинаторные вопросы, о которых я думал, что ответ на них не будет получен при моей жизни, оказались успешно решенными. И это произошло главным образом благодаря усовершенствованию алгоритмов, а не повышению скорости процессоров.

К 1975 году исследования в этой области проводились такими темпами, что значительная доля статей, опубликованных в то время в ведущих журналах по информатике, была посвящена комбинаторным алгоритмам. И этот прорыв был осуществлен не только учеными-кибернетиками; значительный вклад внесли работники в таких областях знаний, как электротехника, искусственный интеллект, исследование операций, математика, физика, статистика и многие другие. Я пытался завершить том 4 *Искусства программирования*, но вместо этого я почувствовал себя сидящим на крышке кипящего чайника: я столкнулся с комбинаторным взрывом другого рода — огромным взрывом новых идей!

Данная серия книг родилась в начале 1962 года, когда я наивно набросал список рабочих названий глав для книги, состоящей из 12 глав. В тот момент я решил включить краткую главу о комбинаторных алгоритмах просто для собственного удовольствия. "Эй, глянь — большинство людей используют компьютеры, чтобы работать с числами, но можно писать программы, которые имеют дело со схемами". В те времена было легко дать достаточно полное описание почти всех известных комбинаторных алгоритмов. И даже к 1966 году, когда я закончил первый черновой вариант из около трех тысяч рукописных страниц (который уже перерос изначально запланированную книгу), к главе 7 относилось менее ста из них. Я совершенно не представлял в тот момент, что то, что я планировал как "легкую закуску", окажется основным блюдом.

Великое комбинаторное брожение 1975 года захватило множество людей. Новые идеи улучшали старые, но при этом редко заменяли их или делали устаревшими. Все это привело к тому, что я был вынужден оставить надежду когда-либо написать достаточно полную книгу, в которой был бы упорядочен весь имеющийся материал и которая была бы достаточно универсальной, чтобы помочь каждому, перед кем

встает та или иная комбинаторная задача. Множество применяемых методов разрослось до таких размеров, что я уже практически не мог рассмотреть некоторую подтему и сказать "Это решение окончательное: на этом мы ставим точку". Вместо этого я должен был ограничиться пояснением наиболее важных принципов, которые, как мне кажется, лежат в основе всех эффективных комбинаторных методов, с которыми я встречался до сих пор. В настоящее время для тома 4 у меня накоплено в два с лишним раза больше материала, чем было собрано для всех предыдущих томов вместе взятых.

Такое количество материала неизбежно ведет к разделению запланированного "тома 4" на несколько физических томов. Сейчас перед вами том 4, А. В один прекрасный день вы увидите тома 4, Б и 4, В, если, конечно, написать их позволит мое здоровье. И, как знать, быть может, наступит очередь томов 4, Г, 4, Д ...? Впрочем, я уверен, что тома 4, Я не будет.

Я планирую по мере возможности систематически разбирать накопленные с 1962 года папки и в меру своих способностей рассказывать истории, которые все еще ждут своего часа. Я не могу гарантировать полноту изложения, но очень хочу воздать должное всем первооткрывателям ключевых идей, а потому я не буду экономить на исторических деталях. Кроме того, всякий раз, когда я знакомлюсь с чем-то, что, как мне кажется, останется важным еще лет 50 и что можно пояснить парой абзацев, я не могу пройти мимо этого и не внести его в книгу. С другой стороны, сложный материал, который требует длительных доказательств, выходит за рамки этих книг (конечно, за исключением действительно фундаментальных тем).

Итак, понятно, что вся область комбинаторных алгоритмов огромна, и охватить ее всю не в моих силах. Какие же темы из тех, которые я опускаю, наиболее важные? Я думаю, мое наибольшее "слепое пятно" — геометрия, потому что я всегда лучше чувствовал себя при работе с визуализацией и алгебраическими формулами, чем с объектами в пространстве. Поэтому я не пытаюсь заниматься в этих книгах комбинаторными задачами, связанными с вычислительной геометрией, такими как плотная упаковка сфер, кластеризация точек данных в n-мерном евклидовом пространстве или деревьями Штайнера на плоскости. Еще более существенно то, что я уклоняюсь от такой темы, как полиэдрическая комбинаторика, и от подходов, основанных, в первую очередь, на линейном программировании, целочисленном программировании или полуопределенном программировании. Эти вопросы рассматриваются во многих других книгах по данной тематике и рассчитаны на геометрическую интуицию читателя. Для моего понимания чисто комбинаторные подходы гораздо проще.

Должен также признаться в некотором предубеждении против алгоритмов, эффективных только в асимптотическом смысле, превосходная производительность которых начинает проявляться, только когда размер задачи превосходит размер вселенной. Алгоритмам такого рода посвящено множество современных публикаций. Я могу понять, почему медитация над беспредельными пределами столь интеллектуально привлекательна и так легко попадает в академическую печать. Однако в *Искусстве программирования* любые методы, которые я сам не испытал в реальных программах, ожидает быстрая расправа. (Конечно же, из этого правила есть исключения, главным образом касающиеся базовых концепций. Одни непрактичные методы просто слишком красивы и/или интеллектуальны, чтобы их можно

было исключить; другие представляют собой поучительные примеры того, как *не* следует поступать.)

Кроме того, как и в предыдущих томах данной серии, я преднамеренно практически полностью ограничиваюсь *последовательными* алгоритмами, несмотря на все возрастающую способность компьютеров к параллельным вычислениям. Я не могу судить, какие из идей, относящихся к параллельным вычислениям, будут полезны через пять или десять лет, не говоря уж о бóльших сроках, а потому оставляю эти вопросы другим, более сведущим авторам. Последовательные методы сами по себе уже являются испытанием пределов моей способности разглядеть, что искусные программисты захотят знать завтра.

Основные решения, которые мне нужно было принять при планировании представления данного материала, — как его лучше организовать: в соответствии с решаемыми задачами или в соответствии с применяемыми методами. Так, например, глава 5 в томе 3 посвящена единственной задаче — сортировке данных; к различным ее аспектам оказались применимыми более двух десятков разнообразных методов. Комбинаторные алгоритмы, напротив, включают множество различных задач, которые, как правило, решаются с помощью существенно меньшего набора методов. В конечном итоге я пришел к выводу, что смешанная стратегия будет работать лучше любого "чистого" подхода. Так, например, в разделе 7.3 рассматривается задача поиска кратчайших путей, а в разделе 7.4.1 — задача связности. При этом многие другие разделы посвящены базовым методам, таким как применение булевой алгебры (раздел 7.1), откат (раздел 7.2.2), теория матроидов (раздел 7.6) или динамическое программирование (раздел 7.7). Знаменитая задача коммивояжера и другие классические комбинаторные задачи, связанные с покрытием, раскраской и упаковкой, не имеют собственных разделов, но неоднократно появляются в разных местах, так как они могут решаться различными способами.

Я упомянул большой прогресс в области комбинаторных вычислений, но это совсем не означает, что все комбинаторные задачи действительно "приручены". Когда время работы компьютерной программы взлетает не хуже баллистической ракеты, ее разработчикам не следует рассчитывать на то, что в этой книге найдется "серебряная пуля", некий универсальный рецепт для решения их проблемы. Описанные здесь методы зачастую будут работать намного быстрее, чем подходы, которые пытались применить эти программисты. Но давайте посмотрим правде в глаза: комбинаторные задачи очень быстро становятся невообразимо огромными. Можно даже строго доказать, что некоторая небольшая, естественная задача *никогда* не будет иметь практического решения, хотя принципиально она разрешима (см. теорему Стокмейера и Мейера в разделе 7.1.2). В других случаях мы знаем, что подходящий алгоритм для решения данной задачи вряд ли существует, хотя и не можем этого доказать. Дело в том, что любой такой эффективный алгоритм позволит решить тысячи других задач, которые давно ставят в тупик крупнейших специалистов в мире (см. описание NP-полноты в разделе 7.9).

Опыт показывает, что изобретение новых комбинаторных алгоритмов будет продолжаться по-прежнему как для новых комбинаторных задач, так и для вновь выявленных вариаций или частных случаев старых. Аппетит программистов к таким алгоритмам со временем будет только расти. Когда программисты сталкиваются с задачами такого рода (а это происходит постоянно), это дает толчок искусству

программирования для достижения новых высот. Однако, скорее всего, сегодняшние методы также останутся в строю.

Бóльшая часть данной книги самодостаточна, хотя и содержит частые ссылки на темы в предыдущих томах. В них достаточно широко рассматривались детали низкоуровневого программирования на машинном языке, так что в данной книге алгоритмы обычно описываются на абстрактном уровне, не зависящем от конкретной машины. Однако некоторые аспекты комбинаторного программирования существенно зависят от низкоуровневых деталей, которые ранее не рассматривались. В таких случаях все примеры в данной книге основаны на компьютере `MMIX`, который заменяет машину `MIX`, описанную в ранних изданиях тома 1. Подробная информация о `MMIX` приведена в дополнении к этому тому, выпущенному в мягкой обложке под названием *The Art of Computer Programming*, Volume 1, Fascicle 1,* в котором содержатся разделы 1.3.1′, 1.3.2′ и т. д. Эти разделы, а также соответствующие ассемблеры и симуляторы, доступны по сети Интернет.

Еще одним загружаемым ресурсом является коллекция программ и данных под названием *The Stanford GraphBase*, постоянно упоминаемая в примерах этой книги. Читателям при изучении комбинаторных алгоритмов настоятельно рекомендуется поработать с ней — мне представляется, что это наиболее эффективный и приятный способ изучения.

Кстати, при написании вводного материала в начале главы 7 мне было приятно отметить, что в ней самым естественным образом оказались упомянутыми некоторые работы моего научного руководителя Маршалла Холла-мл. (Marshall Hall, Jr.) (1910–1990), так же как и некоторые работы *его* научного руководителя Ойстейна Оре (Oystein Ore) (1899–1968) и некоторые работы *его* научного руководителя Торальфа Сколема (Thoralf Skolem) (1887–1963). Научный руководитель Сколема, Аксель Тью (Axel Thue) (1863–1922), уже упомянут в главе 6.

Я чрезвычайно благодарен сотням читателей, которые помогли мне выявить многочисленные ошибки, допущенные в черновых вариантах этого тома, которые изначально были доступны через Интернет, а впоследствии напечатаны в виде брошюр в мягких обложках. В особенности следует отметить комментарии Торстена Далхаймера (Thorsten Dahlheimer), Марка ван Льювена (Marc van Leeuwen) и Удо Вермута (Udo Wermuth). Но, боюсь, в материале скрываются и другие ошибки, которые я хотел бы исправить как можно скорее. Поэтому я с удовольствием заплачу 2 доллара 56 центов тому, кто первым сообщит мне о любой замеченной типографской**, технической или исторической ошибке. На веб-сайте по адресу `http://www-cs-faculty.stanford.edu/~knuth/taocp.html` можно найти текущий список всех поправок к данной книге.

Станфорд, Калифорния D. E. K.
Октябрь 2010

---

* Имеется русский перевод: Кнут Д. Э. *Искусство программирования, том 1, выпуск 1. MMIX — RISC-компьютер для нового тысячелетия.* — М.: ООО “И. Д. Вильямс”, 2007. — *Примеч. пер.*

** Конечно же, имеется в виду оригинальное издание. — *Примеч. пер.*

В предисловии к первому изданию я просил читателей не обращать внимания на ошибки. Теперь я не хочу этого делать и благодарен тем читателям, которые не обратили внимания на мою просьбу.

— СТЮАРТ САЗЕРЛЕНД (STUART SUTHERLAND), *The International Dictionary of Psychology* (1996)

Естественно, я несу ответственность за все оставшиеся ошибки — хотя, как мне кажется, мои друзья могли бы выловить и больше.

— ХРИСТОС Х. ПАПАДИМИТРИУ (CHRISTOS H. PAPADIMITRIOU), *Computational Complexity* (1994)

Мне нравится работать в разных областях, потому что так мои ошибки размазываются более тонким слоем.

— ВИКТОР КЛИ (VICTOR KLEE) (1999)

Примечание к ссылкам. Несколько часто цитируемых журналов и трудов конференций имеют специальные коды, которые встречаются в предметно-именном указателе в конце этой книги. Различные серии *IEEE Transactions* цитируются с использованием буквенного кода, указывающего серию и приводимого полужирным шрифтом перед номером тома. Так, '*IEEE Trans.* **C-35**' означает *IEEE Transactions on Computers*, том 35. IEEE больше не использует эти столь удобные буквенные коды, но их нетрудно расшифровать: '**EC**' означает "Electronic Computers", '**IT**' — "Information Theory", '**SE**' — "Software Engineering", '**SP**' — "Signal Processing" и т. д.; '**CAD**' означает "Computer-Aided Design of Integrated Circuits and Systems".

Перекрестная ссылка вида 'упр. 7.10–00' указывает на будущее упражнение в разделе 7.10, номер которого пока что неизвестен.

Примечание к обозначениям. Простые и интуитивно понятные соглашения и обозначения для алгебраического представления математических концепций всегда были благом для прогресса, в особенности когда большинство исследователей в мире стали использовать единый символьный язык. Текущее состояние дел в комбинаторной математике, к сожалению, в этом отношении находится в беспорядке, так как одни и те же символы различными группами исследователей могут использоваться для совершенно разных целей. Некоторые специалисты, работающие в сравнительно узких областях, непреднамеренно создали противоречивую символику. Информатике — которая по своей природе очень широко взаимодействует с математикой — следует избегать этой опасности, принимая для себя, насколько это возможно, непротиворечивые обозначения. Поэтому мне часто приходилось делать выбор среди конкурирующих систем обозначений, при этом отдавая себе отчет в том, что угодить всем будет невозможно. Я изо всех сил пытался сделать все, чтобы придумать обозначения, которые, как мне кажется, будут наилучшими для будущего применения, зачастую после многих лет экспериментов и обсуждений с коллегами. Не редкостью были и метания между альтернативными вариантами, пока не удавалось остановиться на одном из них. Обычно все же удавалось найти

удобные обозначения, еще не используемые другими учеными способом, противоречащим моему.

Приложение Б представляет собой всеобъемлющий предметный указатель для всех основных обозначений, которые использованы в настоящей книге, включая и те, которые (пока что?) не являются стандартными. Если вы столкнетесь в книге с формулой, которая выглядит для вас странно и/или непонятно, достаточно высоки шансы, что приложение Б направит вас к странице, поясняющей мои намерения. Однако я хотел бы сделать несколько пояснений прямо сейчас, чтобы вы могли ознакомиться с ними еще до первого прочтения книги.

- Шестнадцатеричные константы предваряются знаком фунта (#), т. е., например, $^{\#}\texttt{123}$ означает $(123)_{16}$.
- Обозначение $x \dot{-} y$ применяется для операции "монус", иногда именуемой "минус с точкой" или вычитанием с насыщением, результатом которой является $\max(0, x - y)$.
- Медиана трех чисел $\{x, y, z\}$ обозначается как $\langle xyz \rangle$.
- Множество, такое как $\{x\}$, состоящее из одного элемента, часто обозначается просто как x в контекстах наподобие $X \cup x$ или $X \setminus x$.
- Если n — неотрицательное целое число, количество единичных битов в бинарном представлении n равно νn. Кроме того, если $n > 0$, то крайний слева и крайний справа единичные биты n представляют собой соответственно $2^{\lambda n}$ и $2^{\rho n}$. Например, $\nu 10 = 2$, $\lambda 10 = 3$, $\rho 10 = 1$.
- Декартово произведение графов G и H обозначается как $G \,\square\, H$. Например, $C_m \square C_n$ представляет собой тор $m \times n$, так как C_n является циклом из n вершин.

ПРИМЕЧАНИЯ К УПРАЖНЕНИЯМ

Упражнения, приведенные в этом многотомнике, предназначены как для самостоятельной проработки, так и для семинарских занятий. Очень трудно и, наверное, просто невозможно выучить предмет, только читая теорию и не применяя ее для решения конкретных задач, которые заставляют задуматься о прочитанном. Более того, мы лучше всего заучиваем то, до чего дошли самостоятельно, своим умом. Поэтому упражнения занимают важное место в данном издании. Я приложил немало усилий, чтобы сделать их как можно более информативными, а также отобрать задачи, которые были бы не только поучительными, но и позволяли бы читателю получить удовольствие от их решения.

Во многих книгах простые упражнения даются вперемешку с исключительно сложными. Это не всегда удобно, так как читателю хочется знать заранее, сколько времени ему придется затратить на решение задач (иначе в лучшем случае он их только просмотрит). В качестве классического примера подобной ситуации можно привести книгу Ричарда Беллмана (Richard Bellman) *Динамическое программирование* (М.: Изд-во иностр. лит., 1960). Это очень важная, новаторская работа, но у нее есть один недостаток: в конце некоторых глав в разделе "Упражнения и научные проблемы" среди серьезных, еще нерешенных проблем, попадаются простейшие вопросы. Говорят, что кто-то однажды спросил д-ра Беллмана, как отличить упражнения от научных проблем, и он ответил: "Если вы можете решить задачу, значит, это упражнение; в противном случае это научная проблема".

Совершенно очевидно, что в книге, подобной этой, должны быть приведены и сложные научные проблемы, и простейшие упражнения. Поэтому, чтобы читатель не ломал голову, пытаясь отличить одно от другого, были введены *рейтинги*, которые определяют степень сложности каждого упражнения. Эти рейтинги имеют следующее значение.

Рейтинг	*Объяснение*
00	Чрезвычайно простое упражнение, которое можно выполнить сразу же, если прочитанный материал понят. Упражнения подобного типа почти всегда можно решить "в уме".
10	Простая задача, которая заставляет задуматься над прочитанным, но не представляет особых трудностей. На ее решение вы затратите не больше минуты; в процессе решения могут понадобиться карандаш и бумага.
20	Средняя задача, которая позволяет проверить, понял ли читатель основные положения изложенного материала. Чтобы получить исчерпывающий ответ, может понадобиться примерно 15–20 минут.
30	Задача умеренной сложности. Для ее решения может понадобиться более двух часов (а если при этом включен телевизор, то еще больше).

40 Достаточно сложная или трудоемкая задача, которую вполне можно включить в план семинарских занятий. Предполагается, что студент должен справиться с ней, затратив не слишком много времени, и решение будет нетривиальным.

50 Научная проблема, которая (насколько известно автору в момент написания книги) пока еще не получила удовлетворительного решения, хотя найти его пытались очень многие. Если вы нашли решение подобной проблемы, то опубликуйте его; более того, автор данной книги будет очень признателен, если ему сообщат решение как можно скорее (при условии, что оно правильно).

Интерполируя по этой "логарифмической" шкале, можно понять, что означает любой промежуточный рейтинг. Например, рейтинг *17* говорит о том, что упражнение немного проще, чем задача средней сложности. Если задача с рейтингом *50* будет впоследствии решена каким-либо читателем, то в следующих изданиях данной книги и в списке ошибок, опубликованных в Интернете, она может иметь рейтинг *40*.

Остаток от деления рейтинга на 5 показывает, какой объем рутинной работы потребуется для решения данной задачи. Таким образом, для выполнения упражнения с рейтингом *24* может потребоваться больше времени, чем для упражнения с рейтингом *25*, но для последнего необходим более творческий подход.

Автор очень старался правильно присвоить рейтинги упражнениям, но тому, кто составляет задачи, трудно предвидеть, насколько сложными они окажутся для кого-то другого. К тому же одному человеку некая задача может показаться простой, а другому — сложной. Таким образом, определение рейтингов — дело достаточно субъективное и относительное. Я надеюсь, что рейтинги помогут вам получить правильное представление о степени трудности задач, но их следует воспринимать в качестве ориентира, а не в качестве абсолюта.

Эта книга написана для читателей с различным уровнем математической подготовки и научного кругозора, поэтому некоторые упражнения рассчитаны исключительно на тех, кто серьезно интересуется математикой или занимается ею профессионально. Если рейтингу предшествует буква *M*, значит, математические понятия и обоснования используются в упражнении в большей степени, чем это необходимо тому, кто интересуется в основном программированием алгоритмов. Если же упражнение отмечено буквами *HM*, то для его решения необходимо знание высшей математики в большем объеме, чем дается в настоящей книге. Но пометка *HM* совсем *не*обязательно означает, что упражнение трудное.

Перед некоторыми упражнениями стоит стрелка "▸", которая означает, что они особенно поучительны и их очень рекомендуется выполнить. Само собой разумеется, никто не ожидает, что читатель (или студент) будет решать *все* задачи, поэтому наиболее важные из них и были выделены. Но это ни в коем случае не означает, что другие упражнения выполнять не стоит! Каждый читатель должен хотя бы попытаться решить все задачи, рейтинг которых меньше или равен *10*. Стрелки помогут выбрать задачи с более высокими рейтингами, которые следует решать в первую очередь.

В некоторых разделах имеется более сотни упражнений. Как же найти свой путь среди такого множества? В целом последовательность упражнений стремится

следовать последовательности идей в основном тексте. Идущие подряд упражнения могут основываться одно на другом, как в новаторских задачниках Пойя (Pólya) и Сегё (Szegő). В последних упражнениях разделов часто охватывается материал всего раздела или вводятся дополнительные темы.

К большинству упражнений приведены ответы, помещенные в отдельном разделе в конце книги. Пожалуйста, пользуйтесь ими разумно: ответ смотрите только после того, как приложите все усилия, чтобы решить задачу самостоятельно, либо если у вас совершенно нет времени на ее решение. Ответ будет поучителен и полезен для вас только в том случае, если вы ознакомитесь с ним *после* того, как найдете свое решение или изрядно потрудитесь над задачей. Ответы к задачам излагаются очень кратко и схематично, так как предполагается, что читатель честно пытался решить задачу собственными силами. Иногда в приведенном решении дается меньше информации, чем спрашивалось, но чаще бывает наоборот. Вполне возможно, что полученный вами ответ окажется лучше того, который помещен в книге, или вы найдете ошибку в ответе. В таком случае автор был бы очень признателен, если бы вы как можно скорее подробно сообщили ему об этом; тогда в последующих изданиях книги будет опубликовано более удачное решение, а также имя его автора.

Решая задачи, вы, как правило, можете пользоваться ответами к предыдущим упражнениям, за исключением случаев, когда это будет оговорено особо. Рейтинги упражнениям присваивались в расчете именно на это, и вполне возможно, что рейтинг упражнения $n+1$ ниже рейтинга упражнения n, даже если результат упражнения n является его частным случаем.

Условные обозначения			
		00	Простейшее (ответ дать немедленно)
		10	Простое (на одну минуту)
▸	Рекомендуется	*20*	Средней трудности (на четверть часа)
M	С математическим уклоном	*30*	Повышенной трудности
HM	Требует знания высшей математики	*40*	Высокой трудности
		50	Научная проблема

УПРАЖНЕНИЯ

▸ **1.** [*00*] Что означает рейтинг *M20*?

2. [*10*] В чем ценность упражнений в данной книге для читателя?

3. [*HM45*] Докажите, что всякое односвязное компактное трехмерное многообразие гомеоморфно трехмерной сфере.

Искусство является значительной составляющей такого благотворного занятия, как противостояние предубеждениям.

— ГЕНРИ ДЖЕЙМС (HENRY JAMES), *The Art of Fiction* (1884)

Я признателен всем моим друзьям, но особая моя благодарность тем из них, которые по прошествии времени перестают теребить меня вопросом "Какая книга будет следующей?"

— ПИТЕР Д. ГОМЕС (PETER J. GOMES), *The Good Book* (1996)

Наконец-то после долгих обещаний я представляю миру мою работу. Однако я боюсь, что по отношению к ней были поставлены слишком высокие ожидания. Причина задержки ее публикации в значительной мере кроется в чрезвычайном рвении, которое было проявлено высокопочтенными особами со всех сторон в обеспечении меня дополнительной информацией.

— ДЖЕЙМС БОСВЕЛЛ (JAMES BOSWELL), *The Life of Samuel Johnson, LL. D.* (1791)

Автор особенно признателен издательству Addison-Wesley за терпение, проявленное в течение десятилетия ожидания этой книги после подписания контракта.

— ФРЭНК ХАРАРИ (FRANK HARARY), *Graph Theory* (1969)

Средний парень, ненавидящий квадратные корни или алгебру, приходит в восторг от головоломок, построенных на тех же принципах, и может таким способом развить свои математические и изобретательские шишки, которые приведут в изумление семейного френолога.*

— СЭМ ЛОЙД (SAM LOYD), *The World of Puzzledom* (1896)

Пожалуйста, еще немного!

— Слоган Bitburger Brauerei** (1951)

---

* Френология — псевдонаука о связи психики человека и строения поверхности его черепа. Создателем френологии является австрийский врач и анатом Ф. Й. Галль (F. J. Gall), который утверждал, что все психические свойства, якобы локализующиеся в полушариях мозга, при развитии вызывают разрастание определенного участка мозга, а это, в свою очередь, — образование выпуклости ("шишки") на соответствующем участке черепа. — *Примеч. пер.*

** Одно из крупнейших пивоваренных предприятий Германии. — *Примеч. пер.*

Hommage à Bach.

ГЛАВА 7

КОМБИНАТОРНЫЙ ПОИСК

...Чтобы их найти, надо искать весь день,
*а найдешь — увидишь, что и искать не стоило.**

— БАССАНИО (BASSANIO), в *Венецианский купец* (акт I, сцена 1)

Любые действия или противодействия каждой капли этого людского моря
могут сложиться в самые непредсказуемые и необъяснимые комбинации.
*Иногда происходят поразительные, невообразимые события...***

— ШЕРЛОК ХОЛМС (SHERLOCK HOLMES),
в *The Adventure of the Blue Carbuncle* (1892)

Тема комбинаторных алгоритмов слишком обширна,
чтобы охватить ее в одной статье или даже в одной книге.

— РОБЕРТ Э. ТАРЖАН (ROBERT E. TARJAN) (1976)

Постоянно сталкиваясь с массой людей, я был впечатлен
тем фактом, что наиболее успешными среди них были те,
кто обладал врожденным даром разгадывать головоломки.
Жизнь полна загадок, которые мы должны разгадывать по
мере того, как судьба подбрасывает их нам.

— УОЛТЕР ЛОЙД (СЭМ ЛОЙД-МЛ.) (SAM LOYD, JR.) (1926)

Комбинаторика изучает способы, которыми дискретные объекты могут быть упорядочены в шаблоны различного вида. Например, объекты могут представлять собой $2n$ чисел $\{1, 1, 2, 2, \ldots, n, n\}$, и мы хотим разместить их в строку таким образом, чтобы между двумя появлениями каждого числа k находилось ровно k других чисел. Когда $n = 3$, имеется по сути единственный способ размещения таких "пар Лэнгфорда", а именно 231213 (и его зеркальное отображение 312132); аналогично имеется единственное решение и для $n = 4$. Многие другие типы комбинаторных шаблонов будут рассматриваться далее в этой книге.

Обычно при изучении комбинаторных задач возникает пять основных типов вопросов, одни — потруднее, другие — попроще.

i) *Существование*: существует ли упорядочение X, отвечающее данному шаблону?
ii) *Построение*: если существует, то можно ли быстро найти такое X?

* Перевод Т. Щепкиной-Куперник.

** Перевод В. Михалюка.

iii) *Подсчет*: сколько имеется различных упорядочений X?

iv) *Генерация*: могут ли все упорядочения X_1, X_2, ... быть посещены систематическим образом?

v) *Оптимизация*: какие упорядочения максимизируют или минимизируют $f(X)$ для заданной целевой функции f?

Каждый из приведенных вопросов оказывается достаточно интересным по отношению к парам Лэнгфорда.

Например, рассмотрим вопрос существования. Методом проб и ошибок достаточно быстро выясняется, что, когда $n = 5$, разместить $\{1, 1, 2, 2, \ldots, 5, 5\}$ требуемым образом в десяти позициях невозможно. Обе единицы должны находиться либо в четных позициях, либо в нечетных. Точно так же и тройки и пятерки должны находиться парами либо в четных, либо в нечетных позициях; двойки и четверки используют по одной четной и одной нечетной позиции. Таким образом, мы не можем заполнить по пяти позиций каждой четности. Такие же рассуждения доказывают, что задача не имеет решения и при $n = 6$ или, в общем случае, когда количество нечетных значений в множестве $\{1, 2, \ldots, n\}$ нечетно.

Другими словами, решение задачи Лэнгфорда может существовать только для $n = 4m - 1$ или $n = 4m$, для некоторого целого m. И обратно, когда n имеет указанный вид, существует элегантный способ построения требуемого размещения, найденный Роем О. Дэйвисом (Roy O. Davies) и приведенный в упр. 1.

Сколько же имеется существенно разных решений задачи L_n? С ростом n — много, очень много:

$$\begin{aligned} L_3 &= 1; & L_4 &= 1; \\ L_7 &= 26; & L_8 &= 150; \\ L_{11} &= 17\,792; & L_{12} &= 108\,144; \\ L_{15} &= 39\,809\,640; & L_{16} &= 326\,721\,800; \\ L_{19} &= 256\,814\,891\,280; & L_{20} &= 2\,636\,337\,861\,200; \\ L_{23} &= 3\,799\,455\,942\,515\,488; & L_{24} &= 46\,845\,158\,056\,515\,936. \end{aligned} \tag{1}$$

[Значения L_{23} и L_{24} были найдены в 2004 и 2005 годах М. Краецки (M. Krajecki), К. Джейллетом (C. Jaillet) и А. Бью (A. Bui); см. *Studia Informatica Universalis* **4** (2005), 151–190.] Наброски "на коленке" дают грубую оценку значения L_n (когда оно не равно нулю), равную $(4n/e^3)^{n+1/2}$ (см. упр. 5); и эта оценка оказывается в целом корректной для всех известных значений. Но простой формулы для этих значений, похоже, не существует.

Задача Лэнгфорда представляет собой простой частный случай общего класса комбинаторных задач, называющихся *задачами точного покрытия*. В разделе 7.2.2.1 мы познакомимся с так называемым алгоритмом "танцующих связей", который представляет собой удобное средство генерации всех решений таких задач. Например, при $n = 16$ этот метод требует выполнения всего лишь около 3200 обращений к памяти для каждого найденного решения задачи Лэнгфорда. Таким образом, значение L_{16} может быть вычислено за разумное время путем простой генерации всех решений и их подсчета.

Заметим, однако, что число L_{24} *огромно* — огрубленно оно равно 5×10^{16}, или около 1500 MIP-лет. (Вспомним, что "MIP-год" представляет собой количество

инструкций, которые за год может выполнить машина, выполняющая миллион инструкций в секунду, а именно 31 556 952 000 000.) Следовательно, очевидно, что точное значение L_{24} было определено с помощью иной методики, *не* включающей генерацию всех расстановок. Действительно, имеется гораздо, гораздо более быстрый способ вычисления L_n с помощью полиномиальной алгебры. Поучительный метод, описанный в упр. 6, требует $O(4^n n)$ операций, что может показаться неэффективным; но это лучше метода генерации и подсчета на громадный множитель порядка $\Theta((n/e^3)^{n-1/2})$, так что даже при $n = 16$ этот метод работает примерно в 20 раз быстрее. С другой стороны, точное значение L_{100}, вероятно, никогда не станет известным, несмотря на все бóльшую и бóльшую скорость компьютеров.

Можно также рассмотреть решения задачи Лэнгфорда, *оптимальные* в том или ином отношении. Для примера можно расположить шестнадцать пар разновесов $\{1, 1, 2, 2, \ldots, 16, 16\}$ так, чтобы они удовлетворяли условиям Лэнгфорда и обладали дополнительным свойством "сбалансированности", в том смысле, что, будучи расставленными в соответствующем порядке, они оставят в равновесии рычажные весы:

(2)

Другими словами, $15.5 \cdot 16 + 14.5 \cdot 6 + \cdots + 0.5 \cdot 8 = 0.5 \cdot 11 + \cdots + 14.5 \cdot 4 + 15.5 \cdot 7$; и в данном конкретном примере мы сталкиваемся с еще одним условием равновесия, $16 + 6 + \cdots + 8 = 11 + 16 + \cdots + 7$. Следовательно, справедливо также $16 \cdot 16 + 15 \cdot 6 + \cdots + 1 \cdot 8 = 1 \cdot 11 + \cdots + 15 \cdot 4 + 16 \cdot 7$.

Кроме того, размещение (2) имеет *минимальную ширину* среди всех решений задачи Лэнгфорда порядка 16: соединяющие линии под диаграммой показывают, что в любой точке диаграммы одновременно незавершенными оказываются не более семи пар; можно также показать, что ширина, равная шести, в данном случае недостижима (см. упр. 7).

Какое размещение $a_1 a_2 \ldots a_{32}$ чисел $\{1, 1, \ldots, 16, 16\}$ оказывается *наименее* сбалансированным, в том смысле, что сумма $\sum_{k=1}^{32} k a_k$ максимальна? Оказывается, что максимально возможное значение равно 5268. Одним из таких решений (а всего их 12 016) является

$$2\ 3\ 4\ 2\ 1\ 3\ 1\ 4\ 16\ 13\ 15\ 5\ 14\ 7\ 9\ 6\ 11\ 5\ 12\ 10\ 8\ 7\ 6\ 13\ 9\ 16\ 15\ 14\ 11\ 8\ 10\ 12. \tag{3}$$

Более интересный вопрос — о решениях задачи Лэнгфорда, являющихся минимальными и максимальными в лексикографическом порядке. Ответами для $n = 24$ являются

$$\{\texttt{abacbdecfgdoersfpgqtuwxvjklonhmirpsjqkhltiunmwvx},\\ \texttt{xvwsquntkigrdapaodgiknqsvxwutmrpohljcfbecbhmfejl}\}, \tag{4}$$

если использовать буквы a, b, ..., w, x вместо чисел 1, 2, ..., 23, 24.

Мы рассмотрим множество методов комбинаторной оптимизации в следующих разделах этой главы. Наша цель, конечно же, заключается в том, чтобы решать

Рис. 1. Беспорядок в карточном королевстве: никакого совпадения строк. (Это всего лишь одно из множества решений популярной головоломки XVIII века.)

такие задачи с помощью изучения только небольшой части пространства всех возможных размещений.

Ортогональные латинские квадраты. Давайте ненадолго вернемся в ранние дни комбинаторики. Посмертное издание книги Жака Озанама (Jacques Ozanam) *Recreations mathematiques et physiques* (Paris: 1725) включает занимательную головоломку на с. 434 тома 4: "Возьмите всех тузов, королей, дам и валетов из обычной колоды игральных карт и расположите их в виде квадрата так, чтобы в каждой строке и каждом столбце содержались карты всех мастей и всех достоинств". Сумеете ли вы сделать это? Решение Озанама, показанное на рис. 1, делает даже больше: полный набор достоинств и мастей встречается также на обеих диагоналях квадрата.

В 1779 году подобная головоломка, распространившаяся в Санкт-Петербурге, привлекла внимание великого математика Леонарда Эйлера. "Тридцать шесть офицеров шести различных званий, взятые из шести различных полков, хотят выстроиться квадратным строем 6×6 так, чтобы в каждой строке и каждом столбце этого квадрата имелись офицеры всех званий и полков. Как они должны это сделать?" Никто не мог найти удовлетворительного решения этой задачи, так что Эйлер решил взяться за эту проблему (несмотря на то, что в 1771 году он практически ослеп и диктовал все свои работы ассистентам). Он написал большую работу на эту тему [в конце концов опубликованную в *Verhandelingen uitgegeven door het Zeeuwsch Genootschap der Wetenschappen te Vlissingen* **9** (1782), 85–239], в которой привел решения подобной задачи для n званий и n полков для $n = 1$, 3, 4, 5, 7, 8, 9, 11, 12, 13, 15, 16, ...; ему не поддались только случаи $n \bmod 4 = 2$.

Очевидно, что при $n = 2$ решения не существует. Но Эйлер был озадачен случаем $n = 6$, когда рассмотрел "очень значительное число" квадратных расста-

новок, которые не работали. Он показал, что каждое решение должно приводить ко многим другим решениям, выглядящим иными, и не мог поверить, что все такие решения сумели избежать его внимания. Поэтому он говорил: "Я не сомневаюсь в своем выводе о том, что найти полный квадрат из 36 ячеек невозможно, точно так же, как невозможно найти решения для $n = 10$, $n = 14$ ... в общем случае для всех нечетно-четных чисел".

Эйлер дал 36 офицерам имена $a\alpha$, $a\beta$, $a\gamma$, $a\delta$, $a\epsilon$, $a\zeta$, $b\alpha$, $b\beta$, $b\gamma$, $b\delta$, $b\epsilon$, $b\zeta$, $c\alpha$, $c\beta$, $c\gamma$, $c\delta$, $c\epsilon$, $c\zeta$, $d\alpha$, $d\beta$, $d\gamma$, $d\delta$, $d\epsilon$, $d\zeta$, $e\alpha$, $e\beta$, $e\gamma$, $e\delta$, $e\epsilon$, $e\zeta$, $f\alpha$, $f\beta$, $f\gamma$, $f\delta$, $f\epsilon$, $f\zeta$, основываясь на их полках и званиях. Он заметил, что любое решение состоит из двух *отдельных* квадратов, одного — для латинских букв и второго — для греческих. Каждый из этих квадратов, как предполагается, содержит разные записи в строках и столбцах; так что Эйлер начал свое исследование с изучения возможных конфигураций для $\{a, b, c, d, e, f\}$, которые он назвал *латинскими квадратами*. Латинский квадрат может быть объединен в пару с греческим квадратом, образуя "греко-латинский квадрат" только в том случае, когда квадраты *ортогональны* друг к другу, что означает, что пары букв (латинская, греческая) не могут встретиться более одного раза при наложении квадратов. Например, если положить $a = \mathtt{A}$, $b = \mathtt{K}$, $c = \mathtt{Q}$, $d = \mathtt{J}$, $\alpha = \clubsuit$, $\beta = \spadesuit$, $\gamma = \diamondsuit$ и $\delta = \heartsuit$, то рис. 1 эквивалентен латинскому, греческому и греко-латинскому квадратам

$$\begin{pmatrix} d & a & b & c \\ c & b & a & d \\ a & d & c & b \\ b & c & d & a \end{pmatrix}, \begin{pmatrix} \gamma & \delta & \beta & \alpha \\ \beta & \alpha & \gamma & \delta \\ \alpha & \beta & \delta & \gamma \\ \delta & \gamma & \alpha & \beta \end{pmatrix} \text{ и } \begin{pmatrix} d\gamma & a\delta & b\beta & c\alpha \\ c\beta & b\alpha & a\gamma & d\delta \\ a\alpha & d\beta & c\delta & b\gamma \\ b\delta & c\gamma & d\alpha & a\beta \end{pmatrix}. \tag{5}$$

Конечно, можно воспользоваться *любыми* n различными символами в латинском квадрате размером $n \times n$; все, что имеет значение, — это чтобы ни один символ не встречался дважды в одной строке или в одном столбце. Так что мы вполне можем использовать для записей числовые значения $\{0, 1, \ldots, n-1\}$. Кроме того, мы будем говорить просто о латинских квадратах, не подразделяя их на греческие или латинские, поскольку ортогональность — отношение симметричное.

Утверждение Эйлера о том, что два латинских квадрата размером 6×6 не могут быть ортогональны, было подтверждено Томасом Клаузеном (Thomas Clausen), который свел задачу к проверке 17 фундаментально различных случаев. Об этом в письме, датированном 10 августа 1842 года, К. Ф. Гауссу (C. F. Gauss) сообщил Г. К. Шумахер (H. C. Schumacher). Сам Клаузен свой анализ не публиковал. Первая появившаяся в печати демонстрация этого факта принадлежит Г. Тарри (G. Tarry) [*Comptes rendus, Association francaise pour l'avancement des sciences* **29**, part 2 (1901), 170–203], который, идя собственным путем, открыл, что латинские квадраты размером 6×6 могут быть разделены на 17 семейств. (В разделе 7.2.3 мы рассмотрим, как разложить задачу на комбинаторно неэквивалентные классы размещений.)

Гипотеза Эйлера о значениях $n = 10$, $n = 14$, ... была "доказана" трижды: Ю. Петерсеном (J. Petersen) [*Annuaire des mathématiciens* (Paris: 1902), 413–427], П. Вернике (P. Wernicke) [*Jahresbericht der Deutschen Math.-Vereinigung* **19** (1910), 264–267] и Г. Ф. Мак-Нейшем (H. F. MacNeish) [*Annals of Math.* (2) **23** (1922), 221–227]. Однако во всех трех доказательствах известны недостатки; так что вопрос оставался открытым до тех пор, пока много лет спустя не стали доступны быст-

родействующие компьютеры. Одной из первых комбинаторных задач, решенных с применением ЭВМ, стала загадка греко-латинского квадрата размером 10×10: существует он или нет?

В 1957 году Л. Д. Пейдж (L. J. Paige) и Ч. Б. Томпкинс (C. B. Tompkins) запрограммировали ЭВМ SWAC для поиска контрпримера для предсказания Эйлера. Они выбирали один отдельный латинский квадрат 10×10 "почти случайно", и их программа пыталась найти другой квадрат, ортогональный первому. Но результат оказался обескураживающим, и они решили прекратить вычисления через пять часов работы. За это время машина сгенерировала достаточно данных, чтобы можно было предсказать, что вся работа потребует как минимум 4.8×10^{11} часов машинного времени!

Вскоре после этого трем математикам удалось совершить прорыв, который вынес латинские квадраты на страницы одной из ведущих мировых газет (New York Times от 29 апреля 1959 года): Р. Ч. Бозе (R. C. Bose), Ш. Ш. Шрикханде (S. S. Shrikhande) и Э. Т. Паркер (E. T. Parker) нашли замечательный ряд построений, которые дают ортогональные квадраты размером $n \times n$ для всех $n > 6$ [*Proc. Nat. Acad. Sci.* **45** (1959), 734–737, 859–862; *Canadian J. Math.* **12** (1960), 189–203]. Таким образом, после сопротивления атакам в течение 180 лет гипотеза Эйлера оказалась почти полностью ошибочной.

Их открытие было сделано без привлечения компьютеров. Но Паркер работал на UNIVAC, и его навыки программиста помогли получить решение задачи Пейджа и Томпкинса менее чем за час машинного времени ЭВМ UNIVAC 1206 Military Computer. [См. *Proc. Symp. Applied Math.* **10** (1960), 71–83; **15** (1963), 73–81.]

Давайте взглянем, как работали программисты тех времен и как Паркер превзошел их. Пейдж и Томпкинс начали со следующего квадрата L размером 10×10 и его пока неизвестного ортогонального напарника M:

$$L = \begin{pmatrix} 0&1&2&3&4&5&6&7&8&9 \\ 1&8&3&2&5&4&7&6&9&0 \\ 2&9&5&6&3&0&8&4&7&1 \\ 3&7&0&9&8&6&1&5&2&4 \\ 4&6&7&5&2&9&0&8&1&3 \\ 5&0&9&4&7&8&3&1&6&2 \\ 6&5&4&7&1&3&2&9&0&8 \\ 7&4&1&8&0&2&9&3&5&6 \\ 8&3&6&0&9&1&5&2&4&7 \\ 9&2&8&1&6&7&4&0&3&5 \end{pmatrix} \quad \text{и} \quad M = \begin{pmatrix} 0&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 1&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 2&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 3&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 4&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 5&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 6&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 7&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 8&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \\ 9&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup&\sqcup \end{pmatrix}. \tag{6}$$

Без потери общности можно считать, что строки M начинаются, как показано, с 0, 1, ..., 9. Задача заключается в заполнении 90 остающихся пустыми записей, и исходная программа для SWAC работала сверху вниз и слева направо. Верхняя левая пустая запись $\sqcup$ не может содержать 0, поскольку 0 уже имеется в верхней левой строке M. Это не может быть и 1, так как пара $(1,1)$ уже имеется слева в следующей строке квадрата (L, M). Однако мы можем умозрительно вставить в это место 2. Следующей можно вставить цифру 1; и так постепенно мы найдем лексикографически наименьшую верхнюю строку, которая может быть в M, а именно 0214365897. Аналогично наименьшими строками, которые могут размещаться под 0214365897, являются 1023456789 и 2108537946; очередная наименьшая строка имеет вид 3540619278. К сожалению, после этого мы попадаем в переплет:

нет никакой возможности завершить очередную строку так, чтобы не вступить в конфликт с уже имеющимися. Поэтому заменим 3540619278 на 3540629178 (но и этот выбор не дает возможности продолжать работу), затем на 3540698172, и так еще несколько шагов, пока наконец не достигнем строки 3546109278, за которой может следовать строка 4397028651, после которой мы снова встретимся с неприятностями.

В разделе 7.2.3 мы изучим методы оценки поведения таких поисков без реального их проведения. Такие оценки говорят нам о том, что в данном случае метод Пейджа–Томпкинса, по сути, обходит неявное дерево поиска, содержащее около 2.5×10^{18} узлов. Большинство этих узлов принадлежит только нескольким уровням дерева; больше половины из них работают с выбором правой части шестой строки M, после того как около 50 из 90 пустых мест заполнены. Типичный узел дерева поиска требует для проверки, вероятно, около 75 обращений к памяти. Следовательно, общее время работы на современном компьютере можно оценить как время, необходимое для выполнения 2×10^{20} обращений к памяти.

Паркер же вернулся к методу, который Эйлер использовал для поиска ортогональных пар в 1779 году. Сначала он находил все так называемые *секущие* (*transversals*) матрицы L, а именно все способы выбора некоторых из ее элементов так, чтобы в каждой строке и каждом столбце имелся ровно один элемент, причем чтобы были использованы все возможные значения элементов. Например, одним из сечений в записи Эйлера является 0859734216, т. е. в столбце 0 мы выбираем 0, в столбце 1 — 8, ..., в столбце 9 — 6. Каждая секущая, включающая k в крайнем слева столбце L, представляет корректный способ размещения десяти k в квадрате M. Задача поиска секущих в действительности достаточно проста. Для данной матрицы L, оказывается, их имеется ровно 808; для $k = (0, 1, \ldots, 9)$ имеется соответственно $(79, 96, 76, 87, 70, 84, 83, 75, 95, 63)$ секущих.

После того как нам становятся известными секущие, остается решить задачу точного покрытия из десяти этапов, что существенно проще, чем исходная 90-этапная задача в (6). Все, что нам нужно, — это покрыть квадрат десятью непересекающимися секущими, поскольку каждое такое множество из десяти секущих эквивалентно латинскому квадрату M, ортогональному L.

Конкретный квадрат L из (6) в действительности имеет ровно одну ортогональную пару:

$$\begin{pmatrix} 0&1&2&3&4&5&6&7&8&9\\ 1&8&3&2&5&4&7&6&9&0\\ 2&9&5&6&3&0&8&4&7&1\\ 3&7&0&9&8&6&1&5&2&4\\ 4&6&7&5&2&9&0&8&1&3\\ 5&0&9&4&7&8&3&1&6&2\\ 6&5&4&7&1&3&2&9&0&8\\ 7&4&1&8&0&2&9&3&5&6\\ 8&3&6&0&9&1&5&2&4&7\\ 9&2&8&1&6&7&4&0&3&5 \end{pmatrix} \perp \begin{pmatrix} 0&2&8&5&9&4&7&3&6&1\\ 1&7&4&9&3&6&5&0&2&8\\ 2&5&6&4&8&7&0&1&9&3\\ 3&6&9&0&4&5&8&2&1&7\\ 4&8&1&7&5&3&6&9&0&2\\ 5&1&7&8&0&2&9&4&3&6\\ 6&9&0&2&7&1&3&8&4&5\\ 7&3&5&1&2&0&4&6&8&9\\ 8&0&2&3&6&9&1&7&5&4\\ 9&4&3&6&1&8&2&5&7&0 \end{pmatrix}. \tag{7}$$

Алгоритм танцующих связей находит это решение и доказывает его единственность всего лишь примерно за 1.7×10^8 обращений к памяти для данных 808 секущих. Стоимость же фазы поиска секущих (около пяти миллионов обращений к памяти) пренебрежимо мала по сравнению с поиском матрицы. Таким образом, исходное

время работы, равное 2×10^{20} обращениям к памяти и считавшееся неизбежной платой за решение задачи с 10^{90} возможными способами заполнения пустых слотов, оказалось уменьшено более чем в 10^{12}(!) раз.

Позже мы увидим, что можно внести усовершенствования и в методы решения 90-уровневой задачи наподобие (6). Действительно, оказывается, (6) непосредственно представимо в качестве задачи точного покрытия (см. упр. 17), которая решается процедурой танцующих связей из раздела 7.2.2.1 ценой всего лишь 1.3×10^{11} обращений к памяти. Но даже в этом случае подход Эйлера–Паркера оказывается в тысячу раз лучше подхода Пейджа–Томпкинса. "Разложив" задачу на две отдельные фазы, одну для поиска секущих и вторую — для их объединения, Эйлер и Паркер, по сути, уменьшили стоимость вычислений с произведения T_1T_2 до суммы $T_1 + T_2$.

Мораль этой истории очевидна: комбинаторные задачи могут поставить нас перед лицом огромной совокупности возможностей, но мы не должны сразу же опускать руки. Одна хорошая идея может на много порядков сократить количество требующихся вычислений.

Головоломки и реальный мир. Многие из комбинаторных задач, которые мы будем рассматривать в этой главе, наподобие задач Лэнгфорда о парах или Озанама о шестнадцати картах, изначально представляли собой не более чем головоломки. Некоторых особо серьезно настроенных читателей такая легкомысленность некоторых задач может попросту отпугнуть. В самом деле, сто́ит ли тратить попусту машинное время, неужели ему нет более полезного применения? И не должна ли книга, посвященная компьютерам и программированию, в первую очередь озаботиться важными промышленными и научными приложениями, определяющими мировой прогресс?

Начнем с того, что автор данной книги совершенно не озадачивался ее полезностью для прогресса человечества. Но он твердо убежден, что такая книга, как эта, должна обратить особое внимание на *методы* решения задач, а также на математические идеи и *модели*, которые помогут решать многие различные проблемы, а не на причины, по которым такие методы и модели могут быть полезными. Мы изучим множество красивых и эффективных способов решения комбинаторных задач, и главной причиной их изучения будет элегантность этих методов. Комбинаторные задачи встречаются повсюду, и ежедневно появляются новые способы применения описываемых в этой главе методик. Так что давайте не будем заранее ограничивать сами себя, пытаясь заранее классифицировать, для чего годятся те или иные идеи.

Например, оказывается, что ортогональные латинские квадраты чрезвычайно полезны, в особенности при планировании экспериментов. Уже в 1788 году Франсуа Кретте де Паллюль (Francois Cretté de Palluel) использовал латинский квадрат размером 4×4 для изучения, что будет с шестнадцатью овцами четырех различных пород при применении четырех различных кормов и четырех разных периодов стрижки. [*Mémoires d'Agriculture* (Paris: Société Royale d'Agriculture, trimestre d'été, 1788), 17–23.] Латинский квадрат позволил ему ограничиться 16 овцами вместо 64; при использовании греко-латинского квадрата он мог бы варьировать еще один параметр, например количество корма или вид пастбища.

Если бы мы сосредоточились на его воззрениях на животноводство, то могли бы увязнуть в деталях разведения овец, сравнении корнеплодов с зерновыми

и расходами на их выращивание и т. д. Читатели, не интересующиеся сельским хозяйством, могли бы пропустить весь материал, несмотря на то, что латинские квадраты применимы к широкому спектру исследований (представьте, например, проверку действия лекарства на пациентов на пяти стадиях некоторого заболевания, пяти возрастных и пяти весовых группах). Кроме того, концентрация на опытно-исследовательском применении латинских квадратов может привести к тому, что читатели упустят тот факт, что латинские квадраты имеют важные приложения в дискретной геометрии и кодах с исправлением ошибок (см. упр. 18–24).

Даже задача Лэнгфорда, на первый взгляд кажущаяся чистым развлечением, также имеет практическую ценность. Т. Сколем (T. Skolem) использовал последовательности Лэнгфорда для построения Штейнеровской системы троек, которую мы применяли для запросов к базам данных в разделе 6.5 [см. *Math. Scandinavica* **6** (1958), 273–280]; а в 1960-х годах Э. Д. Грот (E. J. Groth) из Motorola Corporation применил пары Лэнгфорда в разработке электронных схем для умножения. Кроме того, алгоритмы, которые эффективно решают задачу Лэнгфорда и секущих латинских квадратов, такие как метод танцующих связей, применимы к решению задач точного покрытия в общем случае; а задача точного покрытия имеет большое значение для таких важнейших проблем, как, например, справедливое распределение участков для избирательных округов и т. п.

Ни приложения, ни головоломки не являются наиболее важными вещами. Наша главная цель — внедрить в свои головы базовые концепции наподобие понятий латинских квадратов или точного покрытия. Эти концепции обеспечат нас строительными блоками, словарем и пониманием, необходимыми для решения *завтрашних* задач.

Глупо обсуждать решение задач без практического решения каких бы то ни было задач. Для стимуляции нашего мышления и воображения, упорядочения серых клеточек и знакомства с базовыми концепциями нужны подходящие задачи. И для этой цели часто лучше всего подходят именно головоломки, поскольку их можно изложить буквально несколькими словами и они не требуют сложных базовых знаний.

Жизнь — слишком сложная вещь, чтобы относиться к ней, как к очередной головоломке, которую следует решить и которую можно решить, просто снабдив машину соответствующими инструкциями и данными. Но, к счастью, есть загадки, которые *могут* быть решены! Головоломки сто́ят того, чтобы причислить их к наибольшим удовольствиям жизни, которыми, как и любыми другими удовольствиями, следует пользоваться умеренно и без злоупотреблений.

Конечно, Лэнгфорд и Озанам предназначали свои головоломки для людей, а не для компьютеров. Разве смысл не будет потерян, если мы просто передадим задачу машинам, которые будут решать ее методом грубой силы, а не с помощью искусного применения рационального мышления? Джордж Брюстер (George Brewster) в письме Мартину Гарднеру (Martin Gardner) в 1963 году выразил широко распространенное мнение следующим образом: “Скармливать головоломки для отдыха компьютерам — немногим лучше, чем глушить форель в горной реке динамитом. Оставьте человеку возможность расслабиться”.

Хотя с этим трудно не согласиться, указанная точка зрения упускает один важный момент: простые головоломки часто имеют обобщения, которые выходят за

рамки человеческих возможностей и вызывают наше любопытство. Изучение этих обобщений часто предполагает наличие поучительных методов, которые могут применяться к ряду других проблем и иметь неожиданные последствия. Действительно, многие из ключевых методов, которые мы будем изучать, родились в попытках решения различных головоломок. При написании этой главы автор не мог не наслаждаться тем фактом, что теперь, когда компьютеры становятся все более и более быстрыми, головоломки доставляют еще большее удовольствие, ведь теперь в наших руках оказывается еще более мощный динамит, с которым можно вволю поиграться. [Более подробно точка зрения автора изложена в его эссе "Полезны ли задачи-головоломки?", написанном еще в 1976 году; см. *Selected Papers on Computer Science* (1996), 169–183.]

Опасность головоломок в том, что они могут оказаться *слишком* элегантными. Хорошие головоломки, как правило, математически чисты и хорошо структурированы, но нам надо научиться систематически работать с тем грязным, хаотическим материалом, который ежедневно нас окружает. Фактически некоторые вычислительные методы важны, в первую очередь, потому, что они предоставляют мощные способы преодоления этих сложностей. Вот почему, например, в начале главы 5 рассказывалось о загадочных правилах упорядочения библиотечных каталогов, а в разделе 2.2.5 моделировалась работа реального лифта.

Чтобы в экспериментах с комбинаторными алгоритмами можно было воспользоваться данными из реального мира, подготовлена коллекция программ и данных под названием Stanford GraphBase (SGB). Она включает, например, данные об американских автострадах и модель "затраты–выпуск" американской экономики; в ней описано распределение персонажей в *Илиаде* Гомера, *Анне Карениной* Льва Толстого и ряде других романов; здесь можно найти структуру *Thesaurus* 1879 года Роджета (Roget) и результаты сотен футбольных матчей разных колледжей; в ней есть даже значения пикселей черно-белого варианта *Джоконды* (Моны Лизы) Леонардо да Винчи. Но, пожалуй, самое важное то, что SGB содержит набор пятибуквенных английских слов, которые мы и рассмотрим следующими.

Пятибуквенные английские слова. Множество примеров в этой главе будут основываться на списке пятибуквенных слов

$$\texttt{aargh}, \texttt{abaca}, \texttt{abaci}, \texttt{aback}, \texttt{abaft}, \texttt{abase}, \texttt{abash}, \ldots, \texttt{zooms}, \texttt{zowie}. \tag{8}$$

(Всего в нем содержится 5757 слов — слишком много, чтобы перечислить здесь их все; но вы легко можете представить себе отсутствующие в (8) слова.) Это список, лично составленный автором книги между 1972 и 1992 годами, начиная с того времени, когда он понял, что такие слова представляют собой идеальные (`ideal`) данные для тестирования комбинаторных алгоритмов различных видов.

Этот список преднамеренно ограничен только истинными словами английского языка, в том смысле, что автор встречался с их реальным применением. Полные словари содержат тысячи слов, известных лишь особо посвященным, такие как `aalii`, `abamp`, ..., `zymin` и `zyxst`; но такие слова нужны, в первую очередь, игрокам в Scrabble®*. Однако применение таких слов уменьшает удовольствие тех, кто

* Скрэббл (от англ. *Scrabble — рыться в поисках чего-либо*) — настольная игра, в которую могут играть от 2 до 4 человек, выкладывая слова из имеющихся у них букв в поле размером 15×15. В русскоязычной среде известна под названием *Эрудит*. — *Примеч. пер.*

с ними не знаком. Поэтому в течение двадцати лет автор систематически записывал все слова, которые казались ему подходящими для дидактических целей *Искусства программирования*.

Наконец, эту коллекцию стало просто необходимо "заморозить" и прекратить пополнять хотя бы для того, чтобы иметь возможность получать воспроизводимые результаты экспериментов. Английский язык продолжает эволюционировать, но 5757 слов всегда останутся неизменными — несмотря на то, что временами автор порывался добавить к списку слова, с которыми он не был знаком в 1992 году, такие как `chads`, `stent`, `blogs`, `ditzy`, `phish`, `bling` и, возможно, `tetch`. Но это не было сделано — время для внесения изменений в SGB закончилось.

Этот словарь предназначен для хранения всех хорошо известных английских слов... которые можно использовать в приличном обществе и которые могут служить в качестве звеньев.

— ЛЬЮИС КЭРРОЛЛ (LEWIS CARROLL), *Doublets: A Word-Puzzle* (1879)

Собственные имена наподобие `Knuth` не рассматривались как корректные пятибуквенные слова. Однако, например, такое слово, как `gauss`, корректно, поскольку "gauss" представляет собой единицу измерения магнитной индукции. Слова в SGB состоят только из обычных строчных букв; в списке нет слов с переносами, сокращений или слов, требующих ударения, наподобие `blasé`. Таким образом, каждое слово можно рассматривать и как вектор с пятью компонентами в диапазоне $[0\,..\,26)$. В векторном смысле слова `yucca` и `abuzz` отстоят друг от друга дальше всех: евклидово расстояние между ними равно

$$\|(24,20,2,2,0)-(0,1,20,25,25)\|_2=\sqrt{24^2+19^2+18^2+23^2+25^2}=\sqrt{2415}.$$

Полностью Stanford GraphBase со всеми программами и данными можно загрузить с веб-сайта автора `http://www-cs-faculty.stanford.edu/~knuth/sgb.html`. Получить список всех слов SGB еще проще, так как он находится по тому же адресу, но в файле '`sgb-words.txt`'. Этот файл содержит 5757 строк, в каждой из которых имеется одно слово, начиная с '`which`' и заканчивая '`pupal`'. Слова расположены в порядке, соответствующем их частоте употребления; например, словами с рангами 1000, 2000, 3000, 4000 и 5000 являются соответственно `ditch`, `galls`, `visas`, `faker` и `pismo`. Запись '`WORDS`(n)' в этой главе будет использоваться для обозначения n наиболее часто встречающихся слов, в соответствии с их рангами.

Кстати, пятибуквенные слова часто включают множественное число *четырехбуквенных слов*, что, конечно, недопустимо при строгом подходе к делу, и такие слова не встретятся вам в *The Official Scrabble® Players Dictionary*, но для словаря SGB они вполне годятся. Один из способов гарантировать, что семантически неподходящие термины не попадут в статью, основанную на списке слов SGB, — ограничить рассмотрение множеством `WORDS`(n), где n равно, скажем, 3000.

Упр. 26–37 могут использоваться в качестве разминки для начального изучения слов SGB, с которыми мы встретимся в этой главе в самых разных комбинаторных контекстах. Например, пока мы еще недалеко ушли от задач покрытия, мы можем заметить, что четыре слова '`third flock began jumps`' охватывают 20 из первой 21 буквы алфавита. Пять же слов могут охватить не более 24 различных букв, как

в случае {becks, fjord, glitz, nymph, squaw}, если только мы не обратимся к редкому, отсутствующему в SGB слову наподобие waqfs (пожертвование для мусульманских религиозных или благотворительных целей), которое можно скомбинировать со словами {gyved, bronx, chimp, klutz} для охвата 25 букв.

Простых слов из WORDS(400) достаточно для образования *квадрата из слов*:

```
class
light
agree .
sheep
steps
```
(9)

Однако для получения *куба из слов* требуется добраться почти до WORDS(3000):

```
types  yeast  pasta  ester  start
yeast  earth  armor  stove  three
pasta  armor  smoke  token  arena .
ester  stove  token  event  rents
start  three  arena  rents  tease
```
(10)

Здесь каждый "срез" 5×5 представляет собой квадрат из слов. С помощью простого расширения базового алгоритма танцующих связей (см. раздел 7.2.2.2) можно показать ценой около 390 миллиардов обращений к памяти, что WORDS(3000) дает возможность получить только три симметричных куба из слов, таких как (10); в упр. 36 вы найдете оставшиеся два. Интересно, что из полного множества WORDS(5757) можно создать 83 576 симметричных кубов.

Графы из слов. Конечно, интересно и важно помещать объекты в строки, квадраты, кубы и прочие фигуры; но в практических приложениях имеется еще *более* интересная и важная комбинаторная структура, а именно *граф*. Вспомним из раздела 2.3.4.1, что граф представляет собой множество точек, именуемых *вершинами*, вместе с множеством линий, называющихся *ребрами*, которые соединяют определенные пары вершин. Графы применяются повсеместно, и имеется масса красивых алгоритмов, предназначенных для работы с ними, так что графы естественным образом оказываются в центре внимания многих разделов данной главы. Фактически Stanford GraphBase, в первую очередь, посвящена графам, что отражено даже в ее названии; а слова SGB собирались автором главным образом потому, что их можно использовать для определения интересных и поучительных графов.

Льюис Кэрролл (Lewis Carroll) указал путь, изобретя в конце 1877 года игру "Дублеты" (Doublets). [См. Martin Gardner, *The Universe in a Handkerchief* (1996), Chapter 6.] Идея Кэрролла, которая быстро завоевала популярность, заключалась в преобразовании одного слова в другое путем замены на каждом шаге одной буквы; например, вот как слезы (tears) превращаются в улыбку (smile):*

tears — sears — stars — stare — stale — stile — smile. (11)

Кратчайшая такая трансформация является кратчайшим *путем* в графе, в котором вершинами являются английские слова, а ребра соединяют пары слов, расстояние

* Вот как по тем же правилам в русском языке муха превращается в слона: муха — муза — луза — лоза — коза — кора — кара — каре — кафе — кафр — каюр — каюк — крюк — урюк — урок — срок — сток — стон — слон. — *Примеч. пер.*

Хэмминга между которыми равно 1 (т. е. они отличаются одно от другого только одной буквой).

Если ограничиться только словами из SGB, правило Кэрролла дает граф из Stanford GraphBase, официальное название которого — $words(5757, 0, 0, 0)$. Каждый граф, определяемый SGB, имеет свой уникальный *идентификатор*, и "кэрроллообразный" граф, порожденный из слов SGB, идентифицируется как $words(n, l, t, s)$. Здесь n — количество вершин графа; l либо равно 0, либо представляет собой список весов, используемый для акцентирования различных видов словаря; t представляет собой пороговое значение, с тем чтобы можно было отвергнуть слова с малыми весами; а s — исходное значение для генерации псевдослучайных чисел, которые могут потребоваться для разбиения связей между словами с одинаковыми весами. Детальная информация в данном случае нас не интересует; нескольких примеров вполне достаточно для представления об общей идее.

- $words(n, 0, 0, 0)$ — граф, получающийся при применении правила Кэрролла к множеству $\texttt{WORDS}(n)$, для $1 \le n \le 5757$.
- $words(1000, \{0, 0, 0, 0, 0, 0, 0, 0, 0\}, 0, s)$ содержит 1000 выбранных случайным образом слов SGB, обычно разных при разных значениях s.
- $words(766, \{0, 0, 0, 0, 0, 0, 0, 1, 0\}, 1, 0)$ содержит все пятибуквенные слова из книг автора о TEX и METAFONT.

В последнем графе всего лишь 766 слов, так что мы не сможем образовать очень много длинных путей наподобие (11), хотя

$$\texttt{basic} — \texttt{basis} — \texttt{bases} — \texttt{based} — \texttt{baked} — \texttt{naked} — \texttt{named} — \texttt{names} — \texttt{games} \tag{12}$$

является одним из сто́ящих внимания примеров.

Конечно, имеется много других способов определения ребер графа, вершины которого представляют пятибуквенные слова. Можно, например, вместо расстояния Хэмминга использовать расстояние Евклида. Можно также объявить два слова смежными, если у них имеется общее подслово длиной 4 буквы; эта стратегия существенно обогащает граф, делая возможным превращение `chaos` в `peace`, даже ограничиваясь 766 словами, связанными с TEX:

$$\texttt{chaos} — \texttt{chose} — \texttt{whose} — \texttt{whole} — \texttt{holes} — \texttt{hopes} — \texttt{copes} — \texttt{scope} — \texttt{score} — \texttt{store} — \texttt{stare} — \texttt{spare} — \texttt{space} — \texttt{paces} — \texttt{peace}. \tag{13}$$

(Использованное правило позволяет удалить букву, а затем вставить другую — возможно, в другом месте.) Мы можем выбрать и совершенно иную стратегию, такую как размещение ребра между словами-векторами $a_1a_2a_3a_4a_5$ и $b_1b_2b_3b_4b_5$ тогда и только тогда, когда их скалярное произведение $a_1b_1+a_2b_2+a_3b_3+a_4b_4+a_5b_5$ кратно некоторому параметру m. Алгоритмы для работы с графами умеют одинаково успешно справляться с разными видами данных.

Слова SGB приводят также к интересному семейству *ориентированных* графов, если создавать ребра $a_1a_2a_3a_4a_5 \to b_1b_2b_3b_4b_5$ тогда, когда $\{a_2, a_3, a_4, a_5\} \subseteq \{b_1, b_2, b_3, b_4, b_5\}$ при рассмотрении слов как мультимножеств. (Удалить первую букву, добавить другую и переупорядочить.) Применение этого правила позволяет, например, преобразовать `words` в `graph` с помощью кратчайшего ориентированного

пути длиной 6:

$$\texttt{words} \to \texttt{dross} \to \texttt{soars} \to \texttt{orcas} \to \texttt{crash} \to \texttt{sharp} \to \texttt{graph}. \tag{14}$$

Теория — первый член ряда Тейлора для практики.

— ТОМАС М. КОВЕР (THOMAS M. COVER) (1992)

Количество терминологических систем, используемых в настоящее время в теории графов, неплохо аппроксимируется количеством теоретиков в этой области.

— РИЧАРД П. СТЕНЛИ (RICHARD P. STANLEY) (1986)

Основы теории графов. Граф G состоит из множества вершин V и множества ребер E, которые представляют собой пары различных вершин. Мы будем считать, что V и E являются *конечными* множествами, если только явно не указано иное. Мы пишем u — v, если u и v являются вершинами, такими, что $\{u, v\} \in E$, и $u \not\text{—} v$, если u и v являются вершинами, такими, что $\{u, v\} \notin E$. Вершины с u — v называются соседними, или смежными, в G. Одно из следствий этого определения заключается в том, что u — v тогда и только тогда, когда v — u. Другое следствие заключается в том, что $v \not\text{—} v$ для всех $v \in V$; другими словами, ни одна из вершин не смежна сама себе. (Однако ниже мы рассмотрим мультиграфы, в которых разрешены петли, идущие из вершины в саму себя.)

Граф $G' = (V', E')$ является *подграфом* $G = (V, E)$, если $V' \subseteq V$ и $E' \subseteq E$. Это *остовный* подграф G, если в действительности $V' = V$. И это *порожденный* (*индуцированный*) подграф G, если, когда V' представляет собой заданное подмножество вершин, E' имеет максимально возможное количество ребер. Другими словами, когда $V' \subseteq V$, подграфом $G = (V, E)$, порожденным V', является $G' = (V', E')$, где

$$E' \;=\; \bigl\{\, \{u, v\} \;\bigm|\; u \in V',\, v \in V' \text{ и } \{u, v\} \in E \,\bigr\}. \tag{15}$$

Этот подграф G' обозначается как $G \mid V'$ и часто называется "G, ограниченный V'". В распространенном случае, когда $V' = V \setminus \{v\}$, мы записываем просто $G \setminus v$ ("G минус вершина v") как сокращение для $G \mid (V \setminus \{v\})$. Аналогичная запись $G \setminus e$ (где $e \in E$) используется для обозначения подграфа $G' = (V, E \setminus \{e\})$, получаемого путем удаления ребра, а не вершины. Заметим, что все описанные ранее SGB-графы *words* (n, l, t, s) являются порожденными подграфами основного графа *words* $(5757, 0, 0, 0)$; в этих рассмотренных графах изменяется только словарь, но не правило смежности.

Граф с n вершинами и e ребрами называется имеющим *порядок* n и *размер* e. Простейшими и наиболее важными графами порядка n являются *полный граф* K_n, *путь* P_n и *цикл* C_n. Предположим, что вершинами являются $V = \{1, 2, \ldots, n\}$. Тогда

- K_n имеет $\binom{n}{2} = \frac{1}{2}n(n-1)$ ребер u — v для $1 \le u < v \le n$; каждый граф с n вершинами является остовным подграфом K_n;
- P_n имеет $n-1$ ребер v — $(v{+}1)$ для $1 \le v < n$, где $n \ge 1$; это — путь длиной $n{-}1$ от 1 до n;
- C_n имеет n ребер v — $((v \bmod n){+}1)$ для $1 \le v \le n$, где $n \ge 1$; он является графом, только когда $n \ge 3$ (C_1 и C_2 являются мультиграфами).

В действительности можно определить K_n, P_n и C_n на вершинах $\{0, 1, \ldots, n-1\}$ или на *любом* n-элементном множестве V вместо $\{1, 2, \ldots, n\}$, поскольку два графа, отличающихся только именами вершин, но не структурой ребер, комбинаторно эквивалентны.

Формально графы $G = (V, E)$ и $G' = (V', E')$ *изоморфны*, если существует взаимно однозначное соответствие φ между V и V', такое, что u — v в G тогда и только тогда, когда $\varphi(u)$ — $\varphi(v)$ в G'. Для указания изоморфности графов G и G' часто используется обозначение $G \cong G'$; впрочем, часто мы будем менее строги, рассматривая изоморфные графы, как если бы они были эквивалентными, и время от времени записывая $G = G'$, даже когда множества вершин G и G' не строго идентичны.

Небольшие графы можно определить, просто начертив диаграмму, в которой вершины изображены маленькими кружка́ми, а ребра — линиями между ними. На рис. 2 показано несколько важных примеров, свойства которых мы изучим позже. Граф Петерсена на рис. 2, (д) назван по имени Юлиуса Петерсена (Julius Petersen), одного из основателей теории графов, который использовал его для опровержения одного правдоподобного предположения [L'*Intermédiaire des Mathématiciens* **5** (1898), 225–227]; в действительности это замечательная конфигурация, которая служит контрпримером для многих оптимистических предсказаний о том, что может оказаться истинно для всех графов в целом. Граф Чватала (см. рис. 2, (е)) описан Вацлавом Чваталом (Václav Chvátal) в *J. Combinatorial Theory* **9** (1970), 93–94.

Рис. 2. Шесть примеров графов с $(5, 5, 5, 8, 10, 12)$ вершинами и $(4, 5, 10, 12, 15, 24)$ ребрами соответственно.

Линии на диаграмме могут пересекаться одна с другой в точках, не являющихся вершинами. Например, центральная точка на рис. 2, (е) *не* является вершиной графа Чватала. Граф называется *планарным*, если существует способ изобразить его без каких бы то ни было пересечений. Ясно, что P_n и C_n всегда планарны; представленная на рис. 2, (г) диаграмма показывает, что трехмерный куб также является планарным графом. Но K_5 имеет слишком много вершин, чтобы быть планарным (см. упр. 46).

Степенью вершины является количество ее соседей. Если все вершины имеют одну и ту же степень, граф называется *регулярным*. На рис. 2, например, P_5 является нерегулярным графом, так как две его вершины имеют степень 1, а три — степень 2. Остальные пять графов регулярны, со степенями соответственно $(2, 4, 3, 3, 4)$. Регулярный граф степени 3 часто называется кубическим или трехвалентным.

Имеется много способов изобразить тот или иной граф, и одни из них более выразительны, чем другие. Например, каждая из шести диаграмм

(16)

изоморфна трехмерному кубу (рис. 2, (г)). Диаграмма графа Чватала, показанная на рис. 2, (е) и раскрывающая неожиданную симметрию графа, придумана Адрианом Бонди (Adrian Bondy) через много лет после публикации статьи Чватала.

Симметриями графа, известными также как его *автоморфизмы*, являются перестановки его вершин, сохраняющие отношение смежности. Другими словами, перестановка φ является автоморфизмом G, если $\varphi(u)$ — $\varphi(v)$, когда u — v в графе G. Хорошо продуманный чертеж наподобие рис. 2, (е) может выявить симметрию графа, но одна диаграмма не всегда в состоянии показать все существующие симметрии. Например, трехмерный куб имеет 48 автоморфизмов, а граф Петерсена — 120. Мы изучим алгоритмы для работы с изоморфизмами и автоморфизмами в разделе 7.2.3. Симметрии часто помогают избавиться от излишних вычислений, делая алгоритмы при работе с графами, обладающими k автоморфизмами, почти в k раз быстрее.

Графы, связанные с объектами реального мира, обычно существенно отличаются от чисто математических графов наподобие приведенных на рис. 2. Например, вот достаточно знакомый американцам граф, вовсе не имеющий симметрии, но зато обладающий достоинством планарности.

(17)

Он представляет внутренние границы США, и позже мы используем его в нескольких примерах. Для удобства вместо изображения пустых кружков 49 вершин на этой диаграмме помечены двухбуквенными кодами штатов.*

* *От переводчика*: пожалуй, читателям переводного издания книги более знакомыми окажутся подобные географические графы, представляющие, например, республики бывшего СССР:

(17, с)

Пути и циклы. Остовный путь P_n графа называется *гамильтоновым путем*, и остовный цикл C_n — *гамильтоновым циклом*, по имени У. Р. Гамильтона (W. R. Hamilton), предложившего в 1856 году головоломку, состоящую в поиске таких путей по ребрам додекаэдра. Независимо от него эту задачу для многогранников в общем случае изучал Т. П. Киркман (T. P. Kirkman) в *Philosophical Transactions* **146** (1856), 413–418; **148** (1858), 145–161. [См. N. L. Biggs, E. K. Lloyd, R. J. Wilson *Graph Theory 1736–1936* (1998), Chapter 2.] Однако задача поиска остовного пути или цикла гораздо старше; на самом деле ее можно рассматривать как наиболее старую из комбинаторных задач, поскольку поиск путей обхода шахматной доски конем имеет долгую историю, ведущую вглубь веков — в Индию IX века (см. раздел 7.3.3). Граф называется *гамильтоновым*, если он имеет гамильтонов цикл. (Граф Петерсена, кстати, является наименьшим 3-регулярным графом, не являющимся ни планарным, ни гамильтоновым; см. C. de Polignac, *Bull. Soc. Math. de France* **27** (1899), 142–145.)

Обхватом (*girth*) графа называется длина его наименьшего цикла; обхват ацикличного графа (не содержащего циклов) бесконечен. Например, шесть графов на рис. 2 имеют обхваты $(\infty, 5, 3, 4, 5, 4)$ соответственно. Нетрудно доказать, что граф с минимальной степенью k и обхватом 5 должен иметь как минимум k^2+1 вершин. Дальнейший анализ показывает, что фактически этот минимум достижим, только если $k = 2$ (C_5), $k = 3$ (граф Петерсена), $k = 7$ или, вероятно, $k = 57$ (см. упр. 63 и 65.)

Расстоянием $d(u,v)$ между двумя вершинами u и v является минимальная длина пути от u до v в графе; оно равно бесконечности, если такого пути не существует. Ясно, что $d(v,v) = 0$ и что $d(u,v) = d(v,u)$. Справедливо также неравенство треугольника

$$d(u,v) + d(v,w) \;\geq\; d(u,w). \tag{18}$$

Ибо, если $d(u,v) = p$, $d(v,w) = q$, $p < \infty$ и $q < \infty$, то существуют пути

$$u = u_0 — u_1 — \cdots — u_p = v \quad \text{и} \quad v = v_0 — v_1 — \cdots — v_q = w, \tag{19}$$

и можно найти наименьший индекс r, такой, что $u_r = v_s$ для некоторого s. Тогда

$$u_0 — u_1 — \cdots — u_{r-1} — v_s — v_{s+1} — \cdots — v_q \tag{20}$$

представляет собой путь длиной $\leq p+q$ от u до w.

Диаметром графа является максимум $d(u,v)$ по всем вершинам u и v. Граф называется *связным*, если его диаметр конечен. Вершины графа всегда могут быть

или области Украины (соответствующий граф для России занял бы слишком много места):

(17, у)

(Здесь вершины графа помечены областными центрами и столицей автономной республики Крым, а не названиями соответствующих областей.)

разбиты на связные *компоненты*, где две вершины, u и v, принадлежат одному и тому же компоненту тогда и только тогда, когда $d(u,v) < \infty$.

В графе *words*$(5757, 0, 0, 0)$, например, $d(\texttt{tears}, \texttt{smile}) = 6$, поскольку (11) представляет собой кратчайший путь от `tears` к `smile`. Кроме того, $d(\texttt{tears}, \texttt{happy}) = 6$, $d(\texttt{smile}, \texttt{happy}) = 10$ и $d(\texttt{world}, \texttt{court}) = 6$. Но $d(\texttt{world}, \texttt{happy}) = \infty$; таким образом, рассматриваемый граф не является связным. В действительности он содержит 671 слово наподобие `aloof`, которые не имеют соседей и сами по себе образуют связные компоненты порядка 1. Пары слов наподобие `alpha` — `aloha`, `droid` — `druid` и `opium` — `odium` дают еще 103 связных компонента порядка 2. Одни компоненты порядка 3 наподобие `chain` — `chair` — `choir` являются путями; другие, такие как $\{\texttt{getup}, \texttt{letup}, \texttt{setup}\}$, представляют собой циклы. Имеется также немного большее количество небольших компонентов, наподобие занимательного пути

$$\texttt{login} — \texttt{logic} — \texttt{yogic} — \texttt{yogis} — \texttt{yogas} — \texttt{togas}, \tag{21}$$

слова в котором не имеют других соседей. Но подавляющее большинство пятибуквенных слов принадлежит гигантскому компоненту порядка 4493. Если вы можете пройти два шага от имеющегося у вас слова, меняя две различные буквы, то шансы, что это слово принадлежит указанному гигантскому компоненту, превышают 15 к 1.

Аналогично граф *words*$(n, 0, 0, 0)$ имеет при $n = (5000, 4000, 3000, 2000, 1000)$ гигантский компонент порядка $(3825, 2986, 2056, 1186, 224)$ соответственно. Но если n мало́, то ребер для связности оказывается недостаточно. Например, *words*$(500, 0, 0, 0)$ имеет 327 различных компонентов, причем среди них нет ни одного компонента порядка 15 или выше.

Концепцию расстояния можно обобщить и рассматривать величину $d(v_1, v_2, \ldots, v_k)$ для любого значения k, которая представляет собой минимальное количество ребер в связанном подграфе, содержащем вершины $\{v_1, v_2, \ldots, v_k\}$. Например, $d(\texttt{blood}, \texttt{sweat}, \texttt{tears})$ оказывается равным 15, поскольку подграф

$$\begin{array}{ccccccccccccc}
\texttt{blood} & — & \texttt{brood} & — & \texttt{broad} & — & \texttt{bread} & — & \texttt{tread} & — & \texttt{treed} & — & \texttt{tweed} \\
 & & & & & & & & & & | & & | \\
\texttt{tears} & — & \texttt{teams} & — & \texttt{trams} & — & \texttt{trims} & — & \texttt{tries} & — & \texttt{trees} & & \texttt{tweet} \\
 & & & & & & & & & & & & | \\
 & & & & & & & & & & \texttt{sweat} & — & \texttt{sweet}
\end{array} \tag{22}$$

имеет 15 ребер, а подходящего подграфа из 14 ребер не существует.

В разделе 2.3.4.1 мы упоминали, что связный граф с наименьшим количеством ребер называется *свободным деревом*. Подграф, соответствующий обобщенному расстоянию $d(v_1, \ldots, v_k)$, всегда будет свободным деревом. Его ошибочно называют *деревом Штейнера*, поскольку Якоб Штейнер (Jacob Steiner) однажды упомянул случай $k = 3$ для точек $\{v_1, v_2, v_3\}$ на евклидовой плоскости [*Crelle* **13** (1835), 362–363]. Франц Хайнен (Franz Heinen) решил эту задачу в *Über Systeme von Kräften* (1834); Гаусс (Gauss) распространил анализ на случай $k = 4$ в письме к Шумахеру (Schumacher) от 21 марта 1836 года.

Раскраска графа. Граф называется *k-дольным*, или *k-раскрашиваемым*, если его вершины могут быть разбиты на k или менее частей так, что концы каждого ребра принадлежат различным частям — или, что то же самое, если имеется способ раскрасить его вершины не более чем k различными цветами, так, чтобы смежные

вершины не были окрашены в один и тот же цвет. Знаменитая Теорема четырех красок, сформулированная в 1852 году Ф. Гутри (F. Guthrie) и окончательно доказанная с применением компьютера К. Аппелем (K. Appel), В. Хакеном (W. Haken) и Д. Кохом (J. Koch) [*Illinois J. Math.* **21** (1977), 429–567], гласит, что *любой планарный граф является 4-раскрашиваемым.* Простое доказательство этой теоремы неизвестно, но частные случаи наподобие (17) могут быть раскрашены на глаз (см. упр. 45); в общем случае для 4-раскрашивания планарного графа достаточно $O(n^2)$ шагов [N. Robertson, D. P. Sanders, P. Seymour и R. Thomas, *STOC* **28** (1996), 571–575].

На практике особенно важен случай 2-раскрашиваемых графов. Такие графы в общем случае называются *двудольными* или просто *биграфами*; каждое ребро такого графа имеет по одной конечной точке в каждой из частей.

Теорема B. *Граф является двудольным тогда и только тогда, когда он не содержит циклов нечетной длины.*

Доказательство. [См. D. König, *Math. Annalen* **77** (1916), 453–454.] Каждый подграф k-дольного графа является k-дольным. Следовательно, цикл C_n может быть подграфом двудольного графа, только если C_n сам является двудольным, а в этом случае n должно быть четным.

И наоборот, если граф не содержит циклов нечетной длины, мы можем раскрасить его вершины двумя цветами $\{0,1\}$, придерживаясь следующей процедуры. Изначально все вершины не окрашены. Если все соседи окрашенных вершин уже окрашены, выберем неокрашенную вершину w и окрасим ее в цвет 0. В противном случае выберем окрашенную вершину u, которая имеет неокрашенного соседа v; назначим v противоположный u цвет. В упр. 48 доказывается, что в конечном итоге при этом получится корректная раскраска. ▮

Полный двудольный граф $K_{m,n}$ представляет собой наибольший двудольный граф, вершины которого разделены на две части размерами m и n. Его можно определить на множестве вершин $\{1,2,\ldots,m+n\}$, говоря, что u — v тогда, когда $1 \le u \le m < v \le m+n$. Другими словами, $K_{m,n}$ имеет mn ребер, по одному для каждого выбора одной вершины из одной части и второй — из другой. Аналогично *полный k-дольный граф* $K_{n_1,\ldots,n_k}$ имеет $N = n_1 + \cdots + n_k$ вершин, разделенных на части размерами $\{n_1,\ldots,n_k\}$, и имеет ребра между любыми двумя вершинами, не принадлежащими одной и той же части. Вот несколько примеров для $N = 6$:

(23)

Обратите внимание, что $K_{1,n}$ представляет собой свободное дерево; его часто называют *звездой* порядка $n+1$.

Далее я буду говорить "орграф" вместо "ориентированный граф".
Это просто и кратко и явно войдет в моду.

— ГЁРГИ ПОЙА (GYÖRGY PÓLYA),
письмо ФРЭНКУ ХАРАРИ (FRANK HARARY) (ок. 1954 года)

Ориентированные графы. В разделе 2.3.4.2 мы определили *ориентированные графы* (или *орграфы*), которые очень похожи на графы, с тем отличием, что вместо ребер они имеют *дуги*. Дуга $u \longrightarrow v$ направлена от одной вершины к другой, в то время как ребро u — v соединяет две вершины, не различая их. Кроме того, ориентированные графы могут иметь петли $v \longrightarrow v$, идущие из вершины в нее же, а также между двумя вершинами u и v может находиться более одной дуги $u \longrightarrow v$.

Формально ориентированным графом $D = (V, A)$ порядка n и размера m являются множество V из n вершин и мультимножество A из m упорядоченных пар (u, v), где $u \in V$ и $v \in V$. Упорядоченные пары называются дугами, и мы записываем $u \longrightarrow v$, когда $(u, v) \in A$. Ориентированный граф называется *простым*, если A является множеством, а не мультимножеством, т. е. если для всех u и v имеется не более одной дуги (u, v). Каждая дуга (u, v) имеет начальную вершину u и конечную вершину v, именуемую также верхушкой (tip). Каждая вершина имеет *исходящую степень* $d^+(v)$, равную количеству дуг, для которых v является начальной вершиной, и *входящую степень* $d^-(v)$, равную количеству дуг, для которых v является верхушкой. Вершина с нулевой входящей степенью называется "источником", а вершина с нулевой исходящей степенью называется "стоком". Заметим, что $\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v)$, поскольку обе суммы равны m — общему количеству дуг.

Большинство концепций, определенных для графов, естественным путем распространяются и на ориентированные графы. Достаточно просто вставить слово "ориентированный", когда возникает необходимость различать ребра и дуги. Например, ориентированные графы имеют ориентированные подграфы, которые могут быть остовными, порожденными или ни теми, ни другими. Изоморфизмом между ориентированными графами $D = (V, A)$ и $D' = (V', A')$ является взаимно однозначное соответствие φ между V и V', при котором количество дуг $u \longrightarrow v$ в D равно количеству дуг $\varphi(u) \longrightarrow \varphi(v)$ в D' для всех $u, v \in V$.

На диаграммах ориентированных графов вместо линий между вершинами используются стрелки. Простейшими и наиболее важными ориентированными графами порядка n являются ориентированные варианты графов K_n, P_n и C_n, а именно *транзитивный турнир* $\vec{K_n}$, *ориентированный путь* $\vec{P_n}$ и *ориентированный цикл* $\vec{C_n}$. Схематически для $n = 5$ они показаны на следующих диаграммах:

$\vec{K_5}$ $\vec{P_5}$ $\vec{C_5}$ (24)

Имеется также *полный ориентированный граф* J_n, который представляет собой наибольший простой ориентированный граф с n вершинами; он имеет n^2 дуг $u \longrightarrow v$, по одной для каждого варианта выбора u и v.

На рис. 3 показана более сложная диаграмма порядка 17, которую можно назвать отчетливо ориентированной: это ориентированный граф, описанный Эркюлем Пуаро (Hercule Poirot) в детективе Агаты Кристи (Agatha Christie) *Убийство в Восточном экспрессе* (1934).* Вершины соответствуют местам в описанном в детективе

* Непереводимая игра слов: "отчетливо ориентированная" ("expressly oriented") перекликается с оригинальным названием повести А. Кристи *Murder on the Orient Express*. Имена персонажей на диаграмме взяты из перевода Л. Беспаловой (Агата Кристи. *Восточный экспресс. Десять негритят. Романы.* — М., 1990). — *Примеч. пер.*

Условные обозначения
0 — Пьер Мишель, француз, проводник
1 — Эркюль Пуаро, бельгиец, детектив
2 — Сэмьюэл Эдуард Рэтчетт, погибший американец
3 — Каролина Марта Хаббард, американка, домохозяйка
4 — Эдуард Генри Мастермэн, англичанин, слуга
5 — Антонио Фоскарелли, итальянец, агент по продажам
6 — Гектор Уиллард Маккуин, американец, секретарь
7 — Харви Харрис, англичанин, не явился
8 — Хильдегарда Шмидт, немка, горничная
9 — (Пустое место)
10 — Грета Ольсон, шведка, экономка
11 — Мэри Хермиона Дебенхэм, англичанка, гувернантка
12 — Хелена Мария Андрени, прекрасная графиня
13 — Рудольф Андрени, венгр, граф, дипломат
14 — Наталья Драгомирова, русская, княгиня
15 — Полковник Арбэтнот, англичанин, офицер из Индии
16 — Сайрус Бетман Хардман, американец, детектив

Рис. 3. Ориентированный граф порядка 17 размером 18, придуманный Агатой Кристи.

спальном вагоне Стамбул–Кале, а дуги $u \longrightarrow v$ означают, что пассажир с места u подтверждает алиби пассажира с места v. В этом графе имеется шесть связных компонентов, а именно $\{0,1,3,6,8,12,13,14,15,16\}$, $\{2\}$, $\{4,5\}$, $\{7\}$, $\{9\}$ и $\{10,11\}$, поскольку связность ориентированного графа определяется путем рассмотрения дуг как ребер.

Две дуги являются *последовательными*, если верхушка первой является начальной вершиной второй. Ряд последовательных дуг $(a_1, a_2, \ldots, a_k)$ называется *обходом* (*walk*) длиной k; его можно определить с помощью как дуг, так и вершин:

$$v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \cdots v_{k-1} \xrightarrow{a_k} v_k. \tag{25}$$

В простом ориентированном графе достаточно просто указать вершины; например, $1 \longrightarrow 0 \longrightarrow 8 \longrightarrow 14 \longrightarrow 8 \longrightarrow 3$ представляет собой обход для графа на рис. 3. Обход (25) представляет собой ориентированный путь, если вершины $\{v_0, v_1, \ldots, v_k\}$ различны; он является ориентированным циклом, если все вершины различны, за исключением $v_k = v_0$.

В ориентированном графе ориентированное расстояние $d(u,v)$ представляет собой количество дуг в кратчайшем *ориентированном* пути от u к v, которое также равно длине кратчайшего обхода от u к v. Оно может отличаться от $d(v,u)$; но неравенство треугольника (18) остается корректным.

Каждый граф может рассматриваться как ориентированный граф, поскольку ребро u — v по сути эквивалентно соответствующей паре дуг, $u \longrightarrow v$ и $v \longrightarrow u$. Ориентированный граф, полученный таким образом, обладает всеми свойствами исходного графа; например, степень каждой вершины графа становится исходящей степенью ориентированного графа, а также ее входящей степенью. Кроме того, расстояния остаются неизменными.

Мультиграф (V,E) подобен графу, с тем отличием, что его ребра E могут быть *мультимножеством* пар $\{u,v\}$; в мультиграфе разрешены также ребра v — v, представляющие собой петли из вершины в нее же, соответствующие “мультипарам” $\{v,v\}$. Например,

(26)

представляет собой мультиграф порядка 3 с 6 ребрами, $\{1,1\}$, $\{1,2\}$, $\{2,3\}$, $\{2,3\}$, $\{3,3\}$ и $\{3,3\}$. Степени вершин в этом примере равны $d(1) = d(2) = 3$ и $d(3) = 6$, поскольку каждая петля вносит в степень вершины вклад, равный 2. При рассмот-

рении мультиграфа как ориентированного графа петля v — v становится *двумя* дугами-петлями $v \longrightarrow v$.

Представление графов и ориентированных графов. Любой ориентированный граф, а значит, любой граф или мультиграф, полностью описывается его *матрицей смежности* $A = (a_{uv})$, имеющей n строк и n столбцов при наличии у ориентированного графа n вершин. Каждый элемент a_{uv} этой матрицы определяет количество дуг от u до v. Например, матрицами смежности для $K_3^{\rightarrow}$, $P_3^{\rightarrow}$, $C_3^{\rightarrow}$, J_3 и (26) являются соответственно

$$K_3^{\rightarrow} = \begin{pmatrix} 011 \\ 001 \\ 000 \end{pmatrix}, \quad P_3^{\rightarrow} = \begin{pmatrix} 010 \\ 001 \\ 000 \end{pmatrix}, \quad C_3^{\rightarrow} = \begin{pmatrix} 010 \\ 001 \\ 100 \end{pmatrix}, \quad J_3 = \begin{pmatrix} 111 \\ 111 \\ 111 \end{pmatrix}, \quad A = \begin{pmatrix} 210 \\ 102 \\ 024 \end{pmatrix}. \tag{27}$$

Мощный математический инструментарий теории матриц позволяет доказывать многие нетривиальные результаты для графов путем изучения их матриц смежности; в упр. 65 представлены особенно впечатляющие примеры того, что может быть сделано таким путем. Одной из основных причин этого является то, что умножение матриц в контексте ориентированных графов имеет простую интерпретацию. Рассмотрим квадрат A, в котором элемент на пересечении строки u и столбца v по определению равен

$$(A^2)_{uv} \;=\; \sum_{w \in V} a_{uw} a_{wv}. \tag{28}$$

Поскольку a_{uw} — количество дуг от u до w, мы видим, что $a_{uw}a_{wv}$ — количество обходов вида $u \longrightarrow w \longrightarrow v$. Следовательно, $(A^2)_{uv}$ является общим количеством обходов длиной 2 от u до v. Аналогично элементы A^k дают общее количество обходов длиной k между любыми упорядоченными парами вершин для всех $k \geq 0$. Например, матрица A в (27) дает

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 2 & 4 \end{pmatrix}, \qquad A^2 = \begin{pmatrix} 5 & 2 & 2 \\ 2 & 5 & 8 \\ 2 & 8 & 20 \end{pmatrix}, \qquad A^3 = \begin{pmatrix} 12 & 9 & 12 \\ 9 & 18 & 42 \\ 12 & 42 & 96 \end{pmatrix}; \tag{29}$$

таким образом, существует 12 обходов длиной 3 от вершины 1 мультиграфа (26) до вершины 3 и 18 таких обходов из вершины 2 назад в нее же.

Переупорядочение вершин заменяет матрицу смежности A на $P^{-}AP$, где P — матрица перестановки (0–1-матрица, в каждой строке и каждом столбце которой имеется ровно одна единица), а $P^{-} = P^T$ представляет собой матрицу для обратной перестановки. Таким образом,

$$\begin{pmatrix} 210 \\ 102 \\ 024 \end{pmatrix}, \quad \begin{pmatrix} 201 \\ 042 \\ 120 \end{pmatrix}, \quad \begin{pmatrix} 012 \\ 120 \\ 204 \end{pmatrix}, \quad \begin{pmatrix} 021 \\ 240 \\ 102 \end{pmatrix}, \quad \begin{pmatrix} 402 \\ 021 \\ 210 \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} 420 \\ 201 \\ 012 \end{pmatrix} \tag{30}$$

представляют собой все матрицы смежности для (26), и никаких других матриц смежности для этого мультиграфа не существует.

Имеется более чем $2^{n(n-1)/2}/n!$ графов порядка n, при $n > 1$, и почти все они требуют $\Omega(n^2)$ бит данных при наиболее экономичном кодировании. Следовательно, для подавляющего большинства всех возможных графов наилучший способ их представления в компьютере с точки зрения требующейся памяти — это работа с их матрицами смежности.

Однако графы, возникающие в практических задачах, по своим характеристикам существенно отличаются от графов, выбираемых случайным образом из

множества всех возможных. Такие реальные графы, как правило, оказываются "разреженными", имеющими, скажем, $O(n \log n)$ ребер вместо $\Omega(n^2)$, если только n не слишком мало́, поскольку $\Omega(n^2)$ бит данных сложно генерировать. Например, предположим, что вершины соответствуют людям, а ребра — отношению дружбы. Если мы рассмотрим 5 иллиардов людей, мало у кого из них будет более чем 10 тысяч друзей. Но даже если каждый в среднем имеет 10 тысяч друзей, то у графа все равно будет только 2.5×10^{13} ребер, в то время как почти все графы с порядком 5 миллиардов имеют примерно 6.25×10^{18} ребер.

Таким образом, наилучший способ представления графа в машине обычно оказывается отличным от записи n^2 значений a_{uv} матрицы смежности. Вместо этого алгоритмы Stanford GraphBase разработаны для работы со структурой данных, похожей на структуру для связного представления разреженных матриц, рассматривавшейся в разделе 2.2.6, хотя и несколько упрощенной. Такой подход не только доказанно универсален и эффективен, но и прост в использовании.

Представление ориентированного графа в SGB представляет собой комбинацию последовательного и связанного распределения с применением узлов двух базовых типов. Ряд узлов представляет вершины, другие представляют дуги. (Имеется и третий тип узлов, которые представляют весь граф в целом, предназначенный для алгоритмов, работающих одновременно с несколькими графами. Но каждый граф требует только одного узла графа, так что узлы вершин и дуг доминируют.)

Вот как это работает. Каждый ориентированный граф SGB порядка n и размера m построен на последовательном массиве из n узлов вершин, что облегчает доступ к вершине k для $0 \le k < n$. Напротив, m узлов дуг связаны вместе в общем пуле памяти, который, по сути, неструктурирован. Каждый узел вершины обычно занимает 32 байт, а каждый узел дуги — 20 байт (узел графа занимает 220 байт); но размеры узлов могут быть легко изменены. Несколько полей каждого узла фиксированы и имеют определенное значение в любом случае. Остальные поля могут использоваться для различных целей разными алгоритмами или на разных этапах одного и того же алгоритма. Фиксированные части узла называются стандартными полями, а многоцелевые части — сервисными полями.

Каждый узел вершины имеет два стандартных поля, `NAME` и `ARCS`. Если v — переменная, указывающая на узел вершины, мы называем ее *переменной вершины*. Тогда `NAME`(v) указывает на строку символов, которые могут использоваться для идентификации соответствующей вершины удобным для восприятия человеком способом; например, 49 вершин графа (17) имеют имена наподобие `CA`, `WA`, `OR`, ..., `RI`. Другое стандартное поле, `ARCS`(v), существенно важнее для алгоритмов: оно указывает на узел дуги, первый в односвязном списке длиной $d^+(v)$, в котором имеется по одному узлу для каждой дуги, исходящей из вершины v.

Каждый узел дуги имеет два стандартных поля, `TIP` и `NEXT`; переменная a, которая указывает на узел дуги, называется *переменной дуги*. `TIP`(a) указывает на узел вершины, представляющий верхушку дуги a; `NEXT`(a) указывает на узел дуги, представляющий следующую дугу, начальной вершиной которой является та же вершина, что и у дуги a.

Вершина v с исходящей степенью 0 представляется соответствующим узлом путем присваивания `ARCS`$(v) = \Lambda$ (нулевой указатель). В противном случае, если, скажем, исходящая степень равна 3, структура данных содержит три узла дуг

с $\mathtt{ARCS}(v) = a_1$, $\mathtt{NEXT}(a_1) = a_2$, $\mathtt{NEXT}(a_2) = a_3$ и $\mathtt{NEXT}(a_3) = \Lambda$; и эти три дуги из вершины v ведут в $\mathtt{TIP}(a_1)$, $\mathtt{TIP}(a_2)$, $\mathtt{TIP}(a_3)$.

Предположим, например, что мы хотим вычислить исходящую степень вершины v и сохранить ее в сервисном поле ODEG. Это просто.

$$\begin{array}{l}\text{Установить } a \leftarrow \mathtt{ARCS}(v) \text{ и } d \leftarrow 0.\\ \text{Пока } a \neq \Lambda\text{, устанавливать } d \leftarrow d+1 \text{ и } a \leftarrow \mathtt{NEXT}(a).\\ \text{Установить } \mathtt{ODEG}(v) \leftarrow d.\end{array} \tag{31}$$

Когда граф или мультиграф рассматривается как ориентированный граф, как упоминалось выше, каждое его ребро u — v эквивалентно двум дугам, $u \longrightarrow v$ и $v \longrightarrow u$. Эти дуги называются парными (mate) и занимают два узла дуг, скажем, a и a', где a появляется в списке дуг, исходящих из u, а a' — в списке дуг, исходящих из v. Тогда $\mathtt{TIP}(a) = v$ и $\mathtt{TIP}(a') = u$. Мы также записываем

$$\mathtt{MATE}(a) = a' \qquad \text{и} \qquad \mathtt{MATE}(a') = a \tag{32}$$

в алгоритмах, которые должны быстро переходить от одного списка к другому. Однако обычно нам не требуется хранить явно указатель от дуги на парную ей или использовать сервисное поле MATE в каждом узле дуги, поскольку необходимые связи можно вывести *неявно* при правильном построении структур данных.

Неявный вывод работает примерно следующим образом: при создании каждого ребра u — v неориентированного графа или мультиграфа мы вводим *последовательные* узлы дуг для $u \longrightarrow v$ и $v \longrightarrow u$. Например, если узлу дуги отводится 20 байт, мы резервируем 40 последовательных байтов для каждой новой пары. Мы можем также обеспечить, чтобы адрес в памяти первого байта был кратен 8. Тогда, если узел дуги a находится в памяти по адресу α, узел парной дуги находится по адресу

$$\left\{\begin{array}{ll}\alpha+20, & \text{если } \alpha \bmod 8 = 0\\ \alpha-20, & \text{если } \alpha \bmod 8 = 4\end{array}\right\} \;=\; \alpha - 20 + \big(40 \,\&\, ((\alpha \,\&\, 4) - 1)\big). \tag{33}$$

Такой трюк очень полезен в комбинаторных задачах, когда операции могут выполняться триллионы раз, поскольку при этом экономия в 3.6 нс на одной операции приведет к тому, что программа завершит работу на час раньше. Но способ (33) не является непосредственно "переносимым" из одной реализации в другую. Если размер узла дуги изменится, например, с 20 до 24, то числа 40, 20, 8 и 4 в (33) нужно будет заменить на 48, 24, 16 и 8 соответственно.

В алгоритмах в этой книге не делается никаких предположений о размерах узлов. Вместо этого мы примем соглашение языка программирования C и его потомков о том, что если a указывает на узел дуги, то '$a+1$' представляет собой указатель на узел дуги, следующий за первым в памяти. В общем случае

$$\mathtt{LOC}(\mathtt{NODE}(a+k)) \;=\; \mathtt{LOC}(\mathtt{NODE}(a)) + kc, \tag{34}$$

если размер каждого узла дуги равен c байт. Аналогично если v представляет собой переменную вершины, то '$v+k$' будет k-м узлом вершины, следующим за узлом, на который указывает v; фактический адрес памяти этого узла будет равен v плюс умноженное на размер узла вершины значение k.

Стандартные поля узла графа g включают общее количество дуг $\mathtt{M}(g)$ и общее количество вершин $\mathtt{N}(g)$; указатель на первый узел вершины в последовательном

списке всех узлов вершин VERTICES(g); идентификатор графа ID(g), представляющий собой строку наподобие words(5757,0,0,0); а также некоторые другие поля, необходимые для выделения и освобождения памяти при увеличении или уменьшении графа, или для экспорта графа во внешний формат для взаимодействия с другими пользователями и системами для работы с графами. Но нам редко придется как обращаться к этим полям узла графа, так и давать полное описание формата SGB, поскольку в этой главе мы будем описывать почти все алгоритмы для работы с графами с помощью словесного описания естественным языком на достаточно абстрактном уровне, а не на битовом уровне машинных программ.

Простой алгоритм для работы с графом. Для иллюстрации алгоритма средневысокого уровня наподобие тех, с которыми мы будем иметь дело позже, давайте превратим доказательство теоремы B в пошаговую процедуру, окрашивающую вершины данного графа в два цвета, если граф является двудольным.

Алгоритм B (*Проверка двудольности*). Для заданного графа, представленного в формате SGB, этот алгоритм либо находит 2-раскрашивание с COLOR(v) $\in \{0, 1\}$ в каждой вершине v, либо завершается неудачей, если корректное 2-раскрашивание невозможно. Здесь COLOR представляет собой сервисное поле в каждом узле вершины. Другое сервисное поле, LINK(v), представляет собой указатель на вершину и используется для поддержки стека всех окрашенных вершин, соседи которых еще не протестированы. Вспомогательная переменная вершины s указывает на вершину этого стека. Алгоритм также использует переменные u, v, w для вершин и a для дуг. Предполагается, что узлы вершин представляют собой $v_0 + k$ для $0 \le k < n$.

B1. [Инициализация.] Установить COLOR($v_0 + k$) $\leftarrow -1$ для $0 \le k < n$. (Сейчас все вершины не окрашены.) Затем установить $w \leftarrow v_0 + n$.

B2. [Выполнено?] (В этот момент все вершины $\ge w$ окрашены, и то же относится и к соседям всех окрашенных вершин.) Завершить работу алгоритма успешно, если $w = v_0$. В противном случае установить $w \leftarrow w - 1$ (следующему узлу вершины в направлении уменьшения индекса).

B3. [Окраска w при необходимости.] Если COLOR(w) ≥ 0, вернуться к шагу B2. В противном случае установить COLOR(w) $\leftarrow 0$, LINK(w) $\leftarrow \Lambda$ и $s \leftarrow w$.

B4. [Стек $\Rightarrow u$.] Установить $u \leftarrow s$, $s \leftarrow$ LINK(s), $a \leftarrow$ ARCS(u). (Мы будем проверять всех соседей окрашенной вершины u.)

B5. [Завершено с u?] Если $a = \Lambda$, перейти к шагу B8. В противном случае установить $v \leftarrow$ TIP(a).

B6. [Обработка v.] Если COLOR(v) < 0, установить COLOR(v) $\leftarrow 1 -$ COLOR(u), LINK(v) $\leftarrow s$ и $s \leftarrow v$. Если же COLOR(v) $=$ COLOR(u), завершить работу алгоритма неудачей.

B7. [Цикл по a.] Установить $a \leftarrow$ NEXT(a) и вернуться к шагу B5.

B8. [Стек не пустой?] Если $s \ne \Lambda$, вернуться к шагу B4. В противном случае вернуться к шагу B2. ▮

Этот алгоритм представляет собой вариант общей процедуры обхода графа, именуемой поиском в глубину, которую мы детально изучим в разделе 7.4.1. Время его работы составляет $O(m + n)$ при наличии m дуг и n вершин (см. упр. 70);

следовательно, он хорошо приспособлен к распространенному случаю разреженных графов. Путем небольших изменений в случае неудачного завершения алгоритм можно заставить выводить цикл нечетной длины, тем самым доказывая невозможность 2-раскрашивания (см. упр. 72).

Примеры графов. Stanford GraphBase включает библиотеку более чем из трех десятков подпрограмм генерации, способных создавать разнообразные ориентированные и неориентированные графы для использования в экспериментах. Мы уже рассматривали графы *слов*; давайте теперь взглянем на некоторые другие, чтобы увидеть имеющиеся возможности.

• *roget*$(1022, 0, 0, 0)$ представляет собой ориентированный граф с 1022 вершинами и 5075 дугами. Вершины представляют категории слов или концепций, которые П. М. Роджет (P. M. Roget) и Д. Л. Роджет (J. L. Roget) включили в свой знаменитый *Thesaurus* (London: Longmans, Green, 1879). Дуги представляет собой перекрестные ссылки между категориями, имеющиеся в книге. Например, типичными дугами являются `water` $\longrightarrow$ `moisture`, `discovery` $\longrightarrow$ `truth`, `fool` $\longrightarrow$ `sage`, `preparation` $\longrightarrow$ `learning`, `vulgarity` $\longrightarrow$ `ugliness`, `wit` $\longrightarrow$ `amusement`.

• *book*$(\texttt{'jean'}, 80, 0, 1, 356, 0, 0, 0)$ представляет собой граф с 80 вершинами и 254 ребрами. Вершины представляют персонажей романа *Отверженные* (*Les Misérables*) Виктора Гюго (Victor Hugo); ребра соединяют персонажей, встречавшихся в романе друг с другом. Типичными ребрами являются `Fantine` — `Javert`, `Cosette` — `Thénardier`.

• *bi_book*$(\texttt{'jean'}, 80, 0, 1, 356, 0, 0, 0)$ представляет собой двудольный граф с $80+356$ вершинами и 727 ребрами. Вершины представляют персонажей и главы упомянутого романа *Отверженные*; ребра соединяют персонажей с главами, в которых они появляются (например, `Napoleon` — `2.1.8`, `Marius` — `4.14.4`).

• *plane_miles*$(128, 0, 0, 0, 1, 0, 0)$ представляет собой планарный граф со 129 вершинами и 381 ребром. Вершины представляют 128 городов США и Канады, а также специальную вершину `INF` для "точки в бесконечности". Ребра определяют так называемую *триангуляцию Делоне* этих городов, основанную на их широте и долготе на плоскости; это означает, что u — v тогда и только тогда, когда существует окружность, проходящая через u и v и не охватывающая никакую иную вершину. Ребра имеются также между `INF` и всеми вершинами, лежащими на выпуклой оболочке всех местоположений городов. Типичными ребрами являются `Seattle, WA` — `Vancouver, BC` — `INF`; `Toronto, ON` — `Rochester, NY`.

• *plane_lisa*$(360, 250, 15, 0, 360, 0, 250, 0, 2295000)$ представляет собой планарный граф с 3027 вершинами и 5967 ребрами. Он получен с помощью обработки оцифрованного изображения *Моны Лизы* Леонардо да Винчи, имеющего 360 строк и 250 столбцов пикселей, состоящей в огрублении интенсивностей пикселей до 16 уровней серого от 0 (черный) до 15 (белый). Полученные в результате 3027 областей одинаковой яркости, связанных "ходом ладьи", рассматриваются затем как соседние, если имеют общую границу между пикселями (рис. 4).

• *bi_lisa*$(360, 250, 0, 360, 0, 250, 8192, 0)$ представляет собой двудольный граф с $360+250 = 610$ вершинами и 40 923 ребрами. Это другой взгляд на знаменитую картину Леонардо да Винчи, на этот раз связывающий строки и столбцы, на пересечении

Рис. 4. Цифровая копия *Моны Лизы* и крупный план фрагмента (который лучше рассматривать издали).

которых уровень яркости равен как минимум 1/8. Например, ребро `r102` — `c113` приходится прямо на средину "улыбки Джоконды".

• *raman*$(31, 23, 3, 1)$ представляет собой граф совершенно отличной от других рассмотренных ранее графов SGB природы. Вместо связывания объектов языка, литературы или иных продуктов человеческой культуры, этот так называемый "граф-расширитель Рамануджана" ("Ramanujan expander graph") основан на строгих математических принципах. Каждая из его $(23^3 - 23)/2 = 6072$ вершин имеет степень 32; следовательно, он имеет 97 152 ребра. Вершины соответствуют классам эквивалентности матриц 2×2, являющихся несингулярными по модулю 23; типичным ребром является `(2,7;1,1)` — `(4,6;1,3)`. Графы Рамануджана важны, в первую очередь, из-за необычно большого обхвата и малого диаметра для их размеров и степеней. У данного графа и обхват, и диаметр равны 4.

• *raman*$(5, 37, 4, 1)$ — аналогично регулярный граф степени 6 с 50 616 вершинами и 151 848 ребрами. Имеет обхват 10, диаметр 10, а кроме того, этот граф двудолен.

• *random_graph*$(1000, 5000, 0, 0, 0, 0, 0, 0, 0, s)$ представляет собой граф с 1000 вершинами, 5000 ребрами и исходным значением для генерации псевдослучайных чисел s. Граф "эволюционирует", начиная с безреберного состояния, путем многократного выбора псевдослучайных номеров вершин $0 \le u, v < 1000$ и добавления ребра u — v, если только не $u = v$ или такое ребро уже имеется в наличии. При $s = 0$ все вершины принадлежат гигантскому компоненту порядка 999, за исключением одной изолированной вершины 908.

• *random_graph*$(1000, 5000, 0, 0, 1, 0, 0, 0, 0, 0)$ представляет собой ориентированный граф с 1000 вершина и 5000 дугами, полученный с помощью аналогичной эволюции. (В действительности каждая из его дуг является также частью *random_graph*$(1000, 5000, 0, 0, 0, 0, 0, 0, 0, 0)$.)

• *subsets*$(5, 1, -10, 0, 0, 0, {}^{\#}1, 0)$ представляет собой граф с $\binom{11}{5} = 462$ вершинами, по одной для каждого пятиэлементного подмножества множества $\{0, 1, \ldots, 10\}$. Две

вершины являются смежными, если соответствующие подмножества непересекающиеся; таким образом, граф является регулярным со степенью 6 и имеет 1386 ребер. Его можно рассматривать как обобщение графа Петерсена, SGB-имя которого — *subsets*$(2, 1, -4, 0, 0, 0, {}^{\#}\mathtt{1}, 0)$.

• *subsets*$(5, 1, -10, 0, 0, 0, {}^{\#}\mathtt{10}, 0)$ имеет те же 462 вершины, но теперь вершины являются смежными, если соответствующие подмножества имеют четыре общих элемента. Данный граф регулярен со степенью 30 и имеет 6930 ребер.

• *parts*$(30, 10, 30, 0)$ представляет собой еще один SGB-граф с математической основой. Он имеет 3590 вершин, по одной для каждого разбиения 30 на не более чем 10 частей. Два разбиения рассматриваются как смежные, если одно из них получается путем деления части другого; это правило определяет 31 377 ребер. Ориентированный граф *parts*$(30, 10, 30, 1)$ аналогичен, но его 31 377 дуг направлены от более коротких к более длинным разбиениям (например, `13+7+7+3`$\longrightarrow$`7+7+7+6+3`).

• *simplex*$(10, 10, 10, 10, 10, 0, 0)$ представляет собой граф с 286 вершинами и 1320 ребрами. Его вершинами являются целочисленные решения $x_1 + x_2 + x_3 + x_4 = 10$, где $x_i \geq 0$, а именно "композиции 10 на четыре неотрицательные части"; их можно также рассматривать как барицентрические координаты точек внутри тетраэдра. Ребра, такие как `3.1.4.2` — `3.0.4.3`, соединяют композиции, которые максимально близки одна к другой.

• *board*$(8, 8, 0, 0, 5, 0, 0)$ и *board*$(8, 8, 0, 0, -2, 0, 0)$ представляют собой графы с 64 вершинами, 168 или 280 ребер которых соответствуют ходам коня или слона в шахматах.

Несметное количество других примеров можно легко получить, варьируя параметры генераторов графов SGB. Например, на рис. 5 показаны два простых варианта *board* и *simplex*; несколько загадочные правила *board* поясняются в упр. 75.

board$(6, 9, 0, 0, 5, 0, 0)$
(ходы коня на доске размером 6×9)

simplex$(10, 8, 7, 6, 0, 0, 0)$
(усеченная треугольная решетка)

Рис. 5. Образцы SGB-графов, связанных с настольными играми.

Алгебра графов. Мы также можем получать новые графы с помощью операций над уже имеющимися. Например, если $G = (V, E)$ — некоторый граф, то его *дополнение* $\overline{G} = (V, \overline{E})$ получается, если положить

$$u \text{ — } v \text{ в } \overline{G} \qquad \Longleftrightarrow \qquad u \neq v \text{ и } u \not\text{—} v \text{ в } G. \tag{35}$$

Таким образом, не ребра становятся ребрами, и наоборот. Обратите внимание, что $\overline{\overline{G}} = G$ и что $\overline{K_n}$ не имеет ребер. Соответствующие матрицы смежности A и $\overline{A}$ удовлетворяют условию

$$A + \overline{A} \;=\; J - I; \tag{36}$$

здесь J — матрица, все элементы которой является единицами, а I — тождественная матрица, так что J и $J-I$ являются соответственно матрицами смежности для J_n и K_n, когда G имеет порядок n.

Кроме того, каждый граф $G=(V,E)$ приводит к *реберному графу* $L(G)$, вершины которого являются ребрами E; два ребра графа G смежны в $L(G)$, если они имеют общую вершину. Таким образом, например, реберный граф $L(K_n)$ имеет $\binom{n}{2}$ вершин и является регулярным графом степени $2n-4$ при $n\ge 2$ (см. упр. 82). Граф называется *k-реберно-раскрашиваемым*, если его реберный граф является k-раскрашиваемым.

Для двух данных графов $G=(U,E)$ и $H=(V,F)$ их *объединением* (*union*) $G\cup H$ является граф $(U\cup V, E\cup F)$, получаемый путем объединения вершин и ребер. Например, предположим, что G и H представляют собой графы ходов ладьи и слона в шахматах; тогда $G\cup H$ является графом ходов ферзя и его официальное SGB-имя имеет вид

$$gunion(board(8,8,0,0,-1,0,0), board(8,8,0,0,-2,0,0),0,0). \tag{37}$$

В частном случае, когда множества вершин U и V непересекающиеся, объединение $G\cup H$ не требует согласованной идентификации вершин для взаимной корреляции; мы получаем диаграмму $G\cup H$, просто изображая диаграмму G, а после нее диаграмму H. Этот частный случай называется соприкосновением (juxtaposition) или *прямой суммой* G и H, и мы будем обозначать ее как $G\oplus H$. Например, легко увидеть, что

$$K_m\oplus K_n \;\cong\; \overline{K_{m,n}} \tag{38}$$

и что каждый граф является прямой суммой своих связных компонентов.

Уравнение (38) является частным случаем общей формулы

$$K_{n_1}\oplus K_{n_2}\oplus\cdots\oplus K_{n_k} \;\cong\; \overline{K_{n_1,n_2,\ldots,n_k}}\,, \tag{39}$$

которая справедлива для полных k-дольных графов при $k\ge 2$. Но (39) не работает при $k=1$ из-за позорного факта: стандартные обозначения теории графов противоречивы! Действительно, $K_{m,n}$ означает полный двудольный граф, но K_n *не* означает полный однодольный граф. Теоретики, работающие в области теории графов, невероятным образом ухитряются десятилетиями жить с этой аномалией и не обезуметь.

Еще одним важным способом комбинации двух непересекающихся графов G и H является образование их *соединения* (*join*) G — H, которое состоит из $G\oplus H$ вместе со всеми ребрами u — v для $u\in U$ и $v\in V$. [См. А. А. Зыков, *Мат. сборник* **24** (1949), 163–188, §I.3.] А если G и H являются непересекающимися *ориентированными графами*, то их *ориентированное соединение* $G\longrightarrow H$ аналогично, но добавляет к $G\oplus H$ только дуги $u\longrightarrow v$, идущие от U к V.

Прямая сумма двух матриц A и B получается путем помещения B по диагонали ниже и правее A:

$$A\oplus B \;=\; \begin{pmatrix} A & O \\ O & B \end{pmatrix}, \tag{40}$$

где каждое O в данном примере представляет собой матрицу, состоящую из одних нулей, с количеством строк и столбцов, необходимых для корректного выравнивания

получающейся матрицы. Наше обозначение $G \oplus H$ для прямой суммы графов легко запомнить, так как матрица смежности для $G \oplus H$ строго равна прямой сумме соответствующих матриц смежности A и B для G и H. Аналогично матрицами смежности для $G \mathrel{—} H$, $G \mathrel{\longrightarrow} H$ и $G \mathrel{\longleftarrow} H$ являются

$$A \mathrel{—} B = \begin{pmatrix} A & J \\ J & B \end{pmatrix}, \qquad A \mathrel{\longrightarrow} B = \begin{pmatrix} A & J \\ O & B \end{pmatrix}, \qquad A \mathrel{\longleftarrow} B = \begin{pmatrix} A & O \\ J & B \end{pmatrix} \tag{41}$$

соответственно, где J — матрица, состоящая из одних единиц, как в (36). Эти операции ассоциативны и связаны операцией дополнения:

$$A \oplus (B \oplus C) = (A \oplus B) \oplus C, \qquad A \mathrel{—} (B \mathrel{—} C) = (A \mathrel{—} B) \mathrel{—} C; \tag{42}$$

$$A \mathrel{\longrightarrow} (B \mathrel{\longrightarrow} C) = (A \mathrel{\longrightarrow} B) \mathrel{\longrightarrow} C, \qquad A \mathrel{\longleftarrow} (B \mathrel{\longleftarrow} C) = (A \mathrel{\longleftarrow} B) \mathrel{\longleftarrow} C; \tag{43}$$

$$\overline{A \oplus B} = \overline{A} \mathrel{—} \overline{B}, \qquad \overline{A \mathrel{—} B} = \overline{A} \oplus \overline{B}; \tag{44}$$

$$\overline{A \mathrel{\longrightarrow} B} = \overline{A} \mathrel{\longleftarrow} \overline{B}, \qquad \overline{A \mathrel{\longleftarrow} B} = \overline{A} \mathrel{\longrightarrow} \overline{B}; \tag{45}$$

$$(A \oplus B) + (A \mathrel{—} B) \;=\; (A \mathrel{\longrightarrow} B) + (A \mathrel{\longleftarrow} B). \tag{46}$$

Обратите внимание, что, комбинируя (39) с (42) и (44), мы получаем

$$K_{n_1,n_2,\ldots,n_k} \;=\; \overline{K_{n_1}} \mathrel{—} \overline{K_{n_2}} \mathrel{—} \cdots \mathrel{—} \overline{K_{n_k}} \tag{47}$$

при $k \geq 2$. Кроме того,

$$K_n = K_1 \mathrel{—} K_1 \mathrel{—} \cdots \mathrel{—} K_1 \quad \text{и} \quad K_n^{\rightarrow} = K_1 \mathrel{\longrightarrow} K_1 \mathrel{\longrightarrow} \cdots \mathrel{\longrightarrow} K_1\,, \tag{48}$$

с n копиями K_1, что показывает, что $K_n = K_{1,1,\ldots,1}$ является полным n-дольным графом.

Прямые суммы и соединения аналогичны сложению, поскольку $\overline{K_m} \oplus \overline{K_n} = \overline{K_{m+n}}$ и $K_m \mathrel{—} K_n = K_{m+n}$. Можно также скомбинировать графы с помощью алгебраической операции, аналогичной умножению. Например, операция *декартова произведения* образует граф $G \mathbin{\square} H$ порядка mn из графа $G = (U, E)$ порядка m и графа $H = (V, F)$ порядка n. Вершинами $G \mathbin{\square} H$ являются упорядоченные пары (u, v), где $u \in U$ и $v \in V$; ребрами являются $(u, v) \mathrel{—} (u', v)$, когда $u \mathrel{—} u'$ находится в G, вместе с $(u, v) \mathrel{—} (u, v')$, когда $v \mathrel{—} v'$ находится в H. Другими словами, $G \mathbin{\square} H$ образуется путем замены каждой вершины G копией H и замены каждого ребра G ребрами между соответствующими вершинами надлежащих копий:

(49)

Как обычно, простейшие частные случаи этого общего построения оказываются особо важными на практике. Когда и G, и H являются путями или циклами, мы получаем "графы в клеточку", а именно *$m \times n$-решетку* $P_m \mathbin{\square} P_n$, *$m \times n$-цилиндр*

$P_m \mathbin{\square} C_n$ и $m \times n$-*тор* $C_m \mathbin{\square} C_n$, показанные ниже для $m = 3$ и $n = 4$:

$P_3 \mathbin{\square} P_4$ (3 × 4-решетка) $P_3 \mathbin{\square} C_4$ (3 × 4-цилиндр) $C_3 \mathbin{\square} C_4$ (3 × 4-тор) (50)

Четыре других заслуживающих внимания пути определения произведений графов также оказываются полезными. В каждом случае вершинами произведения графов являются упорядоченные пары (u, v).

- *Прямое произведение* $G \otimes H$, именуемое также "конъюнкцией" G и H, или их "категорийное произведение" (categorical product) содержит $(u, v) — (u', v')$, когда $u — u'$ имеется в G, а $v — v'$ имеется в H.
- *Сильное произведение* $G \boxtimes H$ объединяет ребра $G \mathbin{\square} H$ с ребрами $G \otimes H$.
- *Нечетное произведение* $G \triangle H$ содержит ребро $(u, v) — (u', v')$, когда либо $u — u'$ содержится в G, либо $v — v'$ содержится в H, но не одновременно.
- *Лексикографическое произведение* $G \circ H$, именуемое также композицией G и H, содержит $(u, v) — (u', v')$, когда $u — u'$ содержится в G, и $(u, v) — (u, v')$, когда $v — v'$ содержится в H.

Все пять этих операций естественным образом распространяются на произведения $k \geq 2$ графов $G_1 = (V_1, E_1)$, …, $G_k = (V_k, E_k)$, вершины которых представляют собой упорядоченные k-кортежи $(v_1, \ldots, v_k)$, такие, что $v_j \in V_j$ для $1 \leq j \leq k$. Например, когда $k = 3$, декартовы произведения $G_1 \mathbin{\square} (G_2 \mathbin{\square} G_3)$ и $(G_1 \mathbin{\square} G_2) \mathbin{\square} G_3$ изоморфны, если рассматривать составные вершины $(v_1, (v_2, v_3))$ и $((v_1, v_2), v_3)$ как одинаковые с (v_1, v_2, v_3). Следовательно, можно записывать декартово произведение без скобок, как $G_1 \mathbin{\square} G_2 \mathbin{\square} G_3$. Наиболее важным примером декартова произведения с k сомножителями является k-мерный куб,

$$P_2 \mathbin{\square} P_2 \mathbin{\square} \cdots \mathbin{\square} P_2; \tag{51}$$

2^k его вершин $(v_1, \ldots, v_k)$ являются смежными, когда расстояние Хэмминга между ними равно 1.

В общем случае предположим, что $v = (v_1, \ldots, v_k)$ и $v' = (v'_1, \ldots, v'_k)$ являются k-кортежами вершин, где $v_j — v'_j$ в G_j для ровно a из индексов j, а $v_j = v'_j$ в точности для b из индексов. Тогда имеем:

- $v — v'$ в $G_1 \mathbin{\square} \cdots \mathbin{\square} G_k$ тогда и только тогда, когда $a = 1$ и $b = k - 1$;
- $v — v'$ в $G_1 \otimes \cdots \otimes G_k$ тогда и только тогда, когда $a = k$ и $b = 0$;
- $v — v'$ в $G_1 \boxtimes \cdots \boxtimes G_k$ тогда и только тогда, когда $a + b = k$ и $a > 0$;
- $v — v'$ в $G_1 \triangle \cdots \triangle G_k$ тогда и только тогда, когда a нечетно.

Лексикографическое произведение несколько отличается, поскольку оно не коммутативно; в $G_1 \circ \cdots \circ G_k$ мы имеем $v — v'$ для $v \neq v'$ тогда и только тогда, когда $v_j — v'_j$, где j — минимальный индекс, такой, что $v_j \neq v'_j$.

В упр. 91–102 исследуются некоторые фундаментальные свойства произведений графов. См. также книгу *Product Graphs* Вильфреда Имриха (Wilfried Imrich)

и Санди Клавжара (Sandi Klavzar) (2000), в которой содержится исчерпывающее введение в теорию графов, включая алгоритмы для разложения графов на “простые” подграфы.

***Графические последовательности степеней.** Последовательность $d_1 d_2 \ldots d_n$ неотрицательных целых чисел называется *графической* (*graphical*), если существует как минимум один граф с вершинами $\{1, 2, \ldots, n\}$, такой, что вершина k имеет степень d_k. Можно считать, что $d_1 \geq d_2 \geq \cdots \geq d_n$. Ясно, что в любом таком графе $d_1 < n$ и что сумма $m = d_1 + d_2 + \cdots + d_n$ любой графической последовательности всегда четна, поскольку она равна удвоенному количеству ребер. Кроме того, легко увидеть, что последовательность 3311 не является графической; следовательно, графические последовательности должны удовлетворять дополнительным условиям. Каким?

Простой способ выяснить, является ли данная последовательность $d_1 d_2 \ldots d_n$ графической, и построить соответствующий граф в случае, если он существует, был открыт В. Гавелом (V. Havel) [*Casopis pro Pěstování Matematiky* **80** (1955), 477–479]. Мы начинаем с пустой таблицы, имеющей d_k ячеек в строке k; эти ячейки представляют “слоты”, в которые мы будем помещать соседей вершины k в строящемся графе. Пусть c_j — количество ячеек в столбце j; таким образом, $c_1 \geq c_2 \geq \cdots$, и при $1 \leq k \leq n$ мы имеем $c_j \geq k$ тогда и только тогда, когда $d_k \geq j$. Предположим, например, что $n = 8$ и $d_1 \ldots d_8 = 55544322$. Тогда исходная таблица имеет вид

(52)

и мы имеем $c_1 \ldots c_5 = 88653$. Идея Гавела состоит в соединении вершины n с d_n вершин с наивысшими степенями. В данном случае, например, мы создаем два ребра, 8 — 3 и 8 — 2, и таблица принимает следующий вид:

(53)

(Мы не используем 8 — 1, поскольку пустые слоты должны продолжать образовывать таблицу; ячейки каждого столбца должны заполняться снизу вверх.) Затем мы устанавливаем $n \leftarrow 7$ и создаем два очередных ребра, 7 — 1 и 7 — 5. Затем наступает очередь следующих трех ребер, 6 — 4, 6 — 3, 6 — 2, и таблица становится

почти наполовину заполненной:

1					7
2				6	8
3				6	8
4				6	
5				7	
6	2	3	4		
7	5	1			
8	2	3			

. (54)

Мы свели задачу к поиску графа с последовательностью степеней $d_1 \ldots d_5 = 43333$; в этот момент мы также имеем $c_1 \ldots c_4 = 5551$. Читателю предлагается самостоятельно заполнить оставшиеся пустые места, перед тем как посмотреть ответ в упр. 103.

Алгоритм H (*Генератор графа для определенных степеней*). Этот алгоритм для заданных $d_1 \geq \cdots \geq d_n \geq d_{n+1} = 0$ создает ребра между вершинами $\{1, \ldots, n\}$ таким образом, что ровно d_k ребер связаны с вершиной k, для $1 \leq k \leq n$, если, конечно, последовательность $d_1 \ldots d_n$ является графической. Массив $c_1 \ldots c_{d_1}$ используется в качестве вспомогательного.

H1. [Установка c.] Начать с $k \leftarrow d_1$ и $j \leftarrow 0$. Затем, пока $k > 0$, выполнять следующие операции: установить $j \leftarrow j + 1$; пока $k > d_{j+1}$, устанавливать $c_k \leftarrow j$ и $k \leftarrow k - 1$. Успешно завершить работу алгоритма, если $j = 0$ (все d равны нулю).

H2. [Поиск n.] Установить $n \leftarrow c_1$. Успешно завершить работу алгоритма, если $n = 0$; завершить работу алгоритма неудачей, если $d_1 \geq n > 0$.

H3. [Начало цикла по j.] Установить $i \leftarrow 1$, $t \leftarrow d_1$, $r \leftarrow c_t$ и $j \leftarrow d_n$.

H4. [Генерация нового ребра.] Установить $c_j \leftarrow c_j - 1$ и $m \leftarrow c_t$. Создать ребро n — m и установить $d_m \leftarrow d_m - 1$, $c_t \leftarrow m - 1$, $j \leftarrow j - 1$. Если $j = 0$, вернуться к шагу H2. В противном случае если $m = i$, установить $i \leftarrow r+1$, $t \leftarrow d_i$ и $r \leftarrow c_t$ (см. упр. 104); повторить шаг H4. ▮

Когда алгоритм H завершается успешно, он строит граф с интересующими нас ребрами. Но если он завершается неудачно, то как мы можем убедиться, что задача неразрешима? Ключевой факт основан на важной концепции, именуемой "мажоризация": если $d_1 \ldots d_n$ и $d'_1 \ldots d'_n$ являются двумя разбиениями одного и того же числа (т. е. если $d_1 \geq \cdots \geq d_n$, $d'_1 \geq \cdots \geq d'_n$ и $d_1 + \cdots + d_n = d'_1 + \cdots + d'_n$), то мы говорим, что $d_1 \ldots d_n$ *мажоризирует* $d'_1 \ldots d'_n$, если $d_1 + \cdots + d_k \geq d'_1 + \cdots + d'_k$ для $1 \leq k \leq n$.

Лемма M. *Если последовательность $d_1 \ldots d_n$ является графической и $d_1 \ldots d_n$ мажоризирует $d'_1 \ldots d'_n$, то последовательность $d'_1 \ldots d'_n$ также является графической.*

Доказательство. Достаточно доказать утверждение, когда $d_1 \ldots d_n$ и $d'_1 \ldots d'_n$ отличаются только в двух местах,

$$d'_k = d_k - [k = i] + [k = j], \qquad \text{где } i < j, \tag{55}$$

поскольку любая последовательность, мажоризируемая $d_1 \ldots d_n$, может быть получена неоднократной мини-мажоризацией, аналогичной показанной. (Детально мажоризация рассматривается в упр. 7.2.1.4–55.)

Из условия (55) вытекает, что $d_i > d'_i \geq d'_{i+1} \geq d'_j > d_j$. Так что любой граф с последовательностью степеней $d_1 \ldots d_n$ содержит вершину v, такую, что v — i и $v \not-\, j$. Удаление ребра v — i и добавление ребра v — j дает граф с последовательностью степеней $d'_1 \ldots d'_n$, как и требовалось. ▮

Следствие Н. *Алгоритм Н завершается успешно, если последовательность $d_1 \ldots d_n$ является графической.*

Доказательство. Можно считать, что $n > 1$. Предположим, что G — произвольный граф с вершинами $\{1, \ldots, n\}$ с последовательностью степеней $d_1 \ldots d_n$, и пусть G' — подграф, порожденный $\{1, \ldots, n-1\}$; другими словами, G' получается путем удаления вершины n и d_n ребер, связанных с ней. Последовательность степеней $d'_1 \ldots d'_{n-1}$ графа G' получается из $d_1 \ldots d_{n-1}$ путем уменьшения некоторых d_n элементов на 1 и сортировкой их в невозрастающем порядке. По определению последовательность $d'_1 \ldots d'_{n-1}$ графическая. Новая последовательность степеней $d''_1 \ldots d''_{n-1}$, получаемая с помощью стратегии, реализованной на шагах Н3 и Н4, задумана как мажоризируемая каждой такой последовательностью $d'_1 \ldots d'_{n-1}$, поскольку уменьшает наибольшие возможные d_n элементов на 1. Таким образом, новая последовательность $d''_1 \ldots d''_{n-1}$ является графической. Алгоритм Н, устанавливающий $d_1 \ldots d_{n-1} \leftarrow d''_1 \ldots d''_{n-1}$, будет, таким образом, успешен по индукции по n. ▮

Время работы алгоритма Н грубо пропорционально количеству генерируемых ребер, которое может быть порядка n^2. В упр. 105 представлен более быстрый метод, который за $O(n)$ шагов выясняет, является ли данная последовательность $d_1 \ldots d_n$ графической, но сам граф при этом не строится.

Над графами. Если вершины и/или дуги графа или ориентированного графа оснащены дополнительными данными, мы называем его *сетью* (*network*). Например, каждая вершина *words*$(5757, 0, 0, 0)$ имеет связанный с ней ранг, соответствующий популярности соответствующего пятибуквенного слова. Каждая вершина *plane_lisa*$(360, 250, 15, 0, 360, 0, 250, 0, 2295000)$ имеет связанную с ней яркость пикселей между 0 и 15. Каждая дуга *board*$(8, 8, 0, 0, -2, 0, 0)$ имеет связанную с ней длину, отражающую расстояние перемещения фигуры по доске: перемещение слона из угла в угол имеет длину 7. Stanford GraphBase включает несколько генераторов, которые не упоминались выше, поскольку они используются, в первую очередь, для генерации интересных сетей, а не графов с интересной структурой.

- *miles*$(128, 0, 0, 0, 0, 127, 0)$ представляет собой сеть со 128 вершинами, соответствующими тем же городам Северной Америки, что и в графе *plane_miles*, описанном ранее. Но *miles*, в отличие от *plane_miles*, представляет собой полный граф с $\binom{128}{2}$ ребрами. Каждое ребро имеет целую длину, представляющую расстояние в милях, которое автомобиль должен был преодолеть при поездке в 1949 году из одного города в другой. Например, в сети *miles* '`Vancouver, BC`' отстоит на 3496 миль от '`West Palm Beach, FL`'.

- *econ*$(81, 0, 0, 0)$ представляет собой сеть с 81 вершиной и 4902 дугами. Ее вершины представляют отрасли экономики США, а дуги — потоки денег из одной отрасли в другую в 1985 году, измеренные в миллионах долларов. Например, значение потока от `Apparel` в `Household furniture` равно 44, а это означает, что мебельная

промышленность заплатила в этом году швейной промышленности 44 миллиона долларов. Суммы потоков, входящих в каждую вершину, равны суммам исходящих потоков. Дуга имеется только в том случае, если поток ненулевой. Специальная вершина под названием `Users` получает потоки, которые представляют общую потребность в продукте; некоторые из этих потоков отрицательны в связи со способом рассмотрения правительственными экономистами импортируемых товаров.

• $games(120, 0, 0, 0, 0, 0, 128, 0)$ представляет собой сеть со 120 вершинами и 1276 дугами. Вершины этого графа представляют футбольные команды американских колледжей и университетов. Дуги проходят между командами, игравшими одна с другой во время сезона 1990 года, и помечены числами, соответствующими выигранным очкам. Например, дуга `Stanford` $\longrightarrow$ `California` имеет значение 27, а дуга `California` $\longrightarrow$ `Stanford` имеет значение 25, поскольку 17 ноября 1990 года Cardinal из Станфорда победили Golden Bears из Беркли со счетом 27:25.

• $risc(16)$ представляет собой сеть совершенно иного вида. В ней 3240 вершин и 7878 дуг, определяющих *ориентированный ациклический граф*, т. е. ориентированный граф, не содержащий ориентированных циклов. Вершины представляют логические элементы с логическими значениями; дуга наподобие `Z45` $\longrightarrow$ `R0:7~` означает, что значение логического элемента `Z45` является входным для элемента `R0:7~`. Каждый элемент имеет код типа (AND, OR, XOR, NOT, защелка или внешний ввод); каждая дуга имеет длину, описывающую величину задержки. Сеть содержит полную логику небольшой микросхемы RISC, способной подчиняться простым командам, определяемым шестнадцатью регистрами размером 16 бит каждый.

Полное детальное описание всех генераторов SGB можно найти в книге автора *The Stanford GraphBase* (New York: ACM Press, 1994) вместе с десятками небольших демонстрационных программ, поясняющих, как работать с генерируемыми графами и сетями. Например, программа LADDERS показывает, как найти кратчайший путь между двумя пятибуквенными словами. Программа TAKE_RISC демонстрирует, как нанокомпьютер выполняет свою работу, имитируя работу сети, построенной из логических элементов $risc(16)$.

Гиперграфы. Графы и сети могут быть просто обворожительны, но это ни в коем случае не конец истории. Множество важных комбинаторных алгоритмов разработано для работы с *гиперграфами*, которые являются более обобщенными по сравнению с графами, поскольку их ребра могут быть *произвольными* подмножествами вершин.

Например, у нас может быть семь вершин, идентифицируемых ненулевыми бинарными строками $v = a_1a_2a_3$, вместе с семью ребрами, идентифицируемыми ненулевыми бинарными строками в квадратных скобках $e = [b_1b_2b_3]$, где $v \in e$ тогда и только тогда, когда $(a_1b_1 + a_2b_2 + a_3b_3) \bmod 2 = 0$. Каждое из этих ребер содержит ровно три вершины:

$$\begin{gathered}[001] = \{010, 100, 110\}; \quad [010] = \{001, 100, 101\}; \quad [011] = \{011, 100, 111\};\\ [100] = \{001, 010, 011\}; \quad [101] = \{010, 101, 111\};\\ [110] = \{001, 110, 111\}; \quad [111] = \{011, 101, 110\}. \end{gathered} \tag{56}$$

В силу симметрии каждая вершина принадлежит ровно трем ребрам. Ребра, содержащие три и более вершин, иногда называют гиперребрами, чтобы отличать их от ребер обычных графов. Но их можно называть и просто ребрами.

Гиперграф называется *r-однородным*, если каждое ребро содержит ровно r вершин. Таким образом, (56) является 3-однородным гиперграфом, а 2-однородный гиперграф представляет собой ни что иное, как обычный граф. Полный r-однородный гиперграф $K_n^{(r)}$ имеет n вершин и $\binom{n}{r}$ ребер.

Большинство основных концепций теории графов естественным образом может быть распространено на гиперграфы. Например, если $H = (V, E)$ представляет собой гиперграф и если $U \subseteq V$, то подгиперграф $H \mid U$, порожденный U, имеет ребра $\{e \mid e \in E \text{ и } e \subseteq U\}$. Дополнение $\overline{H}$ r-однородного гиперграфа имеет ребра $K_n^{(r)}$, которые не являются ребрами H. k-раскрашиванием гиперграфа называется назначение цветов его вершинам таким образом, чтобы ни одно ребро не было монохромным. И т. д.

Гиперграфы могут иметь много других имен, поскольку одни и те же свойства могут быть сформулированы многими различными способами. Например, каждый гиперграф $H = (V, E)$ представляет собой, по сути, *семейство множеств*, поскольку каждое ребро является подмножеством V. 3-однородный гиперграф называется также *системой троек*. Гиперграф эквивалентен также матрице B, состоящей из нулей и единиц, в которой имеется по одной строке для каждой вершины v и по одному столбцу для каждого ребра e; на пересечении строки v и столбца e этой матрицы находится значение $b_{ve} = [v \in e]$. Матрица B называется *матрицей инциденций* H, и мы говорим, что вершина "v инцидентна e", когда $v \in e$. Кроме того, гиперграф эквивалентен *двудольному графу* с множеством вершин $V \cup E$ и с ребром v — e в случае инцидентности v с e. Гиперграф называется *связным* тогда и только тогда, когда связным является соответствующий двудольный граф. *Цикл* длиной k в гиперграфе определяется как цикл длиной $2k$ в соответствующем двудольном графе.

Например, гиперграф (56) может быть определен эквивалентной матрицей инциденций или эквивалентным двудольным графом следующим образом.

$$\begin{array}{c} \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{array} \begin{array}{c} \begin{array}{ccccccc} [001] & [010] & [011] & [100] & [101] & [110] & [111] \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array} \qquad \text{(57)}$$

Он содержит 28 циклов длиной 3, таких как

$$[101] — 101 — [010] — 001 — [100] — 010 — [101]. \qquad (58)$$

Гиперграф H^T, *дуальный* гиперграфу H, получается путем взаимообмена ролей вершин и ребер, но с неизменным отношением инцидентности. Другими словами,

он соответствует транспонированной матрице инциденций. Обратите внимание, что, например, дуальным к r-регулярному графу является r-однородный гиперграф.

Матрицы инциденций и двудольные графы могут соответствовать гиперграфам, в которых некоторые ребра встречаются больше одного раза, поскольку различные столбцы матрицы могут быть одинаковыми. Если гиперграф $H = (V, E)$ не имеет повторяющихся ребер, он соответствует еще одному комбинаторному объекту, а именно *булевой функции* (логической функции). Ибо если, скажем, множество вершин V равно $\{1, 2, \ldots, n\}$, функция

$$h(x_1, x_2, \ldots, x_n) \;=\; \bigl[\{j \mid x_j = 1\} \in E\bigr] \tag{59}$$

характеризует ребра H. Например, булева формула

$$\begin{aligned}(x_1 \oplus x_2 \oplus x_3) \wedge (x_2 \oplus x_4 \oplus x_6) \wedge (x_3 \oplus x_4 \oplus x_7)\\ \wedge (x_3 \oplus x_5 \oplus x_6) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4)\end{aligned} \tag{60}$$

представляет собой другой способ описания гиперграфа (56) и (57).

Тот факт, что комбинаторные объекты можно рассматривать столькими разными способами, может показаться невероятным. Но это исключительно полезно, поскольку предполагает различные способы решения эквивалентных задач. Когда мы рассматриваем задачу с разных точек зрения, мозг естественным образом думает о разных способах ее решения. Иногда озарение приходит, когда мы думаем о том, как работать со строками и столбцами матрицы. В другой раз прорыв происходит, когда мы представляем вершины и пути или при визуализации кластеров точек в пространстве. Возможно, что наилучшим способом окажется применение булевой алгебры. Если мы застряли на одном из представлений, на помощь может прийти другое.

Покрытие и независимость. Если $H = (V, E)$ является графом или гиперграфом, множество вершин U называется *покрытием* H, если каждое ребро содержит как минимум один из членов U. Множество вершин W называется *независимым* (или устойчивым) в H, если в нем не содержится полностью ни одно ребро.

С точки зрения матрицы инциденций покрытие является множеством строк, сумма которых не равна нулю в каждом столбце. В частном случае, когда H является графом, каждый столбец матрицы содержит только две единицы; следовательно, независимое множество в графе соответствует множеству взаимно ортогональных строк, т. е. множеству, в котором скалярное произведение любых двух разных строк равно нулю.

Эти концепции являются оборотными сторонами одной медали. Если U покрывает H, то $W = V \setminus U$ независимо в H; и наоборот, если W независимо в H, то $U = V \setminus W$ покрывает H. Оба утверждения эквивалентны утверждению, что порожденный гиперграф $H \mid W$ не имеет ребер.

Это дуальное отношение между покрытием и независимостью, которое, похоже, впервые было открыто Клодом Бергом (Claude Berge) [*Proc. National Acad. Sci.* **43** (1957), 842–844], несколько парадоксально. Хотя оно логически очевидно и легко проверяемо, на интуитивном уровне оно кажется удивительным. Когда мы смотрим на граф и пытаемся найти большое независимое множество, обычно мы мыслим совершенно иначе, чем когда смотрим на тот же граф и пытаемся найти малое вершинное покрытие; но цели в обоих случаях одинаковы.

Покрывающее множество U *минимально* (*minimal*), если $U \setminus u$ не является покрытием для всех $u \in U$. Аналогично независимое множество W является *максимальным* (*maximal*), если $W \cup w$ не является независимым для всех $w \notin W$. Вот, например, как выглядит минимальное покрытие 49-вершинного графа США (17) и соответствующее ему максимальное независимое множество.

Минимальное вершинное покрытие из 38 вершин

Максимальное независимое множество из 11 вершин

(61)

Покрытие называется *наименьшим* (*minimum*), если оно имеет наименьший возможный размер, а независимое множество называется *наибольшим* (*maximum*), если оно имеет наибольший возможный размер. Например, в случае графа (17) можно добиться лучшего результата, чем представляет собой (61).

Наименьшее вершинное покрытие из 30 вершин

Наибольшее независимое множество из 19 вершин

(62)

От переводчика. Для графа Украины (17, у) соответствующие результаты имеют следующий вид.

Наименьшее вершинное покрытие из 15 вершин

Наибольшее независимое множество из 10 вершин

Обратите внимание на тонкую разницу между “минимальным” и “наименьшим”: в общем случае люди, работающие с комбинаторными алгоритмами, в отличие от словаря английского языка, используют слова на ‘-al’ (наподобие “minimal” или “optimal”) для комбинаторных конфигураций, которые являются наилучшими *локально*, в том смысле, что минимальные изменения их не улучшают. Слова же на ‘-um’, такие как “minimum” или “optimum”, оставлены для конфигураций, наилучших *глобально*, т. е. при рассмотрении всех возможностей.* Легко найти решение любой задачи оптимизации, являющееся всего лишь оптимальным в слабом локальном смысле, просто забираясь на вершину ближайшего холма. Но обычно

* К счастью, русский язык позволяет использовать по-разному звучащие слова — скажем, “оптимальный” и “наилучший”. — *Примеч. пер.*

гораздо труднее найти наилучшее решение, истинный оптимум, который представляет собой вершину самой высокой горы. Например, из раздела 7.9 вы узнаете, что задача поиска наибольшего независимого множества данного графа принадлежит классу сложных задач, который называется *NP-полным*.

Даже если задача является NP-полной, не стоит опускать руки. Мы рассмотрим методы поиска наименьших покрытий в нескольких частях данной главы, и эти методы неплохо работают для задач небольшого размера; наилучшее решение (62) найдено менее чем за секунду, после исследования очень небольшой доли всех 2^{49} возможных вариантов. Кроме того, нередко частные случаи NP-полных задач оказываются проще, чем задачи в общем случае. В разделе 7.5.1 мы увидим, что можно быстро найти наименьшее вершинное покрытие для любого двудольного графа или для любого гиперграфа, являющегося дуальным к графу. А в разделе 7.5.5 мы изучим эффективные способы наибольшего *паросочетания*, которое представляет собой наибольшее независимое множество в реберном графе данного графа.

Задача максимизации размера независимого множества возникает достаточно часто, чтобы получить собственное обозначение: если H представляет собой некоторый гиперграф, то число

$$\alpha(H) \;=\; \max\bigl\{|W| \;\bigm|\; W \text{ является независимым множеством вершин в } H\bigr\} \tag{63}$$

называется *числом независимости* (или числом устойчивости) гиперграфа H. Аналогично

$$\chi(H) \;=\; \min\{k \mid H \text{ является } k\text{-раскрашиваемым}\} \tag{64}$$

называется *хроматическим числом* H. Обратите внимание, что $\chi(H)$ представляет собой размер наименьшего покрытия H независимыми множествами, поскольку вершины, получающие некоторый определенный цвет, должны быть независимыми в соответствии с нашими определениями.

Эти определения $\alpha(H)$ и $\chi(H)$ применимы, в частности, в случае, когда H представляет собой обычный граф, но, конечно, в этих случаях мы обычно пишем $\alpha(G)$ и $\chi(G)$. Графы имеют другое важное число, именуемое их *кликовым числом*,

$$\omega(G) \;=\; \max\bigl\{|X| \;\bigm|\; X \text{ является кликой в } G\bigr\}, \tag{65}$$

где “клика” представляет собой множество взаимно смежных вершин. Ясно, что

$$\omega(G) \;=\; \alpha(\overline{G}), \tag{66}$$

поскольку клика в G является независимым множеством в дополнении графа. Аналогично мы можем видеть, что $\chi(\overline{G})$ является наименьшим размером “кликового покрытия”, которое представляет собой множество клик, которое покрывает в точности все вершины.

Несколько экземпляров “задач точного покрытия” упоминались ранее в этом разделе без точного пояснения, что же в действительности означает такая задача. Наконец, мы созрели для определения: для заданной матрицы инциденций или гиперграфа H *точное покрытие* H представляет собой множество строк, сумма которых равна $(1\,1\ldots1)$. Другими словами, точное покрытие представляет собой множество вершин, которые соединяются с каждым гиперребром ровно по одному разу; обычное покрытие требует соединения с каждым гиперребром *как минимум* по одному разу.

Упражнения

1. [*25*] Предположим, что $n = 4m-1$. Постройте размещение пар Лэнгфорда для чисел $\{1, 1, \ldots, n, n\}$, обладающее тем свойством, что мы также получим решение для $n = 4m$, заменяя первое '$2m-1$' на '$4m$' и добавляя '$2m-1$ $4m$' справа. *Указание*: поместите слева $m-1$ четных чисел $4m-4$, $4m-6$, $\ldots$, $2m$.

2. [*20*] Для каких n можно расположить числа $\{0, 0, 1, 1, \ldots, n-1, n-1\}$ в виде пар Лэнгфорда?

3. [*22*] Предположим, что мы разместили числа $\{0, 0, 1, 1, \ldots, n-1, n-1\}$ по *окружности*, а не на прямой, с расстоянием k между двумя k. Получим ли мы решение, существенно отличное от решения упр. 2?

4. [*M20*] (Т. Сколем (T. Skolem), 1957.) Покажите, что рассматривающаяся в упр. 1.2.8–36 строка Фибоначчи $S_\infty = babbababbabba\ldots$ приводит непосредственно к бесконечной последовательности $0012132453674\ldots$ пар Лэнгфорда для множества *всех* неотрицательных целых чисел, если просто независимо заменять a и b слева направо на 0, 1, 2 и т. д.

▶ **5.** [*HM22*] Какова вероятность того, что при случайной перестановке чисел $\{1, 1, 2, 2, \ldots, n, n\}$ два k будут находиться ровно на расстоянии k позиций одно от другого, для заданного k? Воспользуйтесь этой формулой для оценки величины чисел Лэнгфорда L_n в (1).

▶ **6.** [*M28*] (М. Годфри (M. Godfrey), 2002.) Пусть

$$f(x_1, \ldots, x_{2n}) = \prod_{k=1}^{n} (x_k x_{n+k} \sum_{j=1}^{2n-k-1} x_j x_{j+k+1}).$$

а) Докажите, что $\sum_{x_1,\ldots,x_{2n} \in \{-1,+1\}} f(x_1, \ldots, x_{2n}) = 2^{2n+1} L_n$.

б) Поясните, как вычислить эту сумму за $O(4^n n)$ шагов. Сколько битов точности потребуют при этом арифметические операции?

в) Усильте результат в восемь раз, используя тождества

$$f(x_1, \ldots, x_{2n}) = f(-x_1, \ldots, -x_{2n}) = f(x_{2n}, \ldots, x_1) = f(x_1, -x_2, \ldots, x_{2n-1}, -x_{2n}).$$

7. [*M22*] Докажите, что любое размещение Лэнгфорда чисел $\{1, 1, \ldots, 16, 16\}$ должно иметь при чтении слева направо в некоторой точке семь незавершенных пар.

8. [*23*] Простейшая последовательность Лэнгфорда не только хорошо сбалансирована, но и *планарна*, в том смысле, что ее пары могут быть соединены без пересечения линий, как в (2): $2\,3\,1\,2\,1\,3$. Найдите все планарные последовательности Лэнгфорда для $n \le 8$.

9. [*24*] (*Тройки Лэнгфорда.*) Сколькими способами можно разместить числа $\{1, 1, 1, 2, 2, 2, \ldots, 9, 9, 9\}$ в строке так, чтобы последовательные k находились на расстоянии k друг от друга для $1 \le k \le 9$?

10. [*M20*] Поясните, как построить *магический квадрат* непосредственно по рис. 1. (Преобразуйте каждую карту в число от 1 до 16 так, чтобы суммы в каждой строке, каждом столбце и каждой диагонали были равны 34.)

11. [*20*] Распространите (5) на "иврито-греко-латинский" квадрат, добавляя по одной из букв $\{\text{א}, \text{ב}, \text{ג}, \text{ד}\}$ к двухбуквенным строкам в каждой ячейке. Никакие пары букв (латинская, греческая), (латинская, иврит) и (греческая, иврит) не должны встречаться более чем в одном месте.

▶ **12.** [*M21*] (Л. Эйлер (L. Euler).) Пусть $L_{ij} = (i + j) \bmod n$ для $0 \le i, j < n$ представляет собой таблицу сложения для целых чисел по модулю n. Докажите, что латинский квадрат, ортогональный L, существует тогда и только тогда, когда n нечетно.

13. [*M25*] Квадрат 10×10 можно разделить на четыре четверти размером по 5×5. Латинский квадрат 10×10, образованный цифрами $\{0, 1, \ldots, 9\}$, имеет k "нарушителей", если его верхняя левая четверть имеет ровно k элементов ≥ 5. (См. в упр. 14, (д) пример с $k = 3$.) Докажите, что квадрат не имеет ортогонального к нему квадрата, если только в нем нет как минимум трех нарушителей.

14. [*29*] Найдите все квадраты, ортогональные следующим латинским квадратам:

(а)	(б)	(в)	(г)	(д)
3145926870	2718459036	0572164938	1680397425	7823456019
2819763504	0287135649	6051298473	8346512097	8234067195
9452307168	7524093168	4867039215	9805761342	2340178956
6208451793	1435962780	1439807652	2754689130	3401289567
8364095217	6390718425	8324756091	0538976214	4012395678
5981274036	4069271853	7203941586	4963820571	5678912340
4627530981	3102684597	5610473829	7192034658	6789523401
0576148329	9871546302	9148625307	6219405783	0195634782
1730689452	8956307214	2795380164	3471258906	1956740823
7093812645	5643820971	3986512740	5027143869	9567801234

15. [*50*] Найдите три латинских квадрата 10×10, взаимно ортогональных друг другу.

16. [*48*] (Г. Д. Райзер (H. J. Ryser), 1967.) Латинский квадрат называется "порядка n", если он имеет n строк, n столбцов и n символов. Каждый ли латинский квадрат нечетного порядка имеет секущую?

17. [*25*] Пусть L — латинский квадрат с элементами L_{ij} для $0 \leq i, j < n$. Покажите, что задачи (а) поиска всех секущих L и (б) поиска всех ортогональных L квадратов являются частными случаями общей задачи точного покрытия.

18. [*M26*] Строка $x_1 x_2 \ldots x_N$ называется n-арной, если каждый элемент x_j принадлежит множеству $\{0, 1, \ldots, n-1\}$ n-арных цифр. Две строки, $x_1 x_2 \ldots x_N$ и $y_1 y_2 \ldots y_N$, называются *ортогональными*, если N пар (x_j, y_j) различны для $1 \leq j \leq N$. (Следовательно, две n-арные строки не могут быть ортогональны, если их длина N превышает n^2.) n-арная матрица с m строками и n^2 столбцами, строки которых ортогональны одна другой, называется *ортогональным массивом* порядка n и глубиной m.

Найдите соответствие между ортогональными массивами глубиной m и списками из $m - 2$ взаимно ортогональных латинских квадратов. Какой ортогональный массив соответствует упр. 11?

▶ **19.** [*M25*] Продолжая выполнение упр. 18, докажите, что ортогональный массив порядка $n > 1$ и глубиной m возможен, только если $m \leq n + 1$. Покажите, что этот верхний предел достижим, когда n является простым числом p. Приведите пример для $p = 5$.

20. [*HM20*] Покажите, что если каждый элемент k в ортогональном массиве заменить на $e^{2\pi k i/n}$, то его строки станут ортогональными векторами в обычном смысле (их скалярное произведение будет равно нулю).

▶ **21.** [*M21*] *Геометрическая сеть* представляет собой систему точек и прямых, подчиняющуюся трем аксиомам.

i) Каждая прямая есть множество точек.

ii) Различные прямые имеют не более одной общей точки.

iii) Если p является точкой, а L — прямая, такая, что $p \notin L$, то существует ровно одна прямая M, такая, что $p \in M$ и $L \cap M = \emptyset$. Если $L \cap M = \emptyset$, мы говорим, что L *параллельна* M, и записываем $L \parallel M$.

а) Докажите, что прямые геометрической сети могут быть разбиты на классы эквивалентности, когда две прямые находятся в одном классе тогда и только тогда, когда они совпадают или параллельны.

б) Покажите, что если имеется как минимум два класса параллельных прямых, то каждая прямая содержит то же количество точек, что и другие прямые в ее классе.

в) Кроме того, если имеется как минимум три класса, существуют числа m и n, такие, что каждая точка принадлежит ровно m прямым, и все прямые содержат ровно n точек.

▶ **22.** [*M22*] Покажите, что каждый ортогональный массив можно рассматривать как геометрическую сеть. Справедливо ли обратное утверждение?

23. [*M23*] (*Коды с коррекцией ошибок.*) "Расстоянием Хэмминга" $d(x, y)$ между двумя строками, $x = x_1 \ldots x_N$ и $y = y_1 \ldots y_N$, является количество позиций j, в которых $x_j \neq y_j$. *b-арный код с n информационными и r контрольными разрядами* представляет собой множество $C(b, n, r)$, состоящее из b^n строк $x = x_1 \ldots x_{n+r}$, где $0 \le x_j < b$ для $1 \le j \le n+r$. При передаче кодового слова x и получении сообщения y величина $d(x, y)$ равна количеству ошибок передачи. Код называется *исправляющим t ошибок*, если можно реконструировать значение x при полученном сообщении y, таком, что $d(x, y) \le t$. *Расстоянием* кода является минимальное значение $d(x, x')$, взятое по всем парам кодовых слов $x \neq x'$.

а) Докажите, что код является исправляющим t ошибок тогда и только тогда, когда его расстояние превышает $2t$.

б) Докажите, что исправляющий одну ошибку b-арный код с двумя информационными и двумя контрольными разрядами эквивалентен паре ортогональных латинских квадратов порядка b.

в) Кроме того, код $C(b, 2, r)$ с расстоянием $r + 1$ эквивалентен множеству r взаимно ортогональных латинских квадратов порядка b.

▶ **24.** [*M30*] Геометрическая сеть с N точками и R прямыми естественным образом приводит к бинарному коду $C(2, N, R)$ с кодовыми словами $x_1 \ldots x_N x_{N+1} \ldots x_{N+R}$, определяемыми битами четности

$$x_{N+k} = f_k(x_1, \ldots, x_N) = \left(\sum\{x_j \mid \text{точка } j \text{ лежит на прямой } k\}\right) \bmod 2.$$

а) Докажите, что этот код имеет расстояние $m + 1$, если сеть имеет m классов параллельных прямых.

б) Найдите эффективный способ исправления до t ошибок в этом коде, в предположении, что $m = 2t$. Проиллюстрируйте процесс декодирования для случая $N = 25$, $R = 30$, $t = 3$.

25. [*27*] Найдите латинский квадрат, строки и столбцы которого представляют собой пятибуквенные слова. (Для выполнения этого упражнения вам понадобятся большие словари.)

▶ **26.** [*25*] Составьте имеющее смысл английское предложение, состоящее только из пятибуквенных слов.

27. [*20*] Сколько SGB-слов содержат ровно k различных букв для $1 \le k \le 5$?

28. [*20*] Имеются ли пары SGB-слов, рассматриваемых как векторы, все соответствующие компоненты которых отличаются на ± 1?

29. [*20*] Найдите все SGB-слова, являющиеся *палиндромами* (остающимися неизменными при чтении справа налево) или зеркальными парами (наподобие `regal lager`).

▶ **30.** [*20*] Буквы слова `first` находятся в алфавитном порядке слева направо. Какое из таких пятибуквенных слов является лексикографически *первым*? Какое последним?

31. [*21*] (К. Мак-Манус (C. McManus).) Найдите все множества из трех SGB-слов, находящихся в арифметической прогрессии и не имеющих общих букв ни в одной фиксированной позиции. (Одним таким примером является {`power`, `slugs`, `visit`}.)

32. [*23*] Содержит ли английский язык хотя бы одно десятибуквенное слово $a_0a_1\ldots a_9$, для которого и $a_0a_2a_4a_6a_8$, и $a_1a_3a_5a_7a_9$ являются SGB-словами?

33. [*20*] (Скот Моррис (Scot Morris).) Завершите следующий список из 26 интересных SGB-слов:

`about`, `bacon`, `faced`, `under`, `chief`, ..., `pizza`.

▶ **34.** [*21*] Для каждого SGB-слова, не включающего букву `y`, получим 5-битовое бинарное число, заменяя гласные {`a`, `e`, `i`, `o`, `u`} единицами, а остальные буквы — нулями. Каковы наиболее распространенные слова для каждых из 32 бинарных значений?

▶ **35.** [*26*] Шестнадцать тщательно отобранных элементов `WORDS`(1000) приводят к ветвящейся структуре

,

которая представляет собой полный бинарный луч слов, начинающихся с буквы `s`. Однако такой структуры для слов, начинающихся с буквы `a`, не существует, даже если рассмотреть всю коллекцию `WORDS`(5757).

Какие буквы алфавита могут быть использованы в качестве начальной буквы шестнадцати слов, образующих полный бинарный луч в `WORDS`(n) для заданного n?

36. [*M17*] Поясните симметрии, наблюдающиеся в кубе из слов (10). Покажите также, что еще два таких куба можно получить изменением только двух слов — {`stove`, `event`}.

37. [*20*] Какие вершины графа *words*$(5757, 0, 0, 0)$ имеют максимальную степень?

38. [*22*] Используя правило ориентированного графа из (14), замените `tears` на `smile` за три шага, *не прибегая к помощи компьютера.*

39. [*M00*] Является ли $G \setminus e$ порожденным подграфом G? Является ли он остовным подграфом?

40. [*M15*] Сколько (а) остовных и (б) порожденных подграфов имеет граф $G = (V, E)$, когда $|V| = n$ и $|E| = e$?

41. [*M10*] Для каких целых чисел n мы имеем: (а) $K_n = P_n$? (б) $K_n = C_n$?

42. [*15*] (Д. Г. Лемер (D. H. Lehmer).) Пусть G — граф с 13 вершинами, каждая вершина которого имеет степень 5. Сформулируйте нетривиальное утверждение о G.

43. [*23*] Является ли какой-либо из приведенных графов графом Петерсена?

44. [*M23*] Сколько симметрий имеет граф Чватала? (См. рис. 2, (е).)

45. [*20*] Найдите простой способ 4-раскрашивания планарного графа (17). Хватит ли вам трех красок?

46. [*M25*] Пусть G представляет собой граф с $n \geq 3$ вершинами, определяемыми планарной диаграммой, и являющийся "максимальным" в том смысле, что между несмежными вершинами нельзя провести никаких дополнительных линий без пересечения уже существующих.

а) Докажите, что диаграмма разбивает плоскость на области, каждая из которых имеет на своих границах ровно три вершины. (Одной из этих областей являются все точки, лежащие за пределами диаграммы.)

б) Следовательно, G имеет ровно $3n-6$ ребер.

47. [*M22*] Докажите, что полный биграф $K_{3,3}$ не является планарным.

48. [*M25*] Завершите доказательство теоремы B, показав, что описанная процедура никогда не дает один и тот же цвет двум соседним вершинам.

49. [*18*] Начертите диаграммы для всех кубических графов не более чем с 6 вершинами.

50. [*M24*] Найдите все двудольные графы, которые могут быть 3-раскрашены ровно 24 способами.

▶ **51.** [*M22*] Для заданной геометрической сети, такой как описанная в упр. 21, постройте двудольный граф, вершинами которого являются точки p и прямые L сети, причем p — L тогда и только тогда, когда $p \in L$. Чему равен *обхват* этого графа?

52. [*M16*] Найдите простое неравенство, которое связывает диаметр и его обхват. (Насколько малым может быть диаметр при большом обхвате?)

53. [*15*] Какое из слов `world` и `happy` принадлежит гигантскому компоненту графа $words(5757, 0, 0, 0)$?

▶ **54.** [*21*] Вот список из 49 почтовых кодов в графе (17) в алфавитном порядке: `AL`, `AR`, `AZ`, `CA`, `CO`, `CT`, `DC`, `DE`, `FL`, `GA`, `IA`, `ID`, `IL`, `IN`, `KS`, `KY`, `LA`, `MA`, `MD`, `ME`, `MI`, `MN`, `MO`, `MS`, `MT`, `NC`, `ND`, `NE`, `NH`, `NJ`, `NM`, `NV`, `NY`, `OH`, `OK`, `OR`, `PA`, `RI`, `SC`, `SD`, `TN`, `TX`, `UT`, `VA`, `VT`, `WA`, `WI`, `WV`, `WY`.

а) Предположим, что мы рассматриваем два штата как соседние, если их почтовые коды совпадают в одной букве (т. е. `AL` — `AR` — `OR` — `OH` и т. д.). Каковы компоненты данного графа?

б) Образуем ориентированный граф с `XY` $\longrightarrow$ `YZ` (например, `AL` $\longrightarrow$ `LA` $\longrightarrow$ `AR` и т. д.). Что собой представляют *сильно связанные компоненты* этого ориентированного графа? (См. раздел 2.3.4.2.)

в) В США, помимо почтовых кодов в (17), имеются дополнительные почтовые коды, а именно `AA`, `AE`, `AK`, `AP`, `AS`, `FM`, `GU`, `HI`, `MH`, `MP`, `PW`, `PR`, `VI`. Рассмотрите заново вопрос из п. (б) с использованием всех 62 почтовых кодов.

55. [*M20*] Сколько ребер имеется в полном k-дольном графе $K_{n_1,\ldots,n_k}$?

▶ **56.** [*M10*] Истинно или ложно утверждение "мультиграф является графом тогда и только тогда, когда соответствующий ориентированный граф является простым"?

57. [*M10*] Истинно или ложно утверждение "вершины u и v находятся в одном и том же связном компоненте ориентированного графа тогда и только тогда, когда либо $d(u,v) < \infty$, либо $d(v,u) < \infty$"?

58. [*M17*] Опишите все (а) графы и (б) мультиграфы, являющиеся регулярными степени 2.

▶ **59.** [*M23*] *Турнир* порядка n представляет собой ориентированный граф из n вершин, имеющий ровно $\binom{n}{2}$ дуг, либо $u \longrightarrow v$, либо $v \longrightarrow u$ для каждой пары различных вершин $\{u, v\}$.

а) Докажите, что каждый турнир содержит ориентированный остовный путь $v_1 \longrightarrow \cdots \longrightarrow v_n$.

б) Рассмотрим турнир на вершинах $\{0, 1, 2, 3, 4\}$, в котором $u \longrightarrow v$ тогда и только тогда, когда $(u - v) \bmod 5 \geq 3$. Сколько ориентированных остовных путей он имеет?

в) Является ли $\vec{K_n}$ единственным турниром порядка n, который имеет единственный ориентированный остовный путь?

▶ **60.** [*M22*] Пусть u — вершина с наибольшей исходящей степенью в турнире и пусть v — любая другая вершина. Докажите, что $d(u,v) \le 2$.

61. [*M16*] Постройте ориентированный граф, имеющий k обходов длиной k из вершины 1 в вершину 2.

62. [*M21*] *Перестановочный ориентированный граф* представляет собой ориентированный граф, каждая вершина которого имеет входящую и исходящую степени, равные 1; следовательно, его компонентами являются ориентированные циклы. Если он имеет n вершин и k компонентов, то мы назовем его *четным* при четном $n-k$ и *нечетным* при $n-k$ нечетном.

а) Пусть G — ориентированный граф с матрицей смежности A. Докажите, что количество остовных перестановочных ориентированных графов G равно $\operatorname{per} A$ (перманенту A).

б) Интерпретируйте определитель $\det A$ в терминах остовных перестановочных ориентированных графов.

63. [*M23*] Пусть G — граф с обхватом g, каждая вершина которого имеет как минимум d соседей. Докажите, что G имеет как минимум N вершин, где

$$N = \begin{cases} 1 + \sum_{0\le k<t} d(d-1)^k, & \text{если } g = 2t+1; \\ 1 + (d-1)^t + \sum_{0\le k<t} d(d-1)^k, & \text{если } g = 2t+2. \end{cases}$$

▶ **64.** [*M21*] Продолжая выполнять упр. 63, покажите, что имеется *единственный* граф с обхватом 4, минимальной степенью d и порядком $2d$ для каждого $d \ge 2$.

▶ **65.** [*HM31*] Предположим, что граф G имеет обхват 5, минимальную степень d и $N = d^2+1$ вершин.

а) Докажите, что матрица смежности A графа G удовлетворяет уравнению $A^2 + A = (d-1)I + J$.

б) Поскольку A — симметричная матрица, она имеет N ортогональных собственных векторов x_j с соответствующими собственными значениями λ_j, такими, что $Ax_j = \lambda_j x_j$ для $1 \le j \le N$. Докажите, что каждое λ_j равно либо d, либо $(-1 \pm \sqrt{4d-3})/2$.

в) Покажите, что если $\sqrt{4d-3}$ иррационально, то $d = 2$. *Указание*: $\lambda_1 + \cdots + \lambda_N = \operatorname{trace}(A) = 0$.

г) А если $\sqrt{4d-3}$ рационально, то $d \in \{3, 7, 57\}$.

66. [*M30*] Продолжая выполнять упр. 65, постройте такой граф для $d = 7$.

67. [*M48*] Существует ли регулярный граф со степенью 57, порядком 3250 и обхватом 5?

68. [*M20*] Сколько различных матриц смежности имеет граф G с n вершинами?

▶ **69.** [*20*] Расширяя (31), поясните, как вычислить и исходящую степень `ODEG(v)`, и входящую степень `IDEG(v)` для *всех* вершин v графа, представленного в SGB-формате.

▶ **70.** [*M20*] Как часто выполняется каждый шаг алгоритма B, когда этот алгоритм успешно выполняет 2-раскрашивание графа с m дугами и n вершинами?

71. [*26*] Реализуйте алгоритм B для компьютера `MMIX` на языке ассемблера `MMIXAL`. Считайте, что в момент начала работы вашей программы регистр `v0` указывает на узел первой вершины, а регистр `n` содержит количество вершин.

▶ **72.** [*M22*] Когда `COLOR(v)` установлен на шаге B6, назовем u *родителем* v; но когда `COLOR(w)` установлен на шаге B3, будем говорить, что w не имеет родителя. Определим (включительно) *предков* вершины v рекурсивно как v вместе с предками родителя v (если таковой имеется).

а) Докажите, что если v в процессе работы алгоритма B находится в стеке ниже u, то родителем v является предок u.

б) Кроме того, если COLOR(v) = COLOR(u) на шаге B6, v находится в данный момент в стеке.

в) Воспользуйтесь этими фактами для расширения алгоритма B таким образом, чтобы, если заданный граф не является двудольным, выводились имена вершин цикла нечетной длины.

73. [*15*] Какое другое имя имеет граф *random_graph*$(10, 45, 0, 0, 0, 0, 0, 0, 0, 0)$?

74. [*21*] Какая вершина *roget*$(1022, 0, 0, 0)$ имеет наибольшую исходящую степень?

75. [*22*] Генератор графов SGB *board*$(n_1, n_2, n_3, n_4, p, w, o)$ создает граф, вершинами которого являются t-мерные целочисленные векторы $(x_1, \ldots, x_t)$ для $0 \le x_i < b_i$, определяемые первыми четырьмя параметрами (n_1, n_2, n_3, n_4) следующим образом. Установим $n_5 \leftarrow 0$, и пусть значение $j \ge 0$ — минимальное такое, что $n_{j+1} \le 0$. Если $j = 0$, установим $b_1 \leftarrow b_2 \leftarrow 8$ и $t \leftarrow 2$; это стандартная шахматная доска размером 8×8. В противном случае если $n_{j+1} = 0$, установим $b_i \leftarrow n_i$ для $1 \le i \le j$ и $t \leftarrow j$. Наконец, если $n_{j+1} < 0$, установим $t \leftarrow |n_{j+1}|$ и установим b_i равным i-му элементу периодической последовательности $(n_1, \ldots, n_j, n_1, \ldots, n_j, n_1, \ldots)$. (Например, значения $(n_1, n_2, n_3, n_4) = (2, 3, 5, -7)$ дают семимерную шахматную доску с $(b_1, \ldots, b_7) = (2, 3, 5, 2, 3, 5, 2)$, следовательно, граф имеет $2 \cdot 3 \cdot 5 \cdot 2 \cdot 3 \cdot 5 \cdot 2 = 1800$ вершин.)

Остальные параметры (p, w, o) указывают часть, завертывание и ориентацию, определяя дуги графа. Положим сначала, что $w = o = 0$. Если $p > 0$, имеем $(x_1, \ldots, x_t) \longrightarrow (y_1, \ldots, y_t)$ тогда и только тогда, когда $y_i = x_i + \delta_i$ для $1 \le i \le t$, где $(\delta_1, \ldots, \delta_t)$ представляет собой целочисленное решение уравнения $\delta_1^2 + \cdots + \delta_t^2 = |p|$. А если $p < 0$, мы также позволяем $y_i = x_i + k\delta_i$ для $k \ge 1$, что соответствует k ходам в том же направлении.

Если $w \ne 0$, пусть в двоичной записи w имеет вид $(w_t \ldots w_1)_2$. Тогда мы разрешаем "завертывание" $y_i = (x_i + \delta_i) \bmod b_i$ или $y_i = (x_i + k\delta_i) \bmod b_i$ по каждой координате i, для которой $w_i = 1$.

Если $o \ne 0$, граф ориентированный; смещения $(\delta_1, \ldots, \delta_t)$ дают дуги только тогда, когда они лексикографически больше, чем $(0, \ldots, 0)$. Но если $o = 0$, граф неориентированный.

Найдите установки $(n_1, n_2, n_3, n_4, p, w, o)$, для которых *board* будет давать следующие фундаментальные графы: (а) полный граф K_n; (б) путь P_n; (в) цикл C_n; (г) транзитивный турнир $K_n^{\rightarrow}$; (д) ориентированный путь $P_n^{\rightarrow}$; (е) ориентированный цикл $C_n^{\rightarrow}$; (ж) $m \times n$-решетку $P_m \square P_n$; (з) $m \times n$-цилиндр $P_m \square C_n$; (и) $m \times n$-тор $C_m \square C_n$; (к) $m \times n$-граф ладьи $K_m \square K_n$; (л) $m \times n$-ориентированный тор $C_m^{\rightarrow} \square C_n^{\rightarrow}$; (м) нулевой граф $\overline{K_n}$; (н) n-мерный куб $P_2 \square \cdots \square P_2$ с 2^n вершинами.

76. [*20*] Может ли *board*$(n_1, n_2, n_3, n_4, p, w, o)$ создавать петли или параллельные (повторяющиеся) ребра?

77. [*M20*] Докажите, что $\overline{G}$ имеет диаметр ≤ 3, если граф G имеет диаметр ≥ 3.

78. [*M27*] Пусть $G = (V, E)$ — граф с $|V| = n$ и $G \cong \overline{G}$. (Другими словами, граф G *самодополнительный*: существует перестановка φ множества V, такая, что u — v тогда и только тогда, когда $\varphi(u) \not\!\!-\!\!\!- \varphi(v)$ и $u \ne v$. Можно представить, что ребра K_n раскрашены черным и белым цветами; белые ребра определяют граф, изоморфный графу с черными ребрами.)

а) Докажите, что $n \bmod 4 = 0$ или 1. Начертите диаграммы всех таких графов с $n < 8$.

б) Докажите, что если $n \bmod 4 = 0$, то каждый цикл перестановки φ имеет длину, кратную 4.

в) И обратно, каждая перестановка φ с такими циклами возникает в некотором таком графе G.

г) Распространите эти результаты на случай $n \bmod 4 = 1$.

▶ **79.** [*M22*] Для заданного $k \geq 0$ постройте граф на вершинах $\{0, 1, \ldots, 4k\}$, который одновременно является регулярным и самодополнительным.

▶ **80.** [*M22*] Самодополнительный граф должен иметь диаметр, равный 2 или 3 в соответствии с упр. 77. Для заданного $k \geq 2$ постройте самодополнительные графы обоих возможных диаметров, когда (а) $V = \{1, 2, \ldots, 4k\}$; (б) $V = \{0, 1, 2, \ldots, 4k\}$.

81. [*20*] Дополнение простого ориентированного графа без циклов определяется путем расширения (35) и (36), так что мы имеем $u \to v$ в $\overline{D}$ тогда и только тогда, когда $u \neq v$ и $u \not\to v$ в D. Что собой представляют самодополнительные ориентированные графы порядка 3?

82. [*M21*] Истинны или ложны приведенные далее утверждения о реберных графах?

а) Если G содержится в G', то $L(G)$ является порожденным подграфом $L(G')$.
б) Если G является регулярным графом, то же справедливо и для $L(G)$.
в) $L(K_{m,n})$ является регулярным для всех $m, n > 0$.
г) $L(K_{m,n,r})$ является регулярным для всех $m, n, r > 0$.
д) $L(K_{m,n}) \cong K_m \,\square\, K_n$.
е) $L(K_4) \cong K_{2,2,2}$.
ж) $L(P_{n+1}) \cong P_n$.
з) Графы G и $L(G)$ имеют одно и то же количество компонентов.

83. [*16*] Начертите граф $\overline{L(K_5)}$.

▶ **84.** [*M21*] Является ли $L(K_{3,3})$ самодополнительным?

85. [*M22*] (O. Ore (O. Ore), 1962.) Для каких графов G выполняется $G \cong L(G)$?

86. [*M20*] (Р. Д. Вильсон (R. J. Wilson).) Найдите граф G порядка 6, для которого $\overline{G} \cong L(G)$.

87. [*20*] Является ли граф Петерсена (а) 3-раскрашиваемым? (б) 3-реберно раскрашиваемым?

88. [*M20*] Граф $W_n = K_1 — C_{n-1}$ называется *колесом* порядка n, где $n \geq 4$. Сколько циклов содержит этот граф в качестве подграфов?

W_8

89. [*M20*] Докажите законы ассоциативности (42) и (43).

▶ **90.** [*M24*] Граф называется *сографом* (*cograph*), если он может быть построен алгебраически из одноэлементных графов посредством операций дополнения и прямого суммирования. Например, имеется четыре неизоморфных графа порядка 3, и все они являются сографами: $\overline{K_3} = K_1 \oplus K_1 \oplus K_1$ и его дополнение, K_3; $\overline{K_{1,2}} = K_1 \oplus K_2$ и его дополнение, $K_{1,2}$, где $K_2 = \overline{K_1 \oplus K_1}$.

Исчерпывающее перечисление показывает, что имеется 11 неизоморфных графов порядка 4. Приведите алгебраические формулы, доказывающие, что 10 из них являются сографами. Какой из них не является сографом?

▶ **91.** [*20*] Начертите диаграммы графов с четырьмя вершинами: (а) $K_2 \,\square\, K_2$; (б) $K_2 \otimes K_2$; (в) $K_2 \boxtimes K_2$; (г) $K_2 \vartriangle K_2$; (д) $K_2 \circ K_2$; (е) $\overline{K_2} \circ K_2$; (ж) $K_2 \circ \overline{K_2}$.

92. [*21*] Пять типов произведений графов, определенных в тексте, хорошо работают как для простых ориентированных графов, так и для обычных графов. Начертите диаграммы для ориентированных графов с четырьмя вершинами (а) $\vec{K_2} \,\square\, \vec{K_2}$; (б) $\vec{K_2} \otimes \vec{K_2}$; (в) $\vec{K_2} \boxtimes \vec{K_2}$; (г) $\vec{K_2} \vartriangle \vec{K_2}$; (д) $\vec{K_2} \circ \vec{K_2}$.

93. [*15*] Какие из пяти произведений графов превращают K_m и K_n в K_{mn}?

94. [*10*] Являются ли SGB-графы *words* порожденными подграфами $P_{26} \,\square\, P_{26} \,\square\, P_{26} \,\square\, P_{26} \,\square\, P_{26}$?

95. [*M20*] Если вершина u графа G имеет степень d_u, а вершина v графа H имеет степень d_v, то какова степень вершины (u, v) в (а) $G \square H$? (б) $G \otimes H$? (в) $G \boxtimes H$? (г) $G \triangle H$? (д) $G \circ H$?

▶ **96.** [*M22*] Пусть A является матрицей $m \times m'$ с элементами $a_{uu'}$ на пересечении строки u и столбца u'; и пусть B является матрицей $n \times n'$ с элементами $b_{vv'}$ на пересечении строки v и столбца v'. *Прямое произведение* $A \otimes B$ представляет собой матрицу $mn \times m'n'$ с элементами $a_{uu'}b_{vv'}$ на пересечении строки (u, v) и столбца (u', v'). Таким образом, $A \otimes B$ представляет собой матрицу смежности для $G \otimes H$, если A и B являются матрицами смежности G и H.

Найдите аналогичные формулы для матриц смежности (а) $G \square H$; (б) $G \boxtimes H$; (в) $G \triangle H$; (г) $G \circ H$.

97. [*M25*] Найдите столько интересных алгебраических соотношений между суммами и произведениями графов, сколько сможете. (Например, закон дистрибутивности $(A \oplus B) \otimes C = (A \otimes C) \oplus (B \otimes C)$ для прямых сумм и произведений матриц влечет за собой $(G \oplus G') \otimes H = (G \otimes H) \oplus (G' \otimes H)$. Мы также имеем $\overline{K_m} \square H = H \oplus \cdots \oplus H$ с m копиями H, и т. д.)

98. [*M20*] Если граф G имеет k компонентов, а граф H имеет l компонентов, то сколько имеется компонентов в графах $G \square H$ и $G \boxtimes H$?

99. [*M20*] Пусть $d_G(u, u')$ — расстояние от вершины u до вершины u' в графе G. Докажите, что $d_{G \square H}((u, v), (u', v')) = d_G(u, u') + d_H(v, v')$, и найдите аналогичную формулу для $d_{G \boxtimes H}((u, v), (u', v'))$.

100. [*M21*] Для каких связных графов $G \otimes H$ связно?

▶ **101.** [*M25*] Найдите все связные графы G и H, такие, что $G \square H \cong G \otimes H$.

102. [*M20*] Какова простейшая алгебраическая формула для графа *ходов королем* (по одной клетке по горизонтали, вертикали или диагонали) на доске $m \times n$?

103. [*20*] Завершите таблицу (54). Примените также алгоритм H к последовательности 866444444.

104. [*18*] Поясните манипуляции переменными i, t и r на шагах H3 и H4.

105. [*M38*] Предположим, что $d_1 \geq \cdots \geq d_n \geq 0$, и пусть $c_1 \geq \cdots \geq c_{d_1}$ — сопряжение этой последовательности, как в алгоритме H. Докажите, что последовательность $d_1 \ldots d_n$ является графической тогда и только тогда, когда $d_1 + \cdots + d_n$ четно и $d_1 + \cdots + d_k \leq c_1 + \cdots + c_k - k$ для $1 \leq k \leq s$, где s — максимальное значение, при котором $d_s \geq s$.

106. [*20*] Истинно или ложно следующее утверждение? Если $d_1 = \cdots = d_n = d < n$ и nd четно, алгоритм H строит *связный* граф.

107. [*M21*] Докажите, что последовательность степеней $d_1 \ldots d_n$ самодополнительного графа удовлетворяет условиям $d_j + d_{n+1-j} = n - 1$ и $d_{2j-1} = d_{2j}$ для $1 \leq j \leq n/2$.

▶ **108.** [*M23*] Разработайте алгоритм, аналогичный алгоритму H, который строит *простой ориентированный граф* на вершинах $\{1, \ldots, n\}$, имеющий определенные значения d_k^- и d_k^+ в качестве входящей и исходящей степеней каждой вершины k, если существует как минимум один такой граф.

109. [*M20*] Разработайте алгоритм, аналогичный алгоритму H, который строит *двудольный граф* на вершинах $\{1, \ldots, m + n\}$, имеющий определенные степени d_k для каждой вершины k, когда это возможно; все ребра j — k должны иметь $j \leq m$ и $k > m$.

110. [*M22*] Не прибегая к алгоритму H, покажите путем непосредственного построения, что последовательность $d_1 \ldots d_n$ является графической, когда $n > d_1 \geq \cdots \geq d_n \geq d_1 - 1$ и $d_1 + \cdots + d_n$ четно.

▶ **111.** [*25*] Пусть G представляет собой граф на вершинах $V = \{1, \ldots, n\}$ со степенью d_k у вершины k и $\max(d_1, \ldots, d_n) = d$. Докажите, что существует целое число N ($n \le N \le 2n$) и граф H на вершинах $\{1, \ldots, N\}$, такой, что H является регулярным графом степени d и $H \mid V = G$. Поясните, как построить такой регулярный граф с наименьшим возможным значением N.

▶ **112.** [*20*] Имеются ли в сети *miles*$(128, 0, 0, 0, 0, 127, 0)$ три эквидистантных города? Если нет, какие три города оказываются ближе всего к вершинам равностороннего треугольника?

113. [*05*] Если H является гиперграфом с m ребрами и n вершинами, то сколько строк и столбцов имеет его матрица инциденций?

114. [*M20*] Предположим, что мультиграф (26) рассматривается как гиперграф. Какова соответствующая матрица инциденций? Какой вид имеет соответствующий двудольный мультиграф?

▶ **115.** [*M20*] Пусть B представляет собой матрицу инциденций графа G. Поясните смысл симметричных матриц B^TB и BB^T.

116. [*M17*] Опишите ребра полного двудольного r-однородного гиперграфа $K_{m,n}^{(r)}$.

117. [*M22*] Сколько неизоморфных 1-однородных гиперграфов имеют m ребер и n вершин? (Ребра могут повторяться.) Перечислите их все для $m = 4$ и $n = 3$.

118. [*M20*] "Гиперлесом" называется гиперграф, не имеющий циклов. Если гиперлес имеет m ребер, n вершин и p компонентов, то чему равна сумма степеней его вершин?

119. [*M18*] Какой гиперграф соответствует (60) без последнего члена $(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_4)$?

120. [*M20*] Определите *ориентированный гиперграф*, обобщая концепцию ориентированных графов.

121. [*M19*] Для заданного гиперграфа $H = (V, E)$ положим $I(H) = (V, F)$, где F — семейство всех максимальных независимых множеств гиперграфа H. Выразите $\chi(H)$ через $|V|$, $|F|$ и $\alpha(I(H)^T)$.

▶ **122.** [*M24*] Найдите наибольшее независимое множество и минимальное раскрашивание следующих систем троек: (а) гиперграф (56); (б) граф, дуальный графу Петерсена.

123. [*17*] Покажите, что оптимальное раскрашивание $K_n \,\square\, K_n$ эквивалентно решению знаменитой комбинаторной задачи.

124. [*M22*] Каково хроматическое число графа Чватала (рис. 2, (е))?

125. [*M48*] Для каких значений g существует 4-регулярный 4-хроматический граф с обхватом g?

▶ **126.** [*M22*] Найдите оптимальное раскрашивание "королевского тора" $C_m \boxtimes C_n$ при значениях $m, n \ge 3$.

127. [*M22*] Докажите, что (а) $\chi(G) + \chi(\overline{G}) \le n + 1$ и (б) $\chi(G)\chi(\overline{G}) \ge n$, когда G является графом порядка n, и найдите графы, для которых выполняется равенство.

128. [*M18*] Выразите $\chi(G \,\square\, H)$ через $\chi(G)$ и $\chi(H)$, когда G и H являются графами.

129. [*23*] Опишите максимальные клики графа ходов ферзя (37).

130. [*M20*] Сколько максимальных клик в полном k-дольном графе?

131. [*M30*] Пусть $N(n)$ — наибольшее количество максимальных клик, которое может иметь граф с n вершинами. Докажите, что $3^{\lfloor n/3 \rfloor} \le N(n) \le 3^{\lceil n/3 \rceil}$.

▶ **132.** [*M20*] Назовем G *плотно раскрашиваемым* (*tightly colorable*), если $\chi(G) = \omega(G)$. Докажите, что $\chi(G \boxtimes H) = \chi(G)\chi(H)$, если G и H плотно раскрашиваемы.

133. [*21*] Показанный "музыкальный граф" предоставляет хорошую возможность наглядно представить многочисленные определения, дававшиеся в этом разделе, поскольку его свойства легко проанализировать. Определите для данного графа (а) порядок; (б) размер; (в) обхват; (г) диаметр; (д) число независимости, $\alpha(G)$; (е) хроматическое число, $\chi(G)$; (ж) реберно-хроматическое число, $\chi(L(G))$; (з) количество клик, $\omega(G)$; (и) алгебраическую формулу графа, выражающую его как произведение хорошо известных меньших графов. Чему равен размер (к) наименьшего вершинного покрытия? (л) наибольшего паросочетания? Является ли граф G (м) регулярным? (н) планарным? (о) связным? (п) ориентированным? (р) свободным деревом? (с) гамильтоновым графом?

134. [*M22*] Сколько автоморфизмов имеет музыкальный граф?

▶ **135.** [*HM26*] Предположим, что композитор осуществляет случайное блуждание в музыкальном графе, начиная с вершины C, и делая на каждом шаге выбор одного из пяти ребер с равной вероятностью. Покажите, что после четного количества шагов блуждание с более высокой вероятностью завершится в вершине C, чем в любой другой. Какова точная вероятность попасть из вершины C в вершину C в результате блуждания в 12 шагов?

136. [*HM23*] *Ориентированный граф Кейли* представляет собой ориентированный граф, вершины V которого являются элементами группы, а его дугами являются $v \longrightarrow v\alpha_j$ для $1 \le j \le d$ и всех вершин v, где $(\alpha_1, \ldots, \alpha_d)$ — фиксированные элементы группы. *Граф Кейли* — это ориентированный граф Кейли, который одновременно представляет собой граф. Является ли граф Петерсена графом Кейли?

▶ **137.** [*M25*] (*Обобщенные торы.*) Тор размером $m \times n$ можно рассматривать как замощение плоскости. Например, можно представить, что бесконечное количество копий торов 3×4 из (50) размещаются вместе в виде решетки, как показано на приведенном выше рисунке слева; из каждой вершины можно перемещаться на север юг, восток или запад в следующую вершину тора. Вершины здесь были пронумерованы таким образом, чтобы перемещение на север из вершины v приводило в вершину $(v + 4) \bmod 12$, перемещение на восток — в вершину $(v + 3) \bmod 12$, и т. д. В правой части рисунка показан тот же тор, но с несколько иной формой мозаичного элемента; *любой* вариант выбора двенадцати ячеек с номерами $\{0, 1, \ldots, 11\}$ будет замащивать плоскость, с точно тем же лежащим в основе замощения графом.

Смещенные копии одной плитки будут также заполнять плоскость, если они образуют *обобщенный* тор, в котором ячейка (x, y) соответствует той же вершине, что и ячейки

$(x+a, y+b)$ и $(x+c, y+d)$, где (a,b) и (c,d) — целочисленные векторы, и $n = ad - bc > 0$. Тогда обобщенный тор будет иметь n точек. Эти векторы (a,b) и (c,d) в приведенном выше примере 3×4 представляют собой $(4,0)$ и $(0,3)$; а когда они становятся соответственно $(5,2)$ и $(1,3)$, мы получаем

Здесь $n = 13$, и перемещение в северном направлении из v дает $(v+4) \bmod 13$; перемещение в восточном направлении приводит в $(v+1) \bmod 13$.

Докажите, что если $\gcd(a,b,c,d) = 1$, то вершинам такого обобщенного тора всегда можно присвоить целочисленные метки $\{0, 1, \ldots, n-1\}$ таким образом, что соседями v являются $(v \pm p) \bmod n$ и $(v \pm q) \bmod n$ для некоторых целых чисел p и q.

138. [*HM27*] Продолжая выполнять упр. 137, придумайте эффективный способ присваивания меток k-мерным вершинам $x = (x_1, \ldots, x_k)$, когда заданы целочисленные векторы α_j, такие, что каждый вектор x эквивалентен $x + \alpha_j$ для $1 \le j \le k$. Проиллюстрируйте свой метод для случая $k = 3$, $\alpha_1 = (3,1,1)$, $\alpha_2 = (1,3,1)$, $\alpha_3 = (1,1,3)$.

▶ **139.** [*M22*] Пусть H представляет собой фиксированный граф порядка h, а $\#(H{:}G)$ — количество раз, когда H оказывается порожденным подграфом данного графа G. Если G выбран случайным образом из множества всех $2^{n(n-1)/2}$ графов на вершинах $V = \{1, 2, \ldots, n\}$, то чему равно среднее значение $\#(H{:}G)$ для случаев, когда H представляет собой (а) K_h; (б) P_h для $h > 1$; (в) C_h для $h > 2$; (г) произвольный граф?

140. [*M30*] Граф G называется *пропорциональным*, если каждое из его количеств порожденных подграфов $\#(K_3{:}G)$, $\#(\overline{K_3}{:}G)$ и $\#(P_3{:}G)$ согласуется с ожидаемыми значениями, полученными в упр. 139.

а) Покажите, что колесо W_8 из упр. 88 пропорционально в данном смысле.

б) Докажите, что граф G пропорционален тогда и только тогда, когда $\#(K_3{:}G) = \frac{1}{8}\binom{n}{3}$ и последовательность степеней $d_1 \ldots d_n$ его вершин удовлетворяет тождествам

$$d_1 + \cdots + d_n = \binom{n}{2}, \qquad d_1^2 + \cdots + d_n^2 = \frac{n}{2}\binom{n}{2}. \qquad (*)$$

141. [*26*] Условия из упр. 140, (б) могут выполняться, только если $n \bmod 16 \in \{0, 1, 8\}$. Напишите программу для поиска всех пропорциональных графов с $n = 8$ вершинами.

142. [*M30*] (С. Янсон (S. Janson) и Я. Кратохвил (J. Kratochvil), 1991.) Докажите, что не существует графа G на четырех или более вершинах, являющегося "сверхпропорциональным" в том смысле, что количество его подграфов $\#(H{:}G)$ согласуется с ожидаемыми значениями из упр. 139 для каждого из одиннадцати неизоморфных графов H порядка 4, которые рассматривались в упр. 90. *Указание*: обратите внимание, что $(n-3)\#(K_3{:}G) = 4\#(K_4{:}G) + 2\#(K_{1,1,2}{:}G) + \#(\overline{K_{1,3}}{:}G) + \#(\overline{K_1 \oplus K_{1,2}}{:}G)$.

▶ **143.** [*M25*] Пусть A является произвольной матрицей с $m > 1$ различными строками и $n \ge m$ столбцами. Докажите, что как минимум один столбец A может быть удален так, что при этом никакие две строки не станут равными.

▶ **144.** [*21*] Пусть X — матрица размером $m \times n$, элементами x_{ij} которой являются 0, 1 или $*$. "Дополнением" X является матрица X^*, в которой каждая $*$ заменена на 0 либо 1.

Покажите, что задача поиска дополнения с наименьшим количеством различных строк эквивалентна задаче поиска хроматического числа графа.

▶ **145.** [*25*] (Р. С. Бойер (R. S. Boyer) и Д. С. Мур (J. S. Moore), 1980.) Предположим, что массив $a_1 \ldots a_n$ содержит *мажоритарный элемент*, т. е. значение, которое встречается более чем $n/2$ раз. Разработайте алгоритм, который находит его менее чем за n сравнений. *Указание*: если $n \geq 3$ и $a_{n-1} \neq a_n$, мажоритарный элемент $a_1 \ldots a_n$ является также мажоритарным элементом $a_1 \ldots a_{n-2}$.

При использовании основания 2 получающиеся единицы можно называть бинарными цифрами (binary digits) или, сокращенно, битами *(*bits*) — словом, предложенным Д. У. Таки (J. W. Tukey).*

— КЛОД Э. ШЕННОН (CLAUDE E. SHANNON), в *Bell System Technical Journal* (1948)

bit (bit), n. . . *[A] boring tool (*буровой инструмент*)* . . .

— *Random House Dictionary of the English Language* (1987)

7.1. НУЛИ И ЕДИНИЦЫ

Комбинаторные алгоритмы часто требуют особого внимания к эффективности, и правильное представление данных — важный способ получения необходимой скорости. Таким образом, разумно расширить наши знания об элементарных методах представления, прежде чем приступать к детальному изучению комбинаторных алгоритмов.

Большинство современных компьютеров основаны на двоичной системе счисления вместо работы непосредственно с десятичными числами, которые предпочитают люди, поскольку машины особенно эффективны при работе с величинами, описываемыми двумя состояниями (включено–выключено), которые обычно обозначаются двумя цифрами — 1 и 0. В главах 1–6 мы почти не использовали тот факт, что двоичные компьютеры могут делать некоторые вещи быстро, чего не могут десятичные компьютеры. Двоичные компьютеры обычно выполняют "логические", или "битовые", операции так же быстро, как сложение или вычитание; однако мы редко используем эту возможность. Мы видели, что для множества применений двоичные и десятичные компьютеры не слишком отличаются, но в некотором смысле можно сказать, что мы просили двоичный компьютер работать с одной рукой, привязанной за спиной.

Удивительная применимость нулей и единиц для кодирования информации, так же как и для кодирования логических отношений между элементами и даже для кодирования алгоритмов обработки информации, делает изучение битов особенно ценным. В действительности мы не только применяем битовые операции для усовершенствования комбинаторных алгоритмов, но и обнаруживаем, что свойства бинарной логики естественным путем приводят к новым комбинаторным задачам, которые представляют большой интерес сами по себе.

Ученые-информатики постепенно укрощают дикие нули и единицы, постоянно совершенствуясь в этой деятельности, и заставляют делать их разные полезные трюки. Но прежде чем приступить к изучению высокоуровневых концепций и методов, желательно хорошо понять низкоуровневые свойства бинарных величин. Так что мы начнем с исследования базовых способов объединения отдельных битов и их последовательностей.

7.1.1. Основы булевой алгебры

Имеется 16 возможных функций $f(x, y)$, которые преобразуют два данных бита x и y в третий бит $z = f(x, y)$, поскольку имеется по два выбора для каждого из значений

$f(0,0)$, $f(0,1)$, $f(1,0)$ и $f(1,1)$. В табл. 1 указаны имена и обозначения, традиционно связанные с этими функциями при изучении формальной логики, и считается, что 1 соответствует истине, а 0 — лжи. Последовательность четырех значений $f(0,0)f(0,1)f(1,0)f(1,1)$ привычно называется *таблицей истинности* функции f.

Давайте представим алгебру, в которой символы
x, y, z *и т. д. принимают значения* 0 *и* 1 *и только их.*
— ДЖОРДЖ БУЛЬ (GEORGE BOOLE),
Исследование законов мышления (*An Investigation of the Laws of Thought*) (1854)

— И задом наперед, совсем наоборот, — подхватил Траляля.
— Если бы это было так, это бы еще ничего, а если бы ничего,
оно бы так и было, но так как это не так, так оно и не этак!
*Такова логика вещей!**
— ЛЬЮИС КЭРРОЛЛ (LEWIS CARROLL),
Алиса в Зазеркалье (*Through the Looking Glass*) (1871)

Такие функции часто называют булевыми операциями в честь Джорджа Буля (George Boole), который первым открыл, что алгебраические операции над нулями и единицами могут использоваться для построения исчисления для логических рассуждений [*The Mathematical Analysis of Logic* (Cambridge: 1847); *An Investigation of the Laws of Thought* (London: 1854)]. Однако Буль никогда не работал с операцией "логическое или" $\vee$; он самоограничился строго арифметическими операциями над нулями и единицами. Таким образом, он записывал $x+y$ для обозначения дизъюнкции, но при этом прилагал массу усилий, чтобы не использовать эту запись никогда, кроме случаев, когда x и y оказывались взаимоисключающими (не были оба одновременно равны 1). При необходимости он писал $x+(1-x)y$, чтобы гарантировать, что результат дизъюнкции никогда не будет равен 2.

Переводя операцию $+$ на английский язык, Буль иногда называл ее "и", а иногда — "или". Такие действия могут показаться странными современным математикам, но только до тех пор, пока мы не поймем, что это фактически обычная практика английского языка; мы говорим, например, что "boys **and** girls are children" ("мальчики и девочки являются детьми"), но "children are boys **or** girls" ("дети — это мальчики или девочки").

Позже булево исчисление было расширено включением У. Стенли Джевонсом (W. Stanley Jevons) необычного правила $x+x=x$ [*Pure Logic* (London: Edward Stanford, 1864), §69], который указал, что при использовании его новой операции $+$ выражение $(x+y)z$ оказывается равным $xz+yz$. Но Джевонс не знал другой дистрибутивный закон $xy+z=(x+z)(y+z)$. Похоже, он пропустил его из-за использовавшихся им обозначений, поскольку второй дистрибутивный закон не имеет знакомого двойника в арифметике; более симметричные обозначения $x\wedge y$, $x\vee y$ в табл. 1 упрощают нам запоминание обоих законов дистрибутивности

$$(x\vee y)\wedge z \;=\; (x\wedge z)\vee(y\wedge z); \tag{1}$$

$$(x\wedge y)\vee z \;=\; (x\vee z)\wedge(y\vee z). \tag{2}$$

* Перевод Н. М. Демуровой.

Таблица 1

Шестнадцать логических операций с двумя переменными

Таблица истинности	Новое и старое обозначения	Символ оператора $\circ$	Название
0000	0	$\bot$	Противоречие; ложное заключение; константа 0
0001	$xy,\ \ x \wedge y,\ \ x \,\&\, y$	$\wedge$	Конъюнкция; и
0010	$x \wedge \bar{y},\ \ x \not\supset y,\ \ [x > y],\ \ x \dot{-} y$	$\overline{\supset}$	Отрицание импликации; разность; но не
0011	x	$\llcorner$	Левая проекция; первый диктатор
0100	$\bar{x} \wedge y,\ \ x \not\subset y,\ \ [x < y],\ \ y \dot{-} x$	$\overline{\subset}$	Обратное отрицание импликации; не... но
0101	y	$\lrcorner$	Правая проекция; второй диктатор
0110	$x \oplus y,\ \ x \not\equiv y,\ \ x\hat{\ }y$	$\oplus$	Разделительная дизъюнкция; неэквивалентность; исключающее или (“xor”)
0111	$x \vee y,\ \ x \mid y$	$\vee$	(Неразделительная) дизъюнкция; или; и/или
1000	$\bar{x} \wedge \bar{y},\ \ \overline{x \vee y},\ \ x \,\overline{\vee}\, y,\ \ x \downarrow y$	$\overline{\vee}$	Отрицание дизъюнкции; конъюнкция отрицаний; ни... ни
1001	$x \equiv y,\ \ x \leftrightarrow y,\ \ x \Leftrightarrow y$	$\equiv$	Эквивалентность; тогда и только тогда, когда
1010	$\bar{y},\ \ \neg y,\ \ !y,\ \ \sim y$	$\bar{\lrcorner}$	Правое дополнение
1011	$x \vee \bar{y},\ x \subset y,\ x \Leftarrow y,\ [x \geq y],\ x^y$	$\subset$	Обратная импликация; если
1100	$\bar{x},\ \ \neg x,\ \ !x,\ \ \sim x$	$\bar{\llcorner}$	Левое дополнение
1101	$\bar{x} \vee y,\ x \supset y,\ x \Rightarrow y,\ [x \leq y],\ y^x$	$\supset$	Импликация; только если; если... то
1110	$\bar{x} \vee \bar{y},\ \ \overline{x \wedge y},\ \ x \,\overline{\wedge}\, y,\ \ x \mid y$	$\overline{\wedge}$	Отрицание конъюнкции; не и... и
1111	1	$\top$	Утверждение; достоверность; тавтология; константа 1

Второй закон (2) был введен Ч. С. Пирсом (C. S. Peirce), который независимо открыл способ расширения булева исчисления [*Proc. Amer. Acad. Arts and Sciences* **7** (1867), 250–261]. Кстати, когда Пирс обсуждал эти ранние разработки несколькими годами позже [*Amer. J. Math.* **3** (1880), 32], он упомянул “булеву (Boolian) алгебру с дополнениями Джевонса”; его (ныне незнакомое) произношение слова “Boolean” использовалось много лет, появившись даже в полном словаре Фанка (Funk) и Вагналлса (Wagnalls) в 1963 году.

Понятие комбинации истинных значений на самом деле гораздо старше булевой алгебры. В действительности логика высказываний была разработана греческими философами еще в IV веке до н. э. В те времена велись значительные споры о том, как присвоить правильное значение “истина” или “ложь” суждению “если x, то y”, когда x и y являются суждениями; Филон из Мегары (Philo of Megara) около 300 года до н. э. определил его с помощью таблицы истинности, приведенной в табл. 1, где, в частности, указано, что импликация истинна, если и x, и y ложны. Многие из этих ранних работ были потеряны, но в работах Галена (Galen) (II век н. э.) имеются отрывки, которые указывают на неразделительную и разделительную дизъюнкции суждений. [См. I. M. Bocheński, *Formale Logik* (1956), перевод на английский Айво Томаса (Ivo Thomas) (1961), где приведен превосходный обзор развития логики от древних времен до XX века.]

Функция от двух переменных часто записывается как $x \circ y$ вместо $f(x,y)$ с применением подходящего символа оператора $\circ$. В табл. 1 показаны шестнадцать символов операторов, которые мы примем для булевых функций от двух переменных; например, $\perp$ символизирует функцию, таблицей истинности которой является 0000, $\wedge$ — символ для таблицы истинности 0001, $\overline{\supset}$ — для 0010 и т. д. Мы имеем $x \perp y = 0$, $x \wedge y = xy$, $x \overline{\supset} y = x \dot{-} y$, $x \llcorner y = x$, $\ldots$, $x \overline{\wedge} y = \bar{x} \vee \bar{y}$, $x \top y = 1$.

Конечно, не все операции в табл. 1 имеют одинаковую важность. Например, первый и последний случаи тривиальны, поскольку имеют константное значение, не зависящее от x и y. Четыре функции являются функциями только от x или только от y. Мы записываем $\bar{x}$ для $1 - x$, *дополнения* x.

Четыре операции, таблица истинности которых содержит только одну единицу, легко выражаются через оператор И ($\wedge$), а именно $x \wedge y$, $x \wedge \bar{y}$, $\bar{x} \wedge y$, $\bar{x} \wedge \bar{y}$. Функции с тремя единицами в таблице истинности выражаются через оператор ИЛИ ($\vee$), а именно $x \vee y$, $x \vee \bar{y}$, $\bar{x} \vee y$, $\bar{x} \vee \bar{y}$. Базовые функции $x \wedge y$ и $x \vee y$ оказались более полезными на практике, чем их дополняющие или полудополняющие двойники, хотя операции НЕ-ИЛИ и НЕ-И $x \overline{\vee} y = \bar{x} \wedge \bar{y}$ и $x \overline{\wedge} y = \bar{x} \vee \bar{y}$ также представляют интерес, поскольку легко реализуются транзисторными схемами.

В 1913 году Г. М. Шеффер (H. M. Sheffer) показал, что все 16 функций могут быть выражены через одну, начиная с операций $\overline{\vee}$ либо $\overline{\wedge}$ в качестве заданной (см. упр. 4). На самом деле Ч. С. Пирс (C. S. Peirce) сделал то же самое открытие около 1880 года, но его работа на эту тему осталась неопубликованной при его жизни [*Collected Papers of Charles Sanders Peirce* **4** (1933), §§12–20, 264]. В табл. 1 показано, что НЕ-И и НЕ-ИЛИ иногда записываются как $x \mid y$ и $x \downarrow y$; иногда их называют чертой Шеффера и стрелкой Пирса. В настоящее время лучше *не* использовать черту Шеффера в качестве оператора НЕ-И, поскольку $x \mid y$ обозначает побитовую операцию $x \vee y$ в языках программирования наподобие C.

К этому моменту мы рассмотрели все функции табл. 1, кроме двух. Последними двумя функциями являются $x \equiv y$ и $x \oplus y$, "эквивалентность" и "исключающее ИЛИ", связанные тождествами

$$x \equiv y \;=\; \bar{x} \oplus y \;=\; x \oplus \bar{y} \;=\; 1 \oplus x \oplus y; \tag{3}$$

$$x \oplus y \;=\; \bar{x} \equiv y \;=\; x \equiv \bar{y} \;=\; 0 \equiv x \equiv y. \tag{4}$$

Обе операции ассоциативны (см. упр. 6). В логике высказываний понятие эквивалентности более важно, чем понятие "исключающего или", означающего неэквивалентность; но при рассмотрении побитовых операций над полными компьютерными словами мы увидим в разделе 7.1.3, что ситуация обратна: в типичных программах "исключающее или" оказывается более полезным, чем экивалентность. Главной причиной важности применения $x \oplus y$, даже в однобитовом случае, является тот факт, что

$$x \oplus y \;=\; (x + y) \bmod 2. \tag{5}$$

Следовательно, $x \oplus y$ и $x \wedge y$ обозначают сложение и вычитание в поле из двух элементов (см. раздел 4.6), а $x \oplus y$ естественным образом наследует многие "чистые" математические свойства.

Основные тождества. Давайте теперь обратимся к взаимодействию между фундаментальными операторами $\wedge$, $\vee$, $\oplus$ и $\bar{\ }$, поскольку прочие операции легко вы-

ражаются через указанные четыре. Каждая из операций $\wedge$, $\vee$, $\oplus$ ассоциативна и коммутативна. Кроме законов дистрибутивности (1) и (2), мы имеем

$$(x \oplus y) \wedge z \;=\; (x \wedge z) \oplus (y \wedge z), \tag{6}$$

а также *законы поглощения*

$$(x \wedge y) \vee x \;=\; (x \vee y) \wedge x \;=\; x. \tag{7}$$

Одним из простейших, но наиболее полезных тождеств является

$$x \oplus x \;=\; 0, \tag{8}$$

поскольку из него, кроме всего прочего, вытекает, что

$$(x \oplus y) \oplus x \;=\; y, \qquad (x \oplus y) \oplus y \;=\; x, \tag{9}$$

если воспользоваться тем очевидным фактом, что $x \oplus 0 = x$. Другими словами, если дано $x \oplus y$ и либо x, либо y, то легко определить другое, неизвестное значение. Нельзя пропустить и простой *закон дополнения*

$$\bar{x} \;=\; x \oplus 1. \tag{10}$$

Другая важная пара тождеств известна как *законы де Моргана*, названные так в честь Огастеса де Моргана (Augustus De Morgan), который дословно писал "The contrary of an aggregate is the compound of the contraries of the aggregants; the contrary of a compound is the aggregate of the contraries of the components. Thus (A, B) and AB have ab and (a, b) for contraries." [*Trans. Cambridge Philos. Soc.* **10** (1858), 208.] Если воспользоваться современными обозначениями, то это не что иное, как правила, которые мы неявно вывели из таблиц истинности в табл. 1 в связи с операциями НЕ-И и НЕ-ИЛИ, а именно

$$\overline{x \wedge y} \;=\; \bar{x} \vee \bar{y}; \tag{11}$$

$$\overline{x \vee y} \;=\; \bar{x} \wedge \bar{y}. \tag{12}$$

Кстати, У. С. Джевонс (W. S. Jevons) знал (12), но не (11); он последовательно писал $\bar{A}B + \bar{B}A + \bar{A}\bar{B}$ вместо $\bar{A} + \bar{B}$ для дополнения AB. Сам де Морган не был первым англичанином, сформулировавшим приведенные законы. И (11), и (12) можно найти в работах начала XIV века двух схоластических философов — Уильяма Оккама (William of Ockham) [*Summa Logicæ* **2** (1323)] и Вальтера Бурлея (Walter Burley) [*De Puritate Artis Logicæ* (c. 1330)].

Законы де Моргана и несколько других тождеств можно использовать для выражения операторов $\wedge$, $\vee$ и $\oplus$ один через другой:

$$x \wedge y \;=\; \overline{\bar{x} \vee \bar{y}} \;=\; x \oplus y \oplus (x \vee y); \tag{13}$$

$$x \vee y \;=\; \overline{\bar{x} \wedge \bar{y}} \;=\; x \oplus y \oplus (x \wedge y); \tag{14}$$

$$x \oplus y \;=\; (x \vee y) \wedge \overline{x \wedge y} \;=\; (x \wedge \bar{y}) \vee (\bar{x} \wedge y). \tag{15}$$

Согласно упр. 7.1.2–77 все вычисления $x_1 \oplus x_2 \oplus \cdots \oplus x_n$, использующие только операции $\wedge$, $\vee$ и $\bar{\ }$, должны состоять как минимум из $4(n-1)$ шагов; таким образом, другие три операции являются не слишком хорошей заменой $\oplus$.

Функции от n переменных. Булева функция $f(x,y,z)$ от трех переменных x, y, z может быть определена своей 8-битовой таблицей истинности $f(0,0,0)f(0,0,1)\ldots f(1,1,1)$; и в общем случае каждая n-арная булева функция $f(x_1,\ldots,x_n)$ соответствует 2^n-битовой таблице истинности, которая перечисляет последовательные значения $f(0,\ldots,0,0)$, $f(0,\ldots,0,1)$, $f(0,\ldots,1,0)$, $\ldots$, $f(1,\ldots,1,1)$.

Для всех этих функций не требуются специальные имена и обозначения, поскольку все они могут быть выражены через бинарные функции, уже изученные нами. Например, как заметил И. И. Жегалкин [*Математический сборник* **35** (1928), 311–369], мы всегда можем записать

$$f(x_1,\ldots,x_n) \;=\; g(x_1,\ldots,x_{n-1}) \;\oplus\; h(x_1,\ldots,x_{n-1})\wedge x_n \tag{16}$$

при $n>0$ для соответствующих функций g и h, полагая

$$\begin{aligned} g(x_1,\ldots,x_{n-1}) &= f(x_1,\ldots,x_{n-1},0);\\ h(x_1,\ldots,x_{n-1}) &= f(x_1,\ldots,x_{n-1},0)\oplus f(x_1,\ldots,x_{n-1},1). \end{aligned} \tag{17}$$

(Операция $\wedge$ традиционно имеет более высокий приоритет, чем $\oplus$, так что нам не требуется заключать в скобки подформулу '$h(x_1,\ldots,x_{n-1})\wedge x_n$' в правой части (16).) Рекурсивно повторяя этот процесс для g и h, пока не достигнем 0-арных функций, мы достигнем выражения, которое включает только операторы $\oplus$ и $\wedge$, и последовательность из 2^n констант вместе с переменными $\{x_1,\ldots,x_n\}$. Кроме того, эти константы обычно можно использовать упрощенным способом, поскольку

$$x\wedge 0=0 \qquad \text{и} \qquad x\wedge 1 = x\oplus 0 = x. \tag{18}$$

После применения законов ассоциативности и дистрибутивности мы прийдем к необходимости константы 0 только в случае, когда $f(x_1,\ldots,x_n)$ тождественно равна нулю, и константы 1, только когда $f(0,\ldots,0)=1$.

Мы можем получить, например,

$$\begin{aligned} f(x,y,z) &= \big((1\oplus 0\wedge x)\oplus(0\oplus 1\wedge x)\wedge y\big)\oplus\big((0\oplus 1\wedge x)\oplus(1\oplus 1\wedge x)\wedge y\big)\wedge z\\ &= (1\oplus x\wedge y)\;\oplus\;(x\oplus y\oplus x\wedge y)\wedge z\\ &= 1\;\oplus\;x\wedge y\;\oplus\;x\wedge z\;\oplus\;y\wedge z\;\oplus\;x\wedge y\wedge z. \end{aligned}$$

А согласно правилу (5) мы видим, что оказываемся просто с полиномом

$$f(x,y,z) \;=\; (1+xy+xz+yz+xyz) \bmod 2, \tag{19}$$

поскольку $x\wedge y = xy$. Обратите внимание, что этот полином линейный (степени ≤ 1) по каждой из своих переменных. В общем случае аналогичные вычисления показывают, что *любая* булева функция $f(x_1,\ldots,x_n)$ имеет единственное такое представление, именуемое *мультилинейным представлением*, или *исключающей нормальной формой* (*exclusive normal form*), которая представляет собой сумму по модулю 2 нуля или большего количества из 2^n возможных членов 1, x_1, x_2, x_1x_2, x_3, x_1x_3, x_2x_3, $x_1x_2x_3$, $\ldots$, $x_1x_2\ldots x_n$.

Джордж Буль (George Boole) разлагал булевы функции иным способом, который зачастую оказывался проще для тех функций, которые встречаются на практике. Вместо (16) он, по сути, записывал

$$f(x_1,\ldots,x_n) \;=\; \big(g(x_1,\ldots,x_{n-1})\wedge\bar{x}_n\big)\;\vee\;\big(h(x_1,\ldots,x_{n-1})\wedge x_n\big) \tag{20}$$

и называл это законом разработки (law of development), где вместо (17) мы имеем просто

$$\begin{aligned} g(x_1,\ldots,x_{n-1}) &= f(x_1,\ldots,x_{n-1},0),\\ h(x_1,\ldots,x_{n-1}) &= f(x_1,\ldots,x_{n-1},1). \end{aligned} \tag{21}$$

Многократно итерируя процедуру Буля, используя закон дистрибутивности (1) и убирая константы, мы остаемся с формулой, являющейся дизъюнкцией нуля или большего количества *минитермов*, где каждый минитерм представляет собой конъюнкцию, такую как $x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \wedge x_4 \wedge x_5$, в которой присутствуют все переменные или их дополнения. Обратите внимание, что минитерм представляет собой булеву функцию, которая истинна ровно в одной точке.

Например, рассмотрим более или менее случайную функцию $f(w,x,y,z)$ с таблицей истинности

$$1100\,1001\,0000\,1111. \tag{22}$$

Разлагая эту функцию способом Буля (20), мы получим дизъюнкцию восьми минитермов, по одному для каждой единицы в таблице истинности:

$$\begin{aligned} f(w,x,y,z) \;=\; & (\bar{w}\wedge\bar{x}\wedge\bar{y}\wedge\bar{z}) \vee (\bar{w}\wedge\bar{x}\wedge\bar{y}\wedge z) \vee (\bar{w}\wedge x\wedge\bar{y}\wedge\bar{z}) \vee (\bar{w}\wedge x\wedge y\wedge z)\\ & \vee (w\wedge x\wedge\bar{y}\wedge\bar{z}) \vee (w\wedge x\wedge\bar{y}\wedge z) \vee (w\wedge x\wedge y\wedge\bar{z}) \vee (w\wedge x\wedge y\wedge z). \end{aligned} \tag{23}$$

В общем случае дизъюнкция минитермов называется *полной дизъюнктивной нормальной формой*. Каждая булева функция может быть выражена таким способом, и результат является единственным — за исключением, конечно, порядка минитермов. *Придирка*: частный случай, возникающий, когда $f(x_1,\ldots,x_n)$ тождественно равна нулю. Мы рассматриваем '0' как пустую дизъюнкцию, без членов, а также рассматриваем '1' как пустую конъюнкцию по той же причине, что и определяли $\sum_{k=1}^{0} a_k = 0$ и $\prod_{k=1}^{0} a_k = 1$ в разделе 1.2.3.

Ч. С. Пирс (C. S. Peirce) заметил в *Amer. J. Math.* **3** (1880), 37–39, что каждая булева функция имеет также *полную конъюнктивную нормальную форму*, которая представляет собой конъюнкцию "минидизъюнктов" наподобие $\bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4 \vee x_5$. Такой минидизъюнкт равен 0 только в одной точке; так что каждый дизъюнкт в такой конъюнкции учитывается только там, где таблица истинности имеет 0. Например, полная конъюнктивная нормальная форма нашей функции в (22) и (23) имеет вид

$$\begin{aligned} f(w,x,y,z) \;=\; & (w\vee x\vee\bar{y}\vee z) \wedge (w\vee x\vee\bar{y}\vee\bar{z}) \wedge (w\vee\bar{x}\vee y\vee\bar{z}) \wedge (w\vee\bar{x}\vee\bar{y}\vee z)\\ & \wedge (\bar{w}\vee x\vee y\vee z) \wedge (\bar{w}\vee x\vee y\vee\bar{z}) \wedge (\bar{w}\vee x\vee\bar{y}\vee z) \wedge (\bar{w}\vee x\vee\bar{y}\vee\bar{z}). \end{aligned} \tag{24}$$

Нет, однако, ничего удивительного в том, что мы часто хотим работать с дизъюнкциями и конъюнкциями, которые *не* обязательно включают все минитермы или минидизъюнкты. Поэтому, следуя терминологии Пауля Бернайса (Paul Bernays) из его *Habilitationsschrift* (1918), мы говорим в общем случае о *дизъюнктивной нормальной форме*, или "DNF", как *любой* дизъюнкции конъюнкций

$$\bigvee_{j=1}^{m} \bigwedge_{k=1}^{s_j} u_{jk} \;=\; (u_{11} \wedge \cdots \wedge u_{1s_1}) \vee \cdots \vee (u_{m1} \wedge \cdots \wedge u_{ms_m}), \tag{25}$$

где каждое u_{jk} является *литералом*, а именно переменной x_i или ее дополнением. Аналогично любая конъюнкция дизъюнкций литералов

$$\bigwedge_{j=1}^{m} \bigvee_{k=1}^{s_j} u_{jk} \;=\; (u_{11} \vee \cdots \vee u_{1s_1}) \wedge \cdots \wedge (u_{m1} \vee \cdots \vee u_{ms_m}) \tag{26}$$

называется *конъюнктивной нормальной формой*, или, для краткости, "CNF".

Очень много электрических схем в современных компьютерных микросхемах составлены из "программируемых логических матриц" (programmable logic arrays, PLA), которые представляют собой ИЛИ, примененные к наборам И (возможно, дополненных) входных сигналов. Другими словами, программируемая логическая матрица в основном вычисляет одну или несколько дизъюнктивных нормальных форм. Такие строительные блоки быстры, универсальны и относительно недороги; в самом деле, DNF играет заметную роль в электронике еще с 1950-х годов, когда коммутируемые схемы реализовывались с помощью сравнительно устаревших устройств наподобие реле или электронных ламп. Поэтому инженеров издавна интересовала задача поиска простейших DNF для классов булевых функций, и можно ожидать, что важность изучения дизъюнктивных нормальных форм не будет снижаться с развитием электронных технологий.

Члены DNF часто называются *импликантами* (*implicants*), поскольку истинность любого члена в дизъюнкции влечет за собой истинность всей формулы. В формуле наподобие

$$f(x,y,z) \;=\; (x \wedge \bar{y} \wedge z) \vee (y \wedge z) \vee (\bar{x} \wedge y \wedge \bar{z}),$$

например, мы знаем, что f истинно, когда истинно $x \wedge \bar{y} \wedge z$, а именно когда $(x,y,z) = (1,0,1)$. Но заметьте, что в этом примере импликантой f является и более короткий (хотя явно и не записанный) член $x \wedge z$, поскольку дополнительный член $y \wedge z$ делает функцию истинной при $x = z = 1$ независимо от значения y. Аналогично импликантой данной конкретной функции является и $\bar{x} \wedge y$. Так что мы можем работать и с более простой формулой

$$f(x,y,z) \;=\; (x \wedge z) \vee (y \wedge z) \vee (\bar{x} \wedge y). \tag{27}$$

Дальнейшие удаления из импликант невозможны, поскольку ни x, ни y, ни z, ни $\bar{x}$ не являются строго достаточными условиями для истинности f.

Импликанта, которая не может быть разложена путем удаления любого из ее литералов без чрезмерного ослабления импликанты, называется *простой импликантой*, следуя терминологии У. В. Куайна (W. V. Quine) из *AMM* **59** (1952), 521–531.

Эти базовые концепции, пожалуй, будут более понятны, если упростить обозначения и встать на более геометрическую точку зрения. Мы можем записать просто '$f(x)$' вместо $f(x_1,\ldots,x_n)$ и рассматривать x как вектор или как бинарную строку $x_1 \ldots x_n$ длиной n. Например, строками $wxyz$, для которых функция (22) принимает истинное значение, являются

$$\{0000, 0001, 0100, 0111, 1100, 1101, 1110, 1111\}, \tag{28}$$

и мы можем рассматривать их как восемь точек четырехмерного гиперкуба $2 \times 2 \times 2 \times 2$. Восемь точек в (28) соответствуют минитермам импликант, которые явно присутствуют в полной дизъюнктивной нормальной форме (23); но никакие

из этих импликант в действительности простыми не являются. Например, первые две точки (28) образуют подкуб 000∗, а последние четыре точки составляют подкуб 11∗∗, если мы используем звездочку для обозначения "групповых символов", как мы делали это при рассмотрении запросов к базам данных в разделе 6.5; следовательно, $\bar{w} \wedge \bar{x} \wedge \bar{y}$ является импликантой f, и то же самое относится и к $w \wedge x$. Аналогично можно увидеть, что подкуб 0∗00 объединяет две из восьми точек (28), делая импликантой $\bar{w} \wedge \bar{y} \wedge \bar{z}$.

В общем случае каждая простая импликанта соответствует таким способом *максимальному* подкубу в пределах множества точек, делающих значение f истинным. (Подкуб максимален в том смысле, что он не содержится в некотором подкубе большего размера с тем же свойством; мы не можем заменить ни один из его явно указанных битов звездочкой. Максимальный подкуб имеет максимальное количество звездочек, следовательно, минимальное количество ограниченных координат, следовательно, минимальное количество переменных в соответствующей импликанте.) Максимальными подкубами восьми точек в (28) являются

$$000*,\ 0*00,\ *100,\ *111,\ 11**; \tag{29}$$

так что простыми импликантами функции $f(w, x, y, z)$ из (23) являются

$$(\bar{w} \wedge \bar{x} \wedge \bar{y}) \vee (\bar{w} \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge \bar{y} \wedge \bar{z}) \vee (x \wedge y \wedge z) \vee (w \wedge x). \tag{30}$$

Дизъюнктивной простой формой булевой функции является дизъюнкция всех ее простых импликант. В упр. 30 приведен алгоритм для поиска всех простых импликант заданной функции на основе списка точек, в которых значения данной функции истинны.

Мы можем определить *простой дизъюнкт* в точности таким же способом, как дизъюнктивное выражение, следующее из f, которое не имеет подвыражений с тем же свойством. А *конъюнктивная простая форма* f представляет собой конъюнкцию всех ее простых дизъюнктов. (Соответствующий пример можно найти в упр. 19.)

Во многих простых случаях дизъюнктивная простая форма представляет собой кратчайшую возможную дизъюнктивную нормальную форму, которую может иметь функция. Но часто можно добиться лучшего результата, поскольку можно оказаться возможным охватить все необходимые точки только несколькими из максимальных подкубов. Например, простая импликанта $(y \wedge z)$ не является необходимой в (27). В выражении (30) нам не требуются одновременно $(\bar{w} \wedge \bar{y} \wedge \bar{z})$ и $(x \wedge \bar{y} \wedge \bar{z})$; при наличии остальных членов достаточно только одного члена этой пары.

К сожалению, из раздела 7.9 вы узнаете, что задача поиска кратчайшей дизъюнктивной нормальной формы NP-сложная, таким образом, очень трудно решаемая в общем случае. Но для достаточно малых задач разработано много полезных упрощений, неплохо поясненных в книге *Introduction to the Theory of Switching Circuits* Э. Д. Мак-Класки (E. J. McCluskey) (New York: McGraw–Hill, 1965). Более новые разработки можно найти в книге Petr Fiser and Jan Hlavicka, *Computing and Informatics* **22** (2003), 19–51.

Однако имеется важный частный случай, для которого кратчайшую DNF легко описать. Булева функция называется *монотонной*, или *положительной*, если ее значение не изменяется с 1 на 0, когда любая из ее переменных изменяется от 0 до 1. Другими словами, функция f монотонна тогда и только тогда, когда $f(x) \le f(y)$

при $x \subseteq y$, где битовая строка $x = x_1 \ldots x_n$ рассматривается как содержащаяся в битовой строке $y = y_1 \ldots y_n$ или равная ей тогда и только тогда, когда $x_j \le y_j$ для всех j. Эквивалентное условие (см. упр. 21) состоит в том, что функция f либо является константой, либо может быть выражена полностью через $\wedge$ и $\vee$, не используя дополнений.

Теорема Q. *Кратчайшей дизъюнктивной нормальной формой монотонной булевой функции является ее дизъюнктивная простая форма.*

Доказательство. [У. В. Куайн (W. V. Quine), *Boletin de la Sociedad Matemática Mexicana* **10** (1953), 64–70.] Пусть функция $f(x_1, \ldots, x_n)$ монотонна и пусть $u_1 \wedge \cdots \wedge u_s$ — одна из ее простых импликант. Мы не можем иметь, скажем, $u_1 = \bar{x}_i$, поскольку в таком случае более краткий член $u_2 \wedge \cdots \wedge u_s$ также был бы импликантой из-за монотонности функции. Следовательно, никакая простая импликанта не может иметь дополненный литерал.

Теперь, если мы установим $u_1 \leftarrow \cdots \leftarrow u_s \leftarrow 1$, а все прочие переменные — равными 0, то значение f будет равно 1, но все другие простые импликанты f примут нулевое значение. Таким образом, член $u_1 \wedge \cdots \wedge u_s$ должен присутствовать в каждой кратчайшей DNF, поскольку очевидно, что каждая импликанта кратчайшей DNF проста. ▮

Следствие Q. *Дизъюнктивная нормальная форма является дизъюнктивной простой формой монотонной булевой функции тогда и только тогда, когда в ней нет дополнений и ни одна из ее импликант не содержится в другой.* ▮

Выполнимость. Булева функция называется *выполнимой*, если она не является тождественно равной нулю, т. е. если она имеет как минимум одну импликанту. Наиболее знаменитой нерешенной задачей всей информатики является поиск эффективного способа решить, является ли данная булева функция выполнимой. Точнее говоря, вопрос заключается в следующем: существует ли алгоритм, который получает булеву формулу длиной N и проверяет ее выполнимость, всегда давая корректный ответ не более чем за $N^{O(1)}$ шагов?

Если вы впервые услышали об этой задаче, у вас может возникнуть соблазн задать встречный вопрос: “Как? Вы всерьез утверждаете, что ученые до сих пор не знают, как сделать такую простую вещь?”

Отлично, если вы считаете проверку выполнимости тривиальной, поделитесь с нами своим методом. Да, задача не всегда трудна; если, например, заданная формула включает всего 30 булевых переменных, простой перебор всех 2^{30} случаев — около миллиарда — действительно решит данную задачу. Но чудовищное количество практических задач, все еще ожидающих своего решения, может быть сформулировано как булевы функции со, скажем, 100 переменными, поскольку математическая логика — очень мощный способ выражения понятий. А решение этих задач соответствует векторам $x = x_1 \ldots x_{100}$, для которых $f(x) = 1$. Так что действительно эффективное решение задачи выполнимости было бы чудесным достижением.

Имеется как минимум один смысл, в котором проверка выполнимости — плевое дело: если функция $f(x_1, \ldots, x_n)$ выбрана случайным образом, так что все 2^n-битовые таблицы истинности равновероятны, то f практически гарантированно

выполнима, и мы можем найти значение x, для которого $f(x) = 1$, после выполнения менее 2 попыток (в среднем). Это похоже на подбрасывание монетки, пока не выпадет орел; обычно этого не приходится долго ждать. Но, конечно, практические задачи не имеют случайных таблиц истинности.

Хорошо, согласимся, что проверка выполнимости кажется очень сложной в общем случае. Фактически выполнимость оказывается сложной, даже если мы пытаемся упростить ее путем требования, чтобы булева функция была представлена как "3CNF-формула", т. е. как конъюнктивная нормальная форма, в каждом дизъюнкте которой имеется только *три* литерала:

$$f(x_1, \ldots, x_n) = (t_1 \vee u_1 \vee v_1) \wedge (t_2 \vee u_2 \vee v_2) \wedge \cdots \wedge (t_m \vee u_m \vee v_m). \tag{31}$$

Здесь каждое t_j, u_j и v_j представляет собой x_k или $\bar{x}_k$ для некоторого k. Задача выяснения выполнимости для 3CNF-формул называется "3SAT", и в упр. 39 поясняется, почему в действительности она не проще, чем задача выполнимости в общем случае.

Мы увидим множество примеров трудных для решения задач 3SAT, например, в разделе 7.2.2.2, где задача выполнимости рассматривается очень подробно. Ситуация, однако, несколько необычна, поскольку формула должна быть достаточно длинной, чтобы нам надо было задумываться над ее выполнимостью. Например, простейшая невыполнимая формула в 3CNF-виде — $(x \vee x \vee x) \wedge (\bar{x} \vee \bar{x} \vee \bar{x})$; но очевидно, что здесь не над чем задумываться. Никаких особых трудностей не предвидится, пока три литерала t_j, u_j, v_j дизъюнкта не станут соответствовать трем различным переменным. А в этом случае каждый дизъюнкт исключает $1/8$ возможностей, поскольку семь различных установок (t_j, u_j, v_j) делают его истинным. Следовательно, каждая такая 3CNF-формула с не более чем семью дизъюнктами автоматически выполнима, и случайная установка ее переменных будет успешной с вероятностью $\geq 1 - 7/8 = 1/8$.

Таким образом, кратчайшая интересная формула в 3CNF-формате имеет как минимум восемь дизъюнктов. Действительно, имеется такая восьмидизъюнктная формула, основанная на конструкции ассоциативного блока Р. Л. Ривеста (R. L. Rivest), рассматривавшегося в 6.5–(13):

$$\begin{aligned}(x_2 \vee x_3 \vee \bar{x}_4) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee x_2 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)\\ \wedge (\bar{x}_2 \vee \bar{x}_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee \bar{x}_4) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_4) \wedge (x_1 \vee x_2 \vee \bar{x}_3).\end{aligned} \tag{32}$$

Любые семь из этих восьми дизъюнктов выполнимы ровно двумя способами и определяют значения трех переменных; например, из первых семи дизъюнктов вытекает, что $x_1 x_2 x_3 = 001$. Но все восемь дизъюнктов одновременно выполнимыми быть не могут.

Простые частные случаи. Можно указать два важных класса булевых формул, для которых задача выполнимости оказывается достаточно простой. Эти частные случаи осуществляются, когда проверяемая конъюнктивная нормальная форма состоит только из "дизъюнктов Хорна" или только из "дизъюнктов Крома". *Дизъюнкт Хорна* представляет собой ИЛИ литералов, среди которых все или почти все литералы являются дополнениями — не более чем один из литералов дизъюнкта является "чистой", недополненной переменной. *Дизъюнкт Крома* представляет собой

ИЛИ ровно двух литералов. Таким образом, например,

$$\bar{x} \lor \bar{y}, \qquad w \lor \bar{y} \lor \bar{z}, \qquad \bar{u} \lor \bar{v} \lor \bar{w} \lor \bar{x} \lor \bar{y} \lor z \quad \text{и} \quad x$$

являются примерами дизъюнктов Хорна; а

$$x \lor x, \qquad \bar{x} \lor \bar{x}, \qquad \bar{x} \lor \bar{y}, \qquad x \lor \bar{y}, \qquad \bar{x} \lor y \quad \text{и} \quad x \lor y$$

представляют собой примеры дизъюнктов Крома, причем только последний из них не является одновременно дизъюнктом Хорна. (Первый пример является дизъюнктом Хорна, поскольку $x \lor x = x$.) Обратите внимание, что дизъюнкт Хорна может содержать сколько угодно литералов, но если мы ограничиваемся дизъюнктами Крома, то, по сути, рассматриваем 2SAT-задачу. В обоих случаях, как мы увидим, вопрос о выполнимости может быть решен за линейное время, т. е. путем выполнения только $O(N)$ простых шагов, если задана формула длиной N.

Начнем с рассмотрения дизъюнктов Хорна. Почему с ними так легко работать? Основная причина этого в том, что дизъюнкт наподобие $\bar{u} \lor \bar{v} \lor \bar{w} \lor \bar{x} \lor \bar{y} \lor z$ может быть переписан в виде $\neg(u \land v \land w \land x \land y) \lor z$, что то же самое, что и

$$u \land v \land w \land x \land y \;\Rightarrow\; z.$$

Другими словами, если все переменные u, v, w, x и y истинны, то и переменная z также должна быть истинна. По этой причине параметризированные дизъюнкты Хорна были выбраны в качестве базового механизма языка программирования Prolog. Кроме того, имеется простой способ точно указать, какие булевы функции могут быть представлены с помощью одних дизъюнктов Хорна.

Теорема H. *Булева функция $f(x_1, \ldots, x_n)$ выражается в виде конъюнкции дизъюнктов Хорна тогда и только тогда, когда*

$$\text{из} \quad f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n) = 1 \quad \text{вытекает} \quad f(x_1 \land y_1, \ldots, x_n \land y_n) = 1 \tag{33}$$

для всех булевых значений x_j и y_j.

Доказательство. [Альфред Хорн (Alfred Horn), *J. Symbolic Logic* **16** (1951), 14–21, лемма 7.] Если мы имеем $x_0 \lor \bar{x}_1 \lor \cdots \lor \bar{x}_k = 1$ и $y_0 \lor \bar{y}_1 \lor \cdots \lor \bar{y}_k = 1$, то

$$\begin{aligned}(x_0 \land y_0) &\lor \overline{x_1 \land y_1} \lor \cdots \lor \overline{x_k \land y_k} \\ &= (x_0 \lor \bar{x}_1 \lor \bar{y}_1 \lor \cdots \lor \bar{x}_k \lor \bar{y}_k) \land (y_0 \lor \bar{x}_1 \lor \bar{y}_1 \lor \cdots \lor \bar{x}_k \lor \bar{y}_k) \\ &\ge (x_0 \lor \bar{x}_1 \lor \cdots \lor \bar{x}_k) \land (y_0 \lor \bar{y}_1 \lor \cdots \lor \bar{y}_k) \;=\; 1;\end{aligned}$$

и аналогичное (но более простое) вычисление применимо, когда отсутствуют положительные (без черты) литералы x_0 и y_0. Следовательно, каждая конъюнкция дизъюнктов Хорна удовлетворяет (33).

И обратно, из условия (33) вытекает, что каждый простой дизъюнкт f представляет собой дизъюнкт Хорна (см. упр. 44). ▮

Назовем *функцией Хорна* булеву функцию, которая удовлетворяет условию (33), и будем также называть ее *определенной* (*definite*), если она удовлетворяет дополнительному условию $f(1, \ldots, 1) = 1$. Легко видеть, что конъюнкция дизъюнктов Хорна является определенной тогда и только тогда, когда каждый дизъюнкт имеет *ровно* один положительный литерал, поскольку только полностью отрицательный

дизъюнкт наподобие $\bar{x} \vee \bar{y}$ будет ложным, когда все переменные истинны. С определенными функциями Хорна немного проще работать, чем с функциями Хорна в общем случае, поскольку очевидно, что они всегда выполнимы. Таким образом, согласно теореме Н они имеют единственный наименьший вектор x, такой, что $f(x) = 1$, а именно побитовое И всех векторов, которые удовлетворяют всем дизъюнктам. *Ядром* определенной функции Хорна является множество всех переменных x_j, являющихся истинными в этом минимальном векторе x. Заметим, что переменные в ядре должны быть истинными, когда истинна f, так что мы, по сути, можем вынести их за скобки.

Определенные функции Хорна возникают в разных ситуациях, например, в анализе игр (см. упр. 51 и 52). Еще один красивый пример — из области компиляторов. Рассмотрим следующую типичную (хотя и упрощенную) грамматику для алгебраических выражений в языке программирования:

$$\begin{aligned}
&\langle\,\text{expression}\,\rangle \to \langle\,\text{term}\,\rangle \mid \langle\,\text{expression}\,\rangle\,\texttt{+}\,\langle\,\text{term}\,\rangle \mid \langle\,\text{expression}\,\rangle\,\texttt{-}\,\langle\,\text{term}\,\rangle\\
&\langle\,\text{term}\,\rangle \to \langle\,\text{factor}\,\rangle \mid \texttt{-}\,\langle\,\text{factor}\,\rangle \mid \langle\,\text{term}\,\rangle\,\texttt{*}\,\langle\,\text{factor}\,\rangle \mid \langle\,\text{term}\,\rangle\,\texttt{/}\,\langle\,\text{factor}\,\rangle\\
&\langle\,\text{factor}\,\rangle \to \langle\,\text{variable}\,\rangle \mid \langle\,\text{constant}\,\rangle \mid \texttt{(}\langle\,\text{expression}\,\rangle\texttt{)}\\
&\langle\,\text{variable}\,\rangle \to \langle\,\text{letter}\,\rangle \mid \langle\,\text{variable}\,\rangle\langle\,\text{letter}\,\rangle \mid \langle\,\text{variable}\,\rangle\langle\,\text{digit}\,\rangle\\
&\langle\,\text{letter}\,\rangle \to \texttt{a} \mid \texttt{b} \mid \texttt{c}\\
&\langle\,\text{constant}\,\rangle \to \langle\,\text{digit}\,\rangle \mid \langle\,\text{constant}\,\rangle\langle\,\text{digit}\,\rangle\\
&\langle\,\text{digit}\,\rangle \to \texttt{0} \mid \texttt{1}
\end{aligned} \tag{34}$$

Например, строка `a/(-b0-10)+cc*cc` отвечает синтаксису для $\langle\,\text{expression}\,\rangle$ и использует каждое из правил грамматики как минимум по одному разу.

Предположим, что мы хотим знать, какие пары символов могут появляться в таких выражениях одна за другой. Определенные дизъюнкты Хорна предоставляют ответ на поставленный вопрос, поскольку мы можем сформулировать задачу следующим образом: пусть величины `Xx`, `xX` и `xy` обозначают булевы "суждения", где `X` — один из символов $\{\texttt{E}, \texttt{T}, \texttt{F}, \texttt{V}, \texttt{L}, \texttt{C}, \texttt{D}\}$, обозначающих соответственно $\langle\,\text{expression}\,\rangle$, $\langle\,\text{term}\,\rangle$, ..., $\langle\,\text{digit}\,\rangle$, и где `x` и `y` — символы из множества $\{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{/}, \texttt{(}, \texttt{)}, \texttt{a}, \texttt{b}, \texttt{c}, \texttt{0}, \texttt{1}\}$. Суждение `Xx` означает "`X` может заканчиваться на `x`"; аналогично `xX` означает "`X` может начинаться с `x`"; а `xy` означает "в выражении непосредственно за символом `x` может следовать символ `y`". (Всего имеется $7 \times 11 + 11 \times 7 + 11 \times 11 = 275$ суждений.) Тогда можно записать

$$\begin{array}{rrrrr}
\texttt{xT} \Rightarrow \texttt{xE} & \Rightarrow \texttt{-T} & \texttt{xC} \Rightarrow \texttt{xF} & \texttt{Vx} \wedge \texttt{yL} \Rightarrow \texttt{xy} & \Rightarrow \texttt{Lc}\\
\texttt{Tx} \Rightarrow \texttt{Ex} & \texttt{xF} \Rightarrow \texttt{-x} & \texttt{Cx} \Rightarrow \texttt{Fx} & \texttt{Vx} \wedge \texttt{yD} \Rightarrow \texttt{xy} & \texttt{xD} \Rightarrow \texttt{xC}\\
\texttt{Ex} \Rightarrow \texttt{x+} & \texttt{Tx} \Rightarrow \texttt{x*} & \Rightarrow \texttt{(F} & \texttt{Dx} \Rightarrow \texttt{Vx} & \texttt{Dx} \Rightarrow \texttt{Cx}\\
\texttt{xT} \Rightarrow \texttt{+x} & \texttt{xF} \Rightarrow \texttt{*x} & \texttt{xE} \Rightarrow \texttt{(x} & \Rightarrow \texttt{aL} & \texttt{Cx} \wedge \texttt{yD} \Rightarrow \texttt{xy}\\
\texttt{Ex} \Rightarrow \texttt{x-} & \texttt{Tx} \Rightarrow \texttt{x/} & \texttt{Ex} \Rightarrow \texttt{x)} & \Rightarrow \texttt{La} & \Rightarrow \texttt{0D}\\
\texttt{xT} \Rightarrow \texttt{-x} & \texttt{xF} \Rightarrow \texttt{/x} & \Rightarrow \texttt{F)} & \Rightarrow \texttt{bL} & \Rightarrow \texttt{D0}\\
\texttt{xF} \Rightarrow \texttt{xT} & \texttt{xV} \Rightarrow \texttt{xF} & \texttt{xL} \Rightarrow \texttt{xV} & \Rightarrow \texttt{Lb} & \Rightarrow \texttt{1D}\\
\texttt{Fx} \Rightarrow \texttt{Tx} & \texttt{Vx} \Rightarrow \texttt{Fx} & \texttt{Lx} \Rightarrow \texttt{Vx} & \Rightarrow \texttt{cL} & \Rightarrow \texttt{D1}
\end{array} \tag{35}$$

где `x` и `y` пробегают одиннадцать терминальных символов $\{\texttt{+}, \ldots, \texttt{1}\}$. Эта схематическая спецификация дает нам определенные дизъюнкты Хорна общим числом $24 \times 11 + 3 \times 11 \times 11 + 13 \times 1 = 640$, которые мы можем формально записать как

$$\big(\overline{\texttt{+T}} \vee \texttt{+E}\big) \wedge \big(\overline{\texttt{-T}} \vee \texttt{-E}\big) \wedge \cdots \wedge \big(\overline{\texttt{V+}} \vee \overline{\texttt{0L}} \vee \texttt{+0}\big) \wedge \cdots \wedge \big(\texttt{D1}\big)$$

если предпочитаем загадочное обозначение булевой алгебры для условного обозначения $\Rightarrow$ из (35).

Почему мы это делаем? Потому что *ядро всех этих дизъюнктов представляет собой множество всех суждений, истинных в данной конкретной грамматике.* Например, можно убедиться, что `-E` истинно, значит, символы `(-` могут следовать в выражении один за другим, а пары символов `++` и `*-` в выражении присутствовать не могут (см. упр. 46).

Кроме того, мы можем без особых сложностей найти ядро любого данного множества определенных дизъюнктов Хорна. Мы просто начинаем с суждений, появляющихся справа от $\Rightarrow$, для которых левая часть пуста; в (35) имеется тринадцать дизъюнктов такого рода. После утверждения об истинности этих суждений мы можем найти один или несколько дизъюнктов, истинность левых частей которых нам известна. Следовательно, их правые части также принадлежат ядру, и мы можем продолжить нашу работу тем же способом. Вся процедура напоминает течение воды вниз, пока она не найдет свое место. На практике при выборе подходящих структур данных этот процесс спуска оказывается достаточно быстрым, требующим только $O(N+n)$ шагов, где N означает общую длину дизъюнктов, а n — количество переменных в суждениях. (Здесь мы считаем, что все дизъюнкты развернуты и не используют сокращенной записи членов с параметрами наподобие `x` и `y`, как показано выше. Более интеллектуальные методы доказательства теорем могут работать с параметризованными дизъюнктами, но они выходят за рамки нашего рассмотрения.)

Алгоритм C (*Вычисление ядра определенных дизъюнктов Хорна*). Для данного множества P переменных суждений и множества дизъюнктов C, каждый из которых имеет вид

$$u_1 \wedge \cdots \wedge u_k \;\Rightarrow\; v, \qquad \text{где } k \geq 0 \text{ и } \{u_1, \ldots, u_k, v\} \subseteq P, \tag{36}$$

данный алгоритм находит множество $Q \subseteq P$ всех переменных, которые обязаны быть истинными, чтобы истинными были все дизъюнкты.

Мы используем для дизъюнктов c, суждений p и гипотез h следующие структуры данных (здесь "гипотеза" представляет собой суждение в левой части дизъюнкта):

`CONCLUSION`(c) является суждением в правой части дизъюнкта c;

`COUNT`(c) является количеством еще не утвержденных гипотез c;

`TRUTH`(p) равно 1, если известно, что p истинно, в противном случае 0;

`LAST`(p) является последней гипотезой, в которой появляется p;

`CLAUSE`(h) является дизъюнктом, в котором h появляется в его левой части;

`PREV`(h) является предыдущей гипотезой, содержащей суждение h.

Мы также поддерживаем стек S_0, S_1, …, S_{s-1} всех суждений, о которых известно, что они истинны, но которые пока не утверждены.

C1. [Инициализация.] Установить `LAST`(p) $\leftarrow \Lambda$ и `TRUTH`(p) $\leftarrow 0$ для каждого суждения p. Также установить $s \leftarrow 0$, так что стек пуст. Затем для каждого дизъюнкта c, имеющего вид (36), установить `CONCLUSION`(c) $\leftarrow v$ и `COUNT`(c) $\leftarrow k$.

Если $k = 0$ и $\texttt{TRUTH}(v) = 0$, установить $\texttt{TRUTH}(v) \leftarrow 1$, $S_s \leftarrow v$ и $s \leftarrow s + 1$. В противном случае для $1 \le j \le k$ создать запись гипотезы h и установить $\texttt{CLAUSE}(h) \leftarrow c$, $\texttt{PREV}(h) \leftarrow \texttt{LAST}(u_j)$, $\texttt{LAST}(u_j) \leftarrow h$.

C2. [Подготовка к утверждению p.] Завершить работу алгоритма, если $s = 0$; искомое ядро сейчас состоит из всех суждений, поле TRUTH которых установлено равным 1. В противном случае установить $s \leftarrow s - 1$, $p \leftarrow S_s$ и $h \leftarrow \texttt{LAST}(p)$.

C3. [Гипотеза обработана?] Если $h = \Lambda$, вернуться к шагу C2.

C4. [Проверка h.] Установить $c \leftarrow \texttt{CLAUSE}(h)$ и $\texttt{COUNT}(c) \leftarrow \texttt{COUNT}(c) - 1$. Если новое значение $\texttt{COUNT}(c)$ все еще ненулевое, перейти к шагу C6.

C5. [Вывод $\texttt{CONCLUSION}(c)$.] Установить $p \leftarrow \texttt{CONCLUSION}(c)$. Если $\texttt{TRUTH}(p) = 0$, установить $\texttt{TRUTH}(p) \leftarrow 1$, $S_s \leftarrow p$, $s \leftarrow s + 1$.

C6. [Цикл по h.] Установить $h \leftarrow \texttt{PREV}(h)$ и вернуться к шагу C3. ▮

Обратите внимание, как согласованно работают структуры данных, позволяя избежать любой необходимости в поиске места, где следует выполнять очередные вычисления. Алгоритм C во многих отношениях подобен алгоритму 2.2.3T (топологической сортировки), который был первым примером многосвязных структур данных, рассмотренным давным-давно, в главе 2; на самом деле алгоритм 2.2.3T можно рассматривать как частный случай алгоритма C, в котором каждое суждение появляется в правой части ровно одного дизъюнкта (см. упр. 47.)

В упр. 48 показано, что небольшая модификация алгоритма C решает задачу выполнимости для дизъюнктов Хорна в общем случае. Дополнительную информацию можно найти в статьях W. F. Dowling and J. H. Gallier, *J. Logic Programming* **1** (1984), 267–284; M. G. Scutellà, *J. Logic Programming* **8** (1990), 265–273.

Теперь мы обратимся к функциям Крома и задаче 2SAT. И вновь для решения задачи имеется алгоритм с линейным временем работы; но, вероятно, будет лучше, если сначала мы рассмотрим упрощенное практическое приложение. Предположим, что семь комиков согласились во время трехдневного фестиваля дать концерты в двух из пяти отелей. При этом у каждого из них один из дней занят другой работой, поэтому вот как выглядят возможные варианты их выступлений в отелях Лас-Вегаса:*

Tomlin может выступить в отелях Aladdin и Caesars в дни 1 и 2;
Unwin может выступить в отелях Bellagio и Excalibur в дни 1 и 2;
Vegas может выступить в отелях Desert и Excalibur в дни 2 и 3;
Williams может выступить в отелях Aladdin и Desert в дни 1 и 3; (37)
Xie может выступить в отелях Caesars и Excalibur в дни 1 и 3;
Yankovic может выступить в отелях Bellagio и Desert в дни 2 и 3;
Zany может выступить в отелях Bellagio и Caesars в дни 1 и 2.

Можно ли составить расписание так, чтобы не возникало никаких конфликтов?

Для решения этой задачи можно ввести семь булевых переменных $\{t, u, v, w, x, y, z\}$, где t, например, означает выступление Tomlin в Aladdin в первый день

* Имена артистов и названия отелей оставлены в оригинальном написании, так как в дальнейшем первые буквы имен артистов и названий отелей соответствуют вводимым переменным и названиям ограничений. — *Примеч. пер.*

и в Caesars во второй, в то время как $\bar t$ означает, что дни соответствуют отелям в обратном порядке: выступление в Aladdin — во второй день, а в Caesars — в первый. Тогда мы можем записать ограничения, означающие, что никакие два комедианта не выступают в одном отеле в один и тот же день, следующим образом.

$$\begin{array}{llllllll}
\neg(t\wedge w) & \text{[A1]} & \neg(y\wedge\bar z) & \text{[B2]} & \neg(t\wedge z) & \text{[C2]} & \neg(w\wedge y) & \text{[D3]}\\
\neg(u\wedge z) & \text{[B1]} & \neg(\bar t\wedge x) & \text{[C1]} & \neg(v\wedge\bar y) & \text{[D2]} & \neg(\bar u\wedge\bar x) & \text{[E1]}\\
\neg(\bar u\wedge y) & \text{[B2]} & \neg(\bar t\wedge\bar z) & \text{[C1]} & \neg(\bar v\wedge w) & \text{[D3]} & \neg(u\wedge\bar v) & \text{[E2]}\\
\neg(\bar u\wedge\bar z) & \text{[B2]} & \neg(x\wedge\bar z) & \text{[C1]} & \neg(\bar v\wedge y) & \text{[D3]} & \neg(v\wedge x) & \text{[E3]}
\end{array} \tag{38}$$

Каждое из этих ограничений, конечно же, представляет собой дизъюнкт Крома; мы должны выполнить

$$\begin{aligned}&(\bar t\vee\bar w)\wedge(\bar u\vee\bar z)\wedge(u\vee\bar y)\wedge(u\vee z)\wedge(\bar y\vee z)\wedge(t\vee\bar x)\wedge(t\vee z)\wedge(\bar x\vee z)\\&\qquad\wedge(\bar t\vee\bar z)\wedge(\bar v\vee y)\wedge(v\vee\bar w)\wedge(v\vee\bar y)\wedge(\bar w\vee\bar y)\wedge(u\vee x)\wedge(\bar u\vee v)\wedge(\bar v\vee\bar x).\end{aligned} \tag{39}$$

Кроме того, дизъюнкты Крома (подобно дизъюнктам Хорна) могут быть записаны в виде импликаций:

$$\begin{aligned}&t\Rightarrow\bar w,\quad u\Rightarrow\bar z,\quad\bar u\Rightarrow\bar y,\quad\bar u\Rightarrow z,\quad y\Rightarrow z,\quad\bar t\Rightarrow\bar x,\quad\bar t\Rightarrow z,\quad x\Rightarrow z,\\&\qquad t\Rightarrow\bar z,\quad v\Rightarrow y,\quad\bar v\Rightarrow\bar w,\quad\bar v\Rightarrow\bar y,\quad w\Rightarrow\bar y,\quad\bar u\Rightarrow x,\quad u\Rightarrow v,\quad v\Rightarrow\bar x.\end{aligned} \tag{40}$$

И каждая такая импликация может также быть представлена в альтернативном, “контрапозитивном” виде:

$$\begin{aligned}&w\Rightarrow\bar t,\quad z\Rightarrow\bar u,\quad y\Rightarrow u,\quad\bar z\Rightarrow u,\quad\bar z\Rightarrow\bar y,\quad x\Rightarrow t,\quad\bar z\Rightarrow t,\quad\bar z\Rightarrow\bar x,\\&\qquad z\Rightarrow\bar t,\quad\bar y\Rightarrow\bar v,\quad w\Rightarrow v,\quad y\Rightarrow v,\quad y\Rightarrow\bar w,\quad\bar x\Rightarrow u,\quad\bar v\Rightarrow\bar u,\quad x\Rightarrow\bar v.\end{aligned} \tag{41}$$

Но — увы и ах! — наблюдается порочный замкнутый круг, вернее, цикл

$$u\ \underset{\text{[B1]}}{\Rightarrow}\ \bar z\ \underset{\text{[B2]}}{\Rightarrow}\ \bar y\ \underset{\text{[D2]}}{\Rightarrow}\ \bar v\ \underset{\text{[E2]}}{\Rightarrow}\ \bar u\ \underset{\text{[B2]}}{\Rightarrow}\ z\ \underset{\text{[C2]}}{\Rightarrow}\ \bar t\ \underset{\text{[C1]}}{\Rightarrow}\ \bar x\ \underset{\text{[E1]}}{\Rightarrow}\ u. \tag{42}$$

Этот цикл говорит о том, что u и $\bar u$ должны иметь одно и то же значение; так что нет никакой возможности удовлетворить все условия (37). Если расписание должно быть составлено любой ценой, организаторам фестиваля придется провести переговоры и пересмотреть соглашение по крайней мере с одним из шести комедиантов $\{t,u,v,x,y,z\}$ (см. упр. 53).

Рис. 6. Ориентированный граф, соответствующий всем импликациям (40) и (41), не включающим v или $\bar v$. Присваивание подходящих значений литералам в каждом сильно связном компоненте решает бинарную задачу составления расписания, которая представляет собой экземпляр 2SAT.

Организаторы могут, например, попытаться временно вывести за рамки картины v. Тогда пять из шестнадцати ограничений (38) исчезнут, и останутся только 22 из импликаций (40) и (41), так что получится ориентированный граф, показанный на рис. 6. Этот ориентированный граф содержит циклы наподобие $z \Rightarrow \bar{u} \Rightarrow x \Rightarrow z$ и $t \Rightarrow \bar{z} \Rightarrow t$; но в нем нет циклов, которые содержат одновременно переменную и ее дополнение. Действительно, на рис. 6 можно увидеть, что значения $tuwxyz = 110000$ выполняют каждый дизъюнкт (39), не включающий ни v, ни $\bar{v}$. Эти значения дают нам расписание, которое выполняет шесть из семи исходных условий (37), начиная с выступления (Tomlin, Unwin, Zany, Williams, Xie) в первый день в (Aladdin, Bellagio, Caesars, Desert, Excalibur).

В общем случае для любой заданной 2SAT-задачи с m дизъюнктами Крома, включающими n булевых переменных, мы можем построить ориентированный граф описанным способом. Имеется $2n$ вершин $\{x_1, \bar{x}_1, \ldots, x_n, \bar{x}_n\}$, по одной для каждого возможного литерала; кроме того, имеется $2m$ дуг вида $\bar{u} \to v$ и $\bar{v} \to u$, по две для каждого дизъюнкта $u \vee v$. Два литерала u и v принадлежат одному и тому же *сильно связному компоненту* этого ориентированного графа тогда и только тогда, когда существуют ориентированные пути из u в v и из v в u. Например, шесть сильно связных компонентов ориентированного графа на рис. 6 указаны контурами из точек. Все литералы в сильно связном компоненте должны иметь одно и то же булево значение в любом решении соответствующей 2SAT-задачи.

Теорема K. *Конъюнктивная нормальная форма с двумя литералами на дизъюнкт выполнима тогда и только тогда, когда ни один сильно связный компонент соответствующего ориентированного графа не содержит одновременно и переменную, и ее дополнение.*

Доказательство. [Melven Krom, *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **13** (1967), 15–20, Corollary 2.2.] Если существуют пути от x до $\bar{x}$ и от $\bar{x}$ до x, формула, определенно, невыполнима.

И обратно, предположим, что таких путей не существует. Любой ориентированный граф имеет как минимум один сильно связный компонент S, являющийся "источником", который не имеет входящих дуг из любого другого сильно связного компонента. Более того, наш ориентированный граф всегда имеет привлекательную антисимметрию, проиллюстрированную на рис. 6: мы имеем $u \to v$ тогда и только тогда, когда $\bar{v} \to \bar{u}$. Следовательно, дополнения литералов в S образуют другой сильно связный компонент $\overline{S} \neq S$, который является "стоком", не имеющим *исходящих* дуг к другим сильно связным компонентам. Следовательно, мы можем назначить значение 0 всем литералам в S и 1 всем литералам в $\overline{S}$, затем удалить их из ориентированного графа и продолжать работать таким образом далее, пока все литералы не получат свои значения. Результирующие значения удовлетворяют неравенству $u \leq v$ при $u \to v$ в ориентированном графе; следовательно, они удовлетворяют $\bar{u} \vee v$, если $\bar{u} \vee v$ представляет собой дизъюнкт в формуле. ▮

Теорема K приводит непосредственно к эффективному решению 2SAT-задачи благодаря алгоритму Р. Э. Таржана (R. E. Tarjan), который находит сильно связные компоненты за линейное время. [См. *SICOMP* **1** (1972), 146–160; D. E. Knuth, *The Stanford GraphBase* (1994), 512–519.] Мы детально изучим алгоритм Таржана в разделе 7.4.1. В упр. 54 показано, что условие теоремы K легко проверяется, когда

алгоритм обнаруживает новый сильно связный компонент. Кроме того, алгоритм первыми обнаруживает "стоки"; таким образом, как простой побочный продукт процедуры Таржана мы можем назначить значения, устанавливающие выполнимость путем выбора значения 1 для каждого литерала в сильно связном компоненте, который встречается до его дополнения.

Медианы. Мы сосредоточились на булевых бинарных операциях наподобие $x \vee y$ или $x \oplus y$. Однако имеется еще и важная *тернарная* операция $\langle xyz \rangle$, именуемая *медианой* x, y и z:

$$\langle xyz \rangle \;=\; (x \wedge y) \vee (y \wedge z) \vee (x \wedge z) = (x \vee y) \wedge (y \vee z) \wedge (x \vee z). \tag{43}$$

Фактически $\langle xyz \rangle$, пожалуй, — наиболее важная тернарная операция во всем мире, поскольку обладает удивительными свойствами, которые постоянно открываются и переоткрываются.

Во-первых, можно легко увидеть, что эта формула для $\langle xyz \rangle$ описывает *мажоритарное* значение любых трех булевых значений x, y и z: $\langle 000 \rangle = \langle 001 \rangle = 0$ и $\langle 011 \rangle = \langle 111 \rangle = 1$. Мы называем $\langle xyz \rangle$ медианой, а не мажоритарным значением, поскольку, если x, y и z — произвольные *действительные* числа, а операции $\wedge$ и $\vee$ в (43) означают min и max, то

$$\langle xyz \rangle \;=\; y, \qquad \text{когда } x \le y \le z. \tag{44}$$

Во-вторых, базовые бинарные операции $\wedge$ и $\vee$ представляют собой частные случаи медиан:

$$x \wedge y \;=\; \langle x0y \rangle; \qquad\qquad x \vee y \;=\; \langle x1y \rangle. \tag{45}$$

Таким образом, *любая* монотонная булева функция может быть выражена только через тернарный оператор медианы и константы 0 и 1. Фактически, если бы мы жили в мире, где нет ничего, кроме медиан, мы могли бы обозначить через $\wedge$ ложь, а $\vee$ использовать для обозначения истины; тогда $x \wedge y = \langle x \wedge y \rangle$ и $x \vee y = \langle x \vee y \rangle$ были бы совершенно естественными выражениями и мы могли бы при желании даже использовать польскую запись наподобие $\langle \wedge xy \rangle$ и $\langle \vee xy \rangle$! Такая же идея применима и к расширенным действительным числам при условии интерпретации $\wedge$ и $\vee$ как минимума и максимума, если брать медианы по отношению к константам $\wedge = -\infty$ и $\vee = +\infty$.

Булева функция $f(x_1, x_2, \ldots, x_n)$ называется *самодуальной*, если она удовлетворяет условию

$$\overline{f(x_1, x_2, \ldots, x_n)} \;=\; f(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n). \tag{46}$$

Мы отмечали, что булева функция является монотонной тогда и только тогда, когда она может быть выражена через $\wedge$ и $\vee$; в соответствии с законами де Моргана (11) и (12) монотонная формула самодуальна тогда и только тогда, когда символы $\wedge$ и $\vee$ могут быть взаимозаменены без изменения значения формулы. Таким образом, операция медианы, определенная в (43), является как монотонной, так и самодуальной. Фактически это простейшая нетривиальная функция такого рода, поскольку ни одна из бинарных операций в табл. 1 не является монотонной и самодуальной, за исключением проекций $\llcorner$ и $\lrcorner$.

Кроме того, *любое* выражение, которое образовано только с помощью оператора медианы, без применения констант, является и монотонным, и самодуальным.

Например, функция $\langle w\langle xyz\rangle\langle w\langle uvw\rangle x\rangle\rangle$ самодуальна, поскольку

$$\begin{aligned}\overline{\langle w\langle xyz\rangle\langle w\langle uvw\rangle x\rangle\rangle} &= \langle \bar w\,\overline{\langle xyz\rangle}\,\overline{\langle w\langle uvw\rangle x\rangle}\rangle\\ &= \langle \bar w\langle \bar x\bar y\bar z\rangle\langle \bar w\,\overline{\langle uvw\rangle}\,\bar x\rangle\rangle \;=\; \langle \bar w\langle \bar x\bar y\bar z\rangle\langle \bar w\langle \bar u\bar v\bar w\rangle\bar x\rangle\rangle.\end{aligned}$$

Эмиль Пост (Emil Post), работая над своей диссертацией (Columbia University, 1920), доказал, что верно и обратное утверждение.

Теорема P. *Каждая монотонная самодуальная булева функция $f(x_1,\ldots,x_n)$ может быть выражена только через операцию медианы $\langle xyz\rangle$.*

Доказательство. [*Annals of Mathematics Studies* **5** (1941), 74–75.] Сначала заметим, что

$$\begin{aligned}&\big\langle x_1 y\langle x_2 y\ldots y\langle x_{s-1}yx_s\rangle\ldots\rangle\big\rangle\\ &\qquad= \big((x_1\vee x_2\vee\cdots\vee x_{s-1}\vee x_s)\wedge y\big)\vee(x_1\wedge x_2\wedge\cdots\wedge x_{s-1}\wedge x_s);\end{aligned}\tag{47}$$

эта формула для многократного применения оператора медианы легко доказывается по индукции по s.

Теперь предположим, что $f(x_1,\ldots,x_n)$ монотонна, самодуальна и имеет дизъюнктивную простую форму

$$f(x_1,\ldots,x_n) \;=\; t_1\vee\cdots\vee t_m,\qquad t_j = x_{j1}\wedge\cdots\wedge x_{js_j},$$

где никакие простые импликанты t_j не содержатся одна в другой (следствие Q). Любые две простые импликанты должны иметь как минимум одну общую переменную. Если мы имеем, скажем, $t_1 = x\wedge y$ и $t_2 = u\wedge v\wedge w$, то значение f будет равно 1, когда $x = y = 1$ и $u = v = w = 0$, как и при $x = y = 0$ и $u = v = w = 1$, что противоречит самодуальности. Следовательно, если любая t_j состоит из одной переменной x, она должна быть единственной простой импликантой, а в этом случае f является тривиальной функцией $f(x_1,\ldots,x_n) = x = \langle xxx\rangle$.

Определим функции g_0, g_1, $\ldots$, g_m путем смешивания медиан следующим образом:

$$\begin{aligned}&g_0(x_1,\ldots,x_n) = x_1;\\ &g_j(x_1,\ldots,x_n) = h(x_{j1},\ldots,x_{js_j}; g_{j-1}(x_1,\ldots,x_n)) \text{ для } 1\le j\le m;\end{aligned}\tag{48}$$

здесь $h(x_1,\ldots,x_s;y)$ обозначает функцию в верхней строке (47). По индукции по j из (47) и (48) можно доказать, что $g_j(x_1,\ldots,x_n) = 1$, если $t_1\vee\cdots\vee t_j = 1$, поскольку $(x_{j1}\vee\cdots\vee x_{js_j})\wedge t_k = t_k$, когда $k < j$.

Наконец, $f(x_1,\ldots,x_n)$ должно быть равно $g_m(x_1,\ldots,x_n)$, поскольку обе функции монотонны и самодуальны, и мы показали, что $f(x_1,\ldots,x_n)\le g_m(x_1,\ldots,x_n)$ для всех комбинаций нулей и единиц. Этого неравенства достаточно для доказательства равенства, поскольку самодуальная функция равна 1 ровно в половине из 2^n возможных случаев. ▮

Одним из следствий из теоремы P является то, что мы можем выразить медиану пяти элементов через медианы трех, поскольку медиана любого нечетного числа булевых переменных очевидным образом является монотонной и самодуальной

булевой функцией. Будем записывать такую медиану как $\langle x_1 \ldots x_{2k-1}\rangle$. Тогда дизъюнктивная простая форма $\langle vwxyz\rangle$ представляет собой

$$(v\wedge w\wedge x)\vee(v\wedge w\wedge y)\vee(v\wedge w\wedge z)\vee(v\wedge x\wedge y)\vee(v\wedge x\wedge z)$$
$$\vee(v\wedge y\wedge z)\vee(w\wedge x\wedge y)\vee(w\wedge x\wedge z)\vee(w\wedge y\wedge z)\vee(x\wedge y\wedge z);$$

так что построение в доказательстве теоремы P выражает $\langle vwxyz\rangle$ как огромного размера формулу $g_{10}(v,w,x,y,z)$, включающую 2046 операций получения медианы из трех элементов с максимальной глубиной вложенности 20. Конечно, это выражение не является кратчайшим; в действительности

$$\langle vwxyz\rangle \;=\; \big\langle v\langle xyz\rangle\langle wx\langle wyz\rangle\rangle\big\rangle. \tag{49}$$

[См. H. S. Miiller and R. O. Winder, *IRE Transactions* **EC-11** (1962), 89–90.]

***Алгебры медиан и медианные графы.** Ранее мы отметили, что тернарная операция $\langle xyz\rangle$ полезна, если x, y и z принадлежат любому упорядоченному множеству наподобие действительных чисел, когда $\wedge$ и $\vee$ рассматриваются как операторы min и max. В действительности операция $\langle xyz\rangle$ играет также полезную роль в гораздо более общих обстоятельствах. *Алгебра медиан* представляет собой произвольное множество M, на котором определена тернарная операция $\langle xyz\rangle$, отображающая элементы M в элементы M и подчиняющаяся трем следующим аксиомам:

$$\langle xxy\rangle = x \quad \text{(закон мажоритарности)}; \tag{50}$$
$$\langle xyz\rangle = \langle xzy\rangle = \langle yxz\rangle = \langle yzx\rangle = \langle zxy\rangle = \langle zyx\rangle \quad \text{(закон коммутативности)}; \tag{51}$$
$$\big\langle xw\langle ywz\rangle\big\rangle = \big\langle\langle xwy\rangle wz\big\rangle \quad \text{(закон ассоциативности)}. \tag{52}$$

В булевом случае, например, закон ассоциативности (52) выполняется при $w=0$ и $w=1$, поскольку $\wedge$ и $\vee$ ассоциативны. В упр. 75 и 76 доказывается, что из этих трех аксиом вытекает также *закон дистрибутивности* для медиан, который имеет как краткую форму

$$\big\langle\langle xyz\rangle uv\big\rangle \;=\; \big\langle x\langle yuv\rangle\langle zuv\rangle\big\rangle, \tag{53}$$

так и более симметричную длинную запись

$$\big\langle\langle xyz\rangle uv\big\rangle \;=\; \big\langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\big\rangle. \tag{54}$$

Простое доказательство этого факта неизвестно, но мы можем как минимум проверить частный случай (53) и (54), когда $y=u$ и $z=v$. Мы имеем

$$\big\langle\langle xyz\rangle yz\big\rangle \;=\; \langle xyz\rangle, \tag{55}$$

поскольку обе части равны $\langle xy\langle zyz\rangle\rangle$. Фактически закон ассоциативности (52) является всего лишь частным случаем $y=u$ закона (53). А с (55) и (52) мы можем также проверить случай $x=u$: $\big\langle\langle uyz\rangle uv\big\rangle = \big\langle vu\langle yuz\rangle\big\rangle = \big\langle\langle vuy\rangle uz\big\rangle = \big\langle\langle yuv\rangle uz\big\rangle = \big\langle\langle\langle yuv\rangle uv\rangle uz\big\rangle = \big\langle\langle yuv\rangle u\langle vuz\rangle\big\rangle = \big\langle u\langle yuv\rangle\langle zuv\rangle\big\rangle$.

Идеалом в алгебре медиан M является множество $C\subseteq M$, для которого мы имеем

$$\langle xyz\rangle \;\in\; C, \qquad \text{когда } x\in C,\ y\in C \text{ и } z\in M. \tag{56}$$

Если u и v являются любыми элементами M, *отрезок* $[u\,..\,v]$ определяется следующим образом:

$$[u\,..\,v] \;=\; \big\{\langle xuv\rangle \;\big|\; x \in M\big\}. \tag{57}$$

Мы говорим, что "x находится между u и v" тогда и только тогда, когда $x \in [u\,..\,v]$. Согласно этим определениям u и v сами принадлежат отрезку $[u\,..\,v]$.

Лемма M. *Каждый отрезок $[u\,..\,v]$ является идеалом, и $x \in [u\,..\,v] \iff x = \langle uxv\rangle$.*

Доказательство. Пусть $\langle xuv\rangle$ и $\langle yuv\rangle$ представляют собой произвольные элементы $[u\,..\,v]$. Тогда

$$\big\langle\langle xuv\rangle\langle yuv\rangle z\big\rangle \;=\; \big\langle\langle xyz\rangle uv\big\rangle \;\in\; [u\,..\,v]$$

для всех $z \in M$ в соответствии с (51) и (53), так что $[u\,..\,v]$ является идеалом. Кроме того, каждый элемент $\langle xuv\rangle \in [u\,..\,v]$ удовлетворяет условию $\langle xuv\rangle = \big\langle u\langle xuv\rangle v\big\rangle$ согласно (51) и (55). ▮

Наши отрезки $[u\,..\,v]$ обладают красивыми свойствами в силу законов медиан:

$$v \in [u\,..\,u] \;\Longrightarrow\; u = v; \tag{58}$$

$$x \in [u\,..\,v] \text{ и } y \in [u\,..\,x] \;\Longrightarrow\; y \in [u\,..\,v]; \tag{59}$$

$$x \in [u\,..\,v] \text{ и } y \in [u\,..\,z], \text{ и } y \in [v\,..\,z] \;\Longrightarrow\; y \in [x\,..\,z]. \tag{60}$$

Равносильно, $[u\,..\,u] = \{u\}$; если $x \in [u\,..\,v]$, то $[u\,..\,x] \subseteq [u\,..\,v]$; и из $x \in [u\,..\,v]$ вытекает также, что $[u\,..\,z] \cap [v\,..\,z] \subseteq [x\,..\,z]$ для всех z. (См. упр. 72.)

Давайте определим граф на множестве вершин M со следующими ребрами:

$$u \text{ — } v \quad\iff\quad u \neq v \text{ и } \langle xuv\rangle \in \{u, v\} \text{ для всех } x \in M. \tag{61}$$

Другими словами, u и v смежны тогда и только тогда, когда отрезок $[u\,..\,v]$ состоит только из двух точек — u и v.

Теорема G. *Если M является любой конечной алгеброй медиан, то граф, определяемый* (61), *является связным. Кроме того, вершина x принадлежит отрезку $[u\,..\,v]$ тогда и только тогда, когда x лежит на кратчайшем пути от u до v.*

Доказательство. Если M не является связным, выберем u и v так, что не существует пути из u в v и отрезок $[u\,..\,v]$ имеет минимально возможное количество элементов. Пусть $x \in [u\,..\,v]$ отлично от u и v. Тогда $\langle xuv\rangle = x \neq v$, так что $v \notin [u\,..\,x]$; аналогично $u \notin [x\,..\,v]$. Но $[u\,..\,x]$ и $[x\,..\,v]$ содержатся в $[u\,..\,v]$ согласно (59). Так что они представляют собой наименьшие отрезки и должен иметься путь из u в x и из x в v. Таким образом, получено противоречие.

Другая половина теоремы доказывается в упр. 73. ▮

Из нашего определения отрезков вытекает, что $\langle xyz\rangle \in [x\,..\,y] \cap [x\,..\,z] \cap [y\,..\,z]$, поскольку $\langle xyz\rangle = \big\langle\langle xyz\rangle xy\big\rangle = \big\langle\langle xyz\rangle xz\big\rangle = \big\langle\langle xyz\rangle yz\big\rangle$ в соответствии с (55). И обратно, если $w \in [x\,..\,y] \cap [x\,..\,z] \cap [y\,..\,z]$, то в упр. 74 доказывается, что $w = \langle xyz\rangle$. Другими словами, *пересечение $[x\,..\,y] \cap [x\,..\,z] \cap [y\,..\,z]$ всегда содержит ровно одну точку*, если x, y и z являются точками M.

На рис. 7 этот принцип проиллюстрирован на кубе размером $4\times4\times4$, где каждая точка x имеет координаты (x_1, x_2, x_3) с $0 \le x_1, x_2, x_3 < 4$. Вершины этого куба образуют алгебру медиан, поскольку $\langle xyz\rangle = \big(\langle x_1y_1z_1\rangle, \langle x_2y_2z_2\rangle, \langle x_3y_3z_3\rangle\big)$; кроме

того, ребра графа на рис. 7 являются ребрами, определенными в (61) и проходящими между вершинами, координаты которых совпадают, за исключением одной координаты, изменяющейся на ± 1. Показаны три типичных отрезка: $[x\,..\,y]$, $[x\,..\,z]$ и $[y\,..\,z]$; единственной общей точкой для всех трех отрезков является вершина $\langle xyz\rangle = (2,2,1)$.

(а) Отрезок $[x\,..\,y]$. (б) Отрезок $[x\,..\,z]$. (в) Отрезок $[y\,..\,z]$.

Рис. 7. Отрезки между вершинами $x = (0,2,1)$, $y = (3,3,3)$ и $z = (2,0,0)$ в кубе размером $4\times 4\times 4$.

До сих пор мы начинали с алгебры медиан и использовали ее для определения графа с некоторыми свойствами. Но можно начать и с графа, который обладает этими свойствами, и воспользоваться им для определения алгебры медиан. Если u и v являются вершинами *любого* графа, определим отрезок $[u\,..\,v]$ как множество всех точек на кратчайших путях между u и v. Назовем конечный граф *медианным графом*, если пересечение $[x\,..\,y]\cap[x\,..\,z]\cap[y\,..\,z]$ трех отрезков, связывающих три данные вершины, x, y и z, представляет собой ровно одну точку; мы будем обозначать эту вершину как $\langle xyz\rangle$. В упр. 75 доказывается, что получающаяся в результате тернарная операция удовлетворяет аксиомам медиан.

Многие важные графы в соответствии с этим определением оказываются медианными графами. Например, любое свободное дерево, как легко увидеть, представляет собой медианный граф; еще одним простым примером является гиперпрямоугольник $n_1\times n_2\times\cdots\times n_m$. Декартово произведение произвольных медианных графов также удовлетворяет указанному условию.

***Медианные метки.** Если u и v представляют собой произвольные элементы алгебры медиан, отображение $f(x)$, которое отображает $x\mapsto\langle xuv\rangle$, является *гомоморфизмом*; т. е. оно удовлетворяет условию

$$f\bigl(\langle xyz\rangle\bigr) \;=\; \bigl\langle f(x)f(y)f(z)\bigr\rangle \tag{62}$$

в силу длинного закона дистрибутивности (54). Эта функция $\langle xuv\rangle$ “проецирует” любую данную точку x в отрезок $[u\,..\,v]$ согласно (57). Это оказывается особенно интересным в случае, когда u — v представляет собой ребро соответствующего графа, поскольку $f(x)$ при этом оказывается функцией с двумя значениями, по сути — булевым отображением.

Например, рассмотрим типичное свободное дерево, показанное ниже, с восемью вершинами и семью ребрами. Мы можем проецировать каждую вершину x в каждый из отрезков ребер $[u\,..\,v]$, решая, находится ли x ближе к u или к v.

$$
\begin{array}{cccccccccc}
 & & ac & bc & cd & de & ef & eg & dh & \\
a & \mapsto & a & c & c & d & e & e & d & 0000000 \\
b & \mapsto & c & b & c & d & e & e & d & 1100000 \\
c & \mapsto & c & c & c & d & e & e & d & 1000000 \\
d & \mapsto & c & c & d & d & e & e & d & 1010000 \\
e & \mapsto & c & c & d & e & e & e & d & 1011000 \\
f & \mapsto & c & c & d & e & f & e & d & 1011100 \\
g & \mapsto & c & c & d & e & e & g & d & 1011010 \\
h & \mapsto & c & c & d & d & e & e & h & 1010001
\end{array}
\tag{63}
$$

Справа мы привели проекции к нулям и единицам, произвольно решив, что $a \mapsto 0000000$. Полученные в результате битовые строки называются *метками* (*labels*) вершин, и мы пишем, например, $l(b) = 1100000$. Поскольку каждая проекция является гомоморфизмом, мы можем вычислить медиану любых трех точек, просто находя булеву медиану для каждого компонента их меток. Например, чтобы вычислить $\langle bgh\rangle$, мы находим побитовую медиану $l(b) = 1100000$, $l(g) = 1011010$ и $l(h) = 1010001$, а именно $1010000 = l(d)$.

При проецировании на все ребра медианного графа можно обнаружить, что два столбца бинарных меток идентичны. Эта ситуация невозможна в случае свободного дерева; но рассмотрим, что произойдет, если добавить к дереву (63) ребро g — h. Получающийся в результате граф остается медианным графом, но столбцы для eg и dh будут идентичны (за исключением $e \leftrightarrow d$ и $g \leftrightarrow h$). Кроме того, новый столбец для gh окажется эквивалентным столбцу для de. В таких случаях следует убрать из меток избыточные компоненты; следовательно, вершины пополненного графа будут иметь не семибитовые, а шестибитовые метки наподобие $l(g) = 101101$ и $l(h) = 101001$.

Элементы любой алгебры медиан всегда могут быть представлены метками таким способом. Следовательно, *любое тождество, выполняющееся в булевом случае, будет истинно во всех алгебрах медиан.* Этот "нуль-один-принцип" делает возможной проверку для двух заданных выражений, построенных из тернарной операции $\langle xyz\rangle$, можно ли показать их эквивалентность как следствие аксиом (50), (51) и (52), хотя мы должны проверить $2^{n-1} - 1$ случаев при тестировании этим методом выражений с n переменными.

Например, закон ассоциативности $\big\langle xw\langle ywz\rangle\big\rangle = \big\langle\langle xwy\rangle wz\big\rangle$ предполагает, что должна иметься симметричная интерпретация обеих частей, не включающая вложенные угловые скобки. И действительно, такая формула существует:

$$\big\langle xw\langle ywz\rangle\big\rangle \;=\; \big\langle\langle xwy\rangle wz\big\rangle \;=\; \langle xwywz\rangle, \tag{64}$$

где $\langle xwywz\rangle$ обозначает медиану пятиэлементного мультимножества $\{x, w, y, w, z\} = \{w, w, x, y, z\}$. Эту формулу можно доказать, воспользовавшись нуль-один-принципом, замечая также, что в булевом случае медиана — это то же самое, что и мажоритарное значение. Аналогично можно доказать (49), и мы можем показать, что

использованная Постом (Post) в (47) функция может быть упрощена до

$$\big\langle x_1 y \langle x_2 y \ldots y \langle x_{s-1} y x_s \rangle \ldots \rangle\big\rangle \;=\; \langle x_1 y x_2 y \ldots y x_{s-1} y x_s \rangle; \tag{65}$$

это медиана $2s-1$ значений, где около половины из них равны y.

Множество C вершин графа называется *выпуклым*, если $[u\,..\,v] \subseteq C$, когда $u \in C$ и $v \in C$. Другими словами, если конечные точки кратчайшего пути принадлежат C, то все вершины пути также должны присутствовать в C. (Следовательно, выпуклое множество идентично тому, что мы назвали "идеал" несколько страниц назад; сейчас наш язык становится геометрическим вместо алгебраического.) *Выпуклая оболочка* для $\{v_1, \ldots, v_m\}$ определяется как наименьшее выпуклое множество, содержащее каждую из вершин $v_1, \ldots, v_m$. Наши теоретические результаты, полученные выше, показывают, что каждый отрезок $[u\,..\,v]$ выпуклый; следовательно, $[u\,..\,v]$ является выпуклой оболочкой двухточечного множества $\{u, v\}$. Но на самом деле истинно гораздо большее.

Теорема C. *Выпуклая оболочка множества $\{v_1, v_2, \ldots, v_m\}$ в медианном графе представляет собой множество всех точек*

$$C \;=\; \big\{\langle v_1 x v_2 x \ldots x v_m \rangle \;\big|\; x \in M\big\}. \tag{66}$$

Кроме того, x находится в C тогда и только тогда, когда $x = \langle v_1 x v_2 x \ldots x v_m \rangle$.

Доказательство. Ясно, что $v_j \in C$ для $1 \le j \le m$. Каждая точка C должна принадлежать выпуклой оболочке, поскольку точка $x' = \langle v_2 x \ldots x v_m \rangle$ находится в оболочке (по индукции по m) и потому что $\langle v_1 x \ldots x v_m \rangle \in [v_1\,..\,x']$. Нуль-один-принцип доказывает, что

$$\big\langle x \langle v_1 y v_2 y \ldots y v_m \rangle \langle v_1 z v_2 z \ldots z v_m \rangle\big\rangle = \big\langle v_1 \langle xyz \rangle v_2 \langle xyz \rangle \ldots \langle xyz \rangle v_m \big\rangle; \tag{67}$$

следовательно, C выпукло. Установка $y = x$ в этой формуле доказывает, что $\langle v_1 x v_2 x \ldots x v_m \rangle$ является точкой C, ближайшей к x, и что $\langle v_1 x v_2 x \ldots x v_m \rangle \in [x\,..\,z]$ для всех $z \in C$. ▌

Следствие C. *Пусть метка v_j представляет собой $v_{j1} \ldots v_{jt}$ для $1 \le j \le m$. Тогда выпуклой оболочкой для $\{v_1, \ldots, v_m\}$ является множество всех $x \in M$, метки которых $x_1 \ldots x_t$ удовлетворяют условию $x_j = c_j$ при $v_{1j} = v_{2j} = \cdots = v_{mj} = c_j$.* ▌

Например, выпуклая оболочка $\{c, g, h\}$ в (63) состоит из всех элементов, метки которых соответствуют шаблону $10{*}{*}0{*}{*}$, а именно $\{c, d, e, g, h\}$.

Когда медианный граф содержит цикл длиной 4 u — x — v — y — u, ребра u — x и v — y эквивалентны в том смысле, что проекция в $[u\,..\,x]$ и проекция в $[v\,..\,y]$ дают одни и те же координаты меток. Причина этого в том, что для любого z с $\langle zux \rangle = u$ мы имеем

$$y = \langle uvy \rangle = \big\langle \langle zux \rangle vy \big\rangle = \big\langle \langle zvy \rangle \langle uvy \rangle \langle xvy \rangle \big\rangle = \big\langle \langle zvy \rangle yv \big\rangle,$$

следовательно, $\langle zvy \rangle = y$; аналогично из $\langle zux \rangle = x$ вытекает $\langle zvy \rangle = v$. Ребра x — v и y — u эквивалентны по той же причине. В упр. 77 среди прочего показано, что два ребра приводят к эквивалентным проекциям тогда и только тогда, когда можно доказать их эквивалентность с помощью цепочек эквивалентности, получае-

мых описанным способом из 4-циклов. Следовательно, количество битов в каждой метке вершины равно количеству классов эквивалентности ребер, порожденных 4-циклами; отсюда следует, что сокращенные метки для вершин определяются единственным образом, если мы указываем вершину с меткой $00\ldots0$.

Интересный способ поиска меток вершин любого медианного графа был предложен П. К. Джхой (P. K. Jha) и Ж. Слуцки (G. Slutzki) [*Ars Combin.* **34** (1992), 75–92] и усовершенствован И. Хагауэром (J. Hagauer), В. Имрихом (W. Imrich) и С. Клавжаром (S. Klavzar) [*Theor. Comp. Sci.* **215** (1999), 123–136].

Алгоритм H (*Медианные метки*). Для данного медианного графа G и исходной вершины a этот алгоритм находит классы эквивалентности, определяемые циклами G длиной 4, и вычисляет метки $l(v) = v_1\ldots v_t$ каждой вершины, где t — количество классов, а $l(a) = 0\ldots0$.

H1. [Инициализация.] Предварительно обработать G, посещая все вершины в порядке их расстояний от a. Для каждого ребра u — v мы говорим, что u является *ранним соседом* v, если a ближе к u, чем к v, в противном случае u является *поздним соседом*; другими словами, ранние соседи v уже посещены к моменту, когда встречается v, а поздние соседи все еще дожидаются своей очереди. Перестроить все списки смежности так, чтобы ранние соседи перечислялись первыми. Поместить сначала каждое ребро в собственный класс эквивалентности; для слияния классов, когда будет выяснена их эквивалентность, будет применяться алгоритм обработки отношений эквивалентности наподобие алгоритма 2.3.3E.

H2. [Вызов подпрограммы.] Установить $j \leftarrow 0$ и вызвать подпрограмму I с параметром a. (Подпрограмма I приведена ниже. Глобальная переменная j будет использоваться для создания главного списка ребер r_j — s_j для $1 \le j < n$, где n — общее количество вершин; для каждой вершины $v \ne a$ будет иметься одна запись с $s_j = v$.)

H3. [Назначение меток.] Пронумеровать классы эквивалентности от 1 до t. Затем установить значение $l(a)$ равным t-битовой строке $0\ldots0$. Для $j = 1, 2, \ldots, n-1$ (в указанном порядке) установить значение $l(s_j)$ равным $l(r_j)$ с битом k, измененным с 0 на 1, где k — класс эквивалентности ребра r_j — s_j. ▮

Подпрограмма I (*Обработка потомков r*). Эта рекурсивная подпрограмма с параметром r и глобальной переменной j выполняет основную работу алгоритма H на графе всех вершин, достижимых в настоящий момент из вершины r. В процессе выполнения все такие вершины будут записаны в главный список, за исключением само́й r, а все ребра между ними будут удалены из текущего графа. Каждая вершина имеет четыре поля, `LINK`, `MARK`, `RANK` и `MATE`, изначально нулевые.

I1. [Цикл по s.] Выбрать вершину s с ребром r — s. Если такой вершины не существует, вернуться из подпрограммы.

I2. [Запись ребра.] Установить $j \leftarrow j+1$, $r_j \leftarrow r$ и $s_j \leftarrow s$.

I3. [Начать поиск в ширину.] (Сейчас мы хотим найти и удалить все ребра текущего графа, эквивалентные r — s.) Установить `MARK`$(s) \leftarrow s$, `RANK`$(s) \leftarrow 1$, `LINK`$(s) \leftarrow \Lambda$ и $v \leftarrow q \leftarrow s$.

I4. [Поиск пары v.] Найти раннего соседа u вершины v, для которого $\texttt{MARK}(u) \neq s$ или $\texttt{RANK}(u) \neq 1$. (Будет иметься ровно одна такая вершина u. Вспомните, что на шаге H1 первыми размещались ранние соседи.) Установить $\texttt{MATE}(v) \leftarrow u$.

I5. [Удаление u — v.] Сделать ребра u — v и r — s эквивалентными путем слияния их классов эквивалентности. Удалить u и v из списков смежности друг друга.

I6. [Классификация соседей v.] Для каждого раннего соседа u вершины v выполнить шаг I7; для каждого позднего соседа u вершины v выполнить шаг I8. Затем перейти к шагу I9.

I7. [Запись возможной эквивалентности.] Если $\texttt{MARK}(u) = s$ и $\texttt{RANK}(u) = 1$, сделать ребро u — v эквивалентным ребру $\texttt{MATE}(u)$ — $\texttt{MATE}(v)$. Вернуться к шагу I6.

I8. [Ранг u.] Если $\texttt{MARK}(u) = s$ и $\texttt{RANK}(u) = 1$, вернуться к шагу I6. В противном случае установить $\texttt{MARK}(u) \leftarrow s$ и $\texttt{RANK}(u) \leftarrow 2$. Установить w равным первому соседу u (это будет ранний сосед). Если $w = v$, сбросить w, сделав его равным второму раннему соседу u; но вернуться к шагу I6, если u имеет только одного раннего соседа. Если $\texttt{MARK}(w) \neq s$ или $\texttt{RANK}(w) \neq 2$, установить $\texttt{RANK}(u) \leftarrow 1$, $\texttt{LINK}(u) \leftarrow \Lambda$, $\texttt{LINK}(q) \leftarrow u$ и $q \leftarrow u$. Вернуться к шагу I6.

I9. [Продолжить поиск в ширину.] Установить $v \leftarrow \texttt{LINK}(v)$. Вернуться к шагу I4, если $v \neq \Lambda$.

I10. [Обработка подграфа s.] Рекурсивно вызвать подпрограмму I с параметром s. Затем вернуться к шагу I1. ▮

Этот алгоритм и подпрограмма описаны в терминах относительно высокоуровневых структур данных; дальнейшая детализация оставлена читателю. Например, списки смежности должны быть дважды связанными, так что ребра на шаге I5 могут быть легко удалены. На этом же шаге может использоваться любой подходящий метод слияния классов эквивалентности.

Теория, делающая данный алгоритм работоспособным, поясняется в упр. 77, а в упр. 78 доказывается, что каждая вершина на шаге I4 встречается не более $\lg n$ раз. Кроме того, в упр. 79 показано, что медианный граф имеет не более $O(n \log n)$ ребер. Следовательно, общее время работы алгоритма H равно $O(n(\log n)^2)$, за исключением, возможно, установки битов на шаге H3.

Читатель может выполнить алгоритм H вручную на медианном графе из табл. 2, вершины которого представляют двенадцать монотонных самодуальных булевых функций от четырех переменных $\{w, x, y, z\}$. Все такие функции, действительно включающие все четыре переменные, могут быть выражены как медианы пяти элементов наподобие (64). Принимая в качестве стартовой вершину $a = w$, алгоритм вычисляет главный список ребер r_j — s_j и бинарные метки, показанные в таблице. (Фактический порядок обработки зависит от порядка, в котором вершины появляются в списках смежности. Однако окончательные метки будут одинаковыми при любом порядке с точностью до перестановки столбцов.)

Заметим, что количество единичных битов в каждой метке $l(v)$ представляет собой расстояние v от стартовой вершины a. В действительности уникальность меток говорит нам о том, что *расстояние между любыми двумя вершинами равно*

Таблица 2

Метки для свободной алгебры медиан на четырех генераторах

j	r_j	s_j	$l(s_j)$
		w	0000000
1	w	$\langle wwxyz\rangle$	0000001
2	$\langle wwxyz\rangle$	$\langle wyz\rangle$	0010001
3	$\langle wyz\rangle$	$\langle wxyzz\rangle$	0010101
4	$\langle wxyzz\rangle$	$\langle xyz\rangle$	0010111
5	$\langle wxyzz\rangle$	z	1010101
6	$\langle wyz\rangle$	$\langle wxyyz\rangle$	0010011
7	$\langle wxyyz\rangle$	y	0110011
8	$\langle wwxyz\rangle$	$\langle wxz\rangle$	0000101
9	$\langle wxz\rangle$	$\langle wxxyz\rangle$	0000111
10	$\langle wxxyz\rangle$	x	0001111
11	$\langle wwxyz\rangle$	$\langle wxy\rangle$	0000011

количеству битовых позиций, в которых отличаются их метки, поскольку мы можем выбрать в качестве стартовой любую отдельную вершину.

Специальный медианный граф в табл. 2 на самом деле можно обработать совершенно иным способом, не прибегая к алгоритму H вообще, поскольку метки в этом случае, по сути, те же, что и *таблицы истинности* соответствующих функций. И вот почему: мы можем сказать, что простые функции w, x, y, z имеют таблицы истинности соответственно $t(w) = 0000000011111111$, $t(x) = 0000111100001111$, $t(y) = 0011001100110011$, $t(z) = 0101010101010101$. Тогда таблица истинности $\langle wwxyz\rangle$ представляет собой побитовую мажоритарную функцию $\bigl\langle t(w)t(w)t(x)t(y)t(z)\bigr\rangle$, а именно строку 0000000101111111; аналогичные вычисления дают таблицы истинности всех остальных вершин.

Вторая половина таблицы истинности любой самодуальной функции такая же, как и первая, но дополненная и обращенная, так что мы можем ее отбросить. Кроме того, крайний слева бит в каждой из наших таблиц истинности всегда равен нулю. Мы всегда остаемся с семибитовыми метками, показанными в табл. 2; а свойство единственности гарантирует, что алгоритм H даст для данного конкретного графа тот же результат с точностью до возможной перестановки столбцов.

Эти рассуждения говорят нам, что ребра графа из табл. 2 соответствуют парам функций, таблицы истинности которых почти одинаковы. Мы перемещаемся между соседними вершинами, меняя только два комплементарных бита их таблиц истинности. Фактически степень каждой вершины оказывается в точности равной количеству простых импликант в дизъюнктивной простой форме монотонной самодуальной функции, представленной этой вершиной (см. упр. 70 и 84).

***Медианные множества.** *Медианное множество* представляет собой коллекцию бинарных векторов X, обладающую тем свойством, что $\langle xyz\rangle \in X$, если $x \in X$, $y \in X$ и $z \in X$, где медианы вычисляются покомпонентно, как в случае меток медиан. Томас Шефер (Thomas Schaefer) заметил в 1978 году, что медианное множество предоставляет привлекательный контрапункт для характеризации функций Хорна в теореме H.

Теорема S. *Булева функция $f(x_1,\ldots,x_n)$ выражаема в виде конъюнкций дизъюнктов Крома тогда и только тогда, когда*

$$\text{из}\quad f(x_1,\ldots,x_n)=f(y_1,\ldots,y_n)=f(z_1,\ldots,z_n)=1$$
$$\text{вытекает}\quad f(\langle x_1y_1z_1\rangle,\ldots,\langle x_ny_nz_n\rangle)=1 \tag{68}$$

для всех булевых значений x_j, y_j и z_j.

Доказательство. [*STOC* **10** (1978), 216–226, Lemma 3.1B.] Если $x_1\vee x_2=y_1\vee y_2=z_1\vee z_2=1$ при, скажем, $x_1\le y_1\le z_1$, то $\langle x_1y_1z_1\rangle\vee\langle x_2y_2z_2\rangle=y_1\vee\langle x_2y_2z_2\rangle=1$, поскольку из $y_1=0$ вытекает, что $x_2=y_2=1$. Таким образом, условие (68) является необходимым.

И наоборот, если выполняется условие (68), пусть $u_1\vee\cdots\vee u_k$ — простой дизъюнкт f, где каждое u_j представляет собой литерал. Тогда для $1\le j\le k$ дизъюнкт $u_1\vee\cdots\vee u_{j-1}\vee u_{j+1}\vee\cdots\vee u_k$ не является дизъюнктом f; так что существует вектор $x^{(j)}$, такой, что $f(x^{(j)})=1$, но $u_i^{(j)}=0$ для всех $i\ne j$. Если $k\ge 3$, медиана $\langle x^{(1)}x^{(2)}x^{(3)}\rangle$ имеет $u_i=0$ для $1\le i\le k$; но это невозможно, поскольку $u_1\vee\cdots\vee u_k$, по нашему предположению, является дизъюнктом. Следовательно, $k\le 2$. ∎

Таким образом, медианные множества представляют собой то же, что и "экземпляры 2SAT", множества точек, удовлетворяющие некоторой формуле f в 2CNF-форме.

Медианное множество называется *приведенным* (*reduced*), если его векторы $x=x_1\ldots x_t$ не содержат избыточных компонентов. Другими словами, для каждой позиции координаты k приведенное медианное множество имеет как минимум два вектора $x^{(k)}$ и $y^{(k)}$, обладающих тем свойством, что $x_k^{(k)}=0$ и $y_k^{(k)}=1$, но $x_i^{(k)}=y_i^{(k)}$ для всех $i\ne k$. Мы видели, что метки медианного графа удовлетворяют этому условию; фактически, если координата k соответствует ребру u — v графа, можно положить $x^{(k)}$ и $y^{(k)}$ метками u и v. И наоборот, любое приведенное медианное множество X определяет медианный граф с одной вершиной для каждого элемента X и со смежностью, определяемой равенствами всех координат, кроме одной. Медианные метки этих вершин должны быть идентичны исходным векторам из X, поскольку мы знаем, что медианные метки, по сути, уникальны.

Медианные метки и приведенные медианные множества можно также охарактеризовать еще одним поучительным способом, который возвращает нас к сетям *модулей компараторов*, которые мы изучали в разделе 5.3.4. В упомянутом разделе мы видели, что такие сети полезны для "рассеянной сортировки" ("oblivious sorting") чисел, а из теоремы 5.3.4Z узнали, что сеть компараторов будет сортировать все $n!$ возможных входных перестановок тогда и только тогда, когда она корректно сортирует все 2^n комбинаций нулей и единиц. Когда модуль компаратора подключается к двум горизонтальным линиям, со входами x и y слева, он дает на выходе те же два значения, но в верхней линии будет значение $\min(x,y)=x\wedge y$, а в нижней — $\max(x,y)=x\vee y$. Давайте немного расширим концепцию, добавив *модули инверторов*, которые заменяют 0 на 1 и наоборот. Вот, например, сравнивающе-инвертирующая сеть (comparator-inverter network, или, для краткости, CI-сеть),

которая преобразует бинарное значение 0010 в 0111.

(69)

(Одиночная точка обозначает инвертор.) Действительно, эта сеть выполняет преобразования

$$\begin{array}{llll}
0000 \mapsto 0110; & 0100 \mapsto 0111; & 1000 \mapsto 0111; & 1100 \mapsto 0110; \\
0001 \mapsto 0111; & 0101 \mapsto 1111; & 1001 \mapsto 0101; & 1101 \mapsto 0111; \\
0010 \mapsto 0111; & 0110 \mapsto 1111; & 1010 \mapsto 0101; & 1110 \mapsto 0111; \\
0011 \mapsto 0110; & 0111 \mapsto 0111; & 1011 \mapsto 0111; & 1111 \mapsto 0110.
\end{array} \tag{70}$$

Предположим, что CI-сеть преобразует битовую строку $x = x_1 \ldots x_t$ в битовую строку $x'_1 \ldots x'_t = f(x)$. Эта функция f, которая отображает t-мерный куб на себя, фактически является *гомоморфизмом графа*. Другими словами, мы имеем $f(x) — f(y)$, когда в t-мерном кубе $x — y$: изменение одного бита x всегда приводит к изменению ровно одного бита $f(x)$, поскольку каждый модуль сети обладает этим поведением. Кроме того, CI-сети имеют замечательную связь с медианными метками.

Теорема F. *Каждое множество X из t-битовых медианных меток множеств может быть представлено CI-сетью, которая вычисляет булеву функцию $f(x)$, обладающую тем свойством, что $f(x) \in X$ для всех битовых векторов $x_1 \ldots x_t$ и $f(x) = x$ для всех $x \in X$.*

Доказательство. [Томас Федер (Tomás Feder), *Memoirs Amer. Math. Soc.* **555** (1995), 1–223, Lemma 3.37; см. также диссертацию Д. Г. Видеманна (D. H. Wiedemann) (University of Waterloo, 1986).] Рассмотрим столбцы i и j медианных меток, где $1 \le i < j \le t$. Любая такая пара столбцов содержит как минимум три из четырех вариантов $\{00, 01, 10, 11\}$, если просматривать все множество меток, поскольку медианные метки не имеют избыточных столбцов. Запишем $\bar{\jmath} \to i$, $j \to i$, $i \to j$ или $i \to \bar{\jmath}$, если значение 00, 01, 10 или 11 соответственно отсутствует среди этих двух столбцов; можно также заметить эквивалентные соотношения $\bar{\imath} \to j$, $\bar{\imath} \to \bar{\jmath}$, $\bar{\jmath} \to \bar{\imath}$ и $j \to \bar{\imath}$ соответственно, которые включают $\bar{\imath}$ вместо i. Например, метки в табл. 2 дают соотношения

$$\begin{array}{ll}
1 \to \overline{2}, 3, \overline{4}, 5, \overline{6}, 7 & 2, \overline{3}, 4, \overline{5}, 6, \overline{7} \to \overline{1}; \\
2 \to 3, \overline{4}, \overline{5}, 6, 7 & \overline{3}, 4, 5, \overline{6}, \overline{7} \to \overline{2}; \\
3 \to \overline{4}, 7 & 4, \overline{7} \to \overline{3}; \\
4 \to 5, 6, 7 & \overline{5}, \overline{6}, \overline{7} \to \overline{4}; \\
5 \to 7 & \overline{7} \to \overline{5}; \\
6 \to 7 & \overline{7} \to \overline{6}.
\end{array} \tag{71}$$

(Между 3 и 5 соотношения нет, поскольку в этих столбцах имеются все четыре возможных варианта. Но мы имеем $3 \to \overline{4}$, так как 11 в столбцах 3 и 4 нет. Вершины, метки которых имеют 1 в столбце 3, — это те, которые в табл. 2 ближе к $\langle wyz \rangle$, чем

к $\langle wwxyz\rangle$; они образуют выпуклое множество, в котором столбец 4 меток всегда равен 0, поскольку они также ближе к $\langle wxxyz\rangle$,чем к x.)

Отношения между литералами $\{1,\overline{1},2,\overline{2},\ldots,t,\overline{t}\}$ не содержат циклов, так что они всегда могут быть топологически отсортированы в антисимметричную последовательность $u_1\,u_2\,\ldots\,u_{2t}$, в которой u_j является дополнением u_{2t+1-j}. Например,

$$1\ \overline{7}\ 4\ 2\ \overline{3}\ \overline{5}\ \overline{6}\ 6\ 5\ 3\ \overline{2}\ \overline{4}\ 7\ \overline{1} \tag{72}$$

является одним таким способом топологической сортировки отношений в (71).

Теперь перейдем к построению сети, начиная с t пустых строк и последовательно рассматривая элементы u_k и u_{k+d} в топологической последовательности для $d = 2t-2,\ 2t-3,\ \ldots,\ 1$ (в указанном порядке) и для $k = 1,\ 2,\ \ldots,\ t-\lceil d/2\rceil$. Если $u_k \to u_{k+d}$ представляет собой отношение между столбцами i и j, где $i<j$, мы добавляем новые модули к линиям i и j сети следующим образом.

Если $i\to j$ Если $i\to\bar{j}$ Если $\bar{i}\to j$ Если $\bar{i}\to\bar{j}$ (73)

Например, из (71) и (72) мы сначала получаем $1\to 7$, затем $1\to\overline{4}$, после $1\to\overline{2}$, $\overline{7}\to\overline{4}$ (т. е. $4\to 7$), и так далее, в результате чего получается следующая сеть.

(74)

(Обратите внимание на то, что нет модулей, полученных, скажем, когда u_k представляет собой $\overline{7}$, а u_{k+d} равно 3, поскольку отношение $\overline{3}\to 7$ в (71) отсутствует.)

В упр. 89 доказывается, что каждый новый кластер модулей (73) сохраняет все предыдущие отношения и добавляет новые. Следовательно, если x — любой входной вектор, $f(x)$ удовлетворяет всем отношениям; так что $f(x)\in X$ согласно теореме S. И обратно, если $x\in X$, то каждый кластер модулей в сети оставляет x неизмененным. ▮

Следствие F. *Предположим, что метки медиан в теореме F замкнуты относительно операций побитового И и ИЛИ, так что $x\,\&\,y\in X$ и $x\mid y\in X$, когда $x\in X$ и $y\in X$. Тогда существует перестановка координат, при которой метки представимы сетью, состоящей только из модулей компараторов.*

Доказательство. Побитовое И всех меток равно $0\ldots0$, а побитовое ИЛИ равно $1\ldots1$. Так что единственными возможными отношениями между столбцами являются $i\to j$ и $j\to i$. Путем топологической сортировки и переименования столбцов можно обеспечить, что, когда $i<j$, встречается только $i\to j$; а в этом случае построение из доказательства никогда не использует инверторы. ▮

В общем случае, когда G представляет собой произвольный граф, гомоморфизм f, который отображает вершины G в подмножество этих вершин X, называется *ретракцией*, если он удовлетворяет условию $f(x)=x$ для всех $x\in X$; и когда такой f существует, мы называем X *ретрактом* G. Важность этой концепции в теории графов впервые отмечена Паволом Хеллом (Pavol Hell) [см. *Lecture Notes*

in Math. **406** (1974), 291–301]. Одним из следствий, например, является то, что расстояние между вершинами в X — количество ребер в кратчайшем пути — остается той же четности, если ограничиться рассмотрением путей, полностью лежащих в X. (См. упр. 93.)

Теорема F демонстрирует, что каждое t-мерное множество меток медиан является ретрактом t-мерного гиперкуба. И обратно, в упр. 94 показано, что ретракты гиперкуба всегда являются медианными графами.

Пороговые функции. Особенно привлекательный и важный класс булевых функций $f(x_1, x_2, \ldots, x_n)$ появляется, когда f можно определить формулой

$$f(x_1, x_2, \ldots, x_n) \;=\; [w_1x_1 + w_2x_2 + \cdots + w_nx_n \geq t], \tag{75}$$

где константы w_1, w_2, $\ldots$, w_n представляют собой целочисленные "веса", а t — целочисленное "пороговое" значение. Например, пороговые функции важны, даже когда все веса одинаковы: мы имеем

$$x_1 \wedge x_2 \wedge \cdots \wedge x_n = [x_1 + x_2 + \cdots + x_n \geq n]; \tag{76}$$

$$x_1 \vee x_2 \vee \cdots \vee x_n = [x_1 + x_2 + \cdots + x_n \geq 1]; \tag{77}$$

$$\langle x_1x_2\ldots x_{2t-1}\rangle = [x_1 + x_2 + \cdots + x_{2t-1} \geq t], \tag{78}$$

где $\langle x_1x_2\ldots x_{2t-1}\rangle$ означает медиану (или мажоритарное значение) мультимножества с нечетным числом булевых значений $\{x_1, x_2, \ldots, x_{2t-1}\}$. В частности, пороговыми функциями являются фундаментальные отображения $x \wedge y$, $x \vee y$ и $\langle xyz\rangle$, как и

$$\bar{x} \;=\; [-x \geq 0]. \tag{79}$$

В случае более общих весов мы получим много других интересных функций, таких как

$$[2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + x_n \geq (t_1t_2\ldots t_n)_2], \tag{80}$$

которая является истинной тогда и только тогда, когда бинарная строка $x_1x_2\ldots x_n$ лексикографически не меньше заданной бинарной строки $t_1t_2\ldots t_n$. Для заданного множества из n объектов размерами w_1, w_2, $\ldots$, w_n, подмножество этих объектов помещается в рюкзак размером $t-1$ тогда и только тогда, когда $f(x_1, x_2, \ldots, x_n) = 0$, где $x_j = 1$ представляет наличие объекта j в подмножестве. Простые модели нейронов, изначально предложенные У. Мак-Каллахом (W. McCulloch) и У. Питтсом (W. Pitts) в *Bull. Math. Biophysics* **5** (1943), 115–133, привели к написанию тысяч исследовательских статей о "нейронных сетях", построенных из пороговых функций.

Можно избавиться от всех отрицательных весов w_j, установив $x_j \leftarrow \bar{x}_j$, $w_j \leftarrow -w_j$ и $t \leftarrow t + |w_j|$. Таким образом, обобщенная пороговая функция может быть приведена к положительной пороговой функции, в которой все веса неотрицательны. Кроме того, любая положительная пороговая функция (75) может быть выражена как частный случай функции медианы (мажоритарной функции) нечетного числа переменных, поскольку мы имеем

$$\langle 0^a1^bx_1^{w_1}x_2^{w_2}\ldots x_n^{w_n}\rangle = [b + w_1x_1 + w_2x_2 + \cdots + w_nx_n \geq b + t], \tag{81}$$

где x^m означает m копий x и где a и b определяются правилами

$$a = \max(0, 2t-1-w), \quad b = \max(0, w+1-2t), \quad w = w_1 + w_2 + \cdots + w_n. \tag{82}$$

Например, когда все веса равны 1, мы имеем

$$\langle 0^{n-1}x_1\ldots x_n\rangle = x_1\wedge\cdots\wedge x_n \quad \text{и} \quad \langle 1^{n-1}x_1\ldots x_n\rangle = x_1\vee\cdots\vee x_n; \tag{83}$$

мы уже видели эти формулы в (45) для $n=2$. В общем случае либо a, либо b равно нулю и левая часть (81) определяет медиану $2T-1$ элементов, где

$$T \;=\; b+t \;=\; \max(t, w_1+w_2+\cdots+w_n+1-t). \tag{84}$$

Точек, в которых и a, и b одновременно больше нуля, нет, так как мажоритарная функция удовлетворяет закону сокращения

$$\langle 01x_1x_2\ldots x_{2t-1}\rangle \;=\; \langle x_1x_2\ldots x_{2t-1}\rangle. \tag{85}$$

Одно важное следствие (81) заключается в том, что каждая положительная пороговая функция получается из чисто мажоритарной функции

$$g(x_0,x_1,x_2,\ldots,x_n) \;=\; \langle x_0^{a+b}x_1^{w_1}x_2^{w_2}\ldots x_n^{w_n}\rangle \tag{86}$$

путем установки $x_0=0$ или 1. Другими словами, мы знаем все пороговые функции n переменных тогда и только тогда, когда мы знаем все различные медианы нечетного количества элементов для $n+1$ или меньшего количества переменных (не содержащих констант). Каждая чисто мажоритарная функция монотонна и самодуальна; таким образом, мы видели чисто мажоритарную функцию от четырех переменных $\{w,x,y,z\}$ в столбце s_j табл. 2 на с. 96, а именно $\langle w\rangle$, $\langle wwxyz\rangle$, $\langle wyz\rangle$, $\langle wxyzz\rangle$, $\langle xyz\rangle$, $\langle z\rangle$, $\langle wxyyz\rangle$, $\langle y\rangle$, $\langle wxz\rangle$, $\langle wxxyz\rangle$, $\langle x\rangle$, $\langle wxy\rangle$. Установив $w=0$ или 1, мы получим все положительные пороговые функции $f(x,y,z)$ трех переменных:

$$\begin{aligned}&\langle 0\rangle,\langle 1\rangle,\langle 00xyz\rangle,\langle 11xyz\rangle,\langle 0yz\rangle,\langle 1yz\rangle,\langle 0xyzz\rangle,\langle 1xyzz\rangle,\langle xyz\rangle,\langle z\rangle,\\ &\qquad\langle 0xyyz\rangle,\langle 1xyyz\rangle,\langle y\rangle,\langle 0xz\rangle,\langle 1xz\rangle,\langle 0xxyz\rangle,\langle 1xxyz\rangle,\langle x\rangle,\langle 0xy\rangle,\langle 1xy\rangle.\end{aligned} \tag{87}$$

Все 150 положительных пороговых функций четырех переменных могут быть получены подобным способом из самодуальных мажоритарных функций в ответе к упр. 84.

Имеется бесконечно много последовательностей весов $(w_1,w_2,\ldots,w_n)$, но пороговых функций для заданного значения n — только конечное число. Так что очевидно, что многие различные последовательности весов эквивалентны. Например, рассмотрим чисто мажоритарную функцию

$$\langle x_1^2x_2^3x_3^5x_4^7x_5^{11}x_6^{13}\rangle,$$

в которой в качестве весов использованы простые числа. Непосредственное рассмотрение всех 2^6 возможных случаев показывает, что

$$\langle x_1^2x_2^3x_3^5x_4^7x_5^{11}x_6^{13}\rangle = \langle x_1x_2^2x_3^2x_4^3x_5^4x_6^5\rangle; \tag{88}$$

таким образом, можно выразить ту же функцию с помощью существенно меньших весов. Аналогично пороговая функция

$$[(x_1x_2\ldots x_{20})_2 \ge (01100100100001111110)_2] = \langle 1^{225028}x_1^{524288}x_2^{262144}x_3^{131072}\ldots x_{20}\rangle,$$

частный случай (80), оказывается равной просто

$$\langle 1^{323}x_1^{764}x_2^{323}x_3^{323}x_4^{118}x_5^{118}x_6^{87}x_7^{31}x_8^{31}x_9^{25}x_{10}^6x_{11}^6x_{12}^6x_{13}^6x_{14}x_{15}x_{16}x_{17}x_{18}x_{19}\rangle. \tag{89}$$

В упр. 103 поясняется, как находить минимальное множество весов с помощью линейного программирования, не прибегая к перебору огромного количества вариантов методом грубой силы.

Красивая схема индексации, которая позволяет назначить каждой пороговой функции свой уникальный идентификатор, была открыта Ч. К. Чжоу (C. K. Chow) [*FOCS* **2** (1961), 34–38]. Пусть для любой заданной булевой функции $f(x_1,\ldots,x_n)$ значение $N(f)$ означает количество векторов $x=(x_1,\ldots,x_n)$, для которых $f(x)=1$, а $\Sigma(f)$ — сумму всех этих векторов. Например, если $f(x_1,x_2)=x_1\vee x_2$, мы имеем $N(f)=3$ и $\Sigma(f)=(0,1)+(1,0)+(1,1)=(2,2)$.

Теорема Т. *Пусть $f(x_1,\ldots,x_n)$ и $g(x_1,\ldots,x_n)$ — булевы функции с $N(f)=N(g)$ и $\Sigma(f)=\Sigma(g)$, где f — пороговая функция. Тогда $f=g$.*

Доказательство. Предположим, что имеется ровно k векторов $x^{(1)},\ldots,x^{(k)}$, таких, что $f(x^{(j)})=1$ и $g(x^{(j)})=0$. Поскольку $N(f)=N(g)$, должно быть ровно k векторов $y^{(1)},\ldots,y^{(k)}$, таких, что $f(y^{(j)})=0$ и $g(y^{(j)})=1$. А поскольку $\Sigma(f)=\Sigma(g)$, мы должны также иметь $x^{(1)}+\cdots+x^{(k)}=y^{(1)}+\cdots+y^{(k)}$.

Теперь предположим, что f — пороговая функция (75); тогда мы имеем $w\cdot x^{(j)}\ge t$ и $w\cdot y^{(j)}<t$ для $1\le j\le k$. Но если $f\ne g$, мы имеем $k>0$ и $w\cdot(x^{(1)}+\cdots+x^{(k)})\ge kt>w\cdot(y^{(1)}+\cdots+y^{(k)})$, что приводит к противоречию. ▌

Пороговые функции обладают многими любопытными свойствами, и некоторые из них рассматриваются ниже в упражнениях. Классическая теория пороговых функций хорошо подытожена в книге Сабуро Мурога (Saburo Muroga) *Threshold Logic and its Applications* (Wiley, 1971).

Симметричные булевы функции. Функция $f(x_1,\ldots,x_n)$ называется *симметричной*, если $f(x_1,\ldots,x_n)$ равно $f(x_{p(1)},\ldots,x_{p(n)})$ для всех перестановок $p(1)\ldots p(n)$ множества $\{1,\ldots,n\}$. Когда все x_j равны 0 или 1, это условие означает, что f зависит только от количества единиц среди ее аргументов, а именно "контрольной суммы" $\nu x=\nu(x_1,\ldots,x_n)=x_1+\cdots+x_n$. Для булевой функции, истинной тогда и только тогда, когда νx равно либо k_1, либо k_2, $\cdots$, либо k_r, обычно используется обозначение $S_{k_1,k_2,\ldots,k_r}(x_1,\ldots,x_n)$. Например, $S_{1,3,5}(v,w,x,y,z)=v\oplus w\oplus x\oplus y\oplus z$; $S_{3,4,5}(v,w,x,y,z)=\langle vwxyz\rangle$; $S_{4,5}(v,w,x,y,z)=\langle 00vwxyz\rangle$.

Многие приложения симметрии включают базовые функции $S_k(x_1,\ldots,x_n)$, истинные только тогда, когда $\nu x=k$. Например, $S_3(x_1,x_2,x_3,x_4,x_5,x_6)$ истинна тогда и только тогда, когда ровно половина ее аргументов $\{x_1,\ldots,x_6\}$ истинны, а вторая половина — ложны. В таких случаях мы, очевидно, имеем

$$S_k(x_1,\ldots,x_n)\;=\;S_{\ge k}(x_1,\ldots,x_n)\wedge\overline{S_{\ge k+1}(x_1,\ldots,x_n)},\tag{90}$$

где $S_{\ge k}(x_1,\ldots,x_n)$ представляет собой сокращение для $S_{k,k+1,\ldots,n}(x_1,\ldots,x_n)$. Функции $S_{\ge k}(x_1,\ldots,x_n)$, конечно, являются пороговыми функциями $[x_1+\cdots+x_n\ge k]$, которые мы уже изучали.

Более сложные случаи могут быть рассмотрены как пороговые функции пороговых функций. Например, мы имеем

$$\begin{aligned}S_{2,3,6,8,9}(x_1,\ldots,x_{12})\;&=\;\bigl[\nu x\ge 2+4[\nu x\ge 4]+2[\nu x\ge 7]+5[\nu x\ge 10]\bigr]\\&=\;\bigl\langle 00x_1\ldots x_{12}\langle 0^5\bar x_1\ldots\bar x_{12}\rangle^4\langle 1\bar x_1\ldots\bar x_{12}\rangle^2\langle 1^7\bar x_1\ldots\bar x_{12}\rangle^5\bigr\rangle,\end{aligned}\tag{91}$$

поскольку количество единиц во внешней мажоритарной функции 25 элементов оказывается равным соответственно $(11, 12, 13, 14, 11, 12, 13, 12, 13, 14, 10, 11, 12)$, когда $x_1 + \cdots + x_{12} = (0, 1, \ldots, 12)$. Аналогичная двухуровневая схема работает в общем случае [R. C. Minnick, *IRE Trans.* **EC-10** (1961), 6–16]; а в случае трех или более уровней логики можно снизить количество пороговых операций еще сильнее. (См. упр. 113.)

Для вычисления симметричных булевых функций открыт ряд искусных методов. Например, С. Мурога (S. Muroga) приписывает следующую замечательную последовательность формул Ф. Сасаки (F. Sasaki):

$$x_0 \oplus x_1 \oplus \cdots \oplus x_{2m} = \langle \bar{x}_0 s_1 s_2 \ldots s_{2m} \rangle,$$
$$\text{где} \quad s_j = \langle x_0 x_j x_{j+1} \ldots x_{j+m-1} \bar{x}_{j+m} \bar{x}_{j+m+1} \ldots \bar{x}_{j+2m-1} \rangle, \tag{92}$$

если $m > 0$ и если рассматривать x_{2m+k} как эквивалентные x_k для $k \geq 1$. В частности, когда $m = 1$ и $m = 2$, мы получаем тождества

$$x_0 \oplus x_1 \oplus x_2 = \langle \bar{x}_0 \langle x_0 x_1 \bar{x}_2 \rangle \langle x_0 x_2 \bar{x}_1 \rangle \rangle; \tag{93}$$
$$x_0 \oplus \cdots \oplus x_4 = \langle \bar{x}_0 \langle x_0 x_1 x_2 \bar{x}_3 \bar{x}_4 \rangle \langle x_0 x_2 x_3 \bar{x}_4 \bar{x}_1 \rangle \langle x_0 x_3 x_4 \bar{x}_1 \bar{x}_2 \rangle \langle x_0 x_4 x_1 \bar{x}_2 \bar{x}_3 \rangle \rangle. \tag{94}$$

Правые части полностью симметричны, но это не так очевидно! (См. упр. 115.)

Канализирующие функции. Булева функция $f(x_1, \ldots, x_n)$ называется *канализирующей* (*canalizing*), или принуждающей (forcing), если можно вывести ее значение при изучении не более одной из ее переменных. Говоря более точно, функция f является канализирующей, если $n = 0$ или если существует индекс j, для которого $f(x)$ имеет константное значение либо когда мы положим $x_j = 0$, либо когда положим $x_j = 1$. Например, функция $f(x, y, z) = (x \oplus z) \vee \bar{y}$ является канализирующей, поскольку она всегда равна 1, когда $y = 0$. (Когда $y = 1$, мы не знаем значение f без изучения значений x и z; но половина — это уже лучше, чем ничего.) Такие функции, введенные Стюартом Кауффманом (Stuart Kauffman) [*Lectures on Mathematics in the Life Sciences* **3** (1972), 63–116; *J. Theoretical Biology* **44** (1974), 167–190], весьма важны во многих приложениях, в особенности в химии и биологии. Некоторые из их свойств рассматриваются в упр. 125–129.

Количественные соображения. Мы изучили много различных типов булевых функций, так что естественным образом возникает вопрос "Сколько имеется функций от n переменных каждого типа?" Ответы, по крайней мере для небольших значений n, приведены в табл. 3–5.

Все возможные функции подсчитаны в табл. 3. Для каждого n имеется 2^{2^n} возможностей, поскольку существует 2^{2^n} возможных таблиц истинности. Некоторые из этих функций самодуальны, некоторые монотонны; есть одновременно и самодуальные, и монотонные функции, как упоминаемые в теореме P. Некоторые функции являются функциями Хорна, как в теореме H; некоторые — функциями Крома (теорема S); и т. д.

Однако в табл. 4 две функции рассматриваются как идентичные, если они отличаются только изменением имен переменных. Таким образом, при $n = 2$ есть только 12 разных случаев, поскольку, например, $x \vee \bar{y}$ и $\bar{x} \vee y$ — по сути, одно и то же.

Табл. 5 идет еще дальше: она позволяет нам дополнять отдельные переменные и даже всю функцию, по сути, не меняя ее. С этой точки зрения 256 булевых

Таблица 3

Булевы функции от n переменных

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
Произвольная	2	4	16	256	65 536	4 294 967 296	18 446 744 073 709 551 616
Самодуальная	0	2	4	16	256	65 536	4 294 967 296
Монотонная	2	3	6	20	168	7581	7 828 354
Монотонная самодуальная	0	1	2	4	12	81	2646
Хорна	2	4	14	122	4960	2 771 104	151 947 502 948
Крома	2	4	16	166	4170	224 716	24 445 368
Пороговая	2	4	14	104	1882	94 572	15 028 134
Симметричная	2	4	8	16	32	64	128
Канализирующая	2	4	14	120	3514	1 292 276	103 071 426 294

Таблица 4

Булевы функции, различные при перестановках переменных

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
Произвольная	2	4	12	80	3984	37 333 248	25 626 412 338 274 304
Самодуальная	0	2	2	8	32	1088	6 385 408
Монотонная	2	3	5	10	30	210	16 353
Монотонная самодуальная	0	1	1	2	3	7	30
Хорна	2	4	10	38	368	29 328	216 591 692
Крома	2	4	12	48	308	3028	49 490
Пороговая	2	4	10	34	178	1720	590 440
Канализирующая	2	4	10	38	294	15 774	149 325 022

Таблица 5

Булевы функции, различные при дополнениях/перестановках

	$n=0$	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
Произвольная	1	2	4	14	222	616 126	200 253 952 527 184
Самодуальная	0	1	1	3	7	83	109 950
Пороговая	1	2	3	6	15	63	567
Пороговая самодуальная	0	1	1	2	3	7	21
Канализирующая	1	2	3	6	22	402	1 228 158

функций от (x, y, z) разбиваются только на 14 различных классов эквивалентности.

Представитель	Размер класса	Представитель	Размер класса
0	2	$x \wedge (y \oplus z)$	24
x	6	$x \oplus (y \wedge z)$	24
$x \wedge y$	24	$(x \wedge y) \vee (\bar{x} \wedge z)$	24
$x \oplus y$	6	$(x \vee y) \wedge (x \oplus z)$	48
$x \wedge y \wedge z$	16	$(x \oplus y) \vee (x \oplus z)$	8
$x \oplus y \oplus z$	2	$\langle xyz \rangle$	8
$x \wedge (y \vee z)$	48	$S_1(x, y, z)$	16

(95)

Мы изучим способы подсчета и перечисления неэквивалентных комбинаторных объектов в разделе 7.2.3.

Упражнения

1. [*15*] (Льюис Кэрролл (Lewis Carroll).) Придайте смысл комментарию Траляля, процитированному в начале данного раздела. [*Указание*: см. табл. 1.]

2. [*17*] Логики на далекой планете Наурика используют символ 1 для представления "ложь", а 0 — для представления "истина". Таким образом, например, у них имеется бинарный оператор под названием "или" со следующими свойствами:

$$1 \text{ или } 1 = 1, \qquad 1 \text{ или } 0 = 0, \qquad 0 \text{ или } 1 = 0, \qquad 0 \text{ или } 0 = 0,$$

которые мы связываем с оператором $\wedge$. Какие операции будут связаны с 16 логическими операторами, которые науриканцы называют соответственно "противоречие", "и", ..., "отрицание конъюнкции", "достоверность" (см. табл. 1)?

▶ **3.** [*13*] Предположим, что логическими значениями вместо 0 и 1 являются соответственно -1 для лжи и $+1$ для истины. Какие операции $\circ$ из табл. 1 будут соответствовать (а) $\max(x, y)$? (б) $\min(x, y)$? (в) $-x$? (г) $x \cdot y$?

4. [*24*] (Г. М. Шеффер (H. M. Sheffer).) Цель этого упражнения — показать, что все операции из табл. 1 могут быть выражены через НЕ-И. (а) Для каждого из 16 операторов $\circ$ этой таблицы найдите формулу, эквивалентную $x \circ y$, но использующую только оператор $\overline{\wedge}$. Ваша формула должна быть краткой, насколько это возможно. Например, ответ для операции $\llcorner$ — просто "x", но ответом для $\overline{\llcorner}$ является "$x \,\overline{\wedge}\, x$". Не используйте в своих формулах константы 0 или 1. (б) Аналогично найдите 16 коротких формул, когда использование констант *разрешено*. Например, $x \,\overline{\llcorner}\, y$ может теперь быть выражено как "$x \,\overline{\wedge}\, 1$".

5. [*24*] Рассмотрите упр. 4 с базовым оператором $\overline{\subset}$ вместо $\overline{\wedge}$.

6. [*21*] (Э. Шрёдер (E. Schröder).) (а) Какие из 16 операций из табл. 1 ассоциативны — другими словами, какие из них удовлетворяют тождеству $x \circ (y \circ z) = (x \circ y) \circ z$? (б) Какие из них удовлетворяют тождеству $(x \circ y) \circ (y \circ z) = x \circ z$?

7. [*20*] Какие операторы из табл. 1 обладают тем свойством, что $x \circ y = z$ тогда и только тогда, когда $y \circ z = x$?

8. [*24*] Какие из 16^2 пар операций $(\circ, \square)$ удовлетворяют закону левой дистрибутивности $x \circ (y \square z) = (x \circ y) \square (x \circ z)$?

9. [*16*] Истинны или ложны следующие утверждения? (а) $(x \oplus y) \vee z = (x \vee z) \oplus (y \vee z)$; (б) $(w \oplus x \oplus y) \vee z = (w \vee z) \oplus (x \vee z) \oplus (y \vee z)$; (в) $(x \oplus y) \vee (y \oplus z) = (x \oplus z) \vee (y \oplus z)$.

10. [*17*] Какой вид имеет мультилинейное представление "случайной" функции (22)?

11. [*M25*] Существует ли интуитивный способ точно понять, когда мультилинейное представление $f(x_1, \ldots, x_n)$ содержит, скажем, член $x_2 x_3 x_6 x_8$? (См. (19).)

▶ **12.** [*M23*] *Целочисленное мультилинейное представление* булевой функции расширяет представления наподобие (19) на полиномы с целыми коэффициентами $f(x_1,\ldots,x_n)$ таким образом, что $f(x_1,\ldots,x_n)$ дает корректное значение (0 или 1) для всех 2^n возможных 0–1-векторов $(x_1,\ldots,x_n)$ *без* взятия остатка по модулю 2. Например, целочисленное мультилинейное представление, соответствующее (19), представляет собой $1-xy-xz-yz+3xyz$.

а) Каково целочисленное мультилинейное представление "случайной" функции (22)?

б) Насколько большими могут быть коэффициенты такого представления $f(x_1,\ldots,x_n)$?

в) Покажите, что в каждом целочисленном мультилинейном представлении $f(x_1,\ldots,x_n)$, когда $x_1,\ldots,x_n$ являются действительными числами, такими, что $0\le x_1,\ldots,x_n\le 1$, выполняются неравенства $0\le f(x_1,\ldots,x_n)\le 1$.

г) Аналогично покажите, что если $f(x_1,\ldots,x_n)\le g(x_1,\ldots,x_n)$, когда $\{x_1,\ldots,x_n\}\subseteq\{0,1\}$, то $f(x_1,\ldots,x_n)\le g(x_1,\ldots,x_n)$, когда $\{x_1,\ldots,x_n\}\subseteq[0\,..\,1]$.

д) Докажите, что если f монотонна и $0\le x_j\le y_j\le 1$ для $1\le j\le n$, то $f(x)\le f(y)$.

▶ **13.** [*20*] Рассмотрим систему, состоящую из n модулей, каждый из которых может быть "исправным" или "поломанным". Если x_j представляет условие "модуль j исправен", то булева функция наподобие $x_1\wedge(\bar x_2\vee\bar x_3)$ представляет утверждение "модуль 1 исправен, но модуль 2 или модуль 3 поломан"; а $S_3(x_1,\ldots,x_n)$ означает "все три модуля исправны".

Предположим, что каждый модуль j исправен с вероятностью p_j, не зависящей от состояния других модулей. Покажите, что булева функция $f(x_1,\ldots,x_n)$ истинна с вероятностью $F(p_1,\ldots,p_n)$, где F — полином от переменных $p_1,\ldots,p_n$.

14. [*20*] Функция вероятности $F(p_1,\ldots,p_n)$ из упр. 13 часто называется *доступностью* системы. Найдите самодуальную функцию $f(x_1,x_2,x_3)$ максимальной доступности, когда вероятности (p_1,p_2,p_3) равны (а) $(0.9,0.8,0.7)$; (б) $(0.8,0.6,0.4)$; (в) $(0.8,0.6,0.1)$.

▶ **15.** [*M20*] Покажите, что если $f(x_1,\ldots,x_n)$ — любая булева функция, то существует полином $F(x)$, обладающий тем свойством, что $F(x)$ целый при целом x, и $f(x_1,\ldots,x_n)=F((x_n\ldots x_1)_2)\bmod 2$. *Указание*: рассмотрите $\binom{x}{k}\bmod 2$.

16. [*13*] Можем ли мы заменить каждый оператор $\vee$ на $\oplus$ в полной дизъюнктивной нормальной форме?

17. [*10*] Согласно законам де Моргана обобщенная дизъюнктивная нормальная форма, такая как (25), представляет собой не только ИЛИ некоторых И, но и НЕ-И некоторых НЕ-И:

$$\overline{\overline{(u_{11}\wedge\cdots\wedge u_{1s_1})}\ \wedge\cdots\wedge\ \overline{(u_{m1}\wedge\cdots\wedge u_{ms_m})}}.$$

Оба уровня логики можно, таким образом, рассматривать как идентичные.

Студент Шустрый переписал это выражение в виде

$$(u_{11}\,\overline{\wedge}\cdots\overline{\wedge}\,u_{1s_1})\ \overline{\wedge}\cdots\overline{\wedge}\ (u_{m1}\,\overline{\wedge}\cdots\overline{\wedge}\,u_{ms_m}).$$

Было ли это хорошей мыслью?

▶ **18.** [*20*] Пусть $u_1\wedge\cdots\wedge u_s$ является импликантой в дизъюнктивной нормальной форме булевой функции f и пусть $v_1\vee\cdots\vee v_t$ является дизъюнктомв конъюнктивной нормальной форме той же функции. Докажите, что $u_i=v_j$ для некоторых i и j.

19. [*20*] Чему равна конъюнктивная простая форма "случайной" функции в (22)?

20. [*M21*] Истинно или ложно утверждение: каждая простая импликанта конъюнкции $f\wedge g$ может быть записана как $f'\wedge g'$, где f' — простая импликанта f, а g' — простая импликанта g.

21. [*M20*] Докажите, что неконстантная булева функция монотонна тогда и только тогда, когда она может быть выражена только через операции $\wedge$ и $\vee$.

22. [*20*] Предположим, что $f(x_1,\ldots,x_n) = g(x_1,\ldots,x_{n-1}) \oplus h(x_1,\ldots,x_{n-1})\wedge x_n$ как в (16). Какие условия необходимо и достаточно наложить на функции g и h, чтобы функция f была монотонной?

23. [*15*] Какова конъюнктивная простая форма для $(v\wedge w\wedge x)\vee(v\wedge x\wedge z)\vee(x\wedge y\wedge z)$?

24. [*M20*] Рассмотрим полное бинарное дерево с 2^k листьями, показанное здесь для $k=3$. Применяя поочередно $\wedge$ и $\vee$ на каждом уровне, назначая корню $\wedge$, для данного примера мы получим $((x_0\wedge x_1)\vee(x_2\wedge x_3))\wedge((x_4\wedge x_5)\vee(x_6\wedge x_7))$. Сколько простых импликант содержит получающаяся функция?

25. [*M21*] Сколько простых импликант имеет $(x_1\vee x_2)\wedge(x_2\vee x_3)\wedge\cdots\wedge(x_{n-1}\vee x_n)$?

26. [*M23*] Пусть $\mathcal{F}$ и $\mathcal{G}$ представляют собой семейства множеств индексов для простых дизъюнктов и простых импликант монотонной конъюнктивной нормальной формы и монотонной дизъюнктивной нормальной формы:

$$f(x) = \bigwedge_{I\in\mathcal{F}}\bigvee_{i\in I} x_i; \qquad g(x) = \bigvee_{J\in\mathcal{G}}\bigwedge_{j\in J} x_j.$$

Эффективно найдите x, такое, что $f(x)\neq g(x)$, если выполняется любое из следующих условий.

а) Существуют $I\in\mathcal{F}$ и $J\in\mathcal{G}$, такие, что $I\cap J=\emptyset$.
б) $\bigcup_{I\in\mathcal{F}} I \neq \bigcup_{J\in\mathcal{G}} J$.
в) Существует $I\in\mathcal{F}$, такое, что $|I|>|\mathcal{G}|$, или $J\in\mathcal{G}$, такое, что $|J|>|\mathcal{F}|$.
г) $\sum_{I\in\mathcal{F}} 2^{n-|I|} + \sum_{J\in\mathcal{G}} 2^{n-|J|} < 2^n$, где $n=|\bigcup_{I\in\mathcal{F}} I|$.

27. [*M31*] Продолжая выполнение предыдущего упражнения, рассмотрим следующий алгоритм $\mathrm{X}(\mathcal{F},\mathcal{G})$, который возвращает либо вектор x, такой, что $f(x)\neq g(x)$, либо значение Λ, если $f=g$.

X1. [Проверка необходимых условий.] Возвращает соответствующее значение x, если применимо условие (а), (б), (в) или (г) из упр. 26.

X2. [Выполнено?] Если $|\mathcal{F}||\mathcal{G}|\le 1$, вернуть Λ.

X3. [Рекурсия.] Вычислить следующие сокращенные семейства для "наилучшего" индекса k:

$$\mathcal{F}_1 = \{I \mid I\in\mathcal{F},\ k\notin I\}, \qquad \mathcal{F}_0 = \mathcal{F}_1\cup\{I \mid k\notin I,\ I\cup\{k\}\in\mathcal{F}\};$$
$$\mathcal{G}_0 = \{J \mid J\in\mathcal{G},\ k\notin J\}, \qquad \mathcal{G}_1 = \mathcal{G}_0\cup\{J \mid k\notin J,\ J\cup\{k\}\in\mathcal{G}\}.$$

Удалить любой член $\mathcal{F}_0$ или $\mathcal{G}_1$, который содержит другой член того же семейства. Индекс k должен быть выбран таким образом, чтобы отношение $\rho=\min(|\mathcal{F}_1|/|\mathcal{F}|, |\mathcal{G}_0|/|\mathcal{G}|)$ было настолько малым, насколько это возможно. Если $\mathrm{X}(\mathcal{F}_0,\mathcal{G}_0)$ возвращает вектор x, вернуть тот же вектор, расширенный значением $x_k=0$. В противном случае, если $\mathrm{X}(\mathcal{F}_1,\mathcal{G}_1)$ возвращает вектор x, вернуть тот же вектор, расширенный значением $x_k=1$. В противном случае вернуть Λ. ▮

Докажите, что если $N=|\mathcal{F}|+|\mathcal{G}|$, то шаг X1 выполняется не более $N^{O(\log N)^2}$ раз. *Указание*: покажите, что на шаге X3 мы всегда имеем $\rho\le 1-1/\lg N$.

28. [*21*] (У. В. Куайн (W. V. Quine), 1952.) Если $f(x_1,\ldots,x_n)$ представляет собой булеву функцию с простыми импликантами $p_1,\ \ldots,\ p_q$, положим $g(y_1,\ldots,y_q)=\bigwedge_{f(x)=1}\bigvee\{y_j \mid p_j(x)=1\}$. Например, "случайная" функция (22) истинна в восьми точках (28) и имеет пять простых импликант, описанных в (29) и (30); так что $g(y_1,\ldots,y_5)$ в этом случае равна

$$(y_1\vee y_2)\wedge(y_1)\wedge(y_2\vee y_3)\wedge(y_4)\wedge(y_3\vee y_5)\wedge(y_5)\wedge(y_5)\wedge(y_4\vee y_5)$$
$$= (y_1\wedge y_2\wedge y_4\wedge y_5)\vee(y_1\wedge y_3\wedge y_4\wedge y_5).$$

Докажите, что каждое кратчайшее DNF-выражение для f соответствует простой импликанте монотонной функции g.

29. [*22*] (Несколько последующих упражнений посвящены алгоритмам, работающим с импликантами булевых функций путем представления точек n-мерного куба как n-битовых чисел $(b_{n-1}\ldots b_1b_0)_2$, а не как битовых строк $x_1\ldots x_n$.) Поясните, как для заданной битовой позиции j и заданных n-битовых значений $v_0 < v_1 < \cdots < v_{m-1}$ найти все пары (k, k'), такие, что $0 \le k < k' < m$ и $v_{k'} = v_k \oplus 2^j$, в возрастающем порядке k. Время работы вашей процедуры должно быть $O(m)$, если битовые операции над n-битовыми словами выполняются за константное время.

▶ **30.** [*27*] В разделе указывалось, что импликанта булевой функции может рассматриваться как подкуб, такой, как $01{*}0{*}$, содержащийся в множестве V всех точек, для которых эта функция истинна. Каждый подкуб может быть представлен в виде пары бинарных чисел $a = (a_{n-1}\ldots a_0)_2$ и $b = (b_{n-1}\ldots b_0)_2$, где a записывает позиции звездочек, а b записывает биты в не-$*$-позициях. Например, числа $a = (00101)_2$ и $b = (01000)_2$ представляют подкуб $c = 01{*}0{*}$. Всегда выполняется условие $a \mathbin{\&} b = 0$.

"j-двойник" ("j-buddy") определен, когда $a_j = 0$, путем изменения b на $b \oplus 2^j$. Например, $01{*}0{*}$ имеет три j-двойника, а именно 4-двойника $11{*}0{*}$, 3-двойника $00{*}0{*}$ и 1-двойника $01{*}1{*}$. Каждому подкубу $c \subseteq V$ может быть присвоено значение дескриптора (tag value) $(t_{n-1}\ldots t_0)_2$, где $t_j = 1$ тогда и только тогда, когда j-двойник c определен и содержится в V. При таком определении c представляет максимальный подкуб (а следовательно, простую импликанту) тогда и только тогда, когда его дескриптор равен нулю.

Используйте эти концепции для разработки алгоритма, который находит все максимальные подкубы (a, b) данного множества V, где V представлено n-битовыми числами $v_0 < v_1 < \cdots < v_{m-1}$.

▶ **31.** [*28*] Алгоритм из упр. 30 требует полного списка всех точек, в которых булева функция истинна, а этот список может быть весьма длинным. Следовательно, мы можем предпочесть работать непосредственно с подкубами, никогда не переходя на уровень явных n-кортежей, если это не является необходимым. Ключом к таким высокоуровневым методам является понятие *консенсуса* между подкубами c и c', обозначаемого как $c \sqcup c'$ и определяемого как наибольший подкуб c'', такой, что

$$c'' \subseteq c \cup c', \qquad c'' \not\subseteq c \quad \text{и} \quad c'' \not\subseteq c'.$$

Такой c'' не всегда существует. Например, если $c = 000{*}$ и $c' = {*}111$, каждый подкуб, содержащийся в $c \cup c'$, содержится либо в c, либо в c'.

а) Докажите, что консенсус, если таковой существует, может быть вычислен покомпонентно с использованием следующих формул в каждой координатной позиции:

$$x \sqcup x = x \sqcup {*} = {*} \sqcup x = x \quad \text{и} \quad x \sqcup \bar{x} = {*} \sqcup {*} = {*} \qquad \text{для } x = 0 \text{ и } x = 1.$$

Кроме того, $c \sqcup c'$ существует тогда и только тогда, когда правило $x \sqcup \bar{x} = {*}$ используется ровно в одном компоненте.

б) Подкуб с k звездочками называется k-кубом. Покажите, что если c является k-кубом, а c' является k'-кубом и если существует консенсус $c'' = c \sqcup c'$, то c'' является k''-кубом, где $1 \le k'' \le \min(k, k') + 1$.

в) Если C и C' являются семействами подкубов, то пусть

$$C \sqcup C' \;=\; \{c \sqcup c' \mid c \in C,\ c' \in C' \text{ и } c \sqcup c' \text{ существует}\}.$$

Поясните, почему работает следующий алгоритм.

Алгоритм E (*Поиск максимальных подкубов*). Для заданного семейства C подкубов n-мерного куба этот алгоритм выводит максимальные подкубы множества $V = \bigcup_{c\in C} c$ без реального вычисления самого множества V.

E1. [Инициализация.] Установить $j \leftarrow 0$. Удалить все подкубы c из C, которые содержатся в других подкубах.

E2. [Выполнено?] (В этот момент каждый j-куб $\subseteq V$ содержится в некотором элементе C и C не содержит k-кубов с $k < j$.) Если C пустое, алгоритм завершает работу.

E3. [Получение консенсусов.] Установить $C' \leftarrow C \sqcup C$ и удалить все подкубы из C', которые являются k-кубами для $k \le j$. В процессе выполнения этого вычисления выводятся также все j-кубы $c \in C$, для которых $c \sqcup C$ не производит $(j+1)$-куба множества C'.

E4. [Продвижение.] Установить $C \leftarrow C \cup C'$, но удалить все j-кубы из этого объединения. Затем удалить все подкубы $c \in C$, которые содержатся в других. Установить $j \leftarrow j+1$ и перейти к шагу E2. ▮

(См. в упр. 7.1.3–142 более эффективный способ выполнения этих вычислений.)

▶ **32.** [*M29*] Пусть $c_1, \ldots, c_m$ — подкубы n-мерного куба.

а) Докажите, что $c_1 \cup \cdots \cup c_m$ содержит не более одного максимального подкуба c, не содержащегося в $c_1 \cup \cdots \cup c_{j-1} \cup c_{j+1} \cup \cdots \cup c_m$ для любого $j \in \{1, \ldots, m\}$. (Если c существует, назовем его *обобщенным консенсусом* $c_1, \ldots, c_m$, поскольку $c = c_1 \sqcup c_2$ при использовании обозначений из упр. 31, когда $m = 2$.)

б) Найдите множество m подкубов, для которых каждое из $2^m - 1$ непустых подмножеств множества $\{c_1, \ldots, c_m\}$ имеет обобщенный консенсус.

в) Докажите, что DNF с m импликантами имеет не более $2^m - 1$ простых импликант.

г) Найдите DNF, которая имеет m импликант и $2^m - 1$ простых импликант.

33. [*M21*] Пусть $f(x_1, \ldots, x_n)$ — одна из $\binom{2^n}{m}$ булевых функций, истинных ровно в m точках. Если f выбирается случайным образом, то чему равна вероятность того, что $x_1 \wedge \cdots \wedge x_k$ является (а) импликантой f? (б) простой импликантой f? [Дайте ответ на п. (б) в виде суммы; но для случая $k = n$ приведите ответ в аналитическом виде.]

▶ **34.** [*HM37*] Продолжая выполнение упр. 33, обозначим через $c(m,n)$ среднее общее количество импликант, а через $p(m,n)$ — среднее общее количество простых импликант.

а) Покажите, что если $0 \le m \le 2^n/n$, то $m \le c(m,n) \le \frac{3}{2}m + O(m/n)$ и $p(m,n) \ge me^{-1} + O(m/n)$; следовательно, в этом диапазоне $p(m,n) = \Theta(c(m,n))$.

б) Пусть теперь $2^n/n \le m \le (1-\epsilon)2^n$, где ϵ — фиксированная положительная константа. Определим числа t и α_{mn} с помощью отношений

$$n^{-4/3} \le \left(\frac{m}{2^n}\right)^{2^t} = \alpha_{mn} < n^{-2/3}, \qquad t \text{ — целое.}$$

Выразите асимптотические значения $c(m,n)$ и $p(m,n)$ через n, t и α_{mn}. [*Указание*: покажите, что почти все импликанты имеют ровно $n-t$ или $n-t-1$ литералов.]

в) Оцените $c(m,n)/p(m,n)$ при $m = 2^{n-1}$ и $n = \lfloor (\ln t - \ln\ln t)2^{2^t} \rfloor$ для целого t.

г) Докажите, что $c(m,n)/p(m,n) = O(\log\log n/\log\log\log n)$ при $m \le (1-\epsilon)2^n$.

▶ **35.** [*M25*] Дизъюнктивная нормальная форма называется *ортогональной*, если ее импликанты соответствуют непересекающимся подкубам. Ортогональные дизъюнктивные нормальные формы особенно полезны при вычислениях или оценках полиномов надежности из упр. 13.

Полная дизъюнктивная нормальная форма каждой функции очевидно ортогональна, поскольку ее подкубы представляют собой отдельные точки. Но мы часто можем найти ортогональную дизъюнктивную нормальную форму, которая имеет значительно меньше импликант, в особенности когда функция монотонна. Например, функция $(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_4)$ истинна в восьми точках и имеет ортогональную дизъюнктивную нормальную форму

$$(x_1 \wedge x_2) \vee (\bar{x}_1 \wedge x_2 \wedge x_3) \vee (\bar{x}_2 \wedge x_3 \wedge x_4).$$

Другими словами, перекрывающиеся подкубы 11**, *11*, **11 могут быть заменены неперекрывающимися кубами 11**, 011*, *011. При использовании для кубов бинарной записи из упр. 30 эти подкубы имеют коды звездочек 0011, 0001, 1000 и битовые коды 1100, 0110, 0011.

Каждая монотонная функция может быть определена с помощью списка битовых кодов $B_1, \ldots, B_p$, когда кодами звездочек являются соответственно $\bar{B}_1, \ldots, \bar{B}_p$. Пусть для такого заданного списка "тенью" S_k кода B_k является побитовое ИЛИ $B_j \mathbin{\&} \bar{B}_k$ для всех $1 \le j < k$, таких, что $\nu(B_j \mathbin{\&} \bar{B}_k) = 1$:

$$S_k = \beta_{1k} \mid \cdots \mid \beta_{(k-1)k}, \quad \beta_{jk} = \big((B_j \mathbin{\&} \bar{B}_k) \oplus ((B_j \mathbin{\&} \bar{B}_k) - 1)\big) \dot{-} ((B_j \mathbin{\&} \bar{B}_k) - 1).$$

Например, когда битовыми кодами являются $(B_1, B_2, B_3) = (1100, 0110, 0011)$, мы получим коды теней $(S_1, S_2, S_3) = (0000, 1000, 0100)$.

а) Покажите, что коды звездочек $A'_j = \bar{B}_j - S_j$ и битовые коды B_j определяют подкубы, которые покрывают те же точки, что и подкубы с кодами звездочек $A_j = \bar{B}_j$.

б) Список битовых кодов $B_1, \ldots, B_p$ называется *разверткой* (*shelling*), если $B_j \mathbin{\&} S_k$ ненулевое для всех $1 \le j < k \le p$. Например, $(1100, 0110, 0011)$ является разверткой; но если мы расположим битовые коды в ином порядке $(1100, 0011, 0110)$, то условие развертки будет нарушено при $j = 1$ и $k = 2$, хотя мы имеем $S_3 = 1001$. Докажите, что подкубы в п. (а) непересекающиеся тогда и только тогда, когда список битовых кодов является разверткой.

в) Согласно теореме Q среди B при представлении монотонной булевой функции таким способом должна появляться каждая простая импликанта. Но иногда нам требуется добавить дополнительные импликанты, если мы хотим, чтобы кубы были непересекающимися. Например, не существует развертки для битовых кодов 1100 и 0011. Покажите, что мы можем, однако, получить развертку для функции $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$, добавляя еще один битовый код. Какой вид имеет получающаяся в результате дизъюнктивная нормальная форма?

г) Переставьте битовые коды $\{11000, 01100, 00110, 00011, 11010\}$ так, чтобы получить развертку.

д) Добавьте два битовых кода к множеству $\{110000, 011000, 001100, 000110, 000011\}$, чтобы получить развертку.

36. [*M21*] Продолжая выполнение упр. 35, пусть f — любая монотонная функция, не идентичная 1. Покажите, что множество битовых векторов

$$B = \{\, x \mid f(x) = 1 \text{ и } f(x') = 0 \,\}, \qquad x' = x \mathbin{\&} (x-1),$$

всегда разворачиваемо при перечислении в уменьшающемся лексикографическом порядке. (Вектор x' получается из x путем обновления крайней справа единицы.) Например, этот метод дает ортогональную дизъюнктивную нормальную форму для $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$ из списка $(1100, 1011, 0111, 0011)$.

▶ **37.** [*M31*] Найдите разворачиваемую дизъюнктивную нормальную форму для $(x_1 \wedge x_2) \vee (x_3 \wedge x_4) \vee \cdots \vee (x_{2n-1} \wedge x_{2n})$, имеющую $2^n - 1$ импликант, и докажите, что для этой функции не имеется ортогональной DNF с меньшим количеством импликант.

38. [*05*] Сложно ли проверить выполнимость функций в *дизъюнктивной* нормальной форме?

▶ **39.** [*25*] Пусть $f(x_1,\ldots,x_n)$ — булева формула, представленная в виде расширенного бинарного дерева с N внутренними узлами и $N+1$ листьями. Каждый лист помечен переменной x_k, а каждый внутренний узел помечен одним из шестнадцати бинарных операторов из табл. 1; применение операторов снизу вверх дает $f(x_1,\ldots,x_n)$ в качестве значения корня.

Поясните, как построить формулу $F(x_1,\ldots,x_n,y_1,\ldots,y_N)$ в виде 3CNF, имеющей ровно $4N+1$ дизъюнктов, таких, что $f(x_1,\ldots,x_n)=\exists y_1\ldots\exists y_N F(x_1,\ldots,x_n,y_1,\ldots,y_N)$. (Таким образом, f выполнима тогда и только тогда, когда выполнима F.)

40. [*23*] Постройте для данного неориентированного графа G следующие дизъюнкты на булевых переменных $\{p_{uv}\mid u\neq v\}\cup\{q_{uvw}\mid u\neq v,\,u\neq w,\,v\neq w,\,u\not\perp w\}$, где u, v и w означают вершины G:

$$A=\bigwedge\{(p_{uv}\vee p_{vu})\wedge(\bar p_{uv}\vee\bar p_{vu})\mid u\neq v\};$$
$$B=\bigwedge\{(\bar p_{uv}\vee\bar p_{vw}\vee p_{uw})\mid u\neq v,\,u\neq w,\,v\neq w\};$$
$$C=\bigwedge\{(\bar q_{uvw}\vee p_{uv})\wedge(\bar q_{uvw}\vee p_{vw})\wedge(q_{uvw}\vee\bar p_{uv}\vee\bar p_{vw})\mid u\neq v,\,u\neq w,\,v\neq w,\,u\not\perp w\};$$
$$D=\bigwedge\{(\textstyle\bigvee_{v\notin\{u,w\}}(q_{uvw}\vee q_{wvu}))\mid u\neq w,\,u\not\perp w\}.$$

Докажите, что формула $A\wedge B\wedge C\wedge D$ выполнима тогда и только тогда, когда G имеет гамильтонов путь. *Указание*: рассмотрите p_{uv} как утверждение '$u<v$'.

41. [*20*] (*Принцип голубиного гнезда.*) На острове Перистери имеется m голубей и n гнезд. Найдите конъюнктивную нормальную форму, которая выполнима тогда и только тогда, когда каждый голубь может сам занимать как минимум одно гнездо.

42. [*20*] Найдите короткую невыполнимую конъюнктивную нормальную форму, которая не является тривиальной, хотя и состоит полностью из дизъюнктов Хорна, являющихся одновременно дизъюнктами Крома.

43. [*20*] Существует ли эффективный способ выяснить выполнимость конъюнктивной нормальной формы, полностью состоящей из дизъюнктов Хорна и/или дизъюнктов Крома (возможно, смешанных)?

44. [*M23*] Завершите доказательство теоремы H, рассматривая следствия (33).

45. [*M20*] (а) Покажите, что ровно половина функций Хорна от n переменных являются определенными. (б) Покажите также, что функций Хорна от n переменных больше, чем монотонных функций от n переменных (за исключением случая $n=0$).

46. [*20*] Какие из 11×11 пар символов `xy` могут следовать одна за другой в контекстно-свободной грамматике (34)?

47. [*20*] Для заданной последовательности отношений $j\prec k$ с $1\le j,k\le n$, как в алгоритме 2.2.3T (топологической сортировки), рассмотрим дизъюнкты

$$x_{j_1}\wedge\cdots\wedge x_{j_t}\Rightarrow x_k\qquad\text{для } 1\le k\le n,$$

где $\{j_1,\ldots,j_t\}$ — множество элементов, такое, что $j_i\prec k$. Сравните поведение алгоритма C над этими дизъюнктами с поведением алгоритма 2.2.3T.

▶ **48.** [*21*] Как протестировать множество дизъюнктов Хорна на выполнимость?

49. [*22*] Покажите, что если и $f(x_1,\ldots,x_n)$, и $g(x_1,\ldots,x_n)$ определяются дизъюнктами Хорна в конъюнктивной нормальной форме, то имеется простой способ проверки выполнения неравенства $f(x_1,\ldots,x_n)\le g(x_1,\ldots,x_n)$ для всех $x_1,\ldots,x_n$.

50. [*HM42*] Существует $(n+2)2^{n-1}$ возможных дизъюнктов Хорна от n переменных. Выберем $c \cdot 2^n$ из них случайным образом, с разрешенными повторениями, где $c > 0$; пусть также $P_n(c)$ — вероятность того, что все выбранные дизъюнкты одновременно выполнимы. Докажите, что

$$\lim_{n\to\infty} P_n(c) \;=\; 1-(1-e^{-c})(1-e^{-2c})(1-e^{-4c})(1-e^{-8c})\ldots.$$

▶ **51.** [*22*] Большое количество игр с двумя игроками может быть определено с помощью ориентированного графа, в котором каждая вершина представляет позицию игры. Имеется два игрока, Алиса и Боб, которые строят ориентированный путь, начиная с определенной вершины, и ходы которых состоят в поочередном добавлении к пути одной дуги. Перед началом игры каждая вершина помечена либо буквой A (означающей победу Алисы), либо буквой B (означающей победу Боба), либо буквой C, или остается непомеченной.

Когда путь достигает вершины v, помеченной буквой A или B, этот игрок выигрывает. Игра заканчивается без победителя, если достигнутая на этом ходу вершина v уже посещалась ранее на ходу того же игрока. Если v помечена буквой C, текущий активный игрок может выбрать ничью; в противном случае он должен выбрать исходящую дугу, продолжающую путь, и активным становится другой игрок. (Если v представляет собой непомеченную вершину с нулевой исходящей степенью, активный игрок проигрывает.)

Назначив четыре переменные суждений, $A^+(v)$, $A^-(v)$, $B^+(v)$ и $B^-(v)$, каждой вершине v графа, поясните, как построить множество определенных дизъюнктов Хорна, таких, что $A^+(v)$ находится в ядре тогда и только тогда, когда Алиса может обеспечить победу в случае начала пути из вершины v, и если она ходит первой; $A^-(v)$ находится в ядре тогда и только тогда, когда Боб может обеспечить ее проигрыш в данной игре; $B^+(v)$ и $B^-(v)$ аналогичны $A^+(v)$ и $A^-(v)$, но с обменянными ролями игроков.

52. [*25*] (*Булевы игры.*) Любая булева функция $f(x_1,\ldots,x_n)$ приводит к игре под названием "два шага вперед или один шаг назад" следующим образом: имеются два игрока, 0 и 1, которые многократно присваивают значения переменным x_j; игрок y пытается сделать значение $f(x_1,\ldots,x_n)$ равным y. Изначально все переменные не имеют значений, а маркер позиции m равен нулю. Игроки делают ходы, и текущий активный игрок устанавливает либо $m \leftarrow m+2$ (если $m+2 \le n$), либо $m \leftarrow m-1$ (если $m-1 \ge 1$), а затем устанавливает

$$\begin{cases} x_m \leftarrow 0 \text{ или } 1, & \text{если } x_m \text{ ранее не было присвоено;} \\ x_m \leftarrow \bar{x}_m, & \text{если } x_m \text{ уже имеет присвоенное значение.} \end{cases}$$

Игра завершается, когда значения присвоены всем переменным; победителем становится игрок $f(x_1,\ldots,x_n)$. Ничья объявляется в том случае, когда некоторое состояние (включая значение m) достигается дважды. Обратите внимание, что в любой момент возможны не более четырех ходов.

Изучите примеры этой игры при $2 \le n \le 9$ для следующих четырех случаев:

а) $f(x_1,\ldots,x_n) = [x_1\ldots x_n < x_n\ldots x_1]$ (в лексикографическом порядке);
б) $f(x_1,\ldots,x_n) = x_1 \oplus \cdots \oplus x_n$;
в) $f(x_1,\ldots,x_n) = [x_1\ldots x_n$ не содержит двух последовательных единиц$]$;
г) $f(x_1,\ldots,x_n) = [(x_1\ldots x_n)_2$ — простое число$]$.

53. [*23*] Покажите, что невозможное расписание фестиваля из (37) *становится возможным*, если изменения вносятся в требования одного лишь (а) Tomlin; (б) Unwin; (в) Vegas; (г) Xie; (д) Yankovic; (е) Zany.

54. [*20*] Пусть $S = \{u_1, u_2, \ldots, u_k\}$ представляет собой множество литералов в некотором сильно связном компоненте ориентированного графа, который соответствует 2CNF-формуле, как показанный на рис. 6. Покажите, что S содержит как переменную, так и ее дополнение тогда и только тогда, когда $u_j = \bar{u}_1$ для некоторого j с $2 \le j \le k$.

▶ **55.** [*30*] Назовем $f(x_1,\ldots,x_n)$ *переименованной функцией Хорна*, если существуют булевы константы y_1, …, y_n, такие, что $f(x_1\oplus y_1,\ldots,x_n\oplus y_n)$ является функцией Хорна.

а) Для заданной функции $f(x_1,\ldots,x_n)$ в конъюнктивной нормальной форме поясните, как построить $g(y_1,\ldots,y_n)$ в форме 2CNF так, чтобы дизъюнкты функции $f(x_1\oplus y_1,\ldots,x_n\oplus y_n)$ являлись дизъюнктами Хорна тогда и только тогда, когда $g(y_1,\ldots,y_n)=1$.

б) Разработайте алгоритм, который за $O(m)$ шагов выясняет, могут ли все дизъюнкты данной CNF длиной m быть преобразованы в дизъюнкты Хорна путем применения дополнения к некоторому подмножеству переменных.

▶ **56.** [*20*] Задача выполнимости булевой функции $f(x_1,x_2,\ldots,x_n)$ формально может быть сформулирована как вопрос о том, является ли истинной формула с кванторами

$$\exists x_1\,\exists x_2\ldots\exists x_n\,f(x_1,x_2,\ldots,x_n);$$

здесь '$\exists x_j\,\alpha$' означает "существует булево значение x_j, такое, что выполняется α".

Гораздо более общая задача получается при замене кванторов существования $\exists x_j$ кванторами всеобщности $\forall x_j$, где '$\forall x_j\,\alpha$' означает "для любых булевых значений x_j выполняется α".

Какие из восьми квантифицированных формул $\exists x\,\exists y\,\exists z\,f(x,y,z)$, $\exists x\,\exists y\,\forall z\,f(x,y,z)$, …, $\forall x\,\forall y\,\forall z\,f(x,y,z)$ истинны, если $f(x,y,z)=(x\vee y)\wedge(\bar x\vee z)\wedge(y\vee\bar z)$?

▶ **57.** [*30*] (Б. Аспвалл (B. Aspvall), М. Ф. Пласс (M. F. Plass) и Р. Э. Таржан (R. E. Tarjan).) Продолжая выполнение упр. 56, разработайте алгоритм, который за линейное время выясняет, является ли истинной данная полностью квантифицированная формула $f(x_1,\ldots,x_n)$, где f представляет собой произвольную 2CNF-формулу (произвольную конъюнкцию дизъюнктов Крома).

▶ **58.** [*37*] Продолжая выполнять упр. 57, разработайте эффективный алгоритм, выясняющий, является ли данная полностью квантифицированная конъюнкция дизъюнктов *Хорна* истинной.

▶ **59.** [*M20*] (Д. Пехушек (D. Pehoushek) и Р. Фраер (R. Fraer), 1997.) Покажите, что если в таблице истинности для $f(x_1,x_2,\ldots,x_n)$ единица имеется ровно в k местах, то ровно k из полностью квантифицированных формул $\mathrm{Q}x_1\,\mathrm{Q}x_2\ldots\mathrm{Q}x_n\,f(x_1,x_2,\ldots,x_n)$, где каждое Q является либо $\exists$, либо $\forall$, истинны.

60. [*12*] Какие из приведенных выражений дают медиану $\langle xyz\rangle$, определенную в (43)? (а) $(x\wedge y)\oplus(y\wedge z)\oplus(x\wedge z)$. (б) $(x\vee y)\oplus(y\vee z)\oplus(x\vee z)$. (в) $(x\oplus y)\wedge(y\oplus z)\wedge(x\oplus z)$. (г) $(x\equiv y)\oplus(y\equiv z)\oplus(x\equiv z)$. (д) $(x\barwedge y)\wedge(y\barwedge z)\wedge(x\barwedge z)$. (е) $(x\barwedge y)\vee(y\barwedge z)\vee(x\barwedge z)$.

61. [*13*] Истинно или ложно утверждение: если $\circ$ представляет собой любой из булевых бинарных операторов из табл. 1, то справедлив закон дистрибутивности $w\circ\langle xyz\rangle=\langle(w\circ x)(w\circ y)(w\circ z)\rangle$.

62. [*25*] (К. Шенстед (C. Schensted).) Докажите, что если $f(x_1,\ldots,x_n)$ представляет собой монотонную булеву функцию и $n\ge 3$, то

$$f(x_1,\ldots,x_n)=\bigl\langle f(x_1,x_1,x_3,x_4,\ldots,x_n)\,f(x_1,x_2,x_2,x_4,\ldots,x_n)\,f(x_3,x_2,x_3,x_4,\ldots,x_n)\bigr\rangle.$$

63. [*20*] Уравнение (49) показывает, как вычислить медиану пяти элементов с помощью медианы трех элементов. Можем ли мы, наоборот, вычислить $\langle xyz\rangle$ с помощью подпрограммы, вычисляющей медиану пяти элементов?

64. [*23*] (Ш. Б. Акерс-мл. (S. B. Akers, Jr.)) (а) Докажите, что булева функция $f(x_1,\ldots,x_n)$ монотонна и самодуальна тогда и только тогда, когда она удовлетворяет следующему условию.

Для всех $x=x_1\ldots x_n$ и $y=y_1\ldots y_n$ существует k, такое, что $f(x)=x_k$ и $f(y)=y_k$.

(б) Предположим, что f не определена для некоторых значений, но указанное условие выполняется, когда и $f(x)$, и $f(y)$ определены. Покажите, что существует монотонная самодуальная булева функция g, для которой $g(x) = f(x)$, когда $f(x)$ определена.

▶ **65.** [*M21*] Любое подмножество X множества $\{1, 2, \ldots, n\}$ соответствует бинарному вектору $x = x_1x_2\ldots x_n$ согласно правилу $x_j = [j \in X]$. А любое семейство $\mathcal{F}$ таких подмножеств соответствует булевой функции $f(x) = f(x_1, x_2, \ldots, x_n)$ от n переменных согласно правилу $f(x) = [X \in \mathcal{F}]$. Следовательно, каждое утверждение о семействах подмножеств соответствует утверждению о булевых функциях, и наоборот.

Семейство $\mathcal{F}$ называется *перекрывающимся*, если $X \cap Y \neq \emptyset$, когда $X, Y \in \mathcal{F}$. Перекрывающееся семейство, теряющее это свойство при попытке добавить другое подмножество, называется *максимальным*. Докажите, что $\mathcal{F}$ является максимальным перекрывающимся семейством тогда и только тогда, когда соответствующая булева функция f монотонна и самодуальна.

▶ **66.** [*M25*] *Комитетом* (*coterie*) $\{1, \ldots, n\}$ является семейство $\mathcal{C}$ подмножеств, называющихся *кворумами* (*quorums*), которые обладают следующими свойствами, когда $Q \in \mathcal{C}$ и $Q' \in \mathcal{C}$: (i) $Q \cap Q' \neq \emptyset$; (ii) из $Q \subseteq Q'$ следует $Q = Q'$. Комитет $\mathcal{C}$ *доминирует* (*dominates*) над комитетом $\mathcal{C}'$, если $\mathcal{C} \neq \mathcal{C}'$ и если для каждого $Q' \in \mathcal{C}'$ имеется $Q \in \mathcal{C}$, такой, что $Q \subseteq Q'$. Например, над комитетом $\{\{1, 2\}, \{2, 3\}\}$ доминирует $\{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$, а также $\{\{2\}\}$. [Комитеты были введены в классических статьях L. Lamport, *CACM* **21** (1978), 558–565; H. Garcia-Molina and D. Barbara, *JACM* **32** (1985), 841–860. Они имеют массу приложений в протоколах распределенных систем, включая взаимоисключения, репликацию данных и сервера имен. В этих приложениях $\mathcal{C}$ предпочитается любому комитету, над которым он доминирует.]

Докажите, что $\mathcal{C}$ является недоминируемым комитетом тогда и только тогда, когда его кворумами являются множества индексов переменных в простых импликантах монотонной самодуальной булевой функции $f(x_1, \ldots, x_n)$. (Таким образом, в табл. 2 иллюстрируются недоминируемые комитеты множества $\{1, 2, 3, 4\}$.)

▶ **67.** [*M30*] (Д. У. Милнор (J. W. Milnor) и К. Шенстед (C. Schensted).) Треугольная сетка порядка n, показанная здесь для $n = 3$, содержит $(n + 2)(n + 1)/2$ точек с неотрицательными "барицентрическими координатами" xyz, где $x + y + z = n$. Две точки являются смежными, если они отличаются на ± 1 ровно в двух координатных позициях. О точке говорят, что она лежит на стороне x, если ее координата x равна нулю, на стороне y, если ее координата y равна нулю, и на стороне z, если ее координата z равна нулю; таким образом, каждая сторона содержит $n + 1$ точек. Если $n > 0$, точка лежит на двух разных сторонах тогда и только тогда, когда она занимает одну из трех угловых позиций.

"Y" представляет собой связное множество точек как минимум с одной точкой на каждой стороне. Предположим, что каждая вершина треугольной сетки накрывается белым или черным камнем. Например, 52 черных камня в показанной на рисунке справа треугольной сетке образуют (несколько деформированную) Y; но если любой черный заменить белым, то получится белая Y. Если немного подумать, становится интуитивно понятно, что при любом размещении черные камни образуют Y тогда и только тогда, когда белые камни ее не образуют.

Мы можем представить цвет каждого камня булевой переменной со значением 0 для белого и 1 для черного камня. Пусть $Y(t) = 1$ тогда и только тогда, когда существует черная Y, где t — треугольная сетка, включающая все булевы переменные. Ясно, что эта функция Y монотонна; а интуитивно понятное утверждение из предыдущего абзаца

эквивалентно утверждению, что Y, кроме того, еще и самодуальна. Цель данного упражнения — доказать утверждение строго, с применением алгебры медиан.

Пусть для данных $a, b, c \geq 0$ t_{abc} представляет собой треугольную подсетку, содержащую все точки, координаты которых xyz удовлетворяют условиям $x \geq a$, $y \geq b$, $z \geq c$. Например, t_{001} обозначает все точки, за исключением тех из них, которые находятся на стороне z (нижняя строка). Заметим, что если $a + b + c = n$, t_{abc} является единственной точкой с координатами abc; а в общем случае t_{abc} представляет собой треугольную сетку порядка $n - a - b - c$.

а) Если $n > 0$, пусть t^* — треугольная сетка порядка $n - 1$, определяемая правилом

$$t^*_{xyz} = \langle t_{(x+1)yz} t_{x(y+1)z} t_{xy(z+1)} \rangle \qquad \text{для } x + y + z = n - 1.$$

Докажите, что $Y(t) = Y(t^*)$. [Другими словами, t^* сжимает каждый маленький треугольник камней, беря медиану их цветов. Повторение этого процесса определяет *пирамиду* камней, на вершине которой будет находиться черный камень тогда и только тогда, когда в нижнем слое имеется черная Y. Очень интересно применить этот принцип сжатия к приведенной на рисунке деформированной Y.]

б) Докажите, что, если $n > 0$, $Y(t) = \langle Y(t_{100}) Y(t_{010}) Y(t_{001}) \rangle$.

68. [*46*] Показанная в предыдущем упражнении конфигурация содержит 52 черных камня. Каково наибольшее количество черных камней может быть в такой конфигурации? (Иначе говоря, сколько переменных может быть в простой импликанте функции $Y(t)$?)

▶ **69.** [*M26*] (К. Шенстед (C. Schensted).) В упр. 67 функция Y выражена через медианы. Пусть, наоборот, функция $f(x_1, \ldots, x_n)$ является любой монотонной самодуальной булевой функцией с $m + 1$ простыми импликантами $p_0, p_1, \ldots, p_m$. Докажите, что $f(x_1, \ldots, x_n) = Y(T)$, где T представляет собой любую треугольную сетку порядка $m - 1$, в которой T_{abc} — переменная, общая для p_a и p_{a+b+1}, при $a + b + c = m - 1$. Например, когда $f(w, x, y, z) = \langle xwywz \rangle$, мы имеем $m = 3$ и

$$f(w, x, y, z) = (w \wedge x) \vee (w \wedge y) \vee (w \wedge z) \vee (x \wedge y \wedge z) = Y\begin{pmatrix} & w & \\ & w\ w & \\ x & y & z \end{pmatrix}.$$

▶ **70.** [*M20*] (А. Мейеровиц (A. Meyerowitz), 1989.) Выберем в данной монотонной самодуальной булевой функции $f(x) = f(x_1, \ldots, x_n)$ любую простую импликанту $x_{j_1} \wedge \cdots \wedge x_{j_s}$ и положим

$$g(x) = \big(f(x) \wedge [x \neq t]\big) \vee [x = \bar{t}],$$

где $t = t_1 \ldots t_n$ — битовый вектор с единицами в позициях $\{j_1, \ldots, j_s\}$. Докажите, что $g(x)$ также монотонна и самодуальна. (Заметим, что $g(x)$ эквивалентна $f(x)$, за исключением двух точек, t и $\bar{t}$.)

▶ **71.** [*M21*] Опираясь на аксиомы алгебры медиан (50)–(52), докажите, что длинный закон дистрибутивности (54) является следствием короткого закона (53).

72. [*M22*] Выведите (58)–(60) из законов медиан (50)–(53).

73. [*M32*] (Ш. П. Аванн (S. P. Avann).) Дана алгебра медиан M, отрезки которой определяются в (57), а соответствующий медианный граф определен в (61), и пусть $d(u, v)$ обозначает расстояние от u до v. Пусть также '$[uxv]$' означает утверждение "x лежит на кратчайшем пути от u до v".

а) Докажите, что $[uxv]$ выполняется тогда и только тогда, когда $d(u, v) = d(u, x) + d(x, v)$.

б) Предположим, что $x \in [u\,..\,v]$ и $u \in [x\,..\,y]$, где $x \neq u$ и y — v представляет собой ребро графа. Покажите, что x — u также является ребром.

в) Докажите по индукции по $d(u, v)$, что если $x \in [u\,..\,v]$, то $[uxv]$.

г) И наоборот, докажите, что из $[uxv]$ вытекает $x \in [u\,..\,v]$.

74. [*M21*] Покажите, что в алгебре медиан, когда $w \in [x\,..\,y]$, $w \in [x\,..\,z]$ и $w \in [y\,..\,z]$, то в соответствии с определением (57) $w = \langle xyz \rangle$.

▶ **75.** [*M36*] (М. Шоландер (M. Sholander), 1954.) Предположим, что M представляет собой множество точек с отношением промежуточности "x лежит между u и v", записываемым как $[uxv]$ и удовлетворяющим трем следующим аксиомам.

i) Если $[uvu]$, то $u = v$.
ii) Если $[uxv]$ и $[xyu]$, то $[vyu]$.
iii) Для заданных x, y и z ровно одна точка $w = \langle xyz \rangle$ удовлетворяет условиям $[xwy]$, $[xwz]$ и $[ywz]$.

Цель данного упражнения — доказать, что M является алгеброй медиан.

а) Докажите закон мажоритарности $\langle xxy \rangle = x$ (50).
б) Докажите закон коммутативности $\langle xyz \rangle = \langle xzy \rangle = \cdots = \langle zyx \rangle$ (51).
в) Докажите, что $[uxv]$ тогда и только тогда, когда $x = \langle uxv \rangle$.
г) Докажите, что если $[uxy]$ и $[uyv]$, то $[xyv]$.
д) Докажите, что если $[uxv]$ и $[uyz]$, и $[vyz]$, то $[xyz]$. *Указание*: постройте точки $w = \langle yuv \rangle$, $p = \langle wux \rangle$, $q = \langle wvx \rangle$, $r = \langle pxz \rangle$, $s = \langle qxz \rangle$ и $t = \langle rsz \rangle$.
е) Наконец выведите короткий закон дистрибутивности (53): $\langle \langle xyz \rangle uv \rangle = \langle x \langle yuv \rangle \langle zuv \rangle \rangle$.

76. [*M33*] Выведите аксиомы промежуточности (i)–(iii) из упр. 75, исходя из трех аксиом медиан (50)–(52), полагая $[uxv]$ сокращенной записью для "$x = \langle uxv \rangle$". Не используйте закон дистрибутивности (53). *Указание*: см. упр. 74.

77. [*M28*] Пусть G представляет собой медианный граф, содержащий ребро r — s. Для каждого ребра u — v будем называть *u ранним соседом v* тогда и только тогда, когда r ближе к u, чем к v. Разобьем вершины на "левые" и "правые", где левые вершины ближе к r, чем к s, а правые ближе к s, чем к r. Каждая правая вершина v имеет *ранг*, который представляет собой кратчайшую дистанцию от v до левой вершины. Аналогично каждая левая вершина u имеет ранг $1 - d$, где d равно кратчайшему расстоянию от u до правой вершины. Таким образом, вершина u имеет нулевой ранг, если она смежна с правой вершиной, в противном случае ее ранг отрицателен. Ясно, что вершина r имеет ранг 0, а вершина s имеет ранг 1.

а) Покажите, что каждая вершина с рангом 1 смежна ровно с одной вершиной с рангом 0.
б) Покажите, что множество всех правых вершин выпукло.
в) Покажите, что множество всех вершин с рангом 1 выпукло.
г) Докажите, что шаги I3–I9 подпрограммы I корректно маркируют все вершины с рангом 1 и 2.
д) Докажите корректность алгоритма H.

▶ **78.** [*M26*] Докажите, что если вершина v рассматривается в процессе выполнения алгоритма H на шаге I4 k раз, то граф имеет как минимум 2^k вершин. *Указание*: имеется k способов начать кратчайший путь от v до a; таким образом, в $l(v)$ имеется как минимум k единиц.

▶ **79.** [*M27*] (Р. Л. Грэхем (R. L. Graham).) Порожденный *подграф гиперкуба* представляет собой граф, вершины которого v могут быть помечены битовыми строками $l(v)$ таким образом, что u — v тогда и только тогда, когда $l(u)$ и $l(v)$ отличаются ровно в одной битовой позиции. (Все метки имеют одну и ту же длину.)

а) Один из способов определить подграф гиперкуба с n вершинами состоит в том, чтобы положить $l(v)$ равными бинарному представлению v для $0 \le v < n$. Покажите, что этот подграф имеет ровно $f(n) = \sum_{k=0}^{n-1} \nu(k)$ ребер, где $\nu(k)$ — функция контрольного суммирования (сумма бинарных цифр k).
б) Докажите, что $f(n) \le n\lceil \lg n \rceil/2$.
в) Докажите, что ни один подграф гиперкуба с n вершинами не имеет больше $f(n)$ ребер.

80. [*27*] *Неполный куб* представляет собой "изометрический" подграф гиперкуба, а именно подграф, в котором расстояния между вершинами те же, что и в полном графе. Следовательно, вершины неполного куба могут быть помечены таким способом, что расстояние от u до v будет являться "расстоянием Хэмминга" между $l(u)$ и $l(v)$, а именно $\nu(l(u) \oplus l(v))$. Алгоритм H показывает, что каждый медианный граф является неполным кубом.

а) Найдите порожденный подграф четырехмерного куба, который не является неполным кубом.

б) Приведите пример неполного куба, который не является медианным графом.

81. [*16*] Каждый ли медианный граф двудольный?

82. [*25*] Пусть дана последовательность вершин $(v_0, v_1, \ldots, v_t)$ в графе G. Рассмотрим задачу поиска другой последовательности $(u_0, u_1, \ldots, u_t)$, для которой $u_0 = v_0$, а сумма

$$\big(d(u_0,u_1) + d(u_1,u_2) + \cdots + d(u_{t-1},u_t)\big) \;+\; \big(d(u_1,v_1) + d(u_2,v_2) + \cdots + d(u_t,v_t)\big)$$

минимизирована (здесь $d(u,v)$ означает расстояние от u до v). (Каждую v_k можно рассматривать как запрос ресурса, необходимого в данной вершине; сервер по мере обработки запросов переходит в u_k.) Докажите, что если G — медианный граф, то мы получим оптимальное решение путем выбора $u_k = \langle u_{k-1} v_k v_{k+1} \rangle$ для $0 < k < t$, и $u_t = v_t$.

▶ **83.** [*38*] Обобщая упр. 82, найдите эффективный способ минимизировать в медианном графе

$$\big(d(u_0,u_1) + d(u_1,u_2) + \cdots + d(u_{t-1},u_t)\big) \;+\; \rho\big(d(u_1,v_1) + d(u_2,v_2) + \cdots + d(u_t,v_t)\big)$$

для заданного положительного отношения ρ.

84. [*30*] Напишите программу для поиска всех монотонных самодуальных функций от пяти переменных. Что собой представляют ребра соответствующего медианного графа? (В табл. 2 проиллюстрирован случай четырех переменных.)

▶ **85.** [*M22*] Теорема S говорит о том, что каждая формула в 2CNF соответствует медианному множеству; следовательно, каждый антисимметричный ориентированный граф, такой как показанный на рис. 6, также соответствует медианному множеству. Какой из этих ориентированных графов точно соответствует *приведенному* медианному множеству?

86. [*15*] Если v, w, x, y и z принадлежат медианному множеству X, то будет ли их вычисленная покомпонентно медиана пяти элементов $\langle vwxyz \rangle$ всегда принадлежать X?

87. [*24*] Какая CI-сеть строится доказательством теоремы F для свободного дерева (63)?

88. [*M21*] Мы можем использовать параллельные вычисления для сжатия сети (74) в

позволяя каждому модулю действовать как можно ранее. Докажите, что хотя сеть, построенная в доказательстве теоремы F, может содержать $\Omega(t^2)$ модулей, она всегда требует не более $O(t \log t)$ уровней задержки.

89. [*24*] Докажите, что когда построение (73) добавляет новый кластер модулей для обеспечения выполнения условия $u \to v$ для некоторых литералов u и v, то оно сохраняет все ранее обеспеченные условия $u' \to v'$.

▶ **90.** [*21*] Постройте CI-сеть со входными битами $x_1 \ldots x_t$ и выходными битами $y_1 \ldots y_t$, где $y_1 = \cdots = y_{t-1} = 0$ и $y_t = x_1 \oplus \cdots \oplus x_t$. Попробуйте обойтись $O(\log t)$ уровнями задержки.

91. [*46*] Может ли ретрагирующее отображение для меток каждого медианного графа размерности t быть вычислено с помощью CI-сети, имеющей только $O(\log t)$ уровней задержки? [Этот вопрос возник в связи с вопросом существования асимптотически оптимальных сетей для аналогичной задачи сортировки; см. M. Ajtai, J. Komlós and E. Szemerédi, *Combinatorica* **3** (1983), 1–19.]

92. [*46*] Может ли CI-сеть сортировать n булевых входов с меньшим количеством модулей, чем "чисто" сортирующая сеть, не содержащая инверторов?

93. [*M20*] Докажите, что каждая ретракция X графа G является изометрическим подграфом G (другими словами, что расстояния в X те же, что и в G; см. упр. 80.)

94. [*M21*] Докажите, что каждая ретракция X гиперкуба является множеством медианных меток, если не рассматривать координаты, являющиеся константами для всех $x \in X$.

95. [*M25*] Истинно или ложно утверждение: множество всех выходных данных, генерируемых сравнивающе-инвертирующей сетью, когда входные данные пробегают все возможные битовые строки, всегда является медианным множеством.

96. [*HM25*] Вместо требования, чтобы константы w_1, w_2, ..., w_n и t в (75) были целыми, мы можем позволить им быть произвольными действительными числами. Увеличит ли это количество пороговых функций?

97. [*10*] Какие медианные/мажорирующие функции получаются в (81) при $n = 2$, $w_1 = w_2 = 1$ и $t = -1$, 0, 1, 2, 3 или 4?

98. [*M23*] Докажите, что любая самодуальная пороговая функция может быть выражена в виде

$$f(x_1, x_2, \ldots, x_n) = [v_1 y_1 + \cdots + v_n y_n > 0],$$

где каждое y_j равно либо x_j, либо $\bar{x}_j$. Например, $2x_1 + 3x_2 + 5x_3 + 7x_4 + 11x_5 + 13x_6 \geq 21$ тогда и только тогда, когда $2x_1 + 3x_2 + 5x_3 - 7\bar{x}_4 + 11x_5 - 13\bar{x}_6 > 0$.

▶ **99.** [*20*] (Й. Э. Мезеи (J. E. Mezei), 1961.) Докажите, что

$$\langle\langle x_1 \ldots x_{2s-1}\rangle\, y_1 \ldots y_{2t-2}\rangle = \langle x_1 \ldots x_{2s-1} y_1^s \ldots y_{2t-2}^s\rangle.$$

100. [*20*] Истинно или ложно утверждение: если $f(x_1, \ldots, x_n)$ является пороговой функцией, то таковыми же являются и функции $f(x_1, \ldots, x_n) \wedge x_{n+1}$ и $f(x_1, \ldots, x_n) \vee x_{n+1}$.

101. [*M23*] *Пороговая функция Фибоначчи* $F_n(x_1, \ldots, x_n)$ определяется формулой $\langle x_1^{F_1} x_2^{F_2} \ldots x_{n-1}^{F_{n-1}} x_n^{F_{n-2}}\rangle$ при $n \geq 3$; например, $F_7(x_1, \ldots, x_7) = \langle x_1 x_2 x_3^2 x_4^3 x_5^5 x_6^8 x_7^5\rangle$.

а) Что собой представляют простые импликанты $F_n(x_1, \ldots, x_n)$?

б) Найдите ортогональную дизъюнктивную нормальную форму для $F_n(x_1, \ldots, x_n)$ (см. упр. 35).

в) Выразите $F_n(x_1, \ldots, x_n)$ через функцию Y (см. упр. 67 и 69).

102. [*M21*] *Самодуализация* булевой функции определяется формулами

$$\begin{aligned}\hat{f}(x_0, x_1, \ldots, x_n) &= (x_0 \wedge f(x_1, \ldots, x_n)) \vee (\bar{x}_0 \wedge \overline{f(\bar{x}_1, \ldots, \bar{x}_n)}) \\ &= (\bar{x}_0 \vee f(x_1, \ldots, x_n)) \wedge (x_0 \vee \overline{f(\bar{x}_1, \ldots, \bar{x}_n)}).\end{aligned}$$

а) Докажите, что если $f(x_1, \ldots, x_n)$ — любая булева функция, то $\hat{f}$ самодуальна.

б) Докажите, что $\hat{f}$ является пороговой функцией тогда и только тогда, когда f является пороговой функцией.

103. [*HM25*] Поясните, как использовать линейное программирование для проверки по заданному списку простых импликант монотонной самодуальной булевой функции, является ли она пороговой функцией. Поясните также, как, если функция является пороговой, минимизировать размер ее представления в виде функции мажоризации $\langle x_1^{w_1} \ldots x_n^{w_n}\rangle$.

104. [*25*] Примените метод из упр. 103 к поиску кратчайших представлений следующих пороговых функций в виде функций мажоризации:

(а) $\langle x_1^2 x_2^3 x_3^5 x_4^7 x_5^{11} x_6^{13} x_7^{17} x_8^{19} \rangle$;

(б) $[(x_1 x_2 x_3 x_4)_2 \geq t]$ для $0 \leq t \leq 16$ (17 случаев);

(в) $\langle x_1^{29} x_2^{25} x_3^{19} x_4^{15} x_5^{12} x_6^8 x_7^8 x_8^3 x_9^3 x_{10} \rangle$.

105. [*M25*] Покажите, что пороговая функция Фибоначчи из упр. 101 не имеет более краткого представления в виде функции мажоризации, чем то, которое использовалось для ее определения.

▶ **106.** [*M25*] Операция медианы трех элементов, $\langle x\bar{y}\bar{z} \rangle$, является истинной тогда и только тогда, когда $x \geq y + z$.

а) Обобщая, покажите, что условие $(x_1 x_2 \ldots x_n)_2 \geq (y_1 y_2 \ldots y_n)_2 + z$ можно протестировать путем вычисления медианы $2^{n+1} - 1$ булевых переменных.

б) Докажите, что медианы менее чем $2^{n+1} - 1$ элементов недостаточно для решения этой задачи.

107. [*17*] Вычислите $N(f)$ и $\Sigma(f)$ для 16 функций из табл. 1. (См. теорему Т.)

108. [*M21*] Пусть $g(x_0, x_1, \ldots, x_n)$ является самодуальной функцией; таким образом, в обозначениях из теоремы Т $N(g) = 2^n$. Выразите $N(f)$ и $\Sigma(f)$ через $\Sigma(g)$, когда $f(x_1, \ldots, x_n)$ представляет собой (а) $g(0, x_1, \ldots, x_n)$; (б) $g(1, x_1, \ldots, x_n)$.

Рис. 8. Бинарная решетка мажоризации для строк длиной 5. (См. упр. 109.)

109. [*M25*] Говорят, что бинарная строка $\alpha = a_1 \ldots a_n$ *мажоризирует* бинарную строку $\beta = b_1 \ldots b_n$, что записывается как $\alpha \succeq \beta$ или $\beta \preceq \alpha$, если $a_1 + \cdots + a_k \geq b_1 + \cdots + b_k$ для $0 \leq k \leq n$.

а) Пусть $\bar{\alpha} = \bar{a}_1 \ldots \bar{a}_n$. Покажите, что $\alpha \succeq \beta$ тогда и только тогда, когда $\bar{\beta} \succeq \bar{\alpha}$.

б) Покажите, что две любые бинарные строки длиной n имеют наибольшую нижнюю границу $\alpha \wedge \beta$, которая обладает тем свойством, что $\alpha \succeq \gamma$ и $\beta \succeq \gamma$ тогда и только тогда, когда $\alpha \wedge \beta \succeq \gamma$. Поясните, как вычислить $\alpha \wedge \beta$ для заданных α и β.

в) Аналогично поясните, как вычислить наименьшую верхнюю границу $\alpha \vee \beta$, обладающую тем свойством, что $\gamma \succeq \alpha$ и $\gamma \succeq \beta$ тогда и только тогда, когда $\gamma \succeq \alpha \vee \beta$.

г) Истинны или ложны утверждения $\alpha \wedge (\beta \vee \gamma) = (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$; $\alpha \vee (\beta \wedge \gamma) = (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$.

д) Будем говорить, что α *покрывает* β, если $\alpha \succeq \beta$ и $\alpha \neq \beta$ и если из $\alpha \succeq \gamma \succeq \beta$ вытекает, что либо $\gamma = \alpha$, либо $\gamma = \beta$. Например, на рис. 8 показано отношение

покрытия между бинарными строками длиной 5. Найдите простой способ описать строки, покрываемые данной бинарной строкой.

е) Покажите, что каждый путь $\alpha = \alpha_0, \alpha_1, \ldots, \alpha_r = 0\ldots0$ от данной строки α до $0\ldots0$, где α_{j-1} покрывает α_j для $1 \le j \le r$, имеет ту же длину $r = r(\alpha)$.

ж) Пусть $m(\alpha)$ — количество бинарных строк β, таких, что $\beta \succeq \alpha$. Докажите, что $m(1\alpha) = m(\alpha)$ и $m(0\alpha) = m(\alpha) + m(\alpha')$, где α' представляет собой α с крайней слева 1 (если таковая имеется), замененной на 0.

з) Сколько строк α длиной n удовлетворяют условию $\bar\alpha \succeq \alpha$?

110. [*M23*] Булева функция называется *регулярной* (*regular*), если из $x \preceq y$ вытекает, что $f(x) \le f(y)$ для всех векторов x и y, где $\preceq$ является отношением мажоризации из упр. 109. Докажите или опровергните следующие утверждения.

а) Каждая регулярная функция монотонна.

б) Если f представляет собой пороговую функцию (75), для которой $w_1 \ge w_2 \ge \cdots \ge w_n$, функция f регулярна.

в) Если f такая, как в п. (б), и $\Sigma(f) = (s_1, \ldots, s_n)$, то $s_1 \ge s_2 \ge \cdots \ge s_n$.

г) Предположим, что f является чистой функцией мажоризации, а именно пороговой функцией вида (86) с $a = b = 0$. Тогда из $s_1 \ge s_2 \ge \cdots \ge s_n$ следует, что $w_1 \ge w_2 \ge \cdots \ge w_n$.

111. [*M36*] *Оптимальным комитетом* для системы с вероятностями работоспособности $(p_1, \ldots, p_n)$ является комитет, соответствующий монотонной самодуальной функции с максимальной доступностью среди всех монотонных самодуальных функций от n переменных. (См. упр. 14 и 66.)

а) Докажите, что если $1 \ge p_1 \ge \cdots \ge p_n \ge \frac12$, то как минимум одна самодуальная функция с максимальной доступностью является регулярной. Опишите ее.

б) Кроме того, достаточно проверить оптимальность регулярной самодуальной функции f в точках y бинарной решетки мажоризации, для которых $f(y) = 1$, но $f(x) = 0$ для всех x, покрываемых y.

в) Какой комитет оптимален, если некоторые из вероятностей $< \frac12$?

▶ **112.** [*M37*] (Й. Хастад (J. Hastad).) Если $f(x_1, x_2, \ldots, x_m)$ представляет собой булеву функцию, то пусть $M(f)$ является ее представлением в виде мультилинейного полинома с целыми коэффициентами (см. упр. 12). Переставим члены в этом полиноме с использованием последовательности Чейза $\alpha_0 = 00\ldots0$, $\alpha_1 = 10\ldots0$, $\ldots$, $\alpha_{2^m-1} = 11\ldots1$ для упорядочения степеней; последовательность Чейза, получаемая путем конкатенации последовательностей A_{m0}, $A_{(m-1)1}$, $\ldots$, A_{0m} из 7.2.1.3–(35), обладает тем красивым свойством, что α_j идентично α_{j+1} за исключением незначительного изменения, либо $0 \to 1$, либо $01 \to 10$, либо $001 \to 100$, либо $10 \to 01$, либо $100 \to 001$. Например, при $m = 4$ последовательность Чейза имеет вид

$$0000, 1000, 0010, 0001, 0100, 1100, 1010, 1001, 0011, 0101, 0110, 1110, 1101, 1011, 0111, 1111,$$

соответствующий членам 1, x_1, x_3, x_4, x_2, x_1x_2, $\ldots$, $x_2x_3x_4$, $x_1x_2x_3x_4$ соответственно; так что соответствующим представлением, скажем, $((x_1 \oplus \bar x_2) \wedge x_3) \vee (x_1 \wedge \bar x_3 \wedge x_4)$ является

$$x_3 - x_1x_3 + x_1x_4 - x_2x_3 + 2x_1x_2x_3 - x_1x_3x_4,$$

когда члены переставлены в указанном порядке. Пусть теперь

$$F(f) = [\text{старший коэффициент } M(f) \text{ положителен}].$$

Например, старшим (последним) ненулевым членом $((x_1 \oplus \bar x_2) \wedge x_3) \vee (x_1 \wedge \bar x_3 \wedge x_4)$ в упорядочении Чейза является $-x_1x_3x_4$, так что в этом случае $F(f) = 0$.

а) Определите $F(f)$ для каждой из 16 функций в табл. 1.

б) Покажите, что $F(f)$ является пороговой функцией от $n = 2^m$ элементов $\{f_{0\ldots00}, f_{0\ldots01}, \ldots, f_{1\ldots11}\}$ таблицы истинности функции f. Запишите эту функцию явно для $m = 2$.

в) Докажите, что когда m велико, все веса любого порогового представления F должны быть огромными: все их абсолютные значения должны превышать

$$\frac{3^{\binom{m}{3}} 7^{\binom{m}{4}} 15^{\binom{m}{5}} \ldots (2^{m-1}-1)^{\binom{m}{m}}}{n}(1 - O(n^{-1})) \;=\; 2^{mn/2-n-2(3/2)^m/\ln 2+O((5/4)^m)}.$$

Указание: рассмотрите дискретное преобразования Фурье элементов таблицы истинности.

113. [*24*] Покажите, что трех следующих пороговых операций достаточно для вычисления функции $S_{2,3,6,8,9}(x_1, \ldots, x_{12})$ из (91):

$$\begin{aligned} g_1(x_1, \ldots, x_{12}) &= [\nu x \geq 6] = \langle 1 x_1 \ldots x_{12} \rangle; \\ g_2(x_1, \ldots, x_{12}) &= [\nu x - 6 g_1 \geq 2] = \langle 1^3 x_1 \ldots x_{12} \bar{g}_1^6 \rangle; \\ g_3(x_1, \ldots, x_{12}) &= [-2\nu x + 13 g_1 + 7 g_2 \geq 1] = \langle 0^5 \bar{x}_1^2 \ldots \bar{x}_{12}^2 g_1^{13} g_2^7 \rangle. \end{aligned}$$

Найдите также четырехпороговую схему, вычисляющую $S_{1,3,5,8}(x_1, \ldots, x_{12})$.

114. [*20*] (Д. А. Хаффман (D. A. Huffman).) Что собой представляет функция $S_{3,6}(x, x, x, x, y, y, z)$?

115. [*M22*] Поясните, почему (92) корректно вычисляет функцию четности $x_0 \oplus x_1 \oplus \cdots \oplus x_{2m}$.

▶ **116.** [*HM28*] (Б. Данхэм (B. Dunham) и Р. Фридшал (R. Fridshal), 1957.) Рассматривая симметричные функции, можно доказать, что булевы функции от n переменных могут иметь много простых импликант.

а) Допустим, что $0 \leq j \leq k \leq n$. Для каких симметричных функций $f(x_1, \ldots, x_n)$ член $x_1 \wedge \cdots \wedge x_j \wedge \bar{x}_{j+1} \wedge \cdots \wedge \bar{x}_k$ является простой импликантой?

б) Сколько простых импликант имеет функция $S_{3,4,5,6}(x_1, \ldots, x_9)$?

в) Пусть $\hat{b}(n)$ — максимальное количество простых импликант среди всех симметричных булевых функций от n переменных. Найдите рекуррентную формулу для $\hat{b}(n)$ и вычислите $\hat{b}(9)$.

г) Докажите, что $\hat{b}(n) = \Theta(3^n/n)$.

д) Покажите, что, кроме того, существуют симметричные функции $f(x_1, \ldots, x_n)$, такие, что и f, и $\bar{f}$ имеют $\Theta(2^{3n/2}/n)$ простых импликант.

117. [*M26*] Дизъюнктивная нормальная форма называется *безызбыточной* (*irredundant*), если ни одна из ее импликант не вытекает из другой. Пусть $b^*(n)$ — максимальное количество импликант в безызбыточной дизъюнктивной нормальной форме, взятое по всем булевым функциям от n переменных. Найдите простую формулу для $b^*(n)$ и определите ее асимптотическое значение.

118. [*29*] Сколько булевых функций $f(x_1, x_2, x_3, x_4)$ имеют ровно m простых импликант для $m = 0, 1, \ldots$?

119. [*M48*] Продолжим выполнение предыдущего упражнения. Пусть $b(n)$ — максимальное количество простых импликант в булевой функции от n переменных. Ясно, что $\hat{b}(n) \leq b(n) < b^*(n)$; чему равно асимптотическое значение $b(n)$?

120. [*23*] Какой вид имеет кратчайшая дизъюнктивная нормальная форма для симметричных функций (а) $x_1 \oplus x_2 \oplus \cdots \oplus x_n$? (б) $S_{0,1,3,4,6,7}(x_1, \ldots, x_7)$? (в) Докажите, что каждая булева функция от n переменных может быть выражена как дизъюнктивная нормальная форма не более чем с 2^{n-1} простыми импликантами.

▶ **121.** [*M23*] Функция $\langle 1(x_1 \oplus x_2) y_1 y_2 y_3 \rangle$ является частично симметричной, поскольку она симметрична относительно переменных $\{x_1, x_2\}$ и $\{y_1, y_2, y_3\}$, но не относительно всех пяти переменных $\{x_1, x_2, y_1, y_2, y_3\}$.

а) Сколько в точности булевых функций $f(x_1, \ldots, x_m, y_1, \ldots, y_n)$ симметричны относительно $\{x_1, \ldots, x_m\}$ и $\{y_1, \ldots, y_n\}$?

б) Сколько из этих функций монотонны?

в) Сколько из этих функций самодуальны?

г) Сколько из этих функций монотонны и самодуальны?

122. [*M25*] Продолжая выполнять упр. 110 и 121, найдите все булевы функции $f(x_1, x_2, x_3, y_1, y_2, y_3, y_4, y_5, y_6)$, которые одновременно симметричны относительно $\{x_1, x_2, x_3\}$, симметричны относительно $\{y_1, y_2, \ldots, y_6\}$, самодуальны и регулярны. Какие из них являются пороговыми функциями?

123. [*46*] Определите точное количество самодуальных булевых функций от десяти переменных, являющихся пороговыми функциями.

124. [*20*] Найдите булеву функцию от четырех переменных, эквивалентную 767 другим функциям в соответствии с правилами табл. 5.

125. [*18*] Какие из классов функций в (95) являются канализирующими?

126. [*23*] (а) Покажите, что булева функция является канализирующей тогда и только тогда, когда множества ее простых импликант и простых дизъюнктов обладают некоторым простым свойством. (б) Покажите, что булева функция является канализирующей тогда и только тогда, когда ее параметры Чжоу $N(f)$ и $\Sigma(f)$ обладают некоторым простым свойством (см. теорему Т). (в) Определим булевы векторы

$$\vee(f) = \bigvee\{x \mid f(x) = 1\} \qquad \text{и} \qquad \wedge(f) = \bigwedge\{x \mid f(x) = 1\}$$

по аналогии с целочисленным вектором $\Sigma(f)$. Покажите, что решить, является ли функция f канализирующей, можно, имея всего лишь четыре вектора — $\vee(f)$, $\vee(\bar{f})$, $\wedge(f)$ и $\wedge(\bar{f})$.

127. [*M25*] Какие канализирующие функции являются (а) самодуальными? (б) определенными функциями Хорна?

▶ **128.** [*20*] Найдите неканализирующую функцию $f(x_1, \ldots, x_n)$, истинную ровно в двух точках.

129. [*M25*] Сколько существует различных канализирующих функций от n переменных?

130. [*M21*] Согласно табл. 3 имеется 168 монотонных булевых функций от четырех переменных. Но некоторые из них, наподобие $x \wedge y$, зависят только от трех или менее переменных.

а) Сколько монотонных булевых функций от четырех переменных действительно зависят от каждой переменной?

б) Сколько из этих функций различны относительно перестановки переменных, как в табл. 4?

131. [*HM42*] Из табл. 3 очевидно, что имеется гораздо больше функций Хорна, чем функций Крома. Каковы асимптотические количества этих функций при $n \to \infty$?

▶ **132.** [*HM30*] Булева функция $g(x) = g(x_1, \ldots, x_n)$ называется *аффинной*, если она может быть записана в виде $y_0 \oplus (x_1 \wedge y_1) \oplus \cdots \oplus (x_n \wedge y_n) = (y_0 + x \cdot y) \bmod 2$ для некоторых булевых констант y_0, y_1, …, y_n.

а) Покажите, что для произвольной заданной булевой функции $f(x)$ некоторая аффинная функция совпадает с $f(x)$ в $2^{n-1} + 2^{n/2-1}$ или большем количестве точек x. *Указание*: положите $s(y) = \sum_x (-1)^{f(x)+x\cdot y}$ и докажите, что $\sum_y s(y)s(y \oplus z) = 2^{2n}[z = 0 \ldots 0]$ для всех n-битовых векторов z.

б) Булева функция $f(x)$ называется *изогнутой*, если нет ни одной аффинной функции, совпадающей с исходной более чем в $2^{n-1}+2^{n/2-1}$ точках. Докажите, что

$$(x_1 \wedge x_2) \oplus (x_3 \wedge x_4) \oplus \cdots \oplus (x_{n-1} \wedge x_n) \oplus h(x_2, x_4, \ldots, x_n)$$

является изогнутой функцией при четном n и произвольной $h(y_1, y_2, \ldots, y_{n/2})$.

в) Докажите, что $f(x)$ является изогнутой функцией тогда и только тогда, когда

$$\sum_x \bigl(f(x) \oplus f(x \oplus y)\bigr) = 2^{n-1} \qquad \text{для всех } y \neq 0\ldots0.$$

г) Покажите, что если изогнутая функция $f(x_1, \ldots, x_n)$ представлена мультилинейным полиномом по модулю 2, как в (19), то она никогда не содержит член $x_1 \ldots x_r$, где $r > n/2 > 1$.

▶ **133.** [*20*] (Марк Э. Смит (Mark A. Smith), 1990.) Предположим, что мы независимо бросаем n монет, чтобы получить n случайных битов, где k-я монета дает бит 1 с вероятностью p_k. Найдите способ выбрать $(p_1, \ldots, p_n)$ так, чтобы $f(x_1, \ldots, x_n) = 1$ с вероятностью $(t_0 t_1 \ldots t_{2^n-1})_2/(2^{2^n}-1)$, где $t_0 t_1 \ldots t_{2^n-1}$ является таблицей истинности булевой функции f. (Таким образом, n подходящих случайных монет могут генерировать вероятность с 2^n-битовой точностью.)

В общем и целом минимизация коммутационных компонентов перевешивает все иные инженерные соображения при проектировании экономичных логических микросхем.

— Г. А. КЕРТИС (H. A. CURTIS),
Новый подход к проектированию коммутирующих схем
(A New Approach to the Design of Switching Circuits) (1962)

Чтобы преуспевать, приходится быть воистину великим вычислителем. Упрощайте, упрощайте!

— ГЕНРИ Д. ТОРО (HENRY D. THOREAU),
Уальден, или Жизнь в лесу (*Walden; or, Life in the Woods*) (1854)

7.1.2. Булевы вычисления

Наша следующая цель — изучение эффективного вычисления булевых функций, похожее на изучение полиномов в разделе 4.6.4. Естественным способом изучения этой темы является рассмотрение цепочек базовых операций, аналогичных цепочкам полиномов, обсуждавшимся в указанном разделе.

Булева цепочка (*Boolean chain*) для функций от n переменных $(x_1, \ldots, x_n)$ представляет собой последовательность $(x_{n+1}, \ldots, x_{n+r})$, обладающую тем свойством, что каждый последующий шаг объединяет два предыдущих:

$$x_i \;=\; x_{j(i)} \circ_i x_{k(i)} \qquad \text{для } n+1 \le i \le n+r, \tag{1}$$

где $1 \le j(i) < i$ и $1 \le k(i) < i$ и где $\circ_i$ — один из шестнадцати бинарных операторов из табл. 7.1.1–1. Например, при $n = 3$ цепочки

$$\begin{aligned} x_4 &= x_1 \wedge x_2 \\ x_5 &= \bar{x}_1 \wedge x_3 \\ x_6 &= x_4 \vee x_5 \end{aligned} \qquad \text{и} \qquad \begin{aligned} x_4 &= x_2 \oplus x_3 \\ x_5 &= x_1 \wedge x_4 \\ x_6 &= x_3 \oplus x_5 \end{aligned} \tag{2}$$

вычисляют “мультиплексорную” функцию (“if-then-else”) $x_6 = (x_1 ?\ x_2 :\ x_3)$, которая возвращает значение x_2 или x_3 в зависимости от того, равно ли значение x_1 единице (истина) или нулю (ложь).

(Обратите внимание, что примере слева в (2) использована упрощенная запись ‘$x_5 = \bar{x}_1 \wedge x_3$’ для операции НЕ-НО вместо записи в виде‘$x_5 = x_1 \overline{\subset} x_3$’, которая была использована в табл. 7.1.1–1. Главное в том, что независимо от использованных обозначений каждый шаг булевой цепочки представляет собой булеву комбинацию двух предыдущих результатов.)

Булевы цепочки естественным образом соответствуют электронным схемам, при этом каждый шаг цепочки соответствует логическому элементу (“вентилю”), имеющему два входа и один выход. Инженеры-электронщики традиционно представляют булевы цепочки (2) электронными схемами:

(3)

Они должны разрабатывать экономичные схемы, зависящие от различных технологических ограничений; например, некоторые элементы могут быть более дорогими, может потребоваться усиление некоторых выходных сигналов, поступающих на несколько разных входов, может быть выдвинуто требование планарности разводки, какие-то цепи должны быть максимально короткими... Но в этой книге нас интересует программное, а не аппаратное, обеспечение, так что мы не будем беспокоиться обо всех этих вещах. Для наших целей все логические элементы рассматриваются как равноценные с точки зрения стоимости, а выходные сигналы могут использоваться в качестве входных в любом количестве. (Иначе говоря, наши булевы цепочки сводятся к схемам, в которых каждый элемент имеет два входа и неограниченно используемый выход.)*

Кроме того, мы будем изображать булевы цепочки вместо электронных схем наподобие (3) в виде бинарных деревьев, таких, как

(4)

Такие бинарные деревья в случаях, когда промежуточные шаги цепочки используются неоднократно, будут иметь перекрывающиеся поддеревья. Каждый внутренний узел имеет метку, которая представляет собой бинарный оператор; внешние узлы помечены целыми числами k, представляющими переменные x_k. Метка ‘$\textcircled{<}$’ в левом дереве (4) означает оператор НЕ-НО, поскольку $\bar{x} \wedge y = [x < y]$; аналогично оператор НО-НЕ $x \wedge \bar{y}$ может быть представлен меткой узла ‘$\textcircled{>}$’.

* Следует заметить, что не все так просто и в случае программного обеспечения. Даже при указанных автором условиях следует учитывать многие другие факторы. Например, на языке программирования C выражение, выясняющее, является ли данный год `y` високосным, часто записывают следующим образом [см., например, Brian W. Kernighan, Rob Pike. *The Practice of Programming* (2004), p. 7]: `((y%4 == 0) && (y%100 != 0)) || (y%400 == 0)`. Но стóит переписать его как `(y%4 == 0) && ((y%100 != 0) || (y%400 == 0))`, что совершенно эквивалентно с точки зрения получаемого результата, как сокращенные логические вычисления C тут же уменьшают среднее время вычисления почти в четыре раза. — *Примеч. пер.*

Несколько различных булевых цепочек могут иметь одну и ту же диаграмму в виде дерева. Например, левое дерево в (4) представляет также цепочку

$$x_4 = \bar{x}_1 \wedge x_3, \qquad x_5 = x_1 \wedge x_2, \qquad x_6 = x_5 \vee x_4.$$

Любая топологическая сортировка узлов дерева дает эквивалентную цепочку.

Для заданной булевой функции f от n переменных нам часто желательно найти булеву цепочку, такую, что $x_{n+r} = f(x_1, \ldots, x_n)$, где r мало́, насколько это возможно. *Комбинационная сложность* $C(f)$ функции f представляет собой длину кратчайшей цепочки, вычисляющей эту функцию. Для краткости мы будем называть $C(f)$ просто стоимостью f. Мультиплексорная функция в приведенных выше примерах имеет стоимость 3, поскольку исчерпывающим перебором можно показать, что она не может быть получена ни одной булевой цепочкой длиной 2.

DNF- и CNF-представления f, рассматривавшиеся нами в разделе 7.1.1, редко дают какую-то информацию о $C(f)$, поскольку обычно оказываются возможны существенно более эффективные схемы вычисления. Например, в обсуждении после 7.1.1–(30) мы выяснили, что более или менее случайная функция от четырех переменных, таблица истинности которой имеет вид 1100 1001 0000 1111, не имеет DNF-выражения, более краткого, чем

$$(\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \vee (\bar{x}_1 \wedge \bar{x}_3 \wedge \bar{x}_4) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2). \tag{5}$$

Эта формула соответствует булевой цепочке из 10 шагов. Но эта же функция может быть выражена и более понятно как

$$\big(((x_2 \wedge \bar{x}_4) \oplus \bar{x}_3) \wedge \bar{x}_1\big) \oplus x_2, \tag{6}$$

так что ее сложность не превышает 4.

Как можно получать неочевидные формулы наподобие (6)? Мы увидим, что компьютер может находить наилучшие цепочки для функций от четырех переменных без выполнения особо большого количества работы. Тем не менее результат может быть достаточно удивительным даже для людей с немалым опытом работы с булевой алгеброй. Типичный пример этого явления можно увидеть на рис. 9, иллюстрирующем функции от четырех переменных, которые, пожалуй, представляют наибольший интерес, а именно функции, симметричные относительно перестановки их переменных.

Рассмотрим, например, функцию $S_2(x_1, x_2, x_3, x_4)$, для которой имеем

$$\begin{array}{rl}
x_1 & 0000\ 0000\ 1111\ 1111 \\
x_2 & 0000\ 1111\ 0000\ 1111 \\
x_3 & 0011\ 0011\ 0011\ 0011 \\
x_4 & 0101\ 0101\ 0101\ 0101 \\
x_5 = x_1 \oplus x_3 & 0011\ 0011\ 1100\ 1100 \\
x_6 = x_1 \oplus x_2 & 0000\ 1111\ 1111\ 0000 \\
x_7 = x_3 \oplus x_4 & 0110\ 0110\ 0110\ 0110 \\
x_8 = x_5 \vee x_6 & 0011\ 1111\ 1111\ 1100 \\
x_9 = x_6 \oplus x_7 & 0110\ 1001\ 1001\ 0110 \\
x_{10} = x_8 \wedge \bar{x}_9 & 0001\ 0110\ 0110\ 1000
\end{array} \tag{7}$$

в соответствии с рис. 9. Здесь же приведена таблица истинности, так что мы легко можем проверить каждый шаг вычислений. Шаг x_8 дает функцию, истинную при

Рис. 9. Оптимальные булевы цепочки симметричных функций от четырех переменных.

$x_1 \neq x_2$ или $x_1 \neq x_3$; а $x_9 = x_1 \oplus x_2 \oplus x_3 \oplus x_4$ представляет собой функцию четности $(x_1 + x_2 + x_3 + x_4) \bmod 2$. Следовательно, конечный результат, x_{10}, истинен только тогда, когда ровно две из переменных $\{x_1, x_2, x_3, x_4\}$ равны 1; это те случаи, которые выполняют x_8 и имеют четную четность.

Некоторые из других вычислительных схем на рис. 9 также могут быть пояснены интуитивно. Но некоторые из цепочек, такие как $S_{1,4}$, оказываются весьма впечатляющими.

Обратите внимание, что в (7) дважды используется промежуточный результат x_6. Фактически для функции $S_2(x_1, x_2, x_3, x_4)$ не существует шестишаговой цепочки, в которой не использовалось бы дважды некоторое промежуточное подвыражение; все кратчайшие алгебраические формулы для S_2, включая красивые симметричные формулы наподобие

$$\big((x_1 \wedge x_2) \vee (x_3 \wedge x_4)\big) \oplus \big((x_1 \vee x_2) \wedge (x_3 \vee x_4)\big), \tag{8}$$

имеют стоимость 7. Но на рис. 9 показано, что все другие симметричные функции от четырех переменных могут быть вычислены оптимальным образом с помощью “чистых” бинарных деревьев без перекрытия поддеревьев, за исключением внешних узлов (которые представляют переменные).

В общем случае, если $f(x_1, \ldots, x_n)$ представляет собой произвольную булеву функцию, мы говорим, что ее *длина* $L(f)$ равна количеству бинарных операторов в кратчайшей формуле для f. Очевидно, что $L(f) \geq C(f)$; мы можем легко убедиться, что $L(f) = C(f)$ при $n \leq 3$, рассматривая четырнадцать базовых типов

функций от трех переменных из 7.1.1–(95). Но мы только что видели, что $L(S_2) = 7$ превышает $C(S_2) = 6$ при $n = 4$. И действительно, $L(f)$ почти всегда существенно больше, чем $C(f)$, при больши́х n (см. упр. 49).

Глубина $D(f)$ булевой функции f представляет собой другую важную меру присущей ей сложности: мы говорим, что глубина булевой цепочки равна длине самого длинного нисходящего пути в ее дереве, и $D(f)$ равна минимально достижимой длине при рассмотрении всех булевых цепочек для f. Все цепочки, показанные на рис. 9, имеют не только минимальную стоимость, но и минимальную глубину — за исключением случаев $S_{2,3}$ и $S_{1,2}$, где мы не можем одновременно достичь стоимости 6 и глубины 3. Формула

$$S_{2,3}(x_1, x_2, x_3, x_4) = \big((x_1 \wedge x_2) \oplus (x_3 \wedge x_4)\big) \vee \big((x_1 \vee x_2) \wedge (x_3 \oplus x_4)\big) \tag{9}$$

показывает, что $D(S_{2,3}) = 3$; подобная формула работает и для $S_{1,2}$.

Оптимальные цепочки для $n = 4$. Исчерпывающие вычисления для функций от 4 переменных вполне реальны, поскольку такие функции имеют только $2^{16} = 65\,536$ возможных таблиц истинности. В действительности необходимо рассмотреть только половину этих таблиц, так как дополнение $\bar{f}$ любой функции f имеет ту же стоимость, длину и глубину, что и сама f.

Будем говорить, что $f(x_1, \ldots, x_n)$ является *нормальной* (*normal*), если $f(0, \ldots, 0) = 0$, и, в общем случае, что

$$f(x_1, \ldots, x_n) \oplus f(0, \ldots, 0) \tag{10}$$

представляет собой "нормализацию" f. Любая булева цепочка может быть нормализована путем нормализации каждого из ее шагов и соответствующих изменений ее операторов; если $(\hat{x}_1, \ldots, \hat{x}_{i-1})$ являются нормализациями $(x_1, \ldots, x_{i-1})$ и если $x_i = x_{j(i)} \circ_i x_{k(i)}$, как в (1), то $\hat{x}_i$, очевидно, является бинарной функцией от $\hat{x}_{j(i)}$ и $\hat{x}_{k(i)}$. (В упр. 7 представлен соответствующий пример.) Следовательно, без потери общности можно ограничиться рассмотрением нормальных булевых цепочек.

Заметим, что булева цепочка является нормальной тогда и только тогда, когда каждый из ее бинарных операторов $\circ_i$ нормален. Имеется только восемь нормальных бинарных операторов, три из которых, а именно $\perp$, $\llcorner$ и $\lrcorner$, тривиальны. Так что можно считать, что все интересующие нас булевы цепочки образованы пятью операторами, $\wedge$, $\overline{\subset}$, $\overline{\supset}$, $\vee$ и $\oplus$, которые на рис. 9 обозначаются соответственно как $\textcircled{\wedge}$, $\textcircled{<}$, $\textcircled{>}$, $\textcircled{\vee}$ и $\textcircled{\oplus}$. Кроме того, мы можем считать, что на каждом шаге $j(i) < k(i)$.

Имеется $2^{15} = 32\,768$ нормальных функций от четырех переменных, и их длины можно без сложности вычислить путем систематического перечисления всех функций длиной 0, 1, 2 и т. д. В самом деле, из $L(f) = r$ вытекает, что $f = g \circ h$ для некоторых g и h, где $L(g) + L(h) = r - 1$, а $\circ$ представляет собой один из пяти нетривиальных нормальных операторов; так что мы можем действовать следующим образом.

Алгоритм L (*Поиск нормальных длин*). Этот алгоритм определяет $L(f)$ для всех нормальных таблиц истинности $0 \le f < 2^{2^n - 1}$ путем построения списков всех ненулевых нормальных функций длиной $r \ge 0$.

L1. [Инициализация.] Пусть $L(0) \leftarrow 0$ и $L(f) \leftarrow \infty$ для $1 \le f < 2^{2^n-1}$. Затем для $1 \le k \le n$ установить $L(x_k) \leftarrow 0$ и поместить x_k в список 0, где

$$x_k \;=\; (2^{2^n}-1)/(2^{2^{n-k}}+1) \tag{11}$$

представляет собой таблицу истинности для x_k (см. упр. 8.) Наконец, установить $c \leftarrow 2^{2^n-1}-n-1$; c — количество мест, где $L(f)=\infty$.

L2. [Цикл по r.] Выполнить шаг L3 для $r = 1, 2, \ldots$; в конечном счете алгоритм будет завершен, когда c станет равным 0.

L3. [Цикл по j и k.] Выполнять шаг L4 для $j = 0, 1, \ldots$ и $k = r-1-j$, пока $j \le k$.

L4. [Цикл по g и h.] Выполнить шаг L5 для всех g в списке j и всех h в списке k. (Если $j = k$, достаточно ограничить h функциями, которые *следуют после* g в списке k.)

L5. [Цикл по f.] Выполнить шаг L6 для $f = g \,\&\, h$, $f = \bar g \,\&\, h$, $f = g \,\&\, \bar h$, $f = g \mid h$ и $f = g \oplus h$. (Здесь $g \,\&\, h$ означает побитовое И над целыми числами g и h; мы представляем таблицы истинности целыми числами в бинарной записи.)

L6. [Функция f новая?] Если $L(f)=\infty$, установить $L(f) \leftarrow r$, $c \leftarrow c-1$ и поместить f в список r. Завершить работу алгоритма, если $c = 0$. ▮

В упр. 10 показано, как подобная процедура вычисляет все длины $D(f)$.

Ценой небольшой дополнительной работы можно модифицировать алгоритм L так, чтобы он находил лучшую верхнюю границу $C(f)$, вычисляя эвристический битовый вектор $\phi(f)$, именуемый отпечатком (footprint) f. Нормальная булева цепочка может начинаться только $5\binom{n}{2}$ различными способами, поскольку первый шаг x_{n+1} должен быть либо $x_1 \wedge x_2$, либо $\bar x_1 \wedge x_2$, либо $x_1 \wedge \bar x_2$, либо $x_1 \vee x_2$, либо $x_1 \oplus x_2$, либо $x_1 \wedge x_3$, либо $\cdots$, либо $x_{n-1} \oplus x_n$. Допустим, что $\phi(f)$ — битовый вектор длиной $5\binom{n}{2}$, а $U(f)$ — верхняя граница $C(f)$ со следующим свойством: каждый единичный бит в $\phi(f)$ соответствует первому шагу некоторой булевой цепочки, вычисляющей f за $U(f)$ шагов.

Такие пары $(U(f), \phi(f))$ могут быть вычислены путем расширения базовой стратегии алгоритма L. Изначально мы устанавливаем $U(f) \leftarrow 1$ и устанавливаем $\phi(f)$ равным соответствующему вектору $0\ldots010\ldots0$, для всех функций f стоимостью 1. Затем, для $r = 2, 3, \ldots$, мы продолжаем просмотр функций $f = g \circ h$, где $U(g) + U(h) = r-1$, как и ранее, но с двумя изменениями: (1) если отпечатки g и h имеют как минимум один общий элемент, а именно если $\phi(g) \,\&\, \phi(h) \ne 0$, то мы знаем, что $C(f) \le r-1$, так что мы можем уменьшить $U(f)$, если оно было $\ge r$. (2) Если стоимость $g \circ h$ равна (но не меньше) нашей текущей верхней границе $U(f)$, мы можем установить $\phi(f) \leftarrow \phi(f) \mid (\phi(g) \mid \phi(h))$, если $U(f) = r$, и $\phi(f) \leftarrow \phi(f) \mid (\phi(g) \,\&\, \phi(h))$, если $U(f) = r-1$. Детали вы можете найти в упр. 11.

Оказывается, что эта “отпечаточная” эвристика достаточно мощная, чтобы находить цепочки оптимальной стоимости $U(f) = C(f)$ для всех функций f, когда $n = 4$. Более того, позже мы увидим, что отпечатки также помогают решать и более сложные задачи, связанные с вычислениями.

Согласно табл. 7.1.1–5, $2^{16} = 65\,536$ функций от четырех переменных принадлежат всего лишь 222 различным классам, если игнорировать мелкие отличия, связанные с перестановкой переменных и/или дополнением значений. Алгоритм L и его варианты приводят к итоговой статистике, показанной в табл. 1.

Таблица 1

Количества функций от четырех переменных с данной сложностью

$C(f)$	Классы	Функции	$L(f)$	Классы	Функции	$D(f)$	Классы	Функции
0	2	10	0	2	10	0	2	10
1	2	60	1	2	60	1	2	60
2	5	456	2	5	456	2	17	1458
3	20	2474	3	20	2474	3	179	56456
4	34	10624	4	34	10624	4	22	7552
5	75	24184	5	75	24184	5	0	0
6	72	25008	6	68	24640	6	0	0
7	12	2720	7	16	3088	7	0	0

***Вычисления с минимальной памятью.** Предположим, что булевы значения x_1, $\ldots$, x_n находятся в n регистрах, и мы хотим вычислить функцию путем выполнения последовательности операций вида

$$x_{j(i)} \;\leftarrow\; x_{j(i)} \circ_i x_{k(i)} \qquad \text{для } 1 \le i \le r, \tag{12}$$

где $1 \le j(i) \le n$ и $1 \le k(i) \le n$, а $\circ_i$ является бинарным оператором. В конце вычислений искомое значение функции должно находиться в одном из регистров. Например, при $n = 3$ четырехшаговая последовательность

$$\begin{array}{ll} x_1 \leftarrow x_1 \oplus x_2 & (x_1 = 00001111 \quad x_2 = 00110011 \quad x_3 = 01010101) \\ x_3 \leftarrow x_3 \wedge x_1 & (x_1 = 00111100 \quad x_2 = 00110011 \quad x_3 = 01010101) \\ x_2 \leftarrow x_2 \wedge \bar{x}_1 & (x_1 = 00111100 \quad x_2 = 00110011 \quad x_3 = 00010100) \\ x_3 \leftarrow x_3 \vee x_2 & (x_1 = 00111100 \quad x_2 = 00000011 \quad x_3 = 00010100) \\ & (x_1 = 00111100 \quad x_2 = 00000011 \quad x_3 = 00010111) \end{array} \tag{13}$$

вычисляет медиану $\langle x_1 x_2 x_3 \rangle$ и помещает ее в позицию, которую изначально занимало значение x_3. (Все восемь возможных вариантов для содержимых регистров приведены здесь в виде таблиц истинности перед и после каждой операции.)

В действительности мы можем проверить вычисления, работая одновременно только с одной таблицей истинности, вместо того чтобы отслеживать все три, если проанализируем ситуацию в обратном направлении. Пусть $f_l(x_1, \ldots, x_n)$ обозначает функцию, вычисляемую шагами $l, l+1, \ldots, r$ последовательности, с пропущенными первыми $l-1$ шагами; таким образом, в нашем примере $f_2(x_1, x_2, x_3)$ будет представлять собой результат в x_3 после шагов $x_3 \leftarrow x_3 \wedge x_1$, $x_2 \leftarrow x_2 \wedge \bar{x}_1$ и $x_3 \leftarrow x_3 \vee x_2$. Тогда функцией, вычисленной в регистре x_3 всеми четырьмя шагами, будет

$$f_1(x_1, x_2, x_3) \;=\; f_2(x_1 \oplus x_2, x_2, x_3). \tag{14}$$

Аналогично $f_2(x_1, x_2, x_3) = f_3(x_1, x_2, x_3 \wedge x_1)$, $f_3(x_1, x_2, x_3) = f_4(x_1, x_2 \wedge \bar{x}_1, x_3)$, $f_4(x_1, x_2, x_3) = f_5(x_1, x_2, x_3 \vee x_2)$ и $f_5(x_1, x_2, x_3) = x_3$. Следовательно, мы можем пройти в обратном направлении от f_5 до f_4, до $\cdots$ до f_1, соответствующим образом работая с таблицами истинности.

Предположим, например, что $f(x_1, x_2, x_3)$ представляет собой функцию, таблица истинности которой имеет вид

$$t \;=\; a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7\,;$$

тогда таблица истинности для $g(x_1, x_2, x_3) = f(x_1 \oplus x_2, x_2, x_3)$,

$$u \;=\; a_0 a_1 a_6 a_7 a_4 a_5 a_2 a_3\,,$$

получается путем замены a_x на $a_{x'}$, где

$$\text{из} \qquad x = (x_1x_2x_3)_2 \qquad \text{вытекает} \qquad x' = ((x_1 \oplus x_2)x_2x_3)_2.$$

Аналогично таблица истинности для, скажем, $h(x_1, x_2, x_3) = f(x_1, x_2, x_3 \wedge x_1)$, представляет собой

$$v \;=\; a_0a_0a_2a_2a_4a_5a_6a_7.$$

Мы можем использовать побитовые операции для вычисления u и v из t (см. 7.1.3–(83)):

$$u = t \oplus \big((t \oplus (t \gg 4) \oplus (t \ll 4)) \,\&\, (00110011)_2\big); \tag{15}$$

$$v = t \oplus \big((t \oplus (t \gg 1)) \,\&\, (01010000)_2\big). \tag{16}$$

Пусть $C_m(f)$ — длина кратчайшего вычисления f с минимальным использованием памяти. Принцип обратного вычисления говорит нам, что если мы знаем таблицы истинности для всех функций f с $C_m(f) < r$, то можем легко найти все таблицы истинности функций с $C_m(f) = r$. Иначе говоря, мы можем, как и ранее, ограничиться рассмотрением нормальных функций. Тогда для всех нормальных функций g, таких, что $C_m(g) = r - 1$, мы можем построить $5n(n-1)$ таблиц истинности для

$$g(x_1, \ldots, x_{j-1}, x_j \circ x_k, x_{j+1}, \ldots, x_n) \tag{17}$$

и маркировать их стоимостью r, если они еще не были маркированы. В упр. 14 показано, что все эти таблицы истинности могут быть вычислены с помощью простых побитовых операций над таблицей истинности функции g.

Когда $n = 4$, все, кроме 13 из 222 базовых типов функций, имеют $C_m(f) = C(f)$, так что они могут быть вычислены с минимальным использованием памяти без повышения стоимости. В частности, все симметричные функции обладают этим свойством, хотя этот факт и не совершенно очевиден из рис. 9. Пять классов функций имеют $C(f) = 5$, но $C_m(f) = 6$; восемь классов имеют $C(f) = 6$, но $C_m(f) = 7$. Наиболее интересным примером последнего типа, вероятно, является функция $(x_1 \vee x_2) \oplus (x_3 \vee x_4) \oplus (x_1 \wedge x_2 \wedge x_3 \wedge x_4)$, которая имеет стоимость 6 благодаря формуле

$$x_1 \oplus (x_3 \vee x_4) \oplus \big(x_2 \wedge (\bar{x}_1 \vee (x_3 \wedge x_4))\big), \tag{18}$$

но не имеет цепочки с минимальным использованием памяти длиной менее 7 (см. упр. 15).

***Определение минимальной стоимости.** Точное значение $C(f)$ можно найти на основе наблюдения, что все оптимальные булевы цепочки $(x_{n+1}, \ldots, x_{n+r})$ для f очевидным образом удовлетворяют как минимум одному из трех условий:

i) $x_{n+r} = x_j \circ x_k$, где x_j и x_k не используют общих промежуточных результатов;
ii) $x_{n+1} = x_j \circ x_k$, где либо x_j, либо x_k не используется на шагах $x_{n+2}, \ldots, x_{n+r}$;
iii) не выполняется ничто из перечисленного выше, даже при перенумерации промежуточных шагов.

В случае (i) мы имеем $f = g \circ h$, где $C(g) + C(h) = r - 1$, и это можно назвать "нисходящим" построением. В случае (ii) мы имеем $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_{j-1}, x_j \circ x_k, x_{j+1}, \ldots, x_n)$, где $C(g) = r - 1$; назовем это построение "восходящим".

Наилучшие цепочки, которые рекурсивно используют только нисходящие построения, соответствуют минимальной длине формулы, $L(f)$. Наилучшие цепочки, которые рекурсивно используют только восходящие построения, соответствуют вычислениям с минимальным использованием памяти и длиной $C_m(f)$. Мы можем поступить еще лучше, объединяя нисходящее и восходящее построения; но при этом мы не знаем, получили ли мы $C(f)$, поскольку цепочка, принадлежащая случаю (iii), может оказаться еще короче.

К счастью, такие особые цепочки редки, поскольку они должны удовлетворять более строгим условиям, и для не слишком больших n и r их можно полностью перечислить. Например, в упр. 19 доказывается, что при $r < n + 2$ таких цепочек не существует; а когда $n = 4$ и $r = 6$, имеется только 25 существенно различных цепочек, которые не могут быть укорочены явным образом.

Систематически испытывая 5^r вариантов размещения операторов для каждой особой цепочки, по одному для каждого способа назначения нормальных операторов внутренним узлам дерева, мы найдем как минимум одну функцию f в каждом классе эквивалентности, для которой минимальная стоимость $C(f)$ оказывается достижима только в случае (iii).

Фактически при $n = 4$ и $r = 6$ эти $25 \cdot 5^6 = 390\,625$ проб дают только один класс функций, который не может быть вычислен за шесть шагов ни одной нисходящей-плюс-восходящей цепочкой. Этот отсутствующий класс, типизируемый частично симметричной функцией $(\langle x_1 x_2 x_3\rangle \lor x_4) \oplus (x_1 \land x_2 \land x_3)$, может быть вычислен за шесть шагов соответствующей специализацией любой из первых пяти цепочек, показанных выше; например, одним из таких способов является способ

$$x_5 = x_1 \land x_2, \quad x_6 = x_1 \lor x_2, \quad x_7 = x_3 \oplus x_5,$$
$$x_8 = x_4 \land \bar{x}_5, \quad x_9 = x_6 \land x_7, \quad x_{10} = x_8 \lor x_9, \tag{19}$$

соответствующий первой особой цепочке. Поскольку у всех прочих функций длина $L(f) \le 7$, эти вычисления методом проб устанавливают истинную минимальную цену для всех случаев.

Историческая справка. Отчет о первых согласованных попытках оптимального вычисления всех булевых функций $f(w,x,y,z)$ появился в *Annals of the Computation Laboratory of Harvard University* **27** (1951), где группа Говарда Айкена (Howard Aiken) представила эвристические методы и обширные таблицы наилучших коммутационных схем, которые они смогли построить. Их мера стоимости $V(f)$ отличалась от рассматриваемой нами стоимости $C(f)$, так как была основана на "управляющих сетках" электровакуумных ламп: у них было четыре типа логических элементов, НЕ(f), НЕ-И(f,g), ИЛИ$(f_1,\ldots,f_k)$ и И$(f_1,\ldots,f_k)$, соответственно со стоимостью 1, 2, k и 0. Каждый вход логического элемента НЕ, НЕ-И или ИЛИ мог быть либо переменной, либо дополнением переменной, либо результатом предыдущего логического элемента; каждый вход И должен был быть выходом либо НЕ, либо НЕ-И, который не мог использоваться где-либо еще.

При таком критерии функция могла иметь стоимость, отличную от стоимости ее дополнения. Можно было, например, вычислить $x \wedge y$ как И$\bigl($НЕ$(\bar{x})$, НЕ$(\bar{y})\bigr)$ со стоимостью 2; однако стоимость $\bar{x} \vee (\bar{y} \wedge \bar{z})$ = НЕ-И$(x,$ ИЛИ$(y,z))$ была равна 4, в отличие от ее дополнения $x \wedge (y \vee z)$ = И$\bigl($НЕ$(\bar{x})$, НЕ-И$(\bar{y},\bar{z})\bigr)$ со стоимостью 3. Таким образом, исследователям из Гарварда потребовалось рассмотреть 402 существенно разных класса функций от 4 переменных вместо 222 (см. ответ к упр. 7.1.1–125). Конечно, в то время они работали в основном вручную. Они нашли, что во всех случаях $V(f) < 20$, за исключением 64 функций, эквивалентных $S_{0,1}(w,x,y,z) \vee \bigl(S_2(w,x,y) \wedge z\bigr)$, которые они вычисляли с помощью 20 управляющих сеток следующим образом:

$$\begin{aligned}
&g_1 = \text{И}(\text{НЕ}(\bar{w}), \text{НЕ}(\bar{x})),\ \ g_2 = \text{НЕ-И}(\bar{y}, z),\\
&\qquad g_3 = \text{И}(\text{НЕ}(w), \text{НЕ}(x));\\
&f = \text{И}\bigl(\text{НЕ-И}(g_1, g_2), \text{НЕ-И}(g_3, \text{И}(\text{НЕ}(\bar{y}), \text{НЕ}(\bar{z}))),\\
&\qquad\qquad \text{НЕ}(\text{И}(\text{НЕ}(g_3), \text{НЕ}(\bar{y}), \text{НЕ}(z))),\\
&\qquad\qquad\qquad \text{НЕ}(\text{И}(\text{НЕ}(g_1), \text{НЕ}(g_2), \text{НЕ}(g_3)))\bigr).
\end{aligned} \tag{20}$$

Первая компьютерная программа для поиска вероятно оптимальных схем была написана Лео Хеллерманом (Leo Hellerman) [*IEEE Transactions* **EC-12** (1963), 198–223], который искал наименьшее количество логических элементов НЕ-ИЛИ, необходимых для вычисления любой данной функции $f(x,y,z)$. Он потребовал, чтобы каждый вход каждого логического элемента был либо недополненной переменной, либо выходом предыдущего элемента; коэффициенты ветвления входов и выходов ограничивались величиной 3. Если две схемы имели одно и то же количество логических элементов, он отдавал предпочтение схеме с меньшей суммой входов. Например, он вычислял $\bar{x}$ = НЕ-ИЛИ(x) со стоимостью 1; $x \vee y \vee z$ = НЕ-ИЛИ(НЕ-ИЛИ(x,y,z)) со стоимостью 2; $\langle xyz \rangle$ = НЕ-ИЛИ(НЕ-ИЛИ(x,y), НЕ-ИЛИ(x,z), НЕ-ИЛИ(y,z)) со стоимостью 4; $S_1(x,y,z)$ = НЕ-ИЛИ$\bigl($НЕ-ИЛИ(x,y,z), $\langle xyz \rangle\bigr)$ со стоимостью 6; и т. д. Поскольку коэффициент ветвления выходов был ограничен значением 3, он обнаружил, что каждая функция трех переменных может быть вычислена со стоимостью 7 или менее, за исключением функции четности $x \oplus y \oplus z = (x \equiv y) \equiv z$, где $x \equiv y$ имеет стоимость 4, поскольку представляет собой НЕ-ИЛИ(НЕ-ИЛИ$(x,$ НЕ-ИЛИ$(x,y))$, НЕ-ИЛИ$(y,$ НЕ-ИЛИ$(x,y)))$.

Таблица 2

Количества функций от пяти переменных с данной сложностью

$C(f)$	Классы	Функции	$L(f)$	Классы	Функции	$D(f)$	Классы	Функции
0	2	12	0	2	12	0	2	12
1	2	100	1	2	100	1	2	100
2	5	1140	2	5	1140	2	17	5350
3	20	11570	3	20	11570	3	1789	6702242
4	93	109826	4	93	109826	4	614316	4288259592
5	389	995240	5	366	936440	5	0	0
6	1988	8430800	6	1730	7236880	6	0	0
7	11382	63401728	7	8782	47739088	7	0	0
8	60713	383877392	8	40297	250674320	8	0	0
9	221541	1519125536	9	141422	955812256	9	0	0
10	293455	2123645248	10	273277	1945383936	10	0	0
11	26535	195366784	11	145707	1055912608	11	0	0
12	1	1920	12	4423	31149120	12	0	0

Инженеры-электронщики продолжали изучение и других критериев стоимости; функции от четырех переменных казались находящимися вне пределов досягаемости до 1977 года, когда Франклин М. Лианг (Frank M. Liang) установил значения $C(f)$, показанные в табл. 1. Неопубликованное решение Лианга основывалось на изучении всех цепочек, которые не могут быть сокращены восходящим построением.

Случай $n = 5$. Согласно табл. 7.1.1–5 имеется 616 126 классов существенно различных функций $f(x_1, x_2, x_3, x_4, x_5)$. В настоящее время компьютеры достаточно быстры, чтобы не пугаться таких чисел; так что автор решил во время написания этого раздела исследовать $C(f)$ для всех булевых функций от пяти переменных. Небольшая доля везения помогла в действительности получить результаты, приведшие к статистике, показанной в табл. 2.

Для проведения этих вычислений алгоритм L и его варианты были модифицированы для работы с представителями классов, а не со всем множеством 2^{31} нормальных таблиц истинности. Метод из упр. 7.2.1.2–20 упрощает генерацию всех функций данного класса по заданной одной из них, приводя к тысячекратному ускорению работы. Восходящий метод был слегка улучшен, позволяя вывести, например, что $f(x_1 \wedge x_2, x_1 \vee x_2, x_3, x_4, x_5)$ имеет стоимость $\le r$, если $C(f) = r-2$. После того как были найдены все классы со стоимостью 10, нисходящий и восходящий методы оказались способными найти цепочки длиной ≤ 11 для всех, кроме семи, классов функций. После этого началась трудоемкая часть работы, в которой были сгенерированы примерно 53 миллиона особых цепочек с $n = 5$ и $r = 11$; каждая такая цепочка вела к $5^{11} = 48\,828\,125$ функциям, некоторые из них, как следовало надеяться, делятся между семью оставшимися загадочными классами. Но, как выяснилось, только шесть из этих классов имеют 11-шаговые решения. Остался один класс, таблица истинности которого имеет вид `169ae443` в шестнадцатеричной записи, и который оказался единственным классом, у которого $C(f) = 12$ (так же, как и $L(f) = 12$).

Получающиеся в результате конструкции для симметричных функций показаны на рис. 10. Некоторые из них потрясающе красивы; некоторые красиво просты; остальные — просто потрясающи. (Взгляните, например, на 8-шаговое вычисление $S_{2,3}(x_1, x_2, x_3, x_4, x_5)$, или на элегантную формулу для $S_{2,3,4}$, или на немонотонные

Рис. 10. Булевы цепочки минимальной стоимости для симметричных функций от пяти переменных.

цепочки для $S_{4,5}$ и $S_{3,4,5}$.) Кстати, в табл. 2 показано, что все функции от пяти переменных имеют глубину ≤ 4, но на рис. 10 не предпринималось никаких попыток минимизировать глубину.

Оказывается, что все эти симметричные функции могут быть вычислены с минимальным использованием памяти без увеличения стоимости. Никакого простого объяснения этому неизвестно.

Многоканальный вывод. Зачастую необходимо вычислить несколько различных булевых функций $f_1(x_1, \ldots, x_n)$, $\ldots$, $f_m(x_1, \ldots, x_n)$ с одними и теми же входными

значениями $x_1, \ldots, x_n$; другими словами, зачастую требуется вычислить мультибитовую функцию $y = f(x)$, где $y = f_1 \ldots f_m$ представляет собой бинарный вектор длиной m, а $x = x_1 \ldots x_n$ является бинарным вектором длиной n. При везении немалая часть работы, выполняемой при вычислении значения одного компонента $f_j(x_1, \ldots, x_n)$, может быть использована в операциях, требуемых для вычисления значений других компонентов $f_k(x_1, \ldots, x_n)$.

Пусть $C(f) = C(f_1 \ldots f_m)$ — длина кратчайшей булевой цепочки, которая вычисляет все нетривиальные функции f_j. Точнее говоря, цепочка $(x_{n+1}, \ldots, x_{n+r})$ должна обладать тем свойством, что для $1 \le j \le m$ либо $f_j(x_1, \ldots, x_n) = x_{l(j)}$, либо $f_j(x_1, \ldots, x_n) = \bar{x}_{l(j)}$, для некоторого $l(j)$, где $0 \le l(j) \le n + r$, и $x_0 = 0$. Ясно, что $C(f) \le C(f_1) + \cdots + C(f_m)$, но может оказаться, что мы можем добиться гораздо большего.

Предположим, например, что мы хотим вычислить функции z_1 и z_0, определяемые

$$(z_1 z_0)_2 \;=\; x_1 + x_2 + x_3, \tag{21}$$

двухбитовой бинарной суммой трех булевых переменных. Мы имеем

$$z_1 \;=\; \langle x_1 x_2 x_3 \rangle \qquad \text{и} \qquad z_0 \;=\; x_1 \oplus x_2 \oplus x_3, \tag{22}$$

так что отдельные стоимости представляют собой $C(z_1) = 4$ и $C(z_0) = 2$. Но легко увидеть, что объединенная стоимость $C(z_1 z_0)$ не превышает 5, поскольку $x_1 \oplus x_2$ представляет собой подходящий первый шаг в вычислении каждого бита z_j:

$$\begin{aligned} &x_4 = x_1 \oplus x_2, \quad z_0 = x_5 = x_3 \oplus x_4; \\ &x_6 = x_3 \wedge x_4, \quad x_7 = x_1 \wedge x_2, \quad z_1 = x_8 = x_6 \vee x_7. \end{aligned} \tag{23}$$

Кроме того, исчерпывающие вычисления показывают, что $C(z_1 z_0) > 4$; следовательно, $C(z_1 z_0) = 5$.

Инженеры-электронщики традиционно называют схему для вычисления (21) *полным сумматором* (*full adder*), поскольку n таких “кирпичиков” можно объединить вместе для сложения двух n-битовых чисел. Частный случай (22), когда $x_3 = 0$, также важен, хотя и упрощается до

$$z_1 \;=\; x_1 \wedge x_2 \qquad \text{и} \qquad z_0 \;=\; x_1 \oplus x_2 \tag{24}$$

и имеет сложность 2; инженеры называют его полусумматором (half adder), несмотря на тот факт, что стоимость полного сумматора превышает стоимость двух полусумматоров.

Общая задача сложения в бинарной системе счисления

$$\begin{array}{r} (x_{n-1} \ldots x_1 x_0)_2 \\ (y_{n-1} \ldots y_1 y_0)_2 \\ \hline (z_n z_{n-1} \ldots z_1 z_0)_2 \end{array} \tag{25}$$

состоит в вычислении $n + 1$ булевых выходов $z_n \ldots z_1 z_0$ для данных $2n$ булевых входов $x_{n-1} \ldots x_1 x_0 y_{n-1} \ldots y_1 y_0$; она легко решается с помощью формул

$$c_{j+1} \;=\; \langle x_j y_j c_j \rangle, \qquad z_j \;=\; x_j \oplus y_j \oplus c_j \qquad \text{для } 0 \le j < n, \tag{26}$$

где c_j представляют собой “биты переноса”, и мы имеем $c_0 = 0$, $z_n = c_n$. Следовательно, можно использовать полусумматор для вычисления c_1 и z_0, за которым следуют

$n-1$ полных сумматоров для вычисления прочих c и z общей стоимостью $5n-3$. На самом деле, Н. П. Редькин [*Проблемы кибернетики* **38** (1981), 181–216] доказал, что $5n-3$ шагов действительно необходимы, приведя скрупулезное 35-страничное доказательство по индукции, завершающееся случаем 2.2.2.3.1.2.3.2.4.3(!). Однако глубина такой схемы, равная $2n-1$, слишком велика для практических параллельных вычислений, так что большие усилия направлены на разработку схем сложения с глубиной $O(\log n)$ и разумной стоимостью. (См. упр. 41–44.)

Теперь давайте расширим (21) и попытаемся вычислить обобщенную "контрольную сумму" ("sideways sum")

$$(z_{\lfloor \lg n \rfloor} \ldots z_1 z_0)_2 \;=\; x_1 + x_2 + \cdots + x_n. \tag{27}$$

Если $n = 2k+1$, мы можем использовать k полных сумматоров, чтобы свести сумму к $(x_1 + \cdots + x_n) \bmod 2$ плюс k бит с весом 2, поскольку каждый полный сумматор уменьшает количество битов с весом 1 на 2. Например, если $n = 9$ и $k = 4$, вычисление имеет вид

$$\begin{array}{llll} x_{10}{=}x_1{\oplus}x_2{\oplus}x_3, & x_{11}{=}x_4{\oplus}x_5{\oplus}x_6, & x_{12}{=}x_7{\oplus}x_8{\oplus}x_9, & x_{13}{=}x_{10}{\oplus}x_{11}{\oplus}x_{12}, \\ y_1{=}\langle x_1x_2x_3\rangle, & y_2{=}\langle x_4x_5x_6\rangle, & y_3{=}\langle x_7x_8x_9\rangle, & y_4{=}\langle x_{10}x_{11}x_{12}\rangle, \end{array}$$

и мы имеем $x_1 + \cdots + x_9 = x_{13} + 2(y_1 + y_2 + y_3 + y_4)$. Если $n = 2k$ четно, применимо аналогичное приведение, но с полусумматором в конце. Биты с весом 2 могут после этого быть просуммированы таким же образом; так что мы получаем рекуррентное соотношение

$$s(n) \;=\; 5\lfloor n/2 \rfloor - 3[n \text{ четно}] + s(\lfloor n/2 \rfloor), \qquad s(0) = 0, \tag{28}$$

для общего количества логических элементов, необходимых для вычисления значения $z_{\lfloor \lg n \rfloor} \ldots z_1 z_0$. (Аналитическая формула для $s(n)$ приведена в упр. 30.) Мы имеем $s(n) < 5n$, а первые значения

$$\begin{array}{rcccccccccccccccccccc} n &=& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ s(n) &=& 0 & 2 & 5 & 9 & 12 & 17 & 20 & 26 & 29 & 34 & 37 & 44 & 47 & 52 & 55 & 63 & 66 & 71 & 74 & 81 \end{array}$$

показывают, что для небольших n этот метод достаточно эффективен. Например, когда $n = 5$, он дает схему

(29)

которая вычисляет три различные симметричные функции, $z_2 = S_{4,5}(x_1, \ldots, x_5)$, $z_1 = S_{2,3}(x_1, \ldots, x_5)$, $z_0 = S_{1,3,5}(x_1, \ldots, x_5)$, всего за 12 шагов. Вычисление $S_{4,5}$ за 10 шагов является, согласно рис. 10, оптимальным; конечно же, оптимальным является и вычисление $S_{1,3,5}$ за 4 шага. Кроме того, хотя $C(S_{2,3}) = 8$, функция $S_{2,3}$ вычисляется здесь с помощью ясной 10-шаговой процедуры, которая использует совместно с $S_{4,5}$ все логические элементы, кроме одного.

Обратите внимание, что теперь мы можем эффективно вычислить *любую* симметричную функцию, поскольку каждая симметричная функция от $\{x_1, \ldots, x_n\}$ является булевой функцией от $z_{\lfloor \lg n \rfloor} \ldots z_1 z_0$. Мы знаем, например, что любая булева функция от четырех переменных имеет сложность ≤ 7; следовательно, любая симметричная функция $S_{k_1,\ldots,k_t}(x_1, \ldots, x_{15})$ имеет стоимость не более $s(15) + 7 = 62$. Сюрприз: симметричные функции от n переменных были среди наиболее сложных для вычисления при малых n и среди простейших при $n \ge 10$.

Мы также можем эффективно вычислять *множества* симметричных функций. Если мы хотим, скажем, вычислить все $n+1$ симметричных функций $S_k(x_1, \ldots, x_n)$ для $0 \le k \le n$ с помощью одной булевой цепочки, нам нужно просто вычислить первые $n+1$ *минитермов* z_0, z_1, $\ldots$, $z_{\lfloor \lg n \rfloor}$. Например, когда $n = 5$, минитермами, дающими все функции S_k, являются соответственно $S_0 = \bar{z}_0 \wedge \bar{z}_1 \wedge \bar{z}_2$, $S_1 = \bar{z}_0 \wedge \bar{z}_1 \wedge z_2$, $\ldots$, $S_5 = z_0 \wedge \bar{z}_1 \wedge z_2$.

Насколько трудно вычислить все 2^n минитермов от n переменных? Инженеры-электронщики называют эту функцию *бинарным декодером* "n-к-2^n", поскольку он преобразует n бит $x_1 \ldots x_n$ в последовательность 2^n бит $d_0 d_1 \ldots d_{2^n-1}$, ровно один из которых равен 1. Принцип "разделяй и властвуй" предполагает, что сначала мы вычисляем все минитермы для первых $\lceil n/2 \rceil$ переменных, как и все минитермы для последних $\lfloor n/2 \rfloor$ переменных; после этого 2^n логических схем И завершают работу. Стоимость данного метода равна $t(n)$, где

$$t(0) = t(1) = 0; \qquad t(n) = 2^n + t(\lceil n/2 \rceil) + t(\lfloor n/2 \rfloor) \quad \text{для } n \ge 2. \tag{30}$$

Так что $t(n) = 2^n + O(2^{n/2})$; это составляет примерно один логический элемент на минитерм. (См. упр. 32.)

Функции с множественным выводом часто помогают нам при построении бóльших функций с одним выводом. Например, мы видели, что сумматор (27) позволяет нам вычислять симметричные функции; декодер "n-к-2^n" также имеет много применений, несмотря на тот факт, что 2^n может быть огромным при большúх значениях n. Рассмотрим 2^m-*канальный мультиплексор* $M_m(x_1, \ldots, x_m; y_0, y_1, \ldots, y_{2^m-1})$, известный также как m-битная *функция обращения к памяти*, которая имеет $n = m + 2^m$ входов и возвращает значение y_k, когда $(x_1 \ldots x_m)_2 = k$. По определению мы имеем

$$M_m(x_1, \ldots, x_m; y_0, y_1, \ldots, y_{2^m-1}) \;=\; \bigvee_{k=0}^{2^m-1} (d_k \wedge y_k), \tag{31}$$

где d_k представляет собой k-й выход бинарного декодера "m-к-2^m". Таким образом, согласно (30) можно вычислить M_m с помощью $2^m + (2^m - 1) + t(m) = 3n + O(\sqrt{n})$ логических элементов. Однако в упр. 39 показано, что в действительности можно уменьшить стоимость до всего лишь $2n + O(\sqrt{n})$. (См. также упр. 79.)

Асимптотика. Когда количество переменных было малó, примененные нами методы исчерпывающего поиска выявили множество случаев, когда булева функция может быть вычислена с удивительной эффективностью. Казалось бы естественным ожидать, что чем больше переменных имеется в наличии, тем больше имеется и возможностей для оригинальных вычислений. Но на самом деле ситуация оказывается обратной, по крайней мере со статистической точки зрения.

Теорема S. *Стоимость почти каждой булевой функции $f(x_1,\ldots,x_n)$ превышает $2^n/n$. Говоря точнее, если $c(n,r)$ булевых функций имеют сложность $\le r$, то*

$$(r-1)!\,c(n,r) \;\le\; 2^{2r+1}(n+r-1)^{2r}. \tag{32}$$

Доказательство. Если функция может быть вычислена за $r-1$ шагов, то она также вычислима r-шаговой цепочкой. (Это утверждение очевидно при $r=1$; в прочих случаях мы можем положить $x_{n+r} = x_{n+r-1} \wedge x_{n+r-1}$.) Мы покажем, что не существует очень много r-шаговых цепочек, следовательно, мы не можем вычислить очень много различных функций со стоимостью $\le r$.

Пусть π — перестановка $\{1,\ldots,n+r\}$, которая отображает $1\mapsto 1$, $\ldots$, $n\mapsto n$ и $n+r\mapsto n+r$; всего имеется $(r-1)!$ таких перестановок. Предположим, что $(x_{n+1},\ldots,x_{n+r})$ — булева цепочка, в которой каждый из промежуточных шагов $x_{n+1},\ldots,x_{n+r-1}$ использовался как минимум в одном из последующих шагов. Тогда переставленные цепочки, определяемые правилом

$$x_i \;=\; x_{j'(i)} \circ'_i\, x_{k'(i)} \;=\; x_{j(i\pi)\pi^-} \circ_{i\pi}\, x_{k(i\pi)\pi^-} \qquad \text{для } n<i\le n+r, \tag{33}$$

отличаются для различных π. (Если π отображает $a\mapsto b$, мы записываем $b=a\pi$ и $a=b\pi^-$.) Например, если π отображает $5\mapsto 6\mapsto 7\mapsto 8\mapsto 9\mapsto 5$, цепочка (7) превращается в

$$\begin{array}{rcl} \text{Исходная} & & \text{Переставленная} \\ x_5 = x_1 \oplus x_3, & & x_5 = x_1 \oplus x_2, \\ x_6 = x_1 \oplus x_2, & & x_6 = x_3 \oplus x_4, \\ x_7 = x_3 \oplus x_4, & & x_7 = x_9 \vee x_5, \\ x_8 = x_5 \vee x_6, & & x_8 = x_5 \oplus x_6, \\ x_9 = x_6 \oplus x_7, & & x_9 = x_1 \oplus x_3, \\ x_{10} = x_8 \wedge \bar{x}_9; & & x_{10} = x_7 \wedge \bar{x}_8. \end{array} \tag{34}$$

Заметим, что у нас может быть $j'(i)\ge k'(i)$ или $j'(i)>i$, или $k'(i)>i$ в противоположность нашим обычным правилам. Но переставленная цепочка вычисляет ту же функцию x_{n+r}, что и ранее, и не имеет никаких циклов, согласно которым некоторый элемент косвенно определяется через самого себя, поскольку переставленное x_i представляет собой исходное $x_{i\pi}$.

Мы можем ограничиться рассмотрением *нормальных* булевых цепочек, как было замечено ранее. Так что $c(n,r)/2$ нормальных булевых функций стоимостью $\le r$ приводят к $(r-1)!\,c(n,r)/2$ различным переставленным цепочкам, где оператор $\circ_i$ на каждом шаге представляет собой $\wedge$, $\vee$, $\overline{\supset}$ или $\oplus$. Имеется не более $4^r(n+r-1)^{2r}$ таких цепочек, поскольку есть четыре варианта выбора $\circ_i$ и $n+r-1$ вариантов выбора каждого из $j(i)$ и $k(i)$ для $n<i\le n+r$. Отсюда следует уравнение (32); исходное утверждение теоремы мы получаем, полагая $r=\lfloor 2^n/n\rfloor$. (См. упр. 46.) ∎

С другой стороны, для любителей бесконечности есть и хорошие новости: оказывается, в действительности любую булеву функцию от n переменных можно вычислить лишь немного больше чем за $2^n/n$ шагов вычислений (даже если отказаться от применения $\oplus$ и $\equiv$) с помощью метода, разработанного К. Э. Шенноном (C. E. Shannon) и усовершенствованного О. Б. Лупановым [*Bell System Tech. J.* **28** (1949), 59–98, Theorem 6; *Известия ВУЗов, Радиофизика* **1** (1958), 120–140].

Фактически подход Шеннона–Лупанова приводит к полезным результатам даже при малых n, так что давайте познакомимся с ним, рассмотрев небольшой пример.

$$f(x_1, x_2, x_3, x_4, x_5, x_6) \;=\; \big[(x_1x_2x_3x_4x_5x_6)_2 \text{ простое}\big] \tag{35}$$

является функцией, которая идентифицирует все 6-битовые простые числа. В ее таблице истинности содержится $2^6 = 64$ бит, и с ними удобно работать, используя массив размером 4×16 вместо привязки к одномерной строке.

$$\begin{array}{r|cccccccccccccccc|l}
x_3 = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
x_4 = & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \\
x_5 = & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & \\
x_6 = & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \\
\hline
x_1x_2 = 00 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \\
x_1x_2 = 01 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \text{Группа 1} \\
x_1x_2 = 10 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & \\
x_1x_2 = 11 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & \text{Группа 2}
\end{array} \tag{36}$$

Строки делятся на две группы по две строки; каждая группа строк имеет 16 столбцов четырех базовых типов, а именно $\begin{smallmatrix}0\\0\end{smallmatrix}$, $\begin{smallmatrix}0\\1\end{smallmatrix}$, $\begin{smallmatrix}1\\0\end{smallmatrix}$ или $\begin{smallmatrix}1\\1\end{smallmatrix}$. Таким образом, мы видим, что функция может быть выражена как

$$\begin{aligned}
f(x_1, \ldots, x_6) \;=\; & \big([x_1x_2 \in \{00\}] \;\wedge [x_3x_4x_5x_6 \in \{0010, 0101, 1011\}]\big) \\
\vee\; & \big([x_1x_2 \in \{01\}] \;\wedge [x_3x_4x_5x_6 \in \{0001, 1111\}]\big) \\
\vee\; & \big([x_1x_2 \in \{00, 01\}] \wedge [x_3x_4x_5x_6 \in \{0011, 0111, 1101\}]\big) \\
\vee\; & \big([x_1x_2 \in \{10\}] \;\wedge [x_3x_4x_5x_6 \in \{1001, 1111\}]\big) \\
\vee\; & \big([x_1x_2 \in \{11\}] \;\wedge [x_3x_4x_5x_6 \in \{1101\}]\big) \\
\vee\; & \big([x_1x_2 \in \{10, 11\}] \wedge [x_3x_4x_5x_6 \in \{0101, 1011\}]\big).
\end{aligned} \tag{37}$$

(Первая строка соответствует группе 1, тип $\begin{smallmatrix}1\\0\end{smallmatrix}$, затем следует группа 1, тип $\begin{smallmatrix}0\\1\end{smallmatrix}$, и т. д.; последняя строка соответствует группе 2 и типу $\begin{smallmatrix}1\\1\end{smallmatrix}$.) Функция наподобие $\big[x_3x_4x_5x_6 \in \{0010, 0101, 1011\}\big]$ представляет собой ИЛИ трех минитермов из $\{x_3, x_4, x_5, x_6\}$.

В общем случае можно рассматривать таблицу истинности как массив $2^k \times 2^{n-k}$ с l группами строк, имеющими либо по $\lfloor 2^k/l \rfloor$, либо по $\lceil 2^k/l \rceil$ строк в каждой группе. Группа размером m будет иметь столбцы 2^m базовых типов. Мы образуем конъюнкцию $(g_{it}(x_1, \ldots, x_k) \wedge h_{it}(x_{k+1}, \ldots, x_n))$ для каждой группы i и каждого ненулевого типа t, где g_{it} представляет собой ИЛИ всех минитермов множества $\{x_1, \ldots, x_k\}$ для строк группы, где t имеет 1, а h_{it} представляет собой ИЛИ всех минитермов множества $\{x_{k+1}, \ldots, x_n\}$ для столбцов, имеющих тип t в группе i. ИЛИ всех этих конъюнкций $(g_{it} \wedge h_{it})$ дает $f(x_1, \ldots, x_n)$.

После того как мы выбрали параметры k и l, где $1 \le k \le n-2$ и $1 \le l \le 2^k$, работа начинается с вычисления всех минитермов множеств $\{x_1, \ldots, x_k\}$ и $\{x_{k+1}, \ldots, x_n\}$ за $t(k)+t(n-k)$ шагов (см. (30)). Затем для $1 \le i \le l$ мы полагаем группу i состоящей из строк для значений $(x_1, \ldots, x_k)$, таких, что $(i-1)2^k/l \le (x_1 \ldots x_k)_2 < i2^k/l$; она содержит $m_i = \lceil i2^k/l \rceil - \lceil (i-1)2^k/l \rceil$ строк. Мы образуем все функции g_{it} для $t \in S_i$, семейства $2^{m_i} - 1$ непустых подмножеств этих строк; эта задача решается с помощью $2^{m_i} - m_i - 1$ операций ИЛИ над ранее вычисленными минитермами. Мы также образуем все функции h_{it}, представляющие строки ненулевого типа t; для этой цели нам потребуется не более 2^{n-k} операций ИЛИ в каждой группе i,

поскольку мы можем добавить каждый минитерм в функцию h соответствующего типа t с помощью операции ИЛИ. Наконец мы вычисляем $f = \bigvee_{i=1}^{l} \bigvee_{t\in S_i} (g_{it} \wedge h_{it})$; каждая операция И компенсируется излишней первой операцией ИЛИ при добавлении минитерма в h_{it}. Так что общая стоимость не превышает

$$t(k) + t(n-k) + (l-1) + \sum_{i=1}^{l} \bigl((2^{m_i} - m_i - 1) + 2^{n-k} + (2^{m_i} - 2)\bigr); \tag{38}$$

мы хотим выбрать k и l таким образом, чтобы минимизировать эту верхнюю границу. В упр. 52 обсуждается наилучший выбор при малых n. Когда же n велико, хороший выбор дает цепочку, доказуемо близкую к оптимальной, по крайней мере для большинства функций.

Теорема L. *Пусть $C(n)$ обозначает стоимость наиболее дорогих функций от n переменных. Тогда при $n \to \infty$ мы имеем*

$$C(n) \;\geq\; \frac{2^n}{n}\Bigl(1 + \frac{\lg n}{n} + O\Bigl(\frac{1}{n}\Bigr)\Bigr); \tag{39}$$

$$C(n) \;\leq\; \frac{2^n}{n}\Bigl(1 + 3\frac{\lg n}{n} + O\Bigl(\frac{1}{n}\Bigr)\Bigr). \tag{40}$$

Доказательство. В упр. 48 показано, что нижняя граница (39) является следствием теоремы S. Для получения верхней границы мы устанавливаем $k = \lfloor 2\lg n \rfloor$ и $l = \lceil 2^k/(n - 3\lg n) \rceil$ в методе Лупанова; см. упр. 53. ▮

Синтез хорошей цепочки. Формула (37) — не лучший способ реализации 6-битового детектора простых чисел, но она подсказывает неплохую стратегию. Например, нам не следует выбирать переменные x_1 и x_2, чтобы возглавить строки: в упр. 51 показано, что получатся лучшие цепочки, если выбрать строки, основанные на x_5x_6, а столбцы — на $x_1x_2x_3x_4$, и в общем случае имеется много способов разбиения таблицы истинности путем выбора множеств из k и $n-k$ переменных.

Кроме того, можно улучшить (37), используя наши полные знания обо всех функциях от четырех переменных; нет необходимости вычислять функцию наподобие $[x_3x_4x_5x_6 \in \{0010, 0101, 1011\}]$ начиная с вычисления минитермов для $\{x_3, x_4, x_5, x_6\}$, если мы знаем лучший способ вычисления каждой такой функции "с нуля". С другой стороны, мы должны вычислять несколько таких функций одновременно, так что подход, основанный на минитермах, в конце концов может оказаться не такой уж плохой идеей. Можем ли мы действительно улучшить его?

Давайте попробуем найти хороший способ синтеза булевой цепочки, которая вычисляет заданное множество функций от 4 переменных. Шесть функций от $x_3x_4x_5x_6$ в (37) достаточно просты (см. упр. 54), так что рассмотрим более интересный пример из повседневной жизни.

Семисегментный дисплей представляет собой вездесущий в наше время способ представления 4-битных чисел $(x_1x_2x_3x_4)_2$ посредством семи точно расположенных сегментов, которые могут быть видимы или невидимы. Эти сегменты традиционно именуются (a, b, c, d, e, f, g), как показано на рисунке; мы получаем '0' при включении сегментов (a, b, c, d, e, f), а для '1' нужно включить только сегменты (b, c). (Кстати, идея таких дисплеев была придумана Ф. В. Вудом (F. W. Wood), *U.S. Patent*

974943 (1910), хотя исходный проект Вуда предусматривал восемь сегментов, так как он считал необходимой диагональную черту в представлении ‘4’.) Обычно семисегментные дисплеи поддерживают только десятичные цифры ‘0’, ‘1’, . . . , ‘9’; но, конечно, часы настоящих программистов должны уметь показывать время в шестнадцатеричном формате. Так что мы разработаем семисегментную логику для вывода шестнадцати цифр

(41)

соответственно для входных данных $x_1x_2x_3x_4 = 0000, 0001, 0010, \ldots, 1111$.

Другими словами, мы хотим вычислять семь булевых функций, таблицами истинности которых являются соответственно

$$\begin{aligned} a &= 1011\ 0111\ 1110\ 0011,\\ b &= 1111\ 1001\ 1110\ 0100,\\ c &= 1101\ 1111\ 1111\ 0100,\\ d &= 1011\ 0110\ 1101\ 1110,\\ e &= 1010\ 0010\ 1011\ 1111,\\ f &= 1000\ 1111\ 1111\ 0011,\\ g &= 0011\ 1110\ 1111\ 1111. \end{aligned} \tag{42}$$

Если бы мы просто хотели вычислять каждую функцию по отдельности, то уже рассматривавшиеся ранее методы могут указать, как сделать это с минимальной стоимостью $C(a) = 5$, $C(b) = C(c) = C(d) = 6$, $C(e) = C(f) = 5$ и $C(g) = 4$; общая стоимость всех семи функций будет в таком случае равна 37. Но мы хотим найти единую булеву цепочку, которая содержит их все, и, по-видимому, такая цепочка должна быть намного эффективнее. Как же нам ее найти?

Ну, задача поиска истинно оптимальной цепочки для $\{a, b, c, d, e, f, g\}$, вероятно, неразрешима с вычислительной точки зрения. Но на удивление хорошее решение может быть найдено с помощью пояснявшейся ранее идеи “отпечатков”. А именно, мы знаем, как вычислить не только минимальную стоимость функции, но и множество всех первых шагов, согласующихся с этой минимальной стоимостью в нормальной цепочке. Функция e, например, имеет стоимость 5, но только если мы вычисляем ее, начиная с одной из инструкций

$$x_5 = x_1 \oplus x_4, \qquad x_5 = x_2 \wedge \bar{x}_3, \qquad x_5 = x_2 \vee x_3.$$

К счастью, один из желательных первых шагов принадлежит четырем из семи отпечатков: функции c, d, f и g могут быть вычислены оптимальным образом начиная с шага $x_5 = x_2 \oplus x_3$. Так что это — естественный выбор; он, по сути, экономит нам три шага, поскольку мы знаем, что теперь для завершения вычислений нам нужны не более чем 33 шага из исходных 37.

Теперь мы можем заново вычислить стоимости и отпечатки всех 2^{16} функций, работая, как и ранее, но теперь инициализируя стоимость новой функции x_5 нулем. В результате стоимости функций c, d, f и g уменьшаются на 1, а также происходит изменение отпечатков. Например, функция a все еще имеет стоимость 5, но ее отпечаток увеличивается от $\{x_1 \oplus x_3, x_2 \wedge x_3\}$ до $\{x_1 \oplus x_3, x_1 \wedge x_4, \bar{x}_1 \wedge x_4, x_2 \wedge x_3, \bar{x}_2 \wedge x_4, x_2 \oplus x_4, x_4 \wedge x_5, x_4 \oplus x_5\}$, если функция $x_5 = x_2 \oplus x_3$ доступна бесплатно.

На поверку шаг $x_6 = \bar{x}_1 \wedge x_4$ является общим для четырех из новых отпечатков, так что у нас вновь имеется естественный путь для дальнейшего продвижения. А после того как будут проведены заново вычисления с нулевыми значениями стоимости x_5 и x_6, следующим желательным шагом в пяти новых отпечатках окажется шаг $x_7 = x_3 \wedge \bar{x}_6$. Продолжая таким "жадным" манером, мы не всегда будем такими же везучими, как до сих пор; тем не менее мы получим цепочку из 22 шагов; а Дэвид Стивенсон (David Stevenson) показал, что в действительности достаточно 21 шага при не жадном выборе x_{10}:

$$\begin{array}{lll}
x_5 = x_2 \oplus x_3, & x_{12} = x_1 \wedge x_2, & \bar{a} = x_{19} = x_{15} \oplus x_{18}, \\
x_6 = \bar{x}_1 \wedge x_4, & x_{13} = x_9 \wedge \bar{x}_{12}, & \bar{b} = x_{20} = x_{11} \wedge \bar{x}_{13}, \\
x_7 = x_3 \wedge \bar{x}_6, & x_{14} = \bar{x}_3 \wedge x_{13}, & \bar{c} = x_{21} = \bar{x}_8 \wedge x_{11}, \\
x_8 = x_1 \oplus x_2, & x_{15} = x_5 \oplus x_{14}, & \bar{d} = x_{22} = x_9 \wedge \bar{x}_{16}, \\
x_9 = x_4 \oplus x_5, & x_{16} = x_1 \oplus x_7, & \bar{e} = x_{23} = x_6 \vee x_{14}, \\
x_{10} = x_3 \vee x_9, & x_{17} = x_1 \vee x_5, & \bar{f} = x_{24} = \bar{x}_8 \wedge x_{15}, \\
x_{11} = x_6 \oplus x_{10}, & x_{18} = x_6 \oplus x_{13}, & g = x_{25} = x_7 \vee x_{17}.
\end{array} \tag{43}$$

(Это *нормальная* цепочка, так что она содержит нормализации $\{\bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{e}, \bar{f}, g\}$ вместо $\{a, b, c, d, e, f, g\}$. Простые преобразования дают ненормализованные функции без изменения стоимости.)

Частично определенные функции. Зачастую на практике выходное значение булевой функции определено только для некоторых входных данных $x_1 \ldots x_n$, а в прочих случаях оно не имеет никакого значения. Мы можем знать, например, что некоторая входная комбинация никогда не может осуществиться на практике. В таких случаях в соответствующей позиции таблицы истинности мы помещаем звездочку, а не 0 или 1, как в прочих местах.

С такой ситуацией можно столкнуться, например, в случае семисегментного дисплея, который нередко используется для вывода лишь десятичных цифр, и его входами могут быть лишь двоично-десятичные числа, для которых $(x_1x_2x_3x_4)_2 \le 9$. Мы можем не беспокоиться о том, какие сегменты будут видны в прочих шести случаях. Так что таблица истинности (42) в действительности принимает вид

$$\begin{array}{rcl}
a & = & 1011\ 0111\ 11{**}\ {****}, \\
b & = & 1111\ 1001\ 11{**}\ {****}, \\
c & = & 1101\ 1111\ 11{**}\ {****}, \\
d & = & 1011\ 0110\ 11{**}\ {****}, \\
e & = & 1010\ 0010\ 10{**}\ {****}, \\
f & = & 1000\ 111{*}\ 11{**}\ {****}, \\
g & = & 0011\ 1110\ 11{**}\ {****}.
\end{array} \tag{44}$$

(Функция f имеет звездочку и в позиции $x_1x_2x_3x_4 = 0111$, поскольку '7' может выводиться и как 7, и как 7. Оба варианта встречались автору при работе над этим разделом одинаково часто. В свое время встречались также урезанные варианты цифр 5 и 9, но, к счастью, они вышли из употребления.)

Звездочки в таблице истинности известны как значение *безразлично* (*don't-cares*) — странный термин, который могли придумать только инженеры-электронщики. В табл. 3 показано преимущество возможности произвольного выбора выхода. Например, имеется $\binom{16}{3}2^{13} = 4\,587\,520$ таблиц истинности с тремя безразличными

Таблица 3

Количество функций от 4 переменных с d безразличными значениями и стоимостью c

	$c=0$	$c=1$	$c=2$	$c=3$	$c=4$	$c=5$	$c=6$	$c=7$
$d=0$	10	60	456	2474	10624	24184	25008	2720
$d=1$	160	960	7296	35040	131904	227296	119072	2560
$d=2$	1200	7200	52736	221840	700512	816448	166144	
$d=3$	5600	33600	228992	831232	2045952	1381952	60192	
$d=4$	18200	108816	666528	2034408	3505344	1118128	3296	
$d=5$	43680	257472	1367776	3351488	3491648	433568	32	
$d=6$	80080	455616	2015072	3648608	1914800	86016		
$d=7$	114400	606944	2115648	2474688	533568	12032		
$d=8$	128660	604756	1528808	960080	71520	896		
$d=9$	114080	440960	707488	197632	4160			
$d=10$	78960	224144	189248	20160				
$d=11$	41440	72064	25472	800				
$d=12$	15480	12360	1280					
$d=13$	3680	800						
$d=14$	480							
$d=15$	32							
$d=16$	1							

значениями; 69% из них имеют стоимость 4 или менее, хотя только 21% таблиц истинности без звездочек допускают такую экономию. С другой стороны, безразличные значения не настолько выгодны, как мы могли бы надеяться; в упр. 63 доказывается, что случайная функция со, скажем, 30% безразличных значений в ее таблице истинности имеет тенденцию к снижению стоимости примерно на 30% по сравнению с полностью заданной функцией.

Какова же кратчайшая булева цепочка, вычисляющая семь частично определенных функций из (44)? Наш жадный метод, основанный на отпечатках, легко адаптируется к наличию безразличных значений, поскольку мы можем объединить с помощью ИЛИ отпечатки всех 2^d функций, которые соответствуют шаблону с d звездочками. Начальные стоимости отдельного вычисления каждой функции теперь уменьшаются до $C(a)=3$, $C(b)=C(c)=2$, $C(d)=5$, $C(e)=2$, $C(f)=3$, $C(g)=4$, что в сумме дает 21 вместо 37. Функция g не становится дешевле, но теперь она имеет больший след. Действуя так же, как и ранее, но с использованием преимуществ, предоставляемых безразличными значениями, теперь мы в состоянии найти подходящую цепочку длиной, равной всего лишь 11, — цепочку, в которой на один выход выполняется менее 1.6 операций(!):

$$\begin{array}{lll} x_5 = x_1 \vee x_2, & \bar d = x_9 = x_6 \oplus x_8, & \bar c = x_{13} = \bar x_4 \wedge x_{10}, \\ x_6 = x_3 \oplus x_5, & \bar f = x_{10} = \bar x_5 \wedge x_8, & \bar e = x_{14} = x_4 \vee x_9, \\ x_7 = \bar x_2 \wedge x_6, & \bar b = x_{11} = x_2 \wedge \bar x_9, & g = x_{15} = x_6 \vee x_{11}. \\ x_8 = x_4 \vee x_7, & \bar a = x_{12} = \bar x_3 \wedge x_9, & \end{array} \tag{45}$$

Эта удивительная цепочка, найденная Кори Пловером (Corey Plover) в 2011 году, выбирает x_7 нежадным способом.

Крестики-нулики. Давайте теперь обратимся к несколько большей задаче, основанной на популярной детской игре. Два игрока по очереди заполняют ячейки решетки 3×3. Один игрок пишет “крестики” ×, а второй — “нулики” ○. Игра продолжается до тех пор, пока не найдется прямая линия с тремя × или с тремя ○ (при этом выигрывает соответствующий игрок) или пока все девять ячеек не будут

Я приступил к изучению игры под названием “крестики-нулики”... чтобы выяснить количество комбинаций всех возможных ходов и позиций. Я обнаружил, что это количество сравнительно незначительно. ...Однако возникла сложность нового рода. Во время хода автомата могло оказаться два хода, в равной мере ведущих его к победе. ...Если только не принять специальные меры, машина будет пытаться сделать два противоречивых хода.

— ЧАРЛЬЗ БЭББИДЖ (CHARLES BABBAGE),
Отрывки из жизни философа
(Passages from the Life of a Philosopher) (1864)

заполнены без победителя (объявляется ничья). Например, игра может протекать следующим образом.

(46)

Выиграл ×. Наша цель — разработать машину для оптимальной игры в крестики-нулики, которая будет делать выигрышный ход в позиции, из которой можно обеспечить победу, и никогда не делает проигрышный ход в позиции, из которой проигрыша можно избежать.

Говоря более точно, мы хотим получить 18 булевых переменных $x_1, \ldots, x_9$, $o_1, \ldots, o_9$, которые будут управлять лампами для подсветки ячеек в текущей позиции. Ячейки пронумерованы цифрами от 1 до 9, как на панели телефона: $\begin{smallmatrix}1&2&3\\4&5&6\\7&8&9\end{smallmatrix}$. В ячейке j выводится ×, если $x_j = 1$, ○, если $o_j = 1$, либо ячейка остается пустой, если $x_j = o_j = 0$.* Мы никогда не должны получить $x_j = o_j = 1$, поскольку это привело бы к выводу ‘⊗’. Будем считать, что переменные $x_1 \ldots x_9 o_1 \ldots o_9$ имеют такие значения, что выводится корректная позиция игры, в которой никто не выиграл. Компьютер ходит крестиками, и в настоящий момент ход компьютера. Мы хотим определить девять функций $y_1, \ldots, y_9$, где y_j означает “изменить x_j с 0 на 1”. Если текущая позиция — ничья, мы должны сделать $y_1 = \cdots = y_9 = 0$; в противном случае ровно одно значение y_j должно быть равно 1, и, конечно, выходное значение $y_j = 1$ может появляться, только если $x_j = o_j = 0$.

При 18 переменных каждая из наших девяти функций y_j будет иметь таблицу истинности размером $2^{18} = 262\,144$. Оказывается, что возможны только 4520 корректных входных данных $x_1 \ldots x_9 o_1 \ldots o_9$, так что эти таблицы истинности на 98.3% заполнены безразличными значениями. Но и 4520 — слишком много, если надеяться разработать и понять имеющую смысл булеву цепочку интуитивно. В разделе 7.1.4 будут рассматриваться альтернативные способы представления булевых функций, при использовании которых зачастую можно работать с сотнями переменных, несмотря на то, что связанные с ними таблицы истинности невозможно велики.

* Эта постановка задачи основана на посещении автором в начале 1950-х годов Научно-промышленного музея в Чикаго, где он впервые познакомился с магией коммутационных схем. Машина в Чикаго, разработанная около 1940 года У. Кейстером (W. Keister) из Bell Telephone Laboratories, позволяла игроку сделать первый ход. Очень быстро я убедился, что победить ее невозможно; тогда я решил делать по возможности самые глупые ходы, надеясь, что разработчик не предусмотрел такого дурацкого поведения игрока. Я позволил машине достичь позиции, в которой у нее было два выигрышных хода, и машина сделала их *оба*! Двойной ход, конечно же, был нарушением правил, так что я одержал моральную победу, несмотря на то, что машина объявила меня проигравшим.

Большинство функций от 18 переменных требуют более $2^{18}/18$ логических элементов, но будем надеяться, что нам удастся достичь лучших результатов. Действительно, правдоподобная стратегия для выполнения очередного хода в крестиках-нуликах напрашивается сама собой. Она выражается через условия, которые не так сложно распознать:

w_j, ×в ячейке j приведет к победе, завершая линию ×;
b_j, ○ в ячейке j приведет к поражению, завершая линию ○;
f_j, ×в ячейке j дает × два способа победить;
d_j, ○ в ячейке j дает ○ два способа победить.

Например, ход × в центр игрового поля в (46) обусловлен необходимостью блокировать ○, так что это ход типа b_5; к счастью, он одновременно является ходом типа f_5, обеспечивающим победу на следующем ходу.

Обозначим множество побеждающих линий как $L = \{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{1,4,7\},\{2,5,8\},\{3,6,9\},\{1,5,9\},\{3,5,7\}\}$. Тогда мы имеем следующее.

$$m_j = \bar{x}_j \wedge \bar{o}_j; \qquad \text{[ход в ячейку } j \text{ корректен]} \tag{47}$$

$$w_j = m_j \wedge \bigvee_{\{i,j,k\}\in L}(x_i \wedge x_k); \qquad \text{[ход в ячейку } j \text{ приносит победу]} \tag{48}$$

$$b_j = m_j \wedge \bigvee_{\{i,j,k\}\in L}(o_i \wedge o_k); \qquad \text{[ход в ячейку } j \text{ блокирующий]} \tag{49}$$

$$f_j = m_j \wedge S_2\big(\{\alpha_{ik} \mid \{i,j,k\} \in L\}\big); \qquad \text{[ход в ячейку } j \text{ ставит вилку]} \tag{50}$$

$$d_j = m_j \wedge S_2\big(\{\beta_{ik} \mid \{i,j,k\} \in L\}\big); \qquad \text{[ход в ячейку } j \text{ оборонительный]} \tag{51}$$

Здесь α_{ik} и β_{ik} означают один × или ○ вместе с пустым местом, а именно

$$\alpha_{ik} = (x_i \wedge m_k) \vee (m_i \wedge x_k), \qquad \beta_{ik} = (o_i \wedge m_k) \vee (m_i \wedge o_k). \tag{52}$$

Например, $b_1 = m_1 \wedge \big((o_2 \wedge o_3) \vee (o_4 \wedge o_7) \vee (o_5 \wedge o_9)\big)$; $f_2 = m_2 \wedge S_2(\alpha_{13}, \alpha_{58}) = m_2 \wedge \alpha_{13} \wedge \alpha_{58}$; $d_5 = m_5 \wedge S_2(\beta_{19}, \beta_{28}, \beta_{37}, \beta_{46})$.

При таких определениях мы можем попытаться упорядочить ходы по рангам следующим образом:

$$\{w_1, \ldots, w_9\} > \{b_1, \ldots, b_9\} > \{f_1, \ldots, f_9\} > \{d_1, \ldots, d_9\} > \{m_1, \ldots, m_9\}. \tag{53}$$

"Если это возможно — победите; в противном случае, если это возможно, — блокируйте противника; в противном случае, если это возможно, — поставьте вилку; в противном случае, если это возможно, — защищайтесь; в противном случае сделайте корректный ход." Кроме того, при выборе между корректными ходами представляется разумным использовать упорядочение

$$m_5 > m_1 > m_3 > m_9 > m_7 > m_2 > m_6 > m_8 > m_4, \tag{54}$$

поскольку средняя ячейка 5 встречается в четырех побеждающих линиях, углы 1, 3, 9 и 7 — в трех, а боковые ячейки 2, 6, 8 и 4 — только в двух. Мы можем принять этот же порядок индексов и во всех группах ходов $\{w_j\}$, $\{b_j\}$, $\{f_j\}$, $\{d_j\}$ и $\{m_j\}$ в (53).

Чтобы гарантировать, что будет выбрано не более одного хода, определим w'_j, b'_j, f'_j, d'_j, m'_j для обозначения "предшествующий выбор лучше последующего." Таким образом, $w'_5 = 0$, $w'_1 = w_5$, $w'_3 = w_1 \vee w'_1$, …, $w'_4 = w_8 \vee w'_8$, $b'_5 = w_4 \vee w'_4$, $b'_1 = b_5 \vee b'_5$,

$\ldots,\ m'_4 = m_8 \vee m'_8$. Тогда мы можем завершить определение автомата для игры в крестики-нулики, полагая

$$y_j = (w_j \wedge \overline{w}'_j) \vee (b_j \wedge \overline{b}'_j) \vee (f_j \wedge \overline{f}'_j) \vee (d_j \wedge \overline{d}'_j) \vee (m_j \wedge \overline{m}'_j) \quad \text{для } 1 \le j \le 9. \tag{55}$$

Итак, мы построили 9 логических элементов для m, 48 для w, 48 для b, 144 для α и β, 35 для f (с помощью рис. 9), 35 для d, 43 для начальных переменных и 80 для y. Кроме того, мы можем использовать наши знания о частично определенных функциях от 4 переменных для уменьшения шести операций в (52) до всего лишь четырех

$$\alpha_{ik} = (x_i \oplus x_k) \vee \overline{(o_i \oplus o_k)}, \qquad \beta_{ik} = \overline{(x_i \oplus x_k)} \vee (o_i \oplus o_k). \tag{56}$$

Эта хитрость экономит нам 48 логических элементов; так что общая стоимость нашего проекта — 396 логических элементов.

Стратегия игры в (47)–(56) хорошо работает в большинстве случаев, но имеет несколько явных проблем. Например, она позорно проигрывает в игре

, (57)

в которой вторым ходом $\times$ является d_3, защита от вилки $\circ$ и который в действительности заставляет $\circ$ поставить вилку в противоположном углу! Еще один сбой возникает, например, после позиции , когда ход m_5 приводит к ничьей , , , , , , вместо победы $\times$, показанной в (46). В упр. 65 вносятся необходимые исправления и получается корректный автомат для игры в крестики-нулики, требующий всего 445 логических элементов.

***Функциональное разложение.** Если функция $f(x_1, \ldots, x_n)$ может быть записана в виде $g(x_1, \ldots, x_k, h(x_{k+1}, \ldots, x_n))$, обычно имеет смысл сначала вычислить $y = h(x_{k+1}, \ldots, x_n)$, а затем — $g(x_1, \ldots, x_k, y)$. Роберт Л. Ашенхарст (Robert L. Ashenhurst) начал изучение таких разложений в 1952 году [см. *Annals Computation Lab. Harvard University* **29** (1957), 74–116] и заметил, что имеется простой способ распознать, что f обладает этим свойством: если записать таблицу истинности f в виде массива $2^k \times 2^{n-k}$, как в (36), со строками для всех наборов значений $x_1 \ldots x_k$ и столбцами для всех наборов значений $x_{k+1} \ldots x_n$, то искомые подфункции g и h существуют тогда и только тогда, когда столбцы этого массива имеют не более двух различных значений. Например, таблица истинности для функции $\langle x_1 x_2 \langle x_3 x_4 x_5 \rangle\rangle$, будучи выражена в описанном двумерном виде, представляет собой

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{matrix}.$$

Один тип столбцов соответствует случаю $h(x_{k+1}, \ldots, x_n) = 0$; другой соответствует $h(x_{k+1}, \ldots, x_n) = 1$.

В общем случае переменные $X = \{x_1, \ldots, x_n\}$ могут быть разбиты на любые непересекающихся подмножества $Y = \{y_1, \ldots, y_k\}$ и $Z = \{z_1, \ldots, z_{n-k}\}$, и мы можем получить $f(x) = g(y, h(z))$. Проверить наличие разложения (Y, Z) можно, рассматривая столбцы таблицы истинности размером $2^k \times 2^{n-k}$, строки которой соответствуют значениям y. Однако имеется 2^n таких способов разбиения X; и все из них являются потенциальными победителями, за исключением тривиальных случаев,

когда $|Y| = 0$ или $|Z| \le 1$. Как же нам избежать изучения такого значительного количества возможностей?

Практичный способ решения этой задачи был открыт В. Ю.-Ш. Шеном (V. Y.-S. Shen), А. Ч. Мак-Келларом (A. C. McKellar) и П. Вайнером (P. Weiner) [*IEEE Transactions* **C-20** (1971), 304–309], метод которых обычно требует только $O(n^2)$ шагов для идентификации любого потенциально полезного разбиения (Y, Z), которое может существовать. Основная идея проста: предположим, что $x_i \in Z$, $x_j \in Z$ и $x_m \in Y$. Определим восемь бинарных векторов δ_l для $l = (l_1 l_2 l_3)_2$, где δ_l имеет (l_1, l_2, l_3) соответственно в компонентах (i, j, m) и нули в остальных местах. Рассмотрим любой выбранный случайным образом вектор $x = x_1 \ldots x_n$ и вычислим $f_l = f(x \oplus \delta_l)$ для $0 \le l \le 7$. Тогда пары

$$\binom{f_0}{f_1} \qquad \binom{f_2}{f_3} \qquad \binom{f_4}{f_5} \qquad \binom{f_6}{f_7} \tag{58}$$

появятся в подматрице размером 2×4 таблицы истинности размером $2^k \times 2^{n-k}$. Так что разложение невозможно, если эти пары различны или если они содержат три различных значения.

Назовем пары хорошими, если все они равны или если они имеют только два разных значения. В противном случае назовем их плохими. Если f имеет, по сути, случайное поведение, то мы быстро найдем плохие пары, если выполним этот эксперимент с несколькими случайным образом выбранными векторами x, поскольку только 88 из 256 возможных вариантов для $f_0 f_1 \ldots f_7$ соответствуют хорошему множеству пар; вероятность найти хорошую пару в строке десять раз подряд составляет всего лишь $(\frac{88}{256})^{10} \approx .00002$. А когда мы найдем плохие пары, мы сможем сделать вывод, что

$$x_i \in Z \quad \text{и} \quad x_j \in Z \implies x_m \in Z, \tag{59}$$

поскольку альтернатива $x_m \in Y$ невозможна.

Предположим, например, что $n = 9$ и что f — функция, таблица истинности которой 11001001000011...00101 состоит из 512 старших битов числа π в бинарной записи. (Это "более-менее случайная функция", которую мы изучали для $n = 4$ в (5) и (6) выше.) Плохие пары этой "π-функции" быстро находятся в каждом из случаев (i, j, m), для которых $m \ne i < j \ne m$. Фактически в экспериментах автора 170 из 252 случаев были разрешены немедленно; среднее количество случайных векторов x на один случай составило лишь 1.52; и только один случай потребовал целых восьми x до того, как появилась плохая пара. Таким образом, (59) выполняется для всех подходящих (i, j, m) и функция, очевидно, неразложима. В действительности, как указывает упр. 73, нам не требуется выполнять 252 теста для установления факта неразложимости π-функции; для этого должно быть достаточно лишь $\binom{n}{2} = 36$ из них.

Обратимся к менее случайной функции. Пусть $f(x_1, \ldots, x_9) = (\det X) \bmod 2$, где

$$X = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix}. \tag{60}$$

Эта функция не удовлетворяет условию (59) при $i = 1$, $j = 2$ и $m = 3$, поскольку в этом случае плохих пар нет. Но она удовлетворяет (59) для $4 \le m \le 9$ при $\{i, j\} = \{1, 2\}$. Мы можем записать это поведение с помощью удобной аббревиатуры '$12 \Rightarrow 456789$'; полное множество импликаций для всех пар $\{i, j\}$ имеет вид

$12 \Rightarrow 456789$	$18 \Rightarrow 34569$	$27 \Rightarrow 34569$	$37 \Rightarrow 24568$	$48 \Rightarrow 12369$	$67 \Rightarrow 12358$
$13 \Rightarrow 456789$	$19 \Rightarrow 24568$	$28 \Rightarrow 134679$	$38 \Rightarrow 14567$	$49 \Rightarrow 12358$	$68 \Rightarrow 12347$
$14 \Rightarrow 235689$	$23 \Rightarrow 456789$	$29 \Rightarrow 14567$	$39 \Rightarrow 124578$	$56 \Rightarrow 123789$	$69 \Rightarrow 124578$
$15 \Rightarrow 36789$	$24 \Rightarrow 36789$	$34 \Rightarrow 25789$	$45 \Rightarrow 123789$	$57 \Rightarrow 12369$	$78 \Rightarrow 123456$
$16 \Rightarrow 25789$	$25 \Rightarrow 134679$	$35 \Rightarrow 14789$	$46 \Rightarrow 123789$	$58 \Rightarrow 134679$	$79 \Rightarrow 123456$
$17 \Rightarrow 235689$	$26 \Rightarrow 14789$	$36 \Rightarrow 124578$	$47 \Rightarrow 235689$	$59 \Rightarrow 12347$	$89 \Rightarrow 123456$

(см. упр. 69). При проверке этой функции случайным образом плохие пары найти несколько сложнее: среднее количество требуемых x в экспериментах автора при этом увеличилось до примерно 3.6 (при существовании плохих пар). И, конечно же, требуется ограничить тестирование, выбирая допустимый порог t; после t последовательных неудачных попыток найти плохие пары дальнейшие попытки прекращаются. Выбор $t = 10$ должен привести к нахождению всех, кроме 8, импликаций из перечисленных выше 198.

Импликации типа (59) являются дизъюнктами Хорна, а из раздела 7.1.1 мы знаем, что из дизъюнктов Хорна легко делать последующие выводы. Действительно, метод из упр. 74 приводит к выводу после рассмотрения менее чем 50 вариантов (i, j, m), что единственным возможным разбиением с $|Z| > 1$ является тривиальное разбиение ($Y = \emptyset$, $Z = \{x_1, \ldots, x_9\}$).

Аналогичные результаты получаются и при $f(x_1, \ldots, x_9) = [\mathrm{per}\, X > 0]$, где 'per' обозначает функцию *перманента*. (В этом случае f говорит нам о том, существует ли совершенное паросочетание в двудольном подграфе $K_{3,3}$, ребра которого определяются переменными $x_1 \ldots x_9$.) Теперь у нас имеется всего 180 импликаций,

$12 \Rightarrow 456789$	$18 \Rightarrow 3459$	$27 \Rightarrow 3459$	$37 \Rightarrow 2468$	$48 \Rightarrow 1269$	$67 \Rightarrow 1358$
$13 \Rightarrow 456789$	$19 \Rightarrow 2468$	$28 \Rightarrow 134679$	$38 \Rightarrow 1567$	$49 \Rightarrow 1358$	$68 \Rightarrow 2347$
$14 \Rightarrow 235689$	$23 \Rightarrow 456789$	$29 \Rightarrow 1567$	$39 \Rightarrow 124578$	$56 \Rightarrow 123789$	$69 \Rightarrow 124578$
$15 \Rightarrow 3678$	$24 \Rightarrow 3678$	$34 \Rightarrow 2579$	$45 \Rightarrow 123789$	$57 \Rightarrow 1269$	$78 \Rightarrow 123456$
$16 \Rightarrow 2579$	$25 \Rightarrow 134679$	$35 \Rightarrow 1489$	$46 \Rightarrow 123789$	$58 \Rightarrow 134679$	$79 \Rightarrow 123456$
$17 \Rightarrow 235689$	$26 \Rightarrow 1489$	$36 \Rightarrow 124578$	$47 \Rightarrow 235689$	$59 \Rightarrow 2347$	$89 \Rightarrow 123456,$

только 122 из которых будут обнаружены при $t = 10$ в качестве порогового значения. (Наилучший выбор t неясен; вероятно, он должен варьироваться динамически.) Тем не менее этих 122 дизъюнктов Хорна более чем достаточно, чтобы установить неразложимость.

А что можно сказать о разложимых функциях? В случае функции $f = \langle x_2 x_3 x_6 x_9 \langle x_1 x_4 x_5 x_7 x_8 \rangle \rangle$ мы получаем $i \wedge j \Rightarrow m$ для всех $m \notin \{i, j\}$, за исключением случаев, когда $\{i, j\} \subseteq \{1, 4, 5, 7, 8\}$; в последнем случае m также должно принадлежать $\{1, 4, 5, 7, 8\}$. Хотя при пороге $t = 10$ находятся только 185 из этих 212 импликаций, очень быстро обнаруживается весьма вероятное разбиение $Y = \{x_2, x_3, x_6, x_9\}$, $Z = \{x_1, x_4, x_5, x_7, x_8\}$.

Когда потенциальное разложение оказывается обоснованным таким образом, необходимо убедиться, что соответствующая таблица истинности $2^k \times 2^{n-k}$ действи-

тельно имеет только один или два различных столбца. Но мы с удовольствием затратим 2^n единиц времени на такую проверку, поскольку очень сильно упростим вычисление f.

Еще одним интересным случаем является функция сравнения $f = \big[(x_1x_2x_3x_4)_2 \geq (x_5x_6x_7x_8)_2 + x_9\big]$. Ее 184 потенциально выводимыми импликациями являются

$12 \Rightarrow 3456789$	$18 \Rightarrow 2345679$	$27 \Rightarrow 34689$	$37 \Rightarrow 489$	$48 \Rightarrow 9$	$67 \Rightarrow 23489$
$13 \Rightarrow 2456789$	$19 \Rightarrow 2345678$	$28 \Rightarrow 34679$	$38 \Rightarrow 479$	$49 \Rightarrow 8$	$68 \Rightarrow 23479$
$14 \Rightarrow 2356789$	$23 \Rightarrow 46789$	$29 \Rightarrow 34678$	$39 \Rightarrow 478$	$56 \Rightarrow 1234789$	$69 \Rightarrow 23478$
$15 \Rightarrow 2346789$	$24 \Rightarrow 36789$	$34 \Rightarrow 789$	$45 \Rightarrow 1236789$	$57 \Rightarrow 1234689$	$78 \Rightarrow 349$
$16 \Rightarrow 2345789$	$25 \Rightarrow 1346789$	$35 \Rightarrow 1246789$	$46 \Rightarrow 23789$	$58 \Rightarrow 1234679$	$79 \Rightarrow 348$
$17 \Rightarrow 2345689$	$26 \Rightarrow 34789$	$36 \Rightarrow 24789$	$47 \Rightarrow 389$	$59 \Rightarrow 1234678$	$89 \Rightarrow 4,$

и 145 из них обнаруживаются при пороге $t = 10$. При этом открываются три потенциальных разложения, имеющих $Z = \{x_4, x_8, x_9\}$, $Z = \{x_3, x_4, x_7, x_8, x_9\}$ и $Z = \{x_2, x_3, x_4, x_6, x_7, x_8, x_9\}$ соответственно. Ашенхарст (Ashenhurst) доказал, что мы можем выполнять приведение функции f немедленно по обнаружении нетривиального разложения; прочие разложения будут обнаружены позже, когда мы будем пытаться приводить более простые функции g и h.

***Разложение частично определенных функций.** Когда функция f определена только частично, разложение с разбиением (Y, Z) опирается на возможность присвоения конкретных значений безразличным значениям таким образом, что в соответствующей таблице истинности $2^k \times 2^{n-k}$ имеется не более двух различных видов столбцов.

Два вектора, $u_1 \ldots u_m$ и $v_1 \ldots v_m$, состоящие из нулей, единиц и звездочек, называются *несовместимыми*, если для некоторого j либо $u_j = 0$ и $v_j = 1$, либо $u_j = 1$ и $v_j = 0$ (или, что то же самое, если подкубы m-мерного куба, определяемые u и v, не имеют общих точек). Рассмотрим граф, вершинами которого являются столбцы таблицы истинности с безразличными значениями, где u — v тогда и только тогда, когда u и v несовместимы. Мы можем присвоить значения звездочкам, чтобы получить не более двух различных видов строк, тогда и только тогда, когда этот граф является *двудольным*. Если $u_1, \ldots, u_l$ взаимно совместимы, их обобщенный консенсус $u_1 \sqcup \cdots \sqcup u_l$, определенный в упр. 7.1.1–32, совместим со всеми ними. [См. S. L. Hight, *IEEE Trans.* **C-22** (1973), 103–110; E. Boros, V. Gurvich, P. L. Hammer, T. Ibaraki, and A. Kogan, *Discrete Applied Math.* **62** (1995), 51–75.] Поскольку граф является двудольным тогда и только тогда, когда он не содержит нечетных циклов, мы можем легко проверить это условие с помощью поиска в глубину (см. раздел 7.4.1).

Следовательно, метод Шена, Мак-Келлара и Вайнера работает и при наличии безразличных значений: четыре пары в (58) рассматриваются как плохие тогда и только тогда, когда три из них взаимно несовместимы. Мы можем работать почти так же, как и ранее, хотя плохие пары, естественно, искать при наличии множества звездочек труднее (см. упр. 72). Однако теорема Ашенхарста больше не применима. Когда имеется несколько разложений, в дальнейшем все они должны быть изучены, поскольку они могут использовать разные настройки для безразличных значений, и некоторые из них могут быть лучше других.

Хотя большинство функций $f(x)$ не имеют простых разложений $g(y, h(z))$, мы не должны терять надежду слишком быстро, поскольку к эффективным цепочкам могут привести и другие виды разложений, такие как $g(y, h_1(z), h_2(z))$. Если, например, функция f симметрична по трем из ее переменных $\{z_1, z_2, z_3\}$, мы всегда можем написать $f(x) = g\bigl(y, S_{1,2}(z_1, z_2, z_3), S_{1,3}(z_1, z_2, z_3)\bigr)$, поскольку $S_{1,2}(z_1, z_2, z_3)$ и $S_{1,3}(z_1, z_2, z_3)$ характеризуют значение $z_1 + z_2 + z_3$. (Заметим, что для вычисления как $S_{1,2}$, так и $S_{1,3}$ достаточно всего четырех шагов.)

В общем случае, как заметил Г. А. Кертис (H. A. Curtis) [*JACM* **8** (1961), 484–496], $f(x)$ может быть выражена в виде $g(y, h_1(z), \ldots, h_r(z))$ тогда и только тогда, когда таблица истинности $2^k \times 2^{n-k}$, соответствующая Y и Z, имеет не более 2^r столбцов различных типов. А при наличии безразличных значений тот же результат справедлив тогда и только тогда, когда граф несовместимости для Y и Z может быть раскрашен не более чем 2^r цветами.

Например, рассматривавшаяся выше функция $f(x) = (\det X) \bmod 2$, как оказывается, имеет восемь различных столбцов, когда $Z = \{x_4, x_5, x_6, x_7, x_8, x_9\}$; это на удивление небольшое число, учитывая, что таблица истинности состоит из 8 строк и 64 столбцов. Этот факт может привести нас к открытию того, как разложить детерминант на алгебраические дополнения первой строки,

$$f(x) \;=\; x_1 \wedge h_1(x_4, \ldots, x_9) \oplus x_2 \wedge h_2(x_4, \ldots, x_9) \oplus x_3 \wedge h_3(x_4, \ldots, x_9),$$

если мы еще незнакомы с этим правилом.

Когда имеется $d \le 2^r$ различных столбцов, мы можем рассматривать $f(x)$ как функцию от y и $h(z)$, где h отображает каждый бинарный вектор $z_1 \ldots z_{n-k}$ в одно из значений $\{0, 1, \ldots, d-1\}$. Таким образом, $(h_1, \ldots, h_r)$, по сути, является кодированием различных типов столбцов, и мы надеемся найти очень простые функции h_1, …, h_r, которые обеспечивают такое кодирование. Более того, если d строго меньше 2^r, функция $g(y, h_1, \ldots, h_r)$ будет иметь много безразличных значений, которые могут существенно снизить ее стоимость.

Различные столбцы могут также предлагать функцию g, для которой h имеют безразличные значения. Например, мы можем использовать $g(y_1, y_2, h_1, h_2) = (y_1 \oplus (h_1 \wedge y_2)) \wedge h_2$, когда все столбцы представляют собой либо $(0,0,0,0)^T$, либо $(0,0,1,1)^T$, либо $(0,1,1,0)^T$; тогда значение $h_1(z)$ произвольно при z, соответствующем ненулевому столбцу. Г. А. Кертис (H. A. Curtis) пояснил, как использовать эту идею при $|Y| = 1$ и $|Z| = n - 1$ [см. *IEEE Transactions* **C-25** (1976), 1033–1044].

Всеобъемлющий обзор методов разложения можно найти в Richard M. Karp, *J. Society for Industrial and Applied Math.* **11** (1963), 291–335.

Бóльшие значения n. Мы рассматривали только весьма малые примеры булевых функций. Теорема S говорит о том, что больши́е, случайные примеры достаточно сложны; но практические примеры могут оказаться совсем не случайными, так что имеет смысл поискать возможность упрощений с помощью эвристических методов.

С ростом n известные лучшие способы работы с булевыми функциями, как правило, начинаются с булевой цепочки (а не с огромной таблицы истинности) и пытаются улучшить ее с помощью "локальных изменений". Цепочка может быть определена как множество уравнений. Тогда, если промежуточный результат используется в сравнительно небольшом количестве последующих шагов, можно попытаться

устранить его, временно делая эти последующие шаги функциями трех переменных, и переформулируя эти функции для получения по возможности лучших цепочек.

Например, предположим, что логический элемент $x_i = x_j \circ x_k$ используется однократно, в логическом элементе $x_l = x_i \,\square\, x_m$, так что $x_l = (x_j \circ x_k) \,\square\, x_m$. Могут существовать и другие логические элементы, вычисляющие иные функции от x_j, x_k и x_m; а из определений x_j, x_k и x_m может вытекать невозможность некоторых объединенных значений (x_j, x_k, x_m). Таким образом, мы можем оказаться способными вычислить x_l из других логических элементов с помощью только одной дополнительной операции. Например, если $x_i = x_j \wedge x_k$ и $x_l = x_i \vee x_m$ и если значения $x_j \vee x_m$ и $x_k \vee x_m$ появляются в цепочке где-то в другом месте, мы можем установить $x_l = (x_j \vee x_m) \wedge (x_k \vee x_m)$; это устранит x_i и снизит стоимость на 1. Или, если, скажем, $x_j \wedge (x_k \oplus x_m)$ появляется где-то в другом месте и мы знаем, что $x_j x_k x_m \neq 101$, то можем установить $x_l = x_m \oplus (x_j \wedge (x_k \oplus x_m))$.

Если x_i используется только в x_l, а x_l используется только в x_p, то логический элемент x_p зависит от четырех переменных, и мы можем снизить стоимость, используя свои знания о функциях от четырех переменных, получая x_p лучшим способом и устраняя при этом x_i и x_l. Аналогично, если x_i появляется только в x_l и x_p, мы можем удалить x_i, если найдем лучший способ вычисления двух различных функций от четырех переменных, возможно, с безразличными значениями и с другими функциями от этих четырех переменных, доступными бесплатно. И опять мы сталкиваемся с тем, что знаем, как решать такие задачи с помощью рассматривавшегося выше метода отпечатков.

Если никакие локальные изменения не в состоянии снизить стоимость, мы можем испытать локальные изменения, сохраняющие или даже увеличивающие стоимость, чтобы обнаружить другие виды цепочек, которые могут быть упрощены иными способами. Такие локальные методы поиска будут широко рассмотрены в разделе 7.10.

Прекрасный обзор методов булевой оптимизации, которую инженеры-электронщики называют задачей “многоуровневого логического синтеза”, был опубликован Р. К. Брейтоном (R. K. Brayton), Г. Д. Хатчелом (G. D. Hachtel) и А. Л. Санджованни-Винсентелли (A. L. Sangiovanni-Vincentelli), *Proceedings of the IEEE* **78** (1990), 264–300, а также в книге Д. де Мичели (G. De Micheli) *Synthesis and Optimization of Digital Circuits* (McGraw–Hill, 1994).

Нижние границы. Теорема S гласит, что почти каждая булева функция от $n \geq 12$ переменных трудна для вычисления и требует цепочки, длина которой превышает $2^n/n$. Тем не менее современные компьютеры, построенные на логических схемах с электрическими сигналами, представляющими тысячи булевых переменных, успешно вычисляют несметное количество булевых функций каждую микросекунду. Очевидно, имеется множество важных функций, которые, невзирая на теорему S, могут быть вычислены очень быстро. Действительно, доказательство этой теоремы было косвенным; мы просто подсчитали случаи с низкой стоимостью, так что мы абсолютно ничего не знаем о конкретных примерах, которые могут возникнуть на практике. Если мы хотим вычислить некоторую функцию и можем думать только о трудоемком способе выполнения этой работы, то как мы можем быть уверены в том, что не существует никакого хитрого способа сократить количество вычислений?

Ответ на этот вопрос почти что позорный: после десятилетий целенаправленных поисков ученые оказались не в состоянии найти *никакого* явного семейства функций $f(x_1, \ldots, x_n)$, стоимость которого была бы с ростом n существенно нелинейна. Истинное поведение представляет собой $2^n/n$, но сильная нижняя граница наподобие $n \log\log\log n$ так и не доказана! Конечно, можно создать искусственные примеры, такие как "лексикографически наименьшая таблица истинности длиной 2^n, недостижимая ни одной булевой цепочкой длиной $\lfloor 2^n/n \rfloor - 1$"; но такие функции, конечно, не являются явно заданными. Таблица истинности явной функции $f(x_1, \ldots, x_n)$ должна быть вычислима за не более чем, скажем, 2^{cn} единиц времени для некоторой константы c; т. е. время, необходимое для указания всех значений функций, должно быть полиномиальным от длины таблицы истинности. При таких базовых правилах в настоящее время неизвестно ни одно семейство функций с одним выходом с комбинационной сложностью, превышающей $3n + O(1)$ при $n \to \infty$. [См. N. Blum, *Theoretical Computer Science* **28** (1984), 337–345.]

В целом картина не столь безрадостна, так как для ряда функций, представляющих практическую важность, доказаны некоторые интересные *линейные* нижние границы. Основной способ получения таких результатов был предложен в 1970 году Н. П. Редькиным. Предположим, что у нас имеется оптимальная цепочка стоимостью r для $f(x_1, \ldots, x_n)$. Установив $x_n \leftarrow 0$ или $x_n \leftarrow 1$, мы получаем сокращенные цепочки для функций $g(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, 0)$ и $h(x_1, \ldots, x_{n-1}) = f(x_1, \ldots, x_{n-1}, 1)$, имеющие стоимость $r - u$, если x_n используется как входные данные для u различных логических элементов. Кроме того, если x_n используется в "канализирующем" логическом элементе $x_i = x_n \circ x_k$, где оператор $\circ$ не является ни $\oplus$, ни $\equiv$, некоторые установки x_n приведут к тому, что x_i будет представлять собой константу, тем самым далее уменьшая цепочку для g или h. Нижние границы для g и/или h, таким образом, приводят к нижней границе для f. (См. упр. 77–81.)

Но где же доказательства нелинейных нижних границ? Почти любая задача с ответом "да-нет" может быть сформулирована как булева функция, так что не имеется недостатка в явных функциях, о которых мы не знаем, как вычислить их за линейное или даже полиномиальное время. Например, любой ориентированный граф G с вершинами $\{v_1, \ldots, v_m\}$ может быть представлен своей матрицей смежности X, где $x_{ij} = [v_i \to v_j]$; тогда

$$f(x_{12}, \ldots, x_{1m}, \ldots, x_{m1}, \ldots, x_{m(m-1)}) \;=\; [G \text{ имеет гамильтонов путь}] \tag{61}$$

представляет собой булеву функцию от $n = m(m-1)$ переменных. Мы бы дорого заплатили за то, чтобы иметь возможность вычислять эту функцию, скажем, за n^4 шагов. Мы знаем, как вычислить таблицу истинности для f за $O(m!\,2^n) = 2^{n+O(\sqrt{n}\log n)}$ шагов, поскольку существуют только $m!$ потенциальных гамильтоновых путей; таким образом, f в действительности "явная". Но никто не знает, как вычислить f за полиномиальное время или как доказать, что не существует цепочки длиной $4n$.

Хотя мы знаем, что могут существовать короткие булевы цепочки для f при каждом n. В конце концов, рис. 9 и 10 свидетельствуют о наличии чертовски хитрых цепочек даже для случаев 4 и 5 переменных. Эффективные цепочки для всех больших задач, которые нам когда-либо понадобится решать, вполне могут оказаться "не от мира сего" — полностью вне нашей досягаемости, поскольку у нас не хватит

времени, чтобы их найти. Даже если бы Всеведущий открыл нам простые цепочки, мы могли бы счесть их непостижимыми, поскольку кратчайшее доказательство их корректности может превышать по размеру количество клеток в нашем мозге.

Теорема S исключает такой сценарий для большинства булевых функций. Но во всей истории нашего мира практическую важность будут иметь менее 2^{100} булевых функций, и теорема S ничего не говорит нам о них.

Однако в 1974 году Ларри Стокмейер (Larry Stockmeyer) и Альберт Мейер (Albert Meyer) сумели построить булеву функцию f с доказуемо огромной сложностью. Их f не является "явной" в точном смысле, описанном выше, но и не является совершенно искусственной — она естественным образом возникает в математической логике. Рассмотрим символические высказывания, такие как

```
048+1015≠1063;                                                    (62)
∀m∃n(m<n+1);                                                      (63)
∀n∃m(m+1<n);                                                      (64)
∀a∀b(b≥a+2⇒∃ab(a<ab∧ab<b));                                       (65)
∀A∀B(A≡B⇔¬∃n(n∈A∧n∉B∨n∈B∧n∉A));                                    (66)
∀A(∃n(n∈A)⇒∃m(m∈A∧∀n(n∈A⇒m≤n)));                                  (67)
∀A(∃n(n∈A)⇒∃m(m∈A∧∀n(n∈A⇒m≥n)));                                  (68)
∃P∀a((a∈P⇔a+3∉P)⇔a<1000);                                         (69)
∀A∀B(∀C∀c(C≡A∧c=1∨C≡B∧c=0⇒(∀n(n∈C⇔n+1∈C)⇔c=1))⇒¬A≡B).             (70)
```

Стокмейер и Мейер определили язык L, используя 63-буквенный алфавит

```
∀∃¬()≡∈∉+∧∨⇒⇔<≤=≠≥>abcdefghijklmnopqABCDEFGHIJKLMNOPQ0123456789
```

с обычными значениями этих символов. Строки строчных букв в предложениях L наподобие '`ab`' в (65) представляют числовые переменные, ограниченные неотрицательными целыми значениями; строки прописных букв представляют множества переменных, ограниченные конечными множествами таких чисел. Например, (66) означает "для всех конечных множеств A и B мы имеем $A = B$ тогда и только тогда, когда не существует такого числа n, которое есть в A, но которого нет в B или которое есть в B и которого нет в A". Одни из этих утверждений истинны, другие — ложны. (См. упр. 82.)

Все строки (62)–(70) принадлежат L, но в действительности язык весьма ограничен: единственная разрешенная арифметическая операция над числом — это прибавление константы; мы можем записать '`a+13`', но не '`a+b`'. Единственным разрешенным отношением между числом и множеством являются отношения принадлежности (∈ или ∉). Единственным разрешенным отношением между множествами является эквивалентность (≡). Кроме того, все переменные должны быть квантифицированы с помощью ∃ или ∀.*

Каждое предложение L, имеющее длину $k \le n$, может быть представлено бинарным вектором длиной $6n$ с нулями в последних $6(n-k)$ битах. Пусть $f(x)$ — булева

* Технически говоря, предложения L принадлежат "слабой монадической логике второго порядка с одним преемником". Слабая логика второго порядка допускает квантификацию на конечных множествах; монадическая логика с k преемниками представляет собой теорию непомеченных k-арных деревьев.

функция от $6n$ переменных, такая, что $f(x) = 1$, если x представляет истинное утверждение L, и $f(x) = 0$, если x представляет ложное утверждение; значение $f(x)$ не определено, если x не является осмысленным предложением. Таблица истинности для такой функции f может быть построена за конечное количество шагов согласно теоремам Бюхи (Büchi) и Элгота (Elgot) [*Zeitschrift für math. Logik und Grundlagen der Math.* **6** (1960), 66–92; *Transactions of the Amer. Math. Soc.* **98** (1961), 21–51]. Но "конечное" не означает "осуществимое": Стокмейер и Мейер доказали, что

$$C(f) > 2^{r-5} \qquad \text{при } n \geq 460 + .302r + 5.08 \ln r \text{ и } r > 36. \tag{71}$$

В частности, мы имеем $C(f) > 2^{426} > 10^{128}$ при $n = 621$. *Булева цепочка с таким количеством логических элементов никогда не может быть построена*, поскольку 10^{128} — заведомо избыточная верхняя граница количества протонов во вселенной. Так что перед нами достаточно небольшая конечная задача, которая никогда не может быть решена.

Детали доказательства Стокмейера и Мейера приведены в *JACM* **49** (2002), 753–784. Основная идея заключается в том, что язык L, несмотря на его ограниченность, достаточно богат для описания таблиц истинности и сложности булевых цепочек с помощью достаточно коротких предложений; следовательно, f должен работать со входными данными, которые, по сути, ссылаются сами на себя.

***Для дальнейшего чтения.** Имеются тысячи статей, посвященных сетям логических ячеек, поскольку такие сети лежат в основе множества аспектов теории и практики. В этом разделе мы сосредоточимся в основном на темах, имеющих отношение к программированию последовательных машин. Однако основательно изучены и другие темы, в первую очередь — имеющие отношение к параллельным вычислениям, такие как схемы с малой глубиной, в которых логические элементы могут иметь любое количество входов ("неограниченный коэффициент объединения по входу"). Книга Инго Вегенера (Ingo Wegener) *The Complexity of Boolean Functions* (Teubner and Wiley, 1987) является хорошим введением во всю рассматриваемую тематику.

Мы в основном рассматривали булевы цепочки, в которых все бинарные операторы имели одинаковую важность. Для наших целей такие логические элементы, как $\oplus$ или $\overline{\subset}$, ничуть не более или менее желательны, чем логические элементы типа $\wedge$ или $\vee$. Но совершенно естественно задаться вопросом, можно ли при вычислении монотонной функции ограничиться только монотонными операторами $\wedge$ и $\vee$. Александр Разборов разработал поразительные методы доказательства, чтобы показать, что в действительности монотонные операторы сами по себе обладают, по сути, ограниченными возможностями. Например, он доказал, что все И-ИЛИ-цепочки для определения того, является ли перманент матрицы $n \times n$, состоящей из 0 и 1, нулевым или единичным, должны иметь стоимость $n^{\Omega(\log n)}$. [См. *Доклады Академии наук СССР* **281** (1985), 798–801; *Математические заметки* **37** (1985), 887–900.] В разделе же 7.5.1 мы видели, что эта задача, эквивалентная "двудольным паросочетаниям", разрешима всего за $O(n^{2.5})$ шагов. Кроме того, эффективные методы из этого раздела могут быть реализованы как булевы цепочки лишь немного большей стоимости, если мы разрешим отрицания или иные булевы операции в дополнение к $\wedge$ и $\vee$. (Воган Пратт (Vaughan Pratt) назвал это "силой отрицательного мышления".) Введение в методы Разборова представлено в упр. 85 и 86.

Упражнения

1. [*24*] "Случайная" функция в формуле (6) соответствует булевой цепочке стоимостью 4 и глубиной 4. Найдите формулу с глубиной 3, имеющую ту же стоимость.

2. [*21*] Покажите, как вычислить (а) $w \oplus \langle xyz \rangle$ и (б) $w \wedge \langle xyz \rangle$ с помощью формул с глубиной 3 и стоимостью 5.

3. [*M23*] (Б. И. Фиников, 1957.) Докажите, что если булева функция $f(x_1, \ldots, x_n)$ истинна ровно в k точках, то $L(f) < 2n + (k-2)2^{k-1}$. *Указание*: подумайте о $k = 3$ и $n = 10^6$.

4. [*M28*] Докажите, что минимальные глубина и длина формулы булевой функции удовлетворяют условию $\lg L(f) < D(f) < e\alpha \lg L(f)$ при $L(f) > 1$, где $\alpha = 1/\lg \chi \approx 2.465965$ связана с "константой пластичности" χ из 7.1.4–(90). *Указание*: если f содержит подформулу g, то мы имеем $f = g$? f_1: f_0 для подходящих f_1 и f_0.

▶ **5.** [*21*] Пороговая функция Фибоначчи $F_n(x_1, \ldots, x_n) = \langle x_1^{F_1} x_2^{F_2} \ldots x_{n-1}^{F_{n-1}} x_n^{F_{n-2}} \rangle$ анализировалась в упр. 7.1.1–101 для $n \geq 3$. Имеется ли эффективный способ ее вычисления?

6. [*20*] Истинно или ложно утверждение: булева функция $f(x_1, \ldots, x_n)$ является нормальной тогда и только тогда, когда она удовлетворяет обобщенному закону дистрибутивности $f(x_1, \ldots, x_n) \wedge y = f(x_1 \wedge y, \ldots, x_n \wedge y)$?

7. [*20*] Преобразуйте булеву цепочку '$x_5 = x_1 \overline{\vee} x_4$, $x_6 = \bar{x}_2 \vee x_5$, $x_7 = \bar{x}_1 \wedge \bar{x}_3$, $x_8 = x_6 \equiv x_7$' в эквивалентную цепочку $(\hat{x}_5, \hat{x}_6, \hat{x}_7, \hat{x}_8)$, в которой каждый шаг является нормальным.

▶ **8.** [*20*] Поясните, почему (11) представляет собой таблицу истинности переменной x_k.

9. [*20*] Алгоритм L определяет длины кратчайших формул для всех функций f, но не предоставляет иной информации. Расширьте алгоритм таким образом, чтобы он предоставлял также фактические формулы с минимальной длиной наподобие (6).

▶ **10.** [*20*] Модифицируйте алгоритм L так, чтобы он вычислял $D(f)$ вместо $L(f)$.

▶ **11.** [*22*] Модифицируйте алгоритм L так, чтобы вместо длин $L(f)$ он вычислял верхние границы $U(f)$ и отпечатки $\phi(f)$, как описано в разделе.

12. [*15*] Какая булева цепочка эквивалентна схеме с минимальным использованием памяти (13)?

13. [*16*] Каковы таблицы истинности f_1, f_2, f_3, f_4 и f_5 в примере (13)?

14. [*22*] Какой имеется удобный способ вычислить $5n(n-1)$ таблиц истинности (17) при заданной таблице истинности для g? (Используйте побитовые операции, как в (15) и (16).)

15. [*28*] Найдите наикратчайшие возможные пути вычисления следующих функций с минимальным использованием памяти: (а) $S_1(x_1, x_2, x_3)$; (б) $S_2(x_1, x_2, x_3, x_4)$; (в) $S_1(x_1, x_2, x_3, x_4)$; (г) функция из (18).

16. [*HM33*] Докажите, что менее чем 2^{118} из 2^{128} булевых функций $f(x_1, \ldots, x_7)$ вычислимы с использованием минимального количества памяти.

▶ **17.** [*25*] (М. С. Патерсон (M. S. Paterson), 1977.) Хотя булевы функции $f(x_1, \ldots, x_n)$ не могут всегда быть вычислены с использованием n регистров, докажите, что $n+1$ регистров всегда достаточно. Другими словами, покажите, что для вычисления $f(x_1, \ldots, x_n)$ всегда имеется последовательность операций наподобие (13), если разрешить $0 \leq j(i), k(i) \leq n$.

▶ **18.** [*35*] Исследуем оптимальные вычисления с использованием минимального количества памяти для $f(x_1, x_2, x_3, x_4, x_5)$. Сколько классов функций от пяти переменных имеют $C_m(f) = r$ для $r = 0, 1, 2, \ldots$?

19. [*M22*] Докажите, что если булева цепочка использует n переменных и имеет длину $r < n + 2$, то она должна быть либо "нисходящим", либо "восходящим" построением.

▶ **20.** [*40*] (Р. Шрёппель (R. Schroeppel), 2004.) Булева цепочка является *канализирующей*, если она не использует операторов $\oplus$ или $\equiv$. Найдите оптимальную стоимость, длину и глубину всех функций от четырех переменных при данном ограничении. Дает ли эвристика на основе отпечатков оптимальные результаты?

21. [*46*] Для какого количества функций от четырех переменных исследователи из Гарварда нашли оптимальные схемы на электровакуумных лампах в 1951 году?

22. [*21*] Поясните цепочку для S_3 на рис. 10, обратив внимание на то, что она включает цепочку для $S_{2,3}$ на рис. 9. Найдите аналогичную цепочку для $S_2(x_1, x_2, x_3, x_4, x_5)$.

▶ **23.** [*23*] На рис. 10 показаны только 16 из 64 симметричных функций от пяти переменных. Поясните, как написать оптимальные цепочки для других функций.

24. [*47*] Для каждой ли симметричной функции f справедливо соотношение $C_m(f) = C(f)$?

▶ **25.** [*17*] Предположим, нам нужна булева цепочка, которая включает *все* функции от n переменных: пусть $f_k(x_1, \ldots, x_n)$ является функцией, таблица истинности которой — бинарное представление k для $0 \le k < m = 2^{2^n}$. Что собой представляет $C(f_0 f_1 \ldots f_{m-1})$?

26. [*25*] Истинно или ложно утверждение: если $f(x_0, \ldots, x_n) = (x_0 \wedge g(x_1, \ldots, x_n)) \oplus h(x_1, \ldots, x_n)$, где g и h представляют собой нетривиальные булевы функции, суммарная стоимость которых равна $C(gh)$, то $C(f) = 2 + C(gh)$?

▶ **27.** [*23*] Можно ли реализовать полный сумматор (22) пятью шагами с использованием минимального количества памяти (т. е. целиком в пределах трех однобитных регистров)?

28. [*26*] Докажите, что $C(u'v') = C(u''v'') = 5$ для булевых функций с двумя выходами, определяемых следующим образом:

$$(u'v')_2 = (x + y - (uv)_2) \bmod 4, \qquad (u''v'')_2 = (-x - y - (uv)_2) \bmod 4.$$

Воспользуйтесь этими функциями для вычисления $[(x_1 + \cdots + x_n) \bmod 4 = 0]$ менее чем за $2.5n$ шагов.

29. [*M28*] Докажите, что приведенная в разделе схема для контрольного суммирования (27) имеет длину $O(\log n)$.

30. [*M25*] Решите бинарное рекуррентное соотношение (28) для стоимости $s(n)$ контрольного суммирования.

31. [*21*] Докажите, что если функция $f(x_1, \ldots, x_n)$ симметрична, то выполняется соотношение $C(f) \le 5n + O(n/\log n)$.

32. [*HM16*] Почему решение (30) удовлетворяет $t(n) = 2^n + O(2^{n/2})$?

33. [*HM22*] Истинно или ложно утверждение: если $1 \le N \le 2^n$, то первые N минитермов $\{x_1, \ldots, x_n\}$ могут быть вычислены за $N + O(\sqrt{N})$ шагов при $n \to \infty$ и $N \to \infty$.

▶ **34.** [*22*] *Приоритетный шифратор* имеет $n = 2^m - 1$ входов $x_1 \ldots x_n$ и m выходов $y_1 \ldots y_m$, где $(y_1 \ldots y_m)_2 = k$ тогда и только тогда, когда $k = \max\{j \mid j = 0 \text{ или } x_j = 1\}$. Разработайте приоритетный шифратор со стоимостью $O(n)$ и с глубиной $O(m)$.

35. [*23*] Покажите, что если $n > 1$, то все конъюнкции $x_1 \wedge \cdots \wedge x_{k-1} \wedge x_{k+1} \wedge \cdots \wedge x_n$ для $1 \le k \le n$ могут быть вычислены из $(x_1, \ldots, x_n)$ с общей стоимостью $\le 3n - 6$.

▶ **36.** [*M28*] (Р. Э. Ладнер (R. E. Ladner) и М. Д. Фишер (M. J. Fischer), 1980.) Пусть y_k представляет собой "префикс" $x_1 \wedge \cdots \wedge x_k$ для $1 \le k \le n$. Ясно, что $C(y_1 \ldots y_n) = n - 1$ и $D(y_1 \ldots y_n) = \lceil \lg n \rceil$; но мы не можем одновременно минимизировать и стоимость, и глубину. Найдите цепочку оптимальной глубины $\lceil \lg n \rceil$ со стоимостью $< 4n$.

37. [*M28*] (Марк Снир (Marc Snir), 1986.) Для заданных $n \geq m \geq 1$ рассмотрим следующий алгоритм.

S1. [Восходящий цикл.] Для $t \leftarrow 1, 2, \ldots, \lceil \lg m \rceil$ установить $x_{\min(m,2^t k)} \leftarrow x_{2^t(k-1/2)} \wedge x_{\min(m,2^t k)}$ для $k \geq 1$ и $2^t(k-1/2) < m$.

S2. [Нисходящий цикл.] Для $t \leftarrow \lceil \lg m \rceil - 1, \lceil \lg m \rceil - 2, \ldots, 1$ установить $x_{2^t(k+1/2)} \leftarrow x_{2^t k} \wedge x_{2^t(k+1/2)}$ для $k \geq 1$ и $2^t(k+1/2) < m$.

S3. [Расширение.] Для $k \leftarrow m+1, m+2, \ldots, n$ установить $x_k \leftarrow x_{k-1} \wedge x_k$. ▮

а) Докажите, что этот алгоритм решает задачу префиксов из упр. 36: он преобразует $(x_1, x_2, \ldots, x_n)$ в $(x_1, x_1 \wedge x_2, \ldots, x_1 \wedge x_2 \wedge \cdots \wedge x_n)$.

б) Пусть $c(m,n)$ и $d(m,n)$ представляют собой стоимость и глубину соответствующей булевой цепочки. Докажите для фиксированного m, что если n достаточно велико, то $c(m,n) + d(m,n) = 2n - 2$.

в) Чему для заданного $n > 1$ равна величина $d(n) = \min_{1 \leq m \leq n} d(m,n)$? Покажите, что $d(n) < 2 \lg n$.

г) Докажите, что существует булева цепочка со стоимостью $2n - 2 - d$ и глубиной d для задачи префиксов при $d(n) \leq d < n$. (Эта стоимость оптимальна согласно упр. 81.)

38. [*25*] В разделе 5.3.4 мы изучали *сортирующие сети*, в которых $\hat{S}(n)$ компараторов могут сортировать n чисел $(x_1, x_2, \ldots, x_n)$ в порядке возрастания. Если входные данные x_j представляют собой нули и единицы, каждый компаратор эквивалентен двум логическим элементам $(x \wedge y, x \vee y)$; так что сортирующая сеть соответствует некоторому виду булевой цепочки, которая вычисляет n функций от $(x_1, x_2, \ldots, x_n)$.

а) Что собой представляют n функций $f_1 f_2 \ldots f_n$, которые вычисляет сортирующая сеть?

б) Покажите, что эти функции $\{f_1, f_2, \ldots, f_n\}$ могут быть вычислены за $O(n)$ шагов цепочкой глубиной $O(\log n)$. (Следовательно, сортирующие сети не являются асимптотически оптимальными в булевом случае.)

▶ **39.** [*M21*] (М. С. Патерсон (M. S. Paterson) и П. Клайн (P. Klein), 1980.) Реализуйте 2^m-канальный мультиплексор $M_m(x_1, \ldots, x_m; y_0, y_1, \ldots, y_{2^m-1})$ из (31) с помощью эффективной цепочки, которая одновременно устанавливает верхние границы $C(M_m) \leq 2n + O(\sqrt{n})$ и $D(M_m) \leq m + O(\log m)$.

40. [*25*] Пусть при $n \geq k \geq 1$ функция $f_{nk}(x_1, \ldots, x_n)$ представляет собой функцию "k единиц подряд"

$$(x_1 \wedge \cdots \wedge x_k) \vee (x_2 \wedge \cdots \wedge x_{k+1}) \vee \cdots \vee (x_{n+1-k} \wedge \cdots \wedge x_n).$$

Покажите, что стоимость $C(f_{nk})$ этой функции меньше $4n - 3k$.

41. [*M23*] (*Сумматоры с условным суммированием.*) Одним из способов выполнить бинарное сложение (25) с глубиной $O(\log n)$ основан на трюке с мультиплексорами из упр. 4: если $(xx')_2 + (yy')_2 = (zz')_2$, где $|x'| = |y'| = |z'|$, мы имеем либо $(x)_2 + (y)_2 = (z)_2$ и $(x')_2 + (y')_2 = (z')_2$, либо $(x)_2 + (y)_2 + 1 = (z)_2$ и $(x')_2 + (y')_2 = (1z')_2$. Чтобы сэкономить время, мы можем вычислить *оба* значения, $(x)_2 + (y)_2$ и $(x)_2 + (y)_2 + 1$, одновременно при вычислении $(x')_2 + (y')_2$. Позже, когда мы узнаем, генерирует ли младшая часть $(x')_2 + (y')_2$ перенос, мы сможем воспользоваться мультиплексорами для выбора корректных битов для старшей части.

Если этот метод рекурсивно используется для построения $2n$-битовых сумматоров на основе n-битовых, то сколько при этом требуется логических элементов при $n = 2^m$? Чему равна соответствующая глубина?

42. [*30*] Положим в бинарном сложении (25) $u_k = x_k \wedge y_k$ и $v_k = x_k \oplus y_k$ для $0 \leq k < n$.

а) Покажите, что $z_k = v_k \oplus c_k$, где биты переноса c_k удовлетворяют условию

$$c_k = u_{k-1} \vee (v_{k-1} \wedge (u_{k-2} \vee (v_{k-2} \wedge (\cdots (v_1 \wedge u_0) \cdots)))).$$

б) Пусть $U_k^k = 0$, $V_k^k = 1$ и $U_j^{k+1} = u_k \vee (v_k \wedge U_j^k)$, $V_j^{k+1} = v_k \wedge V_j^k$ для $k \geq j$. Докажите, что $c_k = U_0^k$ и что $U_i^k = U_j^k \vee (V_j^k \wedge U_i^j)$, $V_i^k = V_j^k \wedge V_i^j$ для $i \leq j \leq k$.

в) Пусть $h(m) = 2^{m(m-1)/2}$. Покажите, что, когда $n = h(m)$, все переносы $c_1, \ldots, c_n$ могут быть вычислены с глубиной $(m+1)m/2 \approx \lg n + \sqrt{2 \lg n}$ и с общей стоимостью $O(2^m n)$.

▶ **43.** [*28*] *Конечный преобразователь* представляет собой абстрактную машину с конечным входным алфавитом A, конечным выходным алфавитом B и конечным множеством внутренних состояний Q. Одно из этих состояний, q_0, называется начальным. Для заданной строки $\alpha = a_1 \ldots a_n$, где каждое $a_j \in A$, машина вычисляет строку $\beta = b_1 \ldots b_n$, где каждое $b_j \in B$, следующим образом.

T1. [Инициализация.] Установить $j \leftarrow 1$ и $q \leftarrow q_0$.

T2. [Выполнено?] Завершить работу алгоритма, если $j > n$.

T3. [Вывод b_j.] Установить $b_j \leftarrow c(q, a_j)$.

T4. [Продвижение j.] Установить $q \leftarrow d(q, a_j)$, $j \leftarrow j + 1$ и вернуться к шагу T2. ▌

Машина имеет встроенные инструкции, которы определяют $c(q, a) \in B$ и $d(q, a) \in Q$ для каждого состояния $q \in Q$ и каждого символа $a \in A$. Цель этого упражнения — показать, что если алфавиты A и B любого конечного преобразователя закодировать в бинарном виде, то строка β может быть вычислена из α с помощью булевой цепочки размером $O(n)$ и глубиной $O(\log n)$.

а) Рассмотрим задачу изменения бинарного вектора $a_1 \ldots a_n$ в $b_1 \ldots b_n$ путем присвоений

$$b_j \;\leftarrow\; a_j \oplus [a_j = a_{j-1} = \cdots = a_{j-k} = 1 \text{ и } a_{j-k-1} = 0, \text{ где } k \geq 1 \text{ нечетно}]$$

в предположении, что $a_0 = 0$. Например, $\alpha = 110010010001111110110101 0 \mapsto \beta = 1000100100010101001001010$. Докажите, что это преобразование может быть выполнено конечным преобразователем, у которого $|A| = |B| = |Q| = 2$.

б) Предположим, что конечный преобразователь с $|Q| = 2$ находится в состоянии q_j после прочтения $a_1 \ldots a_{j-1}$. Поясните, как вычислить последовательность $q_1 \ldots q_n$ с помощью булевой цепочки стоимостью $O(n)$ и глубиной $O(\log n)$ с использованием построения Р. Э. Ладнера (R. E. Ladner) и М. Д. Фишера (M. J. Fischer) из упр. 36. (Из этой последовательности $q_1 \ldots q_n$ легко вычислить $b_1 \ldots b_n$, поскольку $b_j = c(q_j, a_j)$.)

в) Примените метод из п. (б) к решению задачи из п. (а).

▶ **44.** [*26*] (Р. Э. Ладнер (R. E. Ladner) и М. Д. Фишер (M. J. Fischer), 1980.) Покажите, что задача бинарного суммирования (25) может рассматриваться как конечное преобразование. Опишите булеву цепочку, которая является результатом построения из упр. 43 при $n = 2^m$, и сравните ее с сумматором с условным суммированием из упр. 41.

45. [*HM20*] Почему в доказательстве теоремы S не использовано простое рассуждение о том, что количество способов выбрать такие $j(i)$ и $k(i)$, что $1 \leq j(i), k(i) < i$, равно $n^2(n+1)^2 \ldots (n+r-1)^2$?

▶ **46.** [*HM21*] Пусть $\alpha(n) = c(n, \lfloor 2^n/n \rfloor)/2^{2^n}$ — доля булевых функций от n переменных $f(x_1, \ldots, x_n)$, для которых $C(f) \leq 2^n/n$. Докажите, что $\alpha(n) \to 0$ достаточно быстро при $n \to \infty$.

47. [*M23*] Распространите теорему S на функции с n входами и m выходами.

48. [*HM23*] Найдите наименьшее целое число $r = r(n)$, такое, что $(r-1)!\,2^{2^n} \leq 2^{2r+1} \times (n+r-1)^{2r}$: (а) в точности при $1 \leq n \leq 16$; (б) асимптотически при $n \to \infty$.

49. [*HM25*] Докажите, что при $n \to \infty$ почти все булевы функции $f(x_1, \ldots, x_n)$ имеют минимальную длину формулы $L(f) > 2^n/\lg n - 2^{n+2}/(\lg n)^2$.

50. [*24*] Что собой представляют простые импликанты и простые дизъюнкты функции простых чисел (35)? Выразите эту функцию в виде (а) дизъюнктивной нормальной формы; (б) конъюнктивной нормальной формы минимальной длины.

51. [*20*] Какое представление детектора простых чисел будет получено вместо (37), если строки таблицы истинности будут основаны на x_5x_6, а не на x_1x_2?

52. [*23*] Какой выбор k и l минимизирует верхнюю границу (38) при $5 \le n \le 16$?

53. [*HM22*] Оцените (38) при $k = \lfloor 2\lg n \rfloor$ и $l = \lceil 2^k/(n - 3\lg n) \rceil$ и $n \to \infty$.

54. [*29*] Найдите короткую булеву цепочку для вычисления всех шести функций $f_j(x) = [x_1x_2x_3x_4 \in A_j]$, где $A_1 = \{0010, 0101, 1011\}$, $A_2 = \{0001, 1111\}$, $A_3 = \{0011, 0111, 1101\}$, $A_4 = \{1001, 1111\}$, $A_5 = \{1101\}$, $A_6 = \{0101, 1011\}$. (Эти шесть функций появляются в детекторе простых чисел (37).) Сравните свою цепочку со схемой первоначального вычисления минитермов общего метода Лупанова.

55. [*34*] Покажите, что стоимость шестибитовой функции определения простых чисел не превышает 14.

▶ **56.** [*16*] Поясните, почему все функции с 14 или более безразличными значениями в табл. 3 имеют нулевую стоимость.

57. [*19*] Какие семисегментные "цифры" выводятся при $(x_1x_2x_3x_4)_2 > 9$ в (45)?

▶ **58.** [*30*] 4×4-битовый *S-блок* представляет собой перестановку 4-битовых векторов $\{0000, 0001, \ldots, 1111\}$; такие перестановки используются в хорошо известных криптографических системах, таких, как советский стандарт симметричного шифрования ГОСТ 28147-89 (1989). Каждый 4×4-битовый S-блок соответствует последовательности из четырех функций $f_1(x_1, x_2, x_3, x_4)$, ..., $f_4(x_1, x_2, x_3, x_4)$, которые выполняют преобразование $x_1x_2x_3x_4 \mapsto f_1f_2f_3f_4$. Найдите все 4×4-битовые S-блоки, для которых $C(f_1) = C(f_2) = C(f_3) = C(f_4) = 7$.

59. [*29*] Один из S-блоков, удовлетворяющий условиям упр. 58, выполняет отображение $(\mathtt{0}, \ldots, \mathtt{f}) \mapsto (\mathtt{0}, \mathtt{6}, \mathtt{5}, \mathtt{b}, \mathtt{3}, \mathtt{9}, \mathtt{f}, \mathtt{e}, \mathtt{c}, \mathtt{4}, \mathtt{7}, \mathtt{8}, \mathtt{d}, \mathtt{2}, \mathtt{a}, \mathtt{1})$; другими словами, таблицами истинности (f_1, f_2, f_3, f_4) являются соответственно $(\mathtt{179a}, \mathtt{63e8}, \mathtt{5b26}, \mathtt{3e29})$. Найдите булеву цепочку, которая вычисляет эти четыре "максимально трудные" функции менее чем за 20 шагов.

60. [*23*] (Фрэнк Раски (Frank Ruskey).) Предположим, что $z = (x + y) \bmod 3$, где $x = (x_1x_2)_2$, $y = (y_1y_2)_2$, $z = (z_1z_2)_2$ и каждое двухбитовое значение должно быть равно 00, 01 либо 10. Вычислите z_1 и z_2 из x_1, x_2, y_1 и y_2 за шесть булевых шагов.

61. [*34*] Продолжая выполнять упр. 60, найдите эффективный способ вычисления $z = (x + y) \bmod 5$ с использованием трехбитовых значений 000, 001, 010, 011, 100.

62. [*HM23*] Рассмотрим случайную булеву частично определенную функцию от n переменных, которая имеет 2^nc "интересующих" (определенных) и 2nd "безразличных" значений, где $c + d = 1$. Докажите, что стоимость почти всех таких частично определенных функций превышает $2^nc/n$.

63. [*HM35*] (Л. А. Шоломов, 1969.) Продолжая упр. 62, докажите, что все такие функции имеют стоимость $\le 2^nc/n(1 + O(n^{-1}\log n))$. *Указание*: имеется множество $2^m(1 + k)$ векторов $x_1 \ldots x_k$, которое пересекается с каждым $(k - m)$-мерным подкубом k-мерного куба.

64. [*25*] (*Волшебное пятнадцать.*) Два игрока по очереди выбирают цифры от 1 до 9, не используя никакую цифру дважды; победителем, если таковой имеется, становится тот, кто первым выберет три цифры, сумма которых равна 15. Какой должна быть правильная стратегия игрока в эту игру?

▶ **65.** [*35*] Модифицируйте стратегию игры в крестики-нулики из (47)–(56) так, чтобы она всегда играла корректно.

66. [*20*] Раскритикуйте ходы, которые делаются в упр. 65. Всегда ли они оптимальны?

▶ **67.** [*40*] Вместо простого поиска одного корректного хода для каждой позиции в игре в крестики-нулики мы можем предпочесть находить их все. Другими словами, для заданных $x_1 \ldots x_9 o_1 \ldots o_9$ мы можем попытаться вычислить девять выходов $g_1 \ldots g_9$, где $g_j = 1$ тогда и только тогда, когда ход в ячейку j корректен и минимизирует худший исход для ×. Например, восклицательными знаками помечены все корректные ходы × в следующих типичных позициях:

Играть с машиной, которая выбирает ход среди таких возможных вариантов, интереснее, чем с машиной, которая имеет единственную жестко фиксированную стратегию.

Одним привлекательным способом решения задачи всех допустимых ходов является использование того факта, что крестики-нулики имеют восемь симметрий. Представим микросхему с 18 входами $x_1 \ldots x_9 o_1 \ldots o_9$ и тремя выходами (c, s, m) для "угла", "боковой клетки" и "средины", обладающую тем свойством, что требуемые функции g_j могут вычисляться с помощью соответствующего соединения восьми таких микросхем:

$$\begin{aligned}
g_1 &= c(x_1x_2x_3x_4x_5x_6x_7x_8x_9o_1o_2o_3o_4o_5o_6o_7o_8o_9)\\
&\qquad \vee\, c(x_1x_4x_7x_2x_5x_8x_3x_6x_9o_1o_4o_7o_2o_5o_8o_3o_6o_9),\\
g_2 &= s(x_1x_2x_3x_4x_5x_6x_7x_8x_9o_1o_2o_3o_4o_5o_6o_7o_8o_9)\\
&\qquad \vee\, s(x_3x_2x_1x_6x_5x_4x_9x_8x_7o_3o_2o_1o_6o_5o_4o_9o_8o_7),\\
g_3 &= c(x_3x_2x_1x_6x_5x_4x_9x_8x_7o_3o_2o_1o_6o_5o_4o_9o_8o_7)\\
&\qquad \vee\, c(x_3x_6x_9x_2x_5x_8x_1x_4x_7o_3o_6o_9o_2o_5o_8o_1o_4o_7),\\
g_4 &= s(x_1x_4x_7x_2x_5x_8x_3x_6x_9o_1o_4o_7o_2o_5o_8o_3o_6o_9)\\
&\qquad \vee\, s(x_7x_4x_1x_8x_5x_2x_9x_6x_3o_7o_4o_1o_8o_5o_2o_9o_6o_3), \qquad \ldots\\
g_9 &= c(x_9x_8x_7x_6x_5x_4x_3x_2x_1o_9o_8o_7o_6o_5o_4o_3o_2o_1)\\
&\qquad \vee\, c(x_9x_6x_3x_8x_5x_2x_7x_4x_1o_9o_6o_3o_8o_5o_2o_7o_4o_1),
\end{aligned}$$

а g_5 представляет собой ИЛИ m выходов от всех восьми микросхем.

Разработайте логику такой микросхемы, используя менее 2000 логических элементов.

68. [*M25*] Рассмотрим n-битовую π-функцию $\pi_n(x_1 \ldots x_n)$, значение которой представляет собой $(x_1 \ldots x_n)_2$-й бит справа от старшего бита в бинарном представлении π. Доказывает ли метод из упр. 4.3.1–39, описывающий эффективный способ вычисления произвольных битов π, что $C(\pi_n) < 2^n/n$ для достаточно большого n?

69. [*M24*] Пусть мультилинейное представление f имеет вид

$$\alpha_{000} \oplus \alpha_{001}x_m \oplus \alpha_{010}x_j \oplus \alpha_{011}x_jx_m \oplus \alpha_{100}x_i \oplus \alpha_{101}x_ix_m \oplus \alpha_{110}x_ix_j \oplus \alpha_{111}x_ix_jx_m,$$

где каждый коэффициент α_l является функцией от переменных $\{x_1, \ldots, x_n\} \setminus \{x_i, x_j, x_m\}$.

а) Докажите, что пары (58) являются "хорошими" тогда и только тогда, когда коэффициенты удовлетворяют соотношениям

$$\alpha_{010}\alpha_{101} = \alpha_{011}\alpha_{100}, \quad \alpha_{101}\alpha_{110} = \alpha_{100}\alpha_{111} \quad \text{и} \quad \alpha_{110}\alpha_{011} = \alpha_{111}\alpha_{010}.$$

б) Для каких значений (i, j, m) пары оказываются плохими, когда $f = (\det X) \bmod 2$? (См. (60).)

▶ **70.** [*M27*] Пусть X — булева матрица (60) размером 3×3. Найдите эффективные цепочки для булевых функций (а) $(\det X) \bmod 2$; (б) $[\operatorname{per} X > 0]$; (в) $[\det X > 0]$.

▶ **71.** [*M26*] Предположим, что $f(x)$ равна 0 с вероятностью p в каждой точке $x = x_1 \ldots x_n$ независимо от ее значений в других точках.

а) Какова вероятность того, что пары (58) являются хорошими?
б) Какова вероятность того, что плохие пары (58) существуют?
в) Какова вероятность того, что плохие пары (58) будут найдены не более чем при t случайных испытаниях?
г) Каково ожидаемое время проверки случая (i, j, m) как функция от p, t и n?

72. [*M24*] Распространите предыдущее упражнение на случай частично определенных функций, где $f(x) = 0$ с вероятностью p, $f(x) = 1$ с вероятностью q и $f(x) = *$ с вероятностью r.

▶ **73.** [*20*] Покажите, что если плохие пары (58) существуют для всех (i, j, m), таких, что $m \neq i \neq j \neq m$, то неразложимость f может быть выведена после тестирования всего лишь $\binom{n}{2}$ тщательно отобранных троек (i, j, m).

74. [*25*] Расширьте идею из предыдущего упражнения, предложив стратегию для выбора последовательных троек (i, j, m) при использовании метода Шена (Shen), Мак-Келлара (McKellar) и Вайнера (Weiner).

75. [*20*] Что получится при применении описанной в разделе процедуры разложения к функции $S_{0,n}(x_1, \ldots, x_n)$?

▶ **76.** [*M26*] (Д. Улиг (D. Uhlig), 1974.) Целью данного упражнения является доказательство того удивительного факта, что для некоторых функций f лучшая цепочка для вычисления булевой функции

$$F(u_1, \ldots, u_n, v_1, \ldots, v_n) \;=\; f(u_1, \ldots, u_n) \vee f(v_1, \ldots, v_n)$$

сто́ит *меньше* чем $2C(f)$; следовательно, функциональное разложение *не* всегда является хорошей идеей.

Положим $n = m + 2^m$ и запишем $f(i_1, \ldots, i_m, x_0, \ldots, x_{2^m-1}) = f_i(x)$, где i рассматривается как число $(i_1 \ldots i_m)_2$. Тогда $(u_1, \ldots, u_n) = (i_1, \ldots, i_m, x_0, \ldots, x_{2^m-1})$, $(v_1, \ldots, v_n) = (j_1, \ldots, j_m, y_0, \ldots, y_{2^m-1})$ и $F(u, v) = f_i(x) \vee f_j(y)$.

а) Докажите, что цепочки стоимостью $O(n^2)$ достаточно для вычисления $2^m + 1$ функций

$$z_l \;=\; x \oplus \big(([l \le i] \oplus [i \le j]) \wedge (x \oplus y)\big), \qquad 0 \le l \le 2^m,$$

для заданных векторов i, j, x и y; каждое z_l представляет собой вектор длиной 2^m, а однобитовая величина $([l \le i] \oplus [i \le j])$ использована в операции И с каждым из компонентов $x \oplus y$.

б) Пусть $g_i(x) = f_i(x) \oplus f_{i-1}(x)$ для $0 \le i \le 2^m$, где $f_{-1}(x) = f_{2^m}(x) = 0$. Оцените стоимость вычисления $2^m + 1$ значений $c_l = g_l(z_l)$ для заданных векторов z_l, для $0 \le l \le 2^m$.

в) Пусть $c'_l = c_l \wedge ([i \le j] \equiv [l \le i])$ и $c''_l = c_l \wedge ([i \le j] \equiv [j > l])$. Докажите, что

$$f_i(x) = c'_0 \oplus c'_1 \oplus \cdots \oplus c'_{2^m}, \qquad f_j(y) = c''_0 \oplus c''_1 \oplus \cdots \oplus c''_{2^m}.$$

г) Сделайте вывод, что $C(F) \le 2^n/n + O(2^n(\log n)/n^2)$. (Когда n достаточно велико, эта стоимость, определенно, меньше, чем $2^{n+1}/n$, но существуют функции f с $C(f) > 2^n/n$.)

д) Для ясности выпишите цепочку для F при $m = 1$ и $f(i, x_0, x_1) = (i \wedge x_0) \vee x_1$.

▶ **77.** [*35*] (Н. П. Редькин, 1970.) Предположим, что булева цепочка использует только операции И, ИЛИ и НЕ; таким образом, каждый шаг представляет собой $x_i = x_{j(i)} \wedge x_{k(i)}$ либо $x_i = x_{j(i)} \vee x_{k(i)}$, либо $x_i = \bar{x}_{j(i)}$. Докажите, что если такая цепочка вычисляет либо функцию "нечетной четности" $f_n(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$, либо функцию "четной четности" $\bar{f}_n(x_1, \ldots, x_n) = 1 \oplus x_1 \oplus \cdots \oplus x_n$, где $n \ge 2$, то длина цепочки оказывается не меньше $4(n-1)$.

78. [*26*] (В. Я. Поль (W. J. Paul), 1977.) Пусть $f(x_1,\ldots,x_m,y_0,\ldots,y_{2^m-1})$ является произвольной булевой функцией, которая равна y_k при $(x_1\ldots x_m)_2 = k \in S$, для некоторого заданного множества $S \subseteq \{0,1,\ldots,2^m-1\}$; значения f в других точках нам безразличны. Покажите, что $C(f) \geq 2\|S\| - 2$, когда S — непустое множество. (В частности, когда $S = \{0,1,\ldots,2^m-1\}$, мультиплексорная цепочка из упр. 39 является асимптотически оптимальной.)

79. [*32*] (К. П. Шнорр (C. P. Schnorr), 1976.) Будем говорить, что переменные u и v являются "напарниками" в булевой цепочке, если существует ровно один простой путь между ними в соответствующей диаграмме бинарного дерева. Две переменные могут быть напарниками, только если каждая из них используется в цепочке ровно один раз; но этого необходимого условия недостаточно. Например, переменные 2 и 4 являются напарниками в цепочке для $S_{1,2,3}$ на рис. 9, но они не являются напарниками в цепочке для S_2.

а) Докажите, что булева цепочка от n переменных без напарников имеет стоимость $\geq 2n-2$.

б) Докажите, что $C(f) = 2n-3$, если f представляет собой функцию "все равны" $S_{0,n}(x_1,\ldots,x_n)$.

▶ **80.** [*M29*] (Л. Д. Стокмейер (L. J. Stockmeyer), 1977.) Иногда удобно еще одно обозначение для симметричных функций: если $\alpha = a_0a_1\ldots a_n$ представляет собой произвольную бинарную строку, обозначим $S_\alpha(x) = a_{\nu x}$. Например, в этих обозначениях $\langle x_1x_2x_3\rangle = S_{0011}$, а $x_1 \oplus x_2 \oplus x_3 = S_{0101}$. Обратите внимание, что $S_\alpha(0,x_2,\ldots,x_n) = S_{\alpha'}(x_2,\ldots,x_n)$ и $S_\alpha(1,x_2,\ldots,x_n) = S_{'\alpha}(x_2,\ldots,x_n)$, где α' и $'\alpha$ обозначают соответственно α с удаленным последним и первым элементами. Также

$$S_\alpha(f(x_3,\ldots,x_n),\bar f(x_3,\ldots,x_n),x_3,\ldots,x_n) \;=\; S_{'\alpha'}(x_3,\ldots,x_n),$$

если f представляет собой произвольную булеву функцию от $n-2$ переменных.

а) Функция четности имеет $a_0 \neq a_1 \neq a_2 \neq \cdots \neq a_n$. Будем считать, что $n \geq 2$. Докажите, что если S_α не является функцией четности и $S_{'\alpha'}$ не является константой, то

$$C(S_\alpha) \;\geq\; \max\bigl(C(S_{\alpha'})+2,\, C(S_{'\alpha})+2,\, \min\bigl(C(S_{\alpha'})+3,\, C(S_{'\alpha})+3,\, C(S_{'\alpha'})+5\bigr)\bigr).$$

б) Какие нижние границы $C(S_k)$ и $C(S_{\geq k})$ следуют из этого результата при $0 \leq k \leq n$?

81. [*23*] (Марк Снир (Marc Snir), 1986.) Покажите, что любая цепочка со стоимостью c и глубиной d для задачи префиксов из упр. 36 имеет $c + d \geq 2n-2$.

▶ **82.** [*M23*] Поясните логические высказывания (62)–(70). Какие из них истинны?

83. [*21*] Покажите, что если существует булева цепочка для $f(x_1,\ldots,x_n)$, которая содержит p канализирующих операторов, то $C(f) < (p+1)(n+p/2)$.

84. [*M20*] *Монотонная булева цепочка* представляет собой булеву цепочку, в которой каждый оператор $\circ_i$ является монотонным. Длина кратчайшей монотонной цепочки для f обозначается как $C^+(f)$. Покажите, что если существует монотонная булева цепочка для $f(x_1,\ldots,x_n)$, которая содержит p операторов $\wedge$ и q операторов $\vee$, то $C^+(f) < \min\bigl((p+1)(n+p/2),(q+1)(n+q/2)\bigr)$.

▶ **85.** [*M28*] Пусть M_n — множество всех монотонных функций от n переменных. Если L представляет собой семейство функций, содержащихся в M_n, введем обозначения

$$x \sqcup y = \bigwedge\{z \in L \mid z \supseteq x \vee y\} \qquad \text{и} \qquad x \sqcap y = \bigvee\{z \in L \mid z \subseteq x \wedge y\}.$$

Будем называть L законным (legitimate), если оно включает константные функции 0 и 1, а также проецирующие функции x_j для $1 \leq j \leq n$ и если $x \sqcup y \in L$, $x \sqcap y \in L$ при $x, y \in L$.

а) При $n = 3$ можно записать $M_3 = \{$`00`, `01`, `03`, `05`, `11`, `07`, `13`, `15`, `0f`, `33`, `55`, `17`, `1f`, `37`, `57`, `3f`, `5f`, `77`, `7f`, `ff`$\}$, представляя каждую функцию ее таблицей истинности, записанной

в виде шестнадцатеричного числа. Имеется 2^{15} семейств L, таких, что $\{\texttt{00}, \texttt{0f}, \texttt{33}, \texttt{55}, \texttt{ff}\} \subseteq L \subseteq M_3$. Сколько из них законны?

б) Если A представляет собой подмножество $\{1, \ldots, n\}$, то пусть $\lceil A \rceil = \bigvee_{a \in A} x_a$ и пусть также $\lceil \infty \rceil = 1$. Предположим, что $\mathcal{A}$ является семейством подмножеств $\{1, \ldots, n\}$, которые содержат все множества размером ≤ 1 и являются замкнутыми относительно пересечения; другими словами, $A \cap B \in \mathcal{A}$ при $A \in \mathcal{A}$ и $B \in \mathcal{A}$. Докажите, что семейство $L = \{\lceil A \rceil \mid A \in \mathcal{A} \cup \infty\}$ является законным.

в) Пусть $(x_{n+1}, \ldots, x_{n+r})$ является монотонной булевой цепочкой (1). Предположим, что $(\hat{x}_{n+1}, \ldots, \hat{x}_{n+r})$ получается из той же булевой цепочки, но в которой каждый оператор $\wedge$ заменен на $\sqcap$, а каждый оператор $\vee$ — на $\sqcup$ по отношению к некоторому законному семейству L. Докажите, что для $n+1 \leq l \leq n+r$ мы должны иметь

$$\hat{x}_l \subseteq x_l \vee \bigvee_{i=n+1}^{l} \{\hat{x}_i \oplus (\hat{x}_{j(i)} \vee \hat{x}_{k(i)}) \mid \circ_i = \vee\};$$

$$x_l \subseteq \hat{x}_l \vee \bigvee_{i=n+1}^{l} \{\hat{x}_i \oplus (\hat{x}_{j(i)} \wedge \hat{x}_{k(i)}) \mid \circ_i = \wedge\}.$$

86. [*HM37*] Граф G с вершинами $\{1, \ldots, n\}$ может быть определен с помощью $N = \binom{n}{2}$ булевых переменных x_{uv} для $1 \leq u < v \leq n$, где $x_{uv} = [u \text{—} v \text{ в } G]$. Пусть f представляет собой функцию $f(x) = [G \text{ содержит треугольник}]$; например, при $n = 4$, $f(x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{34}) = (x_{12} \wedge x_{13} \wedge x_{23}) \vee (x_{12} \wedge x_{14} \wedge x_{24}) \vee (x_{13} \wedge x_{14} \wedge x_{34}) \vee (x_{23} \wedge x_{24} \wedge x_{34})$. Цель данного упражнения — доказать, что монотонная сложность $C^+(f)$ равна $\Omega(n/\log n)^3$.

а) Если u_j — v_j для $1 \leq j \leq r$ в графе G, назовем $S = \{\{u_1, v_1\}, \ldots, \{u_r, v_r\}\}$ *r-семейством*, и пусть $\Delta(S) = \bigcup_{1 \leq i < j \leq r} (\{u_i, v_i\} \cap \{u_j, v_j\})$ — элементы его попарных пересечений. Будем говорить, что граф G *r-замкнут*, если мы имеем u — v при $\Delta(S) \subseteq \{u, v\}$ для некоторого r-семейства S. Он *сильно* r-замкнут, если, кроме того, мы имеем $|\Delta(S)| \geq 2$ для всех r-семейств S. Докажите, что сильно r-замкнутый граф является также сильно $(r+1)$-замкнутым.

б) Докажите, что полный биграф $K_{m,n}$ является сильно r-замкнутым при $r > \max(m, n)$.

в) Докажите, что сильно r-замкнутый граф имеет не более $(r-1)^2$ ребер.

г) Пусть L представляет собой семейство функций $\{1\} \cup \{\lceil G \rceil \mid G$ является сильно r-замкнутым графом на вершинах $\{1, \ldots, n\}\}$. (См. упр. 85, (б); мы рассматриваем G как множество ребер. Например, когда ребра представляют собой 1 — 3, 1 — 4, 2 — 3, 2 — 4, мы имеем $\lceil G \rceil = x_{13} \vee x_{14} \vee x_{23} \vee x_{24}$.) Является ли L законным?

д) Пусть $x_{N+1}, \ldots, x_{N+p+q} = f$ представляет собой монотонную булеву цепочку с p $\wedge$- и q $\vee$-шагами. Рассмотрим модифицированную цепочку $\hat{x}_{N+1}, \ldots, \hat{x}_{N+p+q} = \hat{f}$, основанную на семействе L из п. (г). Покажите, что если $\hat{f} \neq 1$, то $2(r-1)^3 p + (r-1)^2(n-2) \geq \binom{n}{3}$. *Указание*: воспользуйтесь второй формулой из упр. 85, (в).

е) Кроме того, если $\hat{f} = 1$, мы должны иметь $r^2 q \geq 2^{r+1}$. *Указание*: теперь воспользуйтесь первой формулой.

ж) Следовательно, $p = \Omega(n/\log n)^3$. *Указание*: положите $r \approx 6 \lg n$ и примените результат упр. 84.

87. [*M22*] Покажите, что если разрешены немонотонные операторы, то функция треугольника из упр. 86 имеет стоимость $C(f) = O(n^{\lg 7}(\log n)^2) = O(n^{2.81})$. *Указание*: граф имеет треугольник тогда и только тогда, когда куб его матрицы смежности имеет ненулевую диагональ.

88. [*40*] *Медианная цепочка* аналогична булевой цепочке, но вместо бинарных операций в (1) использует медианы трех значений $x_i = \langle x_{j(i)} x_{k(i)} x_{l(i)} \rangle$ для $n+1 \leq i \leq n+r$.

Изучите оптимальную длину, глубину и стоимость медианных цепочек для всех самодуальных монотонных булевых функций от семи переменных. Какова кратчайшая цепочка для $\langle x_1x_2x_3x_4x_5x_6x_7\rangle$?

7.1.3. Битовые трюки и технологии

А сейчас мы переходим к самой интересной части: использованию булевых операций в наших программах.

Люди обычно лучше знакомы с арифметическими операциями наподобие сложения, вычитания и умножения, чем с битовыми операциями, такими как "и", "исключающее или" и другие, поскольку арифметика имеет очень долгую историю. Но мы увидим, что булевы операции над бинарными числами заслуживают того, чтобы быть широко известными. И действительно, они являются важной частью инструментария любого хорошего программиста.

Проектировщики ранних вычислительных машин предоставляли в них побитовые операции над полными словами, в первую очередь, из-за того, что такие команды могли быть включены в набор машинных инструкций практически бесплатно. Бинарная логика представлялась потенциально полезной, хотя в то время просматривалось только весьма малое количество ее приложений. Например, в вычислительной машине EDSAC, завершенной в 1949 году, имелась команда "collate", которая, по сути, выполняла операцию $z \leftarrow z + (x \mathbin{\&} y)$, где z — аккумулятор, x — регистр множителя, а y — определенное слово в памяти; эта команда использовалась для распаковки данных. Вычислительная машина Manchester Mark I, построенная примерно в то же время, включала не только побитовое И, но также ИЛИ и ЛИБО. Когда Алан Тьюринг (Alan Turing) писал первое руководство по программированию Mark I в 1950 году, он отмечал, что побитовое НЕ можно было получить, применив ЛИБО (обозначавшееся как '$\not\equiv$') в комбинации со строкой единиц. Р. Э. Брукер (R. A. Brooker), расширивший руководство Тьюринга в 1952 году, когда была создана вычислительная машина Mark II, заметил, что ИЛИ можно использовать "для округления числа, устанавливая его бит в младшем разряде равным 1". В это же время машина Mark II, явившаяся прототипом для Ferranti Mercury, получила также новые команды для контрольного суммирования и для поиска позиции старшей 1.

Кейт Точер (Keith Tocher) в 1954 году опубликовал необычное применение И и ИЛИ, которое затем было неоднократно открыто заново (см. упр. 85). В продолжение следующих десятилетий программисты постепенно выясняли, что битовые операции могут быть на удивление полезны. Многие из этих трюков остались частью фольклора; сейчас пришло время воспользоваться тем, что было найдено.

Трюк — это хитрая идея, которая может быть использована один раз, в то время как *технология*, *метод* — это старый трюк, который может быть использован как минимум дважды. В этом разделе мы увидим, что трюки естественным образом эволюционируют в методы.

Улучшенная арифметика. Начнем с официального определения битовых операций над целыми числами. Если $x = (\ldots x_2x_1x_0)_2$, $y = (\ldots y_2y_1y_0)_2$ и $z = (\ldots z_2z_1z_0)_2$ в бинарных обозначениях, то мы имеем

$$x \mathbin{\&} y = z \quad\Longleftrightarrow\quad x_k \wedge y_k = z_k \qquad \text{для всех } k \ge 0; \tag{1}$$

$$x \mid y = z \quad\Longleftrightarrow\quad x_k \vee y_k = z_k \qquad \text{для всех } k \ge 0; \tag{2}$$
$$x \oplus y = z \quad\Longleftrightarrow\quad x_k \oplus y_k = z_k \qquad \text{для всех } k \ge 0. \tag{3}$$

(Было соблазнительно записать '$x \wedge y$' вместо $x \,\&\, y$ и '$x \vee y$' вместо $x \mid y$; но при изучении задач оптимизации мы обнаружим, что обозначения $x \wedge y$ и $x \vee y$ лучше зарезервировать за $\min(x, y)$ и $\max(x, y)$ соответственно.) Таким образом, например,

$$5 \,\&\, 11 = 1, \qquad 5 \mid 11 = 15 \qquad \text{и} \qquad 5 \oplus 11 = 14,$$

поскольку $5 = (0101)_2$, $11 = (1011)_2$, $1 = (0001)_2$, $15 = (1111)_2$ и $14 = (1110)_2$. Отрицательные целые числа можно рассматривать в этой связи как числа с бесконечной точностью в дополнительном двоичном коде, имеющие бесконечно много единиц слева; например, -5 представляет собой $(\ldots 1111011)_2$. Такие числа с бесконечной точностью представляют собой частный случай *2-адических целых чисел*, которые рассматривались в упр. 4.1–31, и в действительности операторы &, $|$, $\oplus$ имеют точный смысл при применении к произвольным 2-адическим числам.

Математики никогда не уделяли большого внимания свойствам & и $|$ как операциям над целыми числами. Но третья операция, $\oplus$, имеет почтенную историю, поскольку она описывает выигрышную стратегию в игре "Ним" (см. упр. 8–16). По этой причине $x \oplus y$ часто называется ним-суммой (nim sum) целых чисел x и y.

Все три базовые битовые операции имеют много полезных свойств. Например, каждое отношение между $\wedge$, $\vee$ и $\oplus$, которое изучалось в разделе 7.1.1, автоматически наследуется &, $|$ и $\oplus$ для целых чисел, поскольку отношения выполняются в каждой битовой позиции. Повторим здесь основные тождества:

$$x \,\&\, y = y \,\&\, x, \qquad x \mid y = y \mid x, \qquad x \oplus y = y \oplus x; \tag{4}$$
$$(x \,\&\, y) \,\&\, z = x \,\&\, (y \,\&\, z), \quad (x \mid y) \mid z = x \mid (y \mid z), \quad (x \oplus y) \oplus z = x \oplus (y \oplus z); \tag{5}$$
$$(x \mid y) \,\&\, z = (x \,\&\, z) \mid (y \,\&\, z), \qquad (x \,\&\, y) \mid z = (x \mid z) \,\&\, (y \mid z); \tag{6}$$
$$(x \oplus y) \,\&\, z = (x \,\&\, z) \oplus (y \,\&\, z); \tag{7}$$
$$(x \,\&\, y) \mid x = x, \qquad (x \mid y) \,\&\, x = x; \tag{8}$$
$$(x \,\&\, y) \oplus (x \mid y) = x \oplus y; \tag{9}$$
$$x \,\&\, 0 = 0, \qquad x \mid 0 = x, \qquad x \oplus 0 = x; \tag{10}$$
$$x \,\&\, x = x, \qquad x \mid x = x, \qquad x \oplus x = 0; \tag{11}$$
$$x \,\&\, -1 = x, \qquad x \mid -1 = -1, \qquad x \oplus -1 = \bar{x}; \tag{12}$$
$$x \,\&\, \bar{x} = 0, \qquad x \mid \bar{x} = -1, \qquad x \oplus \bar{x} = -1; \tag{13}$$
$$\overline{x \,\&\, y} = \bar{x} \mid \bar{y}, \qquad \overline{x \mid y} = \bar{x} \,\&\, \bar{y}, \qquad \overline{x \oplus y} = \bar{x} \oplus y = x \oplus \bar{y}. \tag{14}$$

Обозначение $\bar{x}$ в (12), (13) и (14) означает битовое *дополнение* x, а именно значение $(\ldots \bar{x}_2 \bar{x}_1 \bar{x}_0)_2$, записываемое также как $\sim x$. Заметим, что (12) и (13) — это не то же самое, что 7.1.1–(10) и 7.1.1–(18); здесь мы должны использовать $-1 = (\ldots 1111)_2$ вместо $1 = (\ldots 0001)_2$ для того, чтобы сделать формулы побитово корректными.

Мы говорим, что x *содержится в* y, что записывается как $x \subseteq y$ или $y \supseteq x$, если отдельные биты x и y удовлетворяют условию $x_k \le y_k$ для всех $k \ge 0$. Таким образом,

$$x \subseteq y \quad\Longleftrightarrow\quad x \,\&\, y = x \quad\Longleftrightarrow\quad x \mid y = y \quad\Longleftrightarrow\quad x \,\&\, \bar{y} = 0. \tag{15}$$

Конечно, у нас нет необходимости использовать битовые операции только в связи одна с другой; мы можем комбинировать их со всеми другими обычными арифметическими операциями. Например, из отношения $x+\overline{x}=(\ldots 1111)_2=-1$ можно вывести формулу

$$-x \;=\; \overline{x}+1, \tag{16}$$

которая оказывается чрезвычайно важной. Замена x на $x-1$ дает также

$$-x \;=\; \overline{x-1}; \tag{17}$$

и в общем случае можно свести вычитание к дополнению и сложению:

$$\overline{x-y} \;=\; \overline{x}+y. \tag{18}$$

Нам часто требуется бинарный сдвиг чисел влево или вправо. Эти операции эквивалентны умножению и делению на степени 2 с соответствующим округлением, но для них удобно ввести специальные обозначения:

$$x \ll k \;=\; x \text{ сдвинуто влево на } k \text{ бит} \;=\; \lfloor 2^k x\rfloor; \tag{19}$$

$$x \gg k \;=\; x \text{ сдвинуто вправо на } k \text{ бит} \;=\; \lfloor 2^{-k} x\rfloor. \tag{20}$$

Здесь k может быть любым целым числом, возможно, отрицательным. В частности, мы имеем

$$x \ll (-k) = x \gg k \qquad \text{и} \qquad x \gg (-k) = x \ll k \tag{21}$$

для каждого числа x с бесконечной точностью. Кроме того, $(x \,\&\, y) \ll k = (x \ll k) \,\&\, (y \ll k)$ и т. д.

Когда битовые операции комбинируются со сложением, вычитанием, умножением и/или сдвигом, можно получить исключительно сложные результаты даже при достаточно кратких формулах. Оценить эти возможности можно, например, на рис. 11. Кроме того, такие формулы не только демонстрируют бесцельное, хаотическое поведение: знаменитая цепочка операций, известная как хак Госпера и впервые опубликованная в 1972 году, открыла всем глаза на тот факт, что можно быстро вычислять большое количество полезных и нетривиальных функций (см. упр. 20). Наша цель в данном разделе состоит в изучении того, как можно открывать такие эффективные конструкции.

Рис. 11. Небольшая часть "лоскутного одеяла", которое определяется битовой функцией $f(x,y)=\big((x\oplus\bar{y})\,\&\,((x-350)\gg 3)\big)^2$; квадратная ячейка в строке x и столбце y окрашивается в черный или белый цвет в зависимости от того, чему равно значение $\big((f(x,y)\gg 12)\,\&\,1\big)$ — 0 или 1. (Разработано Д. Слитором (D. Sleator), 1976; см. также упр. 18.)

Упаковка и распаковка. Мы изучали алгоритмы арифметики многократной точности в разделе 4.3.1, работая с ситуациями, когда целые числа оказывались слишком большими для размещения в одном слове памяти или одном регистре компьютера. Однако противоположная ситуация, когда целые числа значительно *меньше* емкости одного слова, в действительности встречаются намного чаще; Д. Г. Лемер (D. H. Lehmer) назвал это явление *дробной точностью*. Зачастую можно работать с несколькими целыми числами одновременно, упаковав их в одно слово.

Например, дата x, которая состоит из номера года y, номера месяца m и номера дня d может быть представлена с использованием 4 бит для m и 5 бит для d:

$$x \;=\; (((y \ll 4) + m) \ll 5) + d. \tag{22}$$

Ниже мы увидим, что многие операции могут выполняться с датами непосредственно в таком упакованном виде. Например, $x < x'$, если дата x предшествует дате x'. Но при необходимости отдельные компоненты (y, m, d) для заданного x можно легко распаковать:

$$d = x \bmod 32, \qquad m = (x \gg 5) \bmod 16, \qquad y = x \gg 9. \tag{23}$$

Эти операции “mod” не требуют применения деления в силу важного закона

$$x \bmod 2^n \;=\; x \,\&\, (2^n - 1) \tag{24}$$

для любого целого числа $n \geq 0$. Мы имеем, например, $d = x \,\&\, 31$ в (22) и (23).

Такая упаковка данных, очевидно, экономит используемую память, но она экономит и время: мы можем быстро перемещать или копировать элементы данных из одного места в другое, если они упакованы вместе. Более того, компьютеры работают существенно быстрее, если имеют дело с числами, которые помещаются в кэш-память ограниченного размера.

Предельная плотность упаковки достигается при однобитовых элементах, поскольку мы можем затолкать в 64-битовое слово целых 64 таких элемента. Предположим, например, что нам нужна таблица всех нечетных простых чисел, меньших 1024, чтобы мы могли легко определить простоту небольшого числа. Никаких проблем; нам для этого потребуется только восемь 64-битовых чисел.

$$\begin{aligned}
P_0 &= 0111011011010011001011010010011001011001010010001011011010000001,\\
P_1 &= 0100110000110010010100100110000110110000010000010110100110000100,\\
P_2 &= 1001001100101100001000000101101000000100100001101001000100100101,\\
P_3 &= 0010001010001000011000011001010010001011010000010001010001010010,\\
P_4 &= 0000110000000001001000010010011001000010010011001001011000010000,\\
P_5 &= 1101001001100000101001000100001000100001000100100101000100101000,\\
P_6 &= 1010000001000010000011000011011000010000000101101000000101101000 0,\\
P_7 &= 0000010100010000100010100100100000010100100100010010000010100110.
\end{aligned}$$

Для проверки, является ли число $2k+1$ $(0 \leq k < 512)$ простым, мы просто вычисляем

$$P_{\lfloor k/64 \rfloor} \ll (k \,\&\, 63) \tag{25}$$

в 64-битовом регистре и смотрим, равен ли крайний слева бит 1. Например, если регистр `pbase` хранит адрес P_0, то указанные действия выполняются с помощью

следующих инструкций MMIX.

```
SRU    $0,k,3          $0 ← ⌊k/8⌋ (т. е. k ≫ 3)
LDOU   $1,pbase,$0     $1 ← P_⌊$0/8⌋ (т. е. P_⌊k/64⌋)
AND    $0,k,#3f        $0 ← k mod 64 (т. е. k & #3f)
SLU    $1,$1,$0        $1 ← ($1 ≪ $0) mod 2^64
BN     $1,PRIME        Переход к PRIME, если s($1) < 0. ▮
```
(26)

Заметим, что крайний слева бит регистра равен 1 тогда и только тогда, когда содержимое регистра отрицательно.

Мы можем точно так же упаковать биты в каждом слове справа налево.

$$
\begin{aligned}
Q_0 &= 1000000101101101000100101001101001100100101101001100101101101110,\\
Q_1 &= 0010000110010110100000100000110110000110010010100100110000110010,\\
Q_2 &= 1010010010001001011000010010000001011010000001000011010011001001,\\
Q_3 &= 0100101000101000100000101101000100101001100001100001000101000100,\\
Q_4 &= 0000100000110100100110010010000100110010010000100100000000110000,\\
Q_5 &= 0001010010001010010010001000010001000010001001010000011001001011,\\
Q_6 &= 0000101101000000101101000000100001101100001100000100001000000101,\\
Q_7 &= 0110010100000010010001001001010000000100100101000100001000100100000.
\end{aligned}
$$

Здесь $Q_j = P_j^R$. Вместо сдвига влево, как в (25), мы теперь выполняем сдвиг вправо,

$$Q_{\lfloor k/64 \rfloor} \gg (k \,\&\, 63), \tag{27}$$

и проверяем *крайний справа* бит результата. Последние две строки (26) превращаются в

```
SRU    $1,$1,$0        $1 ← $1 ≫ $0.
BOD    $1,PRIME        Переход к PRIME, если $1 нечетно. ▮
```
(28)

(И, конечно, мы используем qbase вместо pbase.) В любом случае классическое *решето Эратосфена* легко заполнит элементы базовых таблиц P_j и Q_j (см. упр. 24).

Соглашения о порядке байтов. При упаковке битов или байтов в слова мы должны решить, будем ли мы размещать их слева направо или справа налево. Соглашение о размещении слева направо называется обратным порядком (big-endian), поскольку начальные элементы оказываются в старших позициях; таким образом, при сравнении чисел они будут старше идущих за ними. Соглашение о размещении справа налево называется прямым порядком (little-endian); при этом первые элементы размещаются там, где располагаются меньшие числа.

Обратный порядок во многих случаях представляется более естественным, поскольку мы приучены читать и писать слева направо. Но свои преимущества есть и у прямого порядка размещения. Например, обратимся вновь к задаче простых чисел. Пусть $a_k = [2k{+}1 \text{ — простое число}]$. Наши записи в таблице $\{P_0, P_1, \ldots, P_7\}$ используют обратный порядок, и мы можем рассматривать их как представление одного целого числа с многократной точностью длиной 512 бит:

$$(P_0 P_1 \ldots P_7)_{2^{64}} \;=\; (a_0 a_1 \ldots a_{511})_2. \tag{29}$$

Аналогично наша таблица с прямым порядком записей представляет целое число с многократной точностью

$$(Q_7 \ldots Q_1 Q_0)_{2^{64}} \;=\; (a_{511} \ldots a_1 a_0)_2. \tag{30}$$

Таблица 1

Обратный порядок в случае 32-байтовой памяти

октет 0							
квартет 0				квартет 4			
дуэт 0		дуэт 2		дуэт 4		дуэт 6	
байт 0	байт 1	байт 2	байт 3	байт 4	байт 5	байт 6	байт 7
$a_0 \ldots a_7$	$a_8 \ldots a_{15}$	$a_{16} \ldots a_{23}$	$a_{24} \ldots a_{31}$	$a_{32} \ldots a_{39}$	$a_{40} \ldots a_{47}$	$a_{48} \ldots a_{55}$	$a_{56} \ldots a_{63}$
октет 8							
квартет 8				квартет 12			
дуэт 8		дуэт 10		дуэт 12		дуэт 14	
байт 8	байт 9	байт 10	байт 11	байт 12	байт 13	байт 14	байт 15
$a_{64} \ldots a_{71}$	$a_{72} \ldots a_{79}$	$a_{80} \ldots a_{87}$	$a_{88} \ldots a_{95}$	$a_{96} \ldots a_{103}$	$a_{104} \ldots a_{111}$	$a_{112} \ldots a_{119}$	$a_{120} \ldots a_{127}$
октет 16							
квартет 16				квартет 20			
дуэт 16		дуэт 18		дуэт 20		дуэт 22	
байт 16	байт 17	байт 18	байт 19	байт 20	байт 21	байт 22	байт 23
$a_{128} \ldots a_{135}$	$a_{136} \ldots a_{143}$	$a_{144} \ldots a_{151}$	$a_{152} \ldots a_{159}$	$a_{160} \ldots a_{167}$	$a_{168} \ldots a_{175}$	$a_{176} \ldots a_{183}$	$a_{184} \ldots a_{191}$
октет 24							
квартет 24				квартет 28			
дуэт 24		дуэт 26		дуэт 28		дуэт 30	
байт 24	байт 25	байт 26	байт 27	байт 28	байт 29	байт 30	байт 31
$a_{192} \ldots a_{199}$	$a_{200} \ldots a_{207}$	$a_{208} \ldots a_{215}$	$a_{216} \ldots a_{223}$	$a_{224} \ldots a_{231}$	$a_{232} \ldots a_{239}$	$a_{240} \ldots a_{247}$	$a_{248} \ldots a_{255}$

Последнее целое число математически более красиво, чем первое, поскольку оно представляет собой

$$\sum_{k=0}^{511} 2^k a_k = \sum_{k=0}^{511} 2^k [2k{+}1 \text{ — простое}] = \Bigl(\sum_{k=0}^{\infty} 2^k [2k{+}1 \text{ — простое}]\Bigr) \bmod 2^{512}. \qquad (31)$$

Заметим, однако, что мы используем для получения этого простого результата $(Q_7 \ldots Q_1 Q_0)_{2^{64}}$, а не $(Q_0 Q_1 \ldots Q_7)_{2^{64}}$. Другое число,

$$(Q_0 Q_1 \ldots Q_7)_{2^{64}} = (a_{63} \ldots a_1 a_0 a_{127} \ldots a_{65} a_{64} a_{191} \ldots a_{385} a_{384} a_{511} \ldots a_{449} a_{448})_2,$$

оказывается слишком жутким и не представимым никакой красивой формулой. (См. упр. 25.)

Порядок байтов имеет важные последствия, поскольку большинство компьютеров позволяют как адресовать отдельные байты памяти, так и обращаться к байтам в регистрах. `MMIX` имеет архитектуру с обратным порядком байтов; следовательно, если регистр `x` содержит 64-битовое число `#0123456789abcdef` и если мы воспользуемся командами ‘`STOU x,0; LDBU y,1`’ для сохранения `x` в октете по адресу 0 и считывания байта в позиции 1, то в результате в регистре `y` получим значение `#23`. На машинах с прямым порядком байтов аналогичные команды приведут к установке `y` ← `#cd`; `#23` будет в байте 6.

Таблица 2

Прямой порядок в случае 32-байтовой памяти

октет 24							
квартет 28				квартет 24			
дуэт 30		дуэт 28		дуэт 26		дуэт 24	
байт 31	байт 30	байт 29	байт 28	байт 27	байт 26	байт 25	байт 24
$a_{255} \ldots a_{248}$	$a_{247} \ldots a_{240}$	$a_{239} \ldots a_{232}$	$a_{231} \ldots a_{224}$	$a_{223} \ldots a_{216}$	$a_{215} \ldots a_{208}$	$a_{207} \ldots a_{200}$	$a_{199} \ldots a_{192}$
октет 16							
квартет 20				квартет 16			
дуэт 22		дуэт 20		дуэт 18		дуэт 16	
байт 23	байт 22	байт 21	байт 20	байт 19	байт 18	байт 17	байт 16
$a_{191} \ldots a_{184}$	$a_{183} \ldots a_{176}$	$a_{175} \ldots a_{168}$	$a_{167} \ldots a_{160}$	$a_{159} \ldots a_{152}$	$a_{151} \ldots a_{144}$	$a_{143} \ldots a_{136}$	$a_{135} \ldots a_{128}$
октет 8							
квартет 12				квартет 8			
дуэт 14		дуэт 12		дуэт 10		дуэт 8	
байт 15	байт 14	байт 13	байт 12	байт 11	байт 10	байт 9	байт 8
$a_{127} \ldots a_{120}$	$a_{119} \ldots a_{112}$	$a_{111} \ldots a_{104}$	$a_{103} \ldots a_{96}$	$a_{95} \ldots a_{88}$	$a_{87} \ldots a_{80}$	$a_{79} \ldots a_{72}$	$a_{71} \ldots a_{64}$
октет 0							
квартет 4				квартет 0			
дуэт 6		дуэт 4		дуэт 2		дуэт 0	
байт 7	байт 6	байт 5	байт 4	байт 3	байт 2	байт 1	байт 0
$a_{63} \ldots a_{56}$	$a_{55} \ldots a_{48}$	$a_{47} \ldots a_{40}$	$a_{39} \ldots a_{32}$	$a_{31} \ldots a_{24}$	$a_{23} \ldots a_{16}$	$a_{15} \ldots a_{8}$	$a_{7} \ldots a_{0}$

В табл. 1 и 2 проиллюстрированы взгляды “тупоконечников” и “остроконечников”* на размещение информации в памяти. Подход тупоконечников (сторонников обратного порядка байтов), по существу, нисходящий, с битом 0 и байтом 0 вверху слева; у остроконечников подход, по существу, восходящий, с битом 0 и байтом 0 внизу справа. Из-за этого отличия следует с особой осторожностью относиться к передаче данных с компьютера одного типа на другой и к написанию программ, которые должны давать одинаковые результаты на компьютерах обоих типов. С другой стороны, наш пример с таблицей Q для простых чисел демонстрирует, что мы отлично можем использовать прямую упаковку данных на компьютере с обратным порядком наподобие `MMIX`, и наоборот. Отличие становится заметным только тогда, когда данные загружаются и сохраняются блоками разных размеров, или передаются между машинами.

Работа с крайними справа битами. В общем случае прямой и обратный подходы не столь легко взаимозаменяемы, поскольку законы арифметики передают сигналы влево от битов, находящихся в младших разрядах (least significant). Некоторые из наиболее важных битовых методов основаны на этом факте.

---

* Впервые термины “big-endian” и “little-endian” использованы в *Путешествиях Гулливера* Джонатана Свифта (Jonathan Swift) для обозначения сторонников разбивания яйца с тупого конца и с острого конца соответственно. — *Примеч. пер.*

Если x представляет собой почти любое ненулевое 2-адическое целое число, то его биты можно записать в виде

$$x = (\alpha 01^a 10^b)_2; \tag{32}$$

другими словами, x состоит из некоторой произвольной (но бесконечной) бинарной строки α, за которой следует 0, а за ним — $a+1$ единиц и b нулей для некоторых $a \geq 0$ и $b \geq 0$. (Исключением является $x = -2^b$; тогда $a = \infty$.) Следовательно,

$$\bar{x} = (\bar{\alpha}\, 10^a 01^b)_2, \tag{33}$$

$$x - 1 = (\alpha\, 01^a 01^b)_2, \tag{34}$$

$$-x = (\bar{\alpha}\, 10^a 10^b)_2; \tag{35}$$

и мы видим, что $\bar{x}+1 = -x = \overline{x-1}$, что согласуется с (16) и (17). Таким образом, мы можем вычислить связанные с x значения с помощью двух операций несколькими полезными способами:

$$x \,\&\, (x-1) = (\ \alpha\ 01^a 00^b)_2 \quad \text{[удаление крайней справа единицы]}; \tag{36}$$

$$x \,\&\, -x = (0^\infty 00^a 10^b)_2 \quad \text{[выделение крайней справа единицы]}; \tag{37}$$

$$x \mid -x = (1^\infty 11^a 10^b)_2 \quad \text{[распространение крайней справа единицы влево]}; \tag{38}$$

$$x \oplus -x = (1^\infty 11^a 00^b)_2 \quad \text{[удаление и распространение ее влево]}; \tag{39}$$

$$x \mid (x-1) = (\ \alpha\ 01^a 11^b)_2 \quad \text{[распространение крайней справа единицы вправо]}; \tag{40}$$

$$x \oplus (x-1) = (0^\infty 00^a 11^b)_2 \quad \text{[выделение и распространение ее вправо]}; \tag{41}$$

$$\bar{x} \,\&\, (x-1) = (0^\infty 00^a 01^b)_2 \quad \text{[выделение, удаление и распространение ее вправо]}. \tag{42}$$

Две дополнительные операции дают еще один вариант:

$$((x|(x-1))+1) \,\&\, x = (\ \alpha\ 00^a 00^b)_2 \quad \text{[удаление крайней справа серии единиц]}. \tag{43}$$

При $x = 0$ пять из этих формул дают 0, а три другие дают -1. [Формула (36) разработана Путтило Вейденом (Peter Wegner), *CACM* **3** (1960), 322; а (43) — Х. Т. Гладвином (H. Tim Gladwin), *CACM* **14** (1971), 407–408. См. также Генри С. Уоррен-мл. (Henry S. Warren, Jr.), *CACM* **20** (1977), 439–441.]

В этих формулах величина b, определяющая количество завершающих нулей в x, называется *линеечной функцией* от x и записывается как ρx, поскольку она связана с длиной штриховых меток, часто использующихся для указания долей сантиметров: . В общем случае ρx представляет собой наибольшее целое число k, такое, что 2^k делит x при $x \neq 0$; мы определяем также $\rho 0 = \infty$. Рекуррентные соотношения

$$\rho(2x+1) = 0, \qquad \rho(2x) = \rho(x) + 1 \tag{44}$$

также служат для определения ρx для ненулевых x. Другим удобным соотношением, достойным упоминания, является

$$\rho(x-y) = \rho(x \oplus y). \tag{45}$$

Элегантная формула $x \,\&\, -x$ в (37) позволяет нам красиво *выделить* крайний справа бит 1, но зачастую нам надо точно определить, какой именно это бит. Линеечная функция может быть вычислена многими путями, и наилучший метод часто

сильно зависит от используемого компьютера. Например, последовательность двух команд, придуманная Д. Даллосом (J. Dallos), выполняет эту работу легко и быстро на машине MMIX (см. (42)):

$$\texttt{SUBU t,x,1; SADD rho,t,x.} \tag{46}$$

(Случай $x = 0$ рассматривается в упр. 30.) Здесь мы рассмотрим два подхода, которые не используют экзотические команды наподобие SADD; а позже, после изучения еще нескольких методов, мы рассмотрим и третий способ.

Первый метод общего назначения использует константы "магических масок" μ_k, которые доказанно полезны во многих других приложениях, а именно

$$\begin{aligned}
\mu_0 &= (\ldots 1010101010101010101010101010101)_2 = -1/3,\\
\mu_1 &= (\ldots 1001100110011001100110011001100110011)_2 = -1/5,\\
\mu_2 &= (\ldots 10000111100001111000011110000111 1)_2 = -1/17,\\
\mu_3 &= (\ldots 1000000001111111100000000111111 11)_2 = -1/257
\end{aligned} \tag{47}$$

и т. д. В общем случае μ_k представляет собой бесконечную 2-адическую дробь $-1/(2^{2^k}+1)$, поскольку $(2^{2^k}+1)\mu_k = (\mu_k \ll 2^k) + \mu_k = (\ldots 11111)_2 = -1$. На компьютерах с 2^d-битовыми регистрами нам, конечно, не нужна бесконечная точность, так что мы используем урезанные константы

$$\mu_{d,k} = (2^{2^d}-1)/(2^{2^k}+1) \qquad \text{для } 0 \le k < d. \tag{48}$$

Эти константы знакомы нам по булевым вычислениям, поскольку они представляют собой таблицы истинности проецирующих функций x_{d-k} (см., например, 7.1.2–(7)).

Когда x представляет собой степень 2, мы можем использовать эти маски для вычисления

$$\rho x = [x \,\&\, \mu_0 = 0] + 2[x \,\&\, \mu_1 = 0] + 4[x \,\&\, \mu_2 = 0] + 8[x \,\&\, \mu_3 = 0] + \cdots, \tag{49}$$

поскольку $[2^j \,\&\, \mu_k = 0] = j_k$ при $j = (\ldots j_3 j_2 j_1 j_0)_2$. Таким образом, на 2^d-битовом компьютере можно начать с $\rho \leftarrow 0$ и $y \leftarrow x \,\&\, -x$; затем установить $\rho \leftarrow \rho + 2^k$, если $y \,\&\, \mu_{d,k} = 0$, при $0 \le k < d$. Эта процедура дает $\rho = \rho x$ при $x \ne 0$. (Она также дает $\rho 0 = 2^d - 1$, аномальное значение, которое может потребоваться откорректировать; см. упр. 30.)

Например, соответствующая программа MMIX может выглядеть следующим образом.

```
m0 GREG #5555555555555555 ;m1 GREG #3333333333333333;
m2 GREG #0f0f0f0f0f0f0f0f ;m3 GREG #00ff00ff00ff00ff;
m4 GREG #0000ffff0000ffff ;m5 GREG #00000000ffffffff;
 NEGU y,x; AND y,x,y; AND q,y,m5; ZSZ rho,q,32;
 AND q,y,m4; ADD t,rho,16; CSZ rho,q,t;
 AND q,y,m3; ADD t,rho,8; CSZ rho,q,t;
 AND q,y,m2; ADD t,rho,4; CSZ rho,q,t;
 AND q,y,m1; ADD t,rho,2; CSZ rho,q,t;
 AND q,y,m0; ADD t,rho,1; CSZ rho,q,t;
```
(50)

Общее время работы равно $19v$. Последние три строки можно заменить строками

$$\texttt{SRU y,y,rho; LDB t,rhotab,y; ADD rho,rho,t} \tag{51}$$

где `rhotab` указывает на начало соответствующей 129-байтовой таблицы (реально используются только восемь из ее записей). В этом случае общее время работы составляет $\mu + 13\upsilon$.

Второй подход общего назначения для вычисления ρx существенно отличается от рассмотренного. На 64-битовых машинах он начинается, как и ранее, с $y \leftarrow x \,\&\, -x$; но затем он просто устанавливает

$$\rho \;\leftarrow\; \mathit{decode}\big[((a \cdot y) \bmod 2^{64}) \gg 58\big], \tag{52}$$

где a — подходящий множитель, а *decode* — соответствующая 64-байтовая таблица. Константа $a = (a_{63} \ldots a_1 a_0)_2$ должна обладать тем свойством, что ее 64 подстроки

$$a_{63}a_{62}\ldots a_{58},\; a_{62}a_{61}\ldots a_{57},\; \ldots,\; a_5a_4\ldots a_0,\; a_4a_3a_2a_1a_0 0,\; \ldots,\; a_0 00000$$

различны. В упр. 2.3.4.2–23 показано, что есть много таких "циклов де Брейна"; так, можно использовать константу М. Х. Мартина (M. H. Martin) `#03f79d71b4ca8b09`, которая рассматривается в упр. 3.2.2–17. Таблица декодирования $\mathit{decode}[0], \ldots, \mathit{decode}[63]$ в таком случае представляет собой

$$\begin{array}{l} 00, 01, 56, 02, 57, 49, 28, 03, 61, 58, 42, 50, 38, 29, 17, 04, \\ 62, 47, 59, 36, 45, 43, 51, 22, 53, 39, 33, 30, 24, 18, 12, 05, \\ 63, 55, 48, 27, 60, 41, 37, 16, 46, 35, 44, 21, 52, 32, 23, 11, \\ 54, 26, 40, 15, 34, 20, 31, 10, 25, 14, 19, 09, 13, 08, 07, 06. \end{array} \tag{53}$$

[Этот метод разработан в 1967 году Лютером Вудрамом (Luther Woodrum) из отдела системных разработок IBM (неопубликовано); многие другие программисты впоследствии открыли его независимо.]

Работа с крайними слева битами. Функция $\lambda x = \lfloor \lg x \rfloor$ дуальна к ρx, поскольку устанавливает местоположение *крайней слева* единицы при $x > 0$ и введена в 4.6.3–(6). Она удовлетворяет рекуррентному соотношению

$$\lambda 1 = 0; \qquad \lambda(2x) = \lambda(2x+1) = \lambda(x) + 1 \quad \text{для } x > 0 \tag{54}$$

и не определена, когда x не является положительным целым числом. Каким же способом лучше всего ее вычислять? И вновь `MMIX` предоставляет быстрое решение-трюк:

```
FLOTU y,ROUND_DOWN,x; SUB y,y,fone; SR lam,y,52,
```
(55)

где `fone` = `#3ff0000000000000` — представление с плавающей точкой числа 1.0. (Общее время счета равно 6υ.) Этот код превращает x в число с плавающей точкой, а затем выделяет его степень.

Но если преобразование в число с плавающей точкой не является легко доступным, на 2^d-битовых машинах хорошо работает стратегия бинарного сокращения. Мы можем начать с $\lambda \leftarrow 0$ и $y \leftarrow x$; затем мы устанавливаем $\lambda \leftarrow \lambda + 2^k$ и $y \leftarrow y \gg 2^k$, если $y \gg 2^k \neq 0$, для $k = d-1, \ldots, 1, 0$ (или пока k не снизится до точки, когда для завершения работы можно будет воспользоваться короткой таблицей). Код `MMIX`, аналогичный (50) и (51), теперь представляет собой

```
SRU y,x,32; ZSNZ lam,y,32;
ADD t,lam,16; SRU y,x,t; CSNZ lam,y,t;
ADD t,lam,8; SRU y,x,t; CSNZ lam,y,t;
SRU y,x,lam; LDB t,lamtab,y; ADD lam,lam,t;
```
(56)

а общее время работы составляет $\mu+11\upsilon$. В данном случае таблица `lamtab` имеет 256 элементов, а именно λx для $0 \le x < 256$. Заметим, что вместо команд ветвления здесь и в (50) используются команды "условная установка" ("conditional set" `CS`) и "нуль или установка" ("zero or set" `ZS`).

Похоже, что простого способа выделения крайнего слева бита 1 из регистра, аналогичного трюку с выделением крайнего справа бита 1 в (37), не существует. Для этой цели мы можем вычислить $y \leftarrow \lambda x$, а затем $1 \ll y$, если $x \ne 0$; но бинарное "распространение вправо" оказывается короче и быстрее:

$$\begin{array}{l}\text{Установить } y \leftarrow x, \text{ затем } y \leftarrow y \mid (y \gg 2^k) \text{ для } 0 \le k < d.\\ \text{После этого крайний слева бит 1 в } x \text{ равен } y - (y \gg 1).\end{array} \tag{57}$$

[Эти методы без применения чисел с плавающей точкой предложены Г. С. Уорреном-мл. (H. S. Warren, Jr.).]

Другие операции в левой части регистра, наподобие удаления крайней слева серии единиц, оказываются еще сложнее; см. упр. 39. Однако имеется замечательно простой машинно-независимый способ проверки, выполняется ли соотношение $\lambda x = \lambda y$, для данных беззнаковых целых x и y, несмотря на тот факт, что мы не можем быстро вычислить λx и λy:

$$\lambda x = \lambda y \qquad \text{тогда и только тогда, когда} \qquad x \oplus y \le x \,\&\, y. \tag{58}$$

[См. упр. 40. Это красивое соотношение было открыто в 2006 году У. Ч. Линчем (W. C. Lynch).] Мы будем использовать (58) ниже для разработки другого пути вычисления λx.

Контрольное сложение. Бинарные n-битовые числа $x = (x_{n-1} \ldots x_1 x_0)_2$ часто используются для представления подмножеств X n-элементной совокупности $\{0, 1, \ldots, n-1\}$, где $k \in X$ тогда и только тогда, когда $2^k \subseteq x$. Функции λx и ρx в таком случае представляют наибольший и наименьший элементы X. Функция

$$\nu x \;=\; x_{n-1} + \cdots + x_1 + x_0, \tag{59}$$

которая называется контрольной суммой (sideways sum) или заселенностью (population count) x, также имеет очевидную важность в данной связи, поскольку представляет мощность $|X|$, а именно количество элементов в X. Эта функция, которую мы рассматривали в 4.6.3–(7), удовлетворяет рекуррентному соотношению

$$\nu 0 = 0; \qquad \nu(2x) = \nu(x) \quad \text{и} \quad \nu(2x{+}1) = \nu(x) + 1 \quad \text{для } x \ge 0. \tag{60}$$

Она также имеет интересную связь с линеечной функцией (упр. 1.2.5–11):

$$\rho x = 1 + \nu(x{-}1) - \nu x; \qquad \text{или, что эквивалентно,} \quad \sum_{k=1}^{n} \rho k = n - \nu n. \tag{61}$$

Первый учебник по программированию, *The Preparation of Programs for an Electronic Digital Computer* Уилкиса (Wilkes), Уиллера (Wheeler) и Джилла (Gill), второе издание (Reading, Mass.: Addison–Wesley, 1957), 155, 191–193, содержит интересную подпрограмму для контрольного суммирования, разработанную Д. Б. Джил-

лисом (D. B. Gillies) и Д. Ч. П. Миллером (J. C. P. Miller). Их метод разработан для 35-битового числа машины EDSAC, но легко преобразуется в следующую 64-битовую процедуру для νx, где $x = (x_{63} \ldots x_1 x_0)_2$:

$$\begin{aligned}
&\text{Установить } y \leftarrow x - ((x \gg 1) \,\&\, \mu_0). \text{ (Теперь } y = (u_{31} \ldots u_1 u_0)_4, \text{ где } u_j = x_{2j+1} + x_{2j}.)\\
&\text{Установить } y \leftarrow (y \& \mu_1) + ((y \gg 2) \& \mu_1). \text{ (Теперь } y = (v_{15} \ldots v_1 v_0)_{16}, v_j = u_{2j+1} + u_{2j}.)\\
&\text{Установить } y \leftarrow (y + (y \gg 4)) \,\&\, \mu_2. \text{ (Теперь } y = (w_7 \ldots w_1 w_0)_{256}, w_j = v_{2j+1} + v_{2j}.)\\
&\text{Наконец } \nu \leftarrow ((a \cdot y) \bmod 2^{64}) \gg 56, \text{ где } a = (11111111)_{256}.
\end{aligned} \tag{62}$$

Последний шаг, понятно, вычисляет $y \bmod 255 = w_7 + \cdots + w_1 + w_0$ путем умножения, используя тот факт, что сумма отлично помещается в восемь битов. [Дэвид Мюллер (David Muller) запрограммировал подобный метод для машины ILLIAC I в 1954 году.]

Если ожидается, что значение x "разреженное", содержащее малое количество битов 1, можно воспользоваться более быстрым методом [П. Вейден (P. Wegner), *CACM* **3** (1960), 322]:

$$\begin{aligned}
&\text{Установить } \nu \leftarrow 0,\ y \leftarrow x.\\
&\text{Затем, пока } y \neq 0, \text{ устанавливать } \nu \leftarrow \nu + 1,\ y \leftarrow y \,\&\, (y - 1).
\end{aligned} \tag{63}$$

Аналогичный подход, использующий $y \leftarrow y \mid (y+1)$, работает в случае "плотных" x.

Обращение порядка битов. В нашем следующем трюке выполним замену $x = (x_{63} \ldots x_1 x_0)_2$ его зеркальным отображением, $x^R = (x_0 x_1 \ldots x_{63})_2$. Любой, кто внимательно следил за событиями до этого времени и рассмотрел методы наподобие (50), (56), (57) и (62), вероятно, подумает "Ага, мы снова можем разделять (пополам) и властвовать! Если мы уже знаем, как обращать 32-битовые числа, то сможем обратить и 64-битовое число почти так же быстро, поскольку $(xy)^R = y^R x^R$. Все, что нам надо сделать, — это применить 32-битовый метод параллельно к обеим половинам регистра, а затем поменять местами левую половину и правую".

Правильно. Например, мы можем обратить 8-битовую строку с помощью трех простых шагов.

$$\begin{array}{ll}
\text{Задано} & x_7 x_6 x_5 x_4 x_3 x_2 x_1 x_0\\
\text{Обмен битов} & x_6 x_7 x_4 x_5 x_2 x_3 x_0 x_1\\
\text{Обмен пар битов} & x_4 x_5 x_6 x_7 x_0 x_1 x_2 x_3\\
\text{Обмен полубайтов} & x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7
\end{array} \tag{64}$$

А шесть таких простых шагов обращают 64 бит. К счастью, каждая из таких операций обмена оказывается достаточно простой, если воспользоваться магическими масками μ_k:

$$\begin{aligned}
&y \leftarrow (x \gg 1) \,\&\, \mu_0, \quad z \leftarrow (x \,\&\, \mu_0) \ll 1, \quad x \leftarrow y \mid z;\\
&y \leftarrow (x \gg 2) \,\&\, \mu_1, \quad z \leftarrow (x \,\&\, \mu_1) \ll 2, \quad x \leftarrow y \mid z;\\
&y \leftarrow (x \gg 4) \,\&\, \mu_2, \quad z \leftarrow (x \,\&\, \mu_2) \ll 4, \quad x \leftarrow y \mid z;\\
&y \leftarrow (x \gg 8) \,\&\, \mu_3, \quad z \leftarrow (x \,\&\, \mu_3) \ll 8, \quad x \leftarrow y \mid z;\\
&y \leftarrow (x \gg 16) \,\&\, \mu_4, \quad z \leftarrow (x \,\&\, \mu_4) \ll 16, \quad x \leftarrow y \mid z;\\
&x \leftarrow (x \gg 32) \mid ((x \ll 32) \bmod 2^{64}).
\end{aligned} \tag{65}$$

[Кристофер Страчи (Christopher Strachey) предсказал некоторые аспекты данного построения в *CACM* **4** (1961), 146, а подобный *тернарный* метод был разработан

в 1973 году Брюсом Баумгартом (Bruce Baumgart) (см. упр. 49). Тщательно проработанный алгоритм (65) представлен Генри С. Уорреном-мл. (Henry S. Warren, Jr.) в книге *Hacker's Delight** (Addison-Wesley, 2002), 102.]

MMIX в очередной раз оказался способным превзойти метод общего назначения с помощью менее традиционных команд, которые позволяют выполнить эту работу быстрее. Рассмотрим команды

```
rev GREG #0102040810204080; MOR x,x,rev; MOR x,rev,x;
```
(66)

Первая команда MOR обращает байты x из прямого порядка в обратный и наоборот, а вторая обращает биты в пределах каждого байта.

Обмен битов. Предположим, что мы хотим только лишь обменять два бита в пределах регистра, т. е. выполнить $x_i \leftrightarrow x_j$, где $i > j$. Как сделать это лучше всего? (Дорогой читатель, не спеши! Попытайся решить эту задачу в уме или на бумаге, перед тем как прочесть ответ ниже.)

Пусть $\delta = i - j$. Вот одно из решений (не подсматривайте, пока не решили задачу самостоятельно!):

$$y \leftarrow (x \gg \delta) \mathbin{\&} 2^j,\ z \leftarrow (x \mathbin{\&} 2^j) \ll \delta,\ x \leftarrow (x \mathbin{\&} m) \mid y \mid z,\ \text{где } \overline{m} = 2^i | 2^j. \tag{67}$$

Оно использует два сдвига и пять битовых булевых операций в предположении, что i и j представляют собой заданные константы. Решение похоже на первые строки (65), за исключением того, что требуется новая маска m, поскольку y и z не учитывают все биты x.

Можно, однако, поступить и лучше, сэкономив одну операцию и одну константу:

$$y \leftarrow (x \oplus (x \gg \delta)) \mathbin{\&} 2^j, \qquad x \leftarrow x \oplus y \oplus (y \ll \delta). \tag{68}$$

Первое присваивание теперь помещает $x_i \oplus x_j$ в позицию j; второе изменяет x_i на $x_i \oplus (x_i \oplus x_j)$ и x_j на $x_j \oplus (x_i \oplus x_j)$, как и требовалось. В общем случае часто оказывается разумно преобразовать задачу вида "заменить x на $f(x)$" в задачу "заменить x на $x \oplus g(x)$", поскольку $g(x)$ может быть проще вычислить.

С другой стороны, имеется смысл, в котором (67) может оказаться предпочтительнее (68), поскольку присваивание y и z в (67) иногда может выполняться одновременно. При использовании представления в виде схем (67) имеет глубину 4, в то время как (68) имеет глубину 5.

Операция (68), конечно же, может использоваться и для одновременного обмена нескольких пар битов при использовании маски θ более общего вида, чем 2^j:

$$y \leftarrow (x \oplus (x \gg \delta)) \mathbin{\&} \theta, \qquad x \leftarrow x \oplus y \oplus (y \ll \delta). \tag{69}$$

Назовем эту операцию δ-обменом (δ-swap), поскольку она позволяет выполнять обмен любых неперекрывающихся пар битов, где биты в парах находятся на расстоянии δ один от другого. Маска θ должна иметь 1 в крайней справа позиции каждой пары, в которой должен быть выполнен обмен. Например, (69) может

* Имеется перевод на русский язык: Генри С. Уоррен. *Алгоритмические трюки для программистов.* М.: Издательский дом "Вильямс", 2003. — *Примеч. пер.*

обменивать 25 крайних слева битов 64-битового слова с 25 крайними справа битами, оставляя 14 средних битов нетронутыми, если положить $\delta = 39$ и $\theta = 2^{25} - 1 = {}^{\#}\texttt{1ffffff}$.

В действительности имеется изумительный способ обращения порядка 64 битов с помощью δ-обмена, а именно

$$\begin{aligned}
&y \leftarrow (x \gg 1)\ \&\ \mu_0, \quad z \leftarrow (x\ \&\ \mu_0) \ll 1, \quad x \leftarrow y \mid z,\\
&y \leftarrow (x \oplus (x \gg 4))\ \&\ {}^{\#}\texttt{0300c0303030c303}, \quad x \leftarrow x \oplus y \oplus (y \ll 4),\\
&y \leftarrow (x \oplus (x \gg 8))\ \&\ {}^{\#}\texttt{00c0300c03f0003f}, \quad x \leftarrow x \oplus y \oplus (y \ll 8),\\
&y \leftarrow (x \oplus (x \gg 20))\ \&\ {}^{\#}\texttt{00000ffc00003fff}, \quad x \leftarrow x \oplus y \oplus (y \ll 20),\\
&x \leftarrow (x \gg 34) \mid ((x \ll 30) \bmod 2^{64}),
\end{aligned} \tag{70}$$

который экономит две битовые операции в (65), хотя способ (65) выглядит "оптимальным".

***Перестановка битов в общем случае.** Только что рассмотренные методы применимы и для получения *произвольной* перестановки битов в регистре. В действительности всегда существуют маски $\theta_0, \ldots, \theta_5, \hat\theta_4, \ldots, \hat\theta_0$, такие, что приведенные далее операции преобразуют $x = (x_{63} \ldots x_1 x_0)_2$ в любую требуемую перестановку $x^\pi = (x_{63\pi} \ldots x_{1\pi} x_{0\pi})_2$ его битов:

$$\begin{aligned}
&x \leftarrow 2^k\text{-обмен } x \text{ с маской } \theta_k \text{ для } k = 0, 1, 2, 3, 4, 5;\\
&x \leftarrow 2^k\text{-обмен } x \text{ с маской } \hat\theta_k \text{ для } k = 4, 3, 2, 1, 0.
\end{aligned} \tag{71}$$

В общем случае перестановка 2^d бит может быть достигнута с помощью $2d-1$ таких шагов с использованием соответствующих масок θ_k и $\hat\theta_k$, где расстояния обмена представляют собой соответственно $2^0, 2^1, \ldots, 2^{d-1}, \ldots, 2^1, 2^0$.

Для доказательства этого факта можно воспользоваться частным случаем перестановочных сетей, которые были независимо открыты Э. М. Дугвайдом (A. M. Duguid) и Ж. ле Корром (J. Le Corre) в 1959 году на основе более ранней работы Д. Слепиана (D. Slepian) [см. V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic* (New York: Academic Press, 1965), Section 3.3]. На рис. 12 показана перестановочная сеть $P(2n)$ для $2n$ элементов, построенная из двух перестановочных сетей для n элементов при $n = 4$. Каждое соединение ⚷ между двумя линиями представляет *модуль перекрещивания*, который при потоке данных слева направо либо оставляет содержимое линий неизменным, либо обменивает его. Для начала рекурсии при $n = 1$ мы положим $P(2)$ состоящим из одного перекрещивания. Каждая настройка отдельных перекрещиваний, очевидно, приводит к тому, что $P(2n)$ дает на выходе перестановку входных данных; и обратно, мы покажем, что любая перестановка $2n$ входов может быть получена при соответствующей настройке перекрещиваний.

Конструкцию на рис. 12 легче всего понять на конкретном примере. Предположим, мы хотим направить входы $(0, 1, 2, 3, 4, 5, 6, 7)$ в выходы $(3, 2, 4, 1, 6, 0, 5, 7)$ соответственно. Первая задача состоит в определении содержимого линий непосредственно после первого столбца перекрещиваний и непосредственно перед последним столбцом, поскольку тогда мы сможем применить аналогичный метод и для на-

Рис. 12. Внутреннее устройство черного ящика $P(2n)$, который выполняет перестановку $2n$ элементов всеми возможными способами при $n > 1$. (Иллюстрация для значения $n = 4$.)

стройки перекрещиваний во внутренних $P(4)$. Таким образом, в сети

(72)

необходимо найти перестановки abcdefgh и ABCDEFGH, такие, что $\{\mathtt{a},\mathtt{b}\} = \{0,1\}$, $\{\mathtt{c},\mathtt{d}\} = \{2,3\}$, ..., $\{\mathtt{g},\mathtt{h}\} = \{6,7\}$, $\{\mathtt{a},\mathtt{c},\mathtt{e},\mathtt{g}\} = \{\mathtt{A},\mathtt{C},\mathtt{E},\mathtt{G}\}$, $\{\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{h}\} = \{\mathtt{B},\mathtt{D},\mathtt{F},\mathtt{H}\}$, $\{\mathtt{A},\mathtt{B}\} = \{3,2\}$, $\{\mathtt{C},\mathtt{D}\} = \{4,1\}$, ..., $\{\mathtt{G},\mathtt{H}\} = \{5,7\}$. Начиная снизу, выберем $\mathtt{h} = 7$, поскольку мы не хотим трогать содержимое линий без необходимости. Тогда мы *вынуждены* сделать следующий выбор:

$$\mathtt{H}=7;\ \mathtt{G}=5;\ \mathtt{e}=5;\ \mathtt{f}=4;\ \mathtt{D}=4;\ \mathtt{C}=1;\ \mathtt{a}=1;\ \mathtt{b}=0;\ \mathtt{F}=0;\ \mathtt{E}=6;\ \mathtt{g}=6. \tag{73}$$

Если бы мы выбрали $\mathtt{h} = 6$, то получили бы аналогичный, но обратный шаблон

$$\mathtt{F}=6;\ \mathtt{E}=0;\ \mathtt{a}=0;\ \mathtt{b}=1;\ \mathtt{D}=1;\ \mathtt{C}=4;\ \mathtt{e}=4;\ \mathtt{f}=5;\ \mathtt{H}=5;\ \mathtt{G}=7;\ \mathtt{g}=7. \tag{74}$$

Выбор и (73), и (74) может быть завершен либо выбором $\mathtt{d} = 3$ (а следовательно, $\mathtt{B} = 3$, $\mathtt{A} = 2$, $\mathtt{c} = 2$), либо выбором $\mathtt{d} = 2$ (а следовательно, $\mathtt{B} = 2$, $\mathtt{A} = 3$, $\mathtt{c} = 3$).

В общем случае такой вынужденный шаблон состоит из циклов, независимо от того, с какой перестановки мы начинаем. Чтобы увидеть это, рассмотрим граф из восьми вершин {ab, cd, ef, gh, AB, CD, EF, GH}, который содержит ребро от uv к UV тогда, когда пара на входе, подключенная к uv, имеет общий элемент с парой на выходе, подключенной к UV. Таким образом, в нашем примере ребрами являются ab — EF, ab — CD, cd — AB, cd — AB, ef — CD, ef — GH, gh — EF, gh — GH. Мы имеем "двойную связь" между cd и AB, поскольку входы, подключенные к c и d, представляют собой выходы, подключенные к A и B; результат при этом несколько отклоняется от строгого определения графа. Мы видим, что каждая вершина

является соседней ровно с двумя другими вершинами, и вершины, помеченные строчными буквами, всегда являются смежными с вершинами, помеченными буквами прописными. Следовательно, граф всегда состоит из непересекающихся циклов четной длины. В нашем примере этими циклами являются

$$\begin{array}{ccccccc} & & \mathtt{EF} & - & \mathtt{gh} & & \\ \mathtt{ab} & \diagup & & & & \diagdown & \mathtt{GH} \\ & \diagdown & \mathtt{CD} & - & \mathtt{ef} & \diagup & \end{array} \qquad \mathtt{cd} = \mathtt{AB}, \tag{75}$$

где более длинный цикл соответствует (73) и (74). Если имеется k разных циклов, будет 2^k разных способов определить поведение первого и последнего столбцов перекрещиваний.

Для завершения сети мы можем таким же образом обработать внутренние 4-элементные перестановки; таким же рекурсивным способом можно получить *любую* перестановку 2^d элементов. Получающиеся настройки перекрещиваний определяют маски θ_j и $\hat{\theta}_j$ из (71). Некоторые выборы перекрещиваний могут привести к маскам, состоящим полностью из одних лишь нулей; в таком случае мы можем опустить соответствующий этап вычислений.

Наше построение показывает, как в случае, когда входные и выходные данные в нижних строках сети идентичны, обеспечить, чтобы никакие перекрещивания, касающиеся этих линий, не были активными. Например, 64-битовый алгоритм в (71) может использоваться и с 60-битовым регистром без необходимости в четырех дополнительных битах для любого из промежуточных результатов.

Конечно, зачастую в частных случаях мы можем превзойти процедуру из (71). Например, в упр. 52 показано, что метод (71) требует девяти шагов обмена для транспонирования матрицы 8×8, в то время как в действительности достаточно трех обменов.

Дано	7-обмен	14-обмен	28-обмен
00 01 02 03 04 05 06 07	00 **10** 02 **12** 04 **14** 06 **16**	00 10 **20** **30** 04 14 **24** **34**	00 10 20 30 **40** **50** **60** **70**
10 11 12 13 14 15 16 17	**01** 11 **03** 13 **05** 15 **07** 17	01 11 **21** **31** 05 15 **25** **35**	01 11 21 31 **41** **51** **61** **71**
20 21 22 23 24 25 26 27	20 **30** 22 **32** 24 **34** 26 **36**	**02** **12** 22 32 **06** **16** 26 36	02 12 22 32 **42** **52** **62** **72**
30 31 32 33 34 35 36 37	**21** 31 **23** 33 **25** 35 **27** 37	**03** **13** 23 33 **07** **17** 27 37	03 13 23 33 **43** **53** **63** **73**
40 41 42 43 44 45 46 47	40 **50** 42 **52** 44 **54** 46 **56**	40 50 **60** **70** 44 54 **64** **74**	**04** **14** **24** **34** 44 54 64 74
50 51 52 53 54 55 56 57	**41** 51 **43** 53 **45** 55 **47** 57	41 51 **61** **71** 45 55 **65** **75**	**05** **15** **25** **35** 45 55 65 75
60 61 62 63 64 65 66 67	60 **70** 62 **72** 64 **74** 66 **76**	**42** **52** 62 72 **46** **56** 66 76	**06** **16** **26** **36** 46 56 66 76
70 71 72 73 74 75 76 77	**61** 71 **63** 73 **65** 75 **67** 77	**43** **53** 63 73 **47** **57** 67 77	**07** **17** **27** **37** 47 57 67 77

"Идеальное тасование" представляет собой другую часто возникающую на практике перестановку битов. Если $x = (\ldots x_2x_1x_0)_2$ и $y = (\ldots y_2y_1y_0)_2$ являются любыми 2-адическими целыми числами, определим $x \ddagger y$ ("x zip y", *функцию застежки** от x и y) как чередование их битов:

$$x \ddagger y \;=\; (\ldots x_2y_2x_1y_1x_0y_0)_2. \tag{76}$$

Эта операция имеет важное приложение для представления двумерных данных, поскольку небольшое изменение либо в x, либо в y обычно приводит к небольшому изменению в $x \ddagger y$ (см. упр. 86). Обратите также внимание, что константы магических масок (47) удовлетворяют условию

$$\mu_k \ddagger \mu_k \;=\; \mu_{k+1}. \tag{77}$$

* *Zipper function.* Zipper (англ.) — застежка-молния. — *Примеч. пер.*

Если x находится в левой части регистра, а y — в его правой части, то идеальное тасование представляет собой перестановку, которая заменяет содержимое регистра на $x \ddagger y$.

Последовательность $d-1$ шагов обмена идеально тасует 2^d-битовый регистр; в действительности в упр. 53 показано, что имеется несколько способов решения этой задачи. Следовательно, мы еще раз оказались способными улучшить $(2d-1)$-шаговый метод (71) и рис. 12.

И обратно, предположим, у нас есть перетасованное значение $z = x \ddagger y$ в 2^d-битовом регистре; существует ли эффективный способ выделить исходное значение y? Конечно: если $d-1$ обменов, осуществляющих идеальное тасование, выполнить в обратном порядке, то будут восстановлены значения x и y. Но если требуется восстановить только y, то можно сэкономить половину работы. Начнем с $y \leftarrow z \& \mu_0$; затем установим $y \leftarrow (y + (y \gg 2^{k-1})) \& \mu_k$ для $k = 1, \ldots, d-1$. Например, когда $d = 3$, эта процедура выполняет $(0y_3 0y_2 0y_1 0y_0)_2 \mapsto (00y_3y_2 00y_1y_0)_2 \mapsto (0000y_3y_2y_1y_0)_2$. И вновь работает принцип "разделяй и властвуй".

Рассмотрим теперь более общую задачу, когда мы хотим выделить и сжать *произвольное* подмножество битов регистра. Предположим, что у нас есть 2^d-битовое слово $z = (z_{2^d-1} \ldots z_1 z_0)_2$ и маска $\chi = (\chi_{2^d-1} \ldots \chi_1 \chi_0)_2$, в которой s бит равны 1; таким образом, $\nu\chi = s$. Задача заключается в сборке компактного подслова

$$y \;=\; (y_{s-1} \ldots y_1 y_0)_2 \;=\; (z_{j_{s-1}} \ldots z_{j_1} z_{j_0})_2, \tag{78}$$

где $j_{s-1} > \cdots > j_1 > j_0$ — индексы, для которых $\chi_j = 1$. Например, если $d = 3$ и $\chi = (10110010)_2$, мы хотим выполнить преобразование $z = (y_3x_3y_2y_1x_2x_1y_0x_0)_2$ в $y = (y_3y_2y_1y_0)_2$. (Задача перехода от $x \ddagger y$ к y, рассмотренная выше, представляет собой частный случай $\chi = \mu_0$.) Из (71) мы знаем, что y может быть найдено с помощью не более $2d-1$ δ-обменов; но в нашей задаче необходимые данные всегда перемещаются только вправо, так что можно ускорить работу путем применения *сдвигов* вместо обменов.

Назовем δ-сдвигом x с маской θ операцию

$$x \;\leftarrow\; x \oplus \big((x \oplus (x \gg \delta)) \,\&\, \theta\big), \tag{79}$$

которая изменяет бит x_j на $x_{j+\delta}$, если θ содержит 1 в позиции j, но оставляет x_j неизменным в противном случае. Гай Стил (Guy Steele) открыл, что всегда существуют маски $\theta_0, \theta_1, \ldots, \theta_{d-1}$, такие, что общая задача выделения (78) может быть решена с помощью небольшого количества δ-сдвигов:

$$\begin{gathered}\text{Начать с } x \leftarrow z\text{; затем выполнить } 2^k\text{-сдвиг } x \text{ с маской } \theta_k \\ \text{для } k = 0, 1, \ldots, d-1\text{; наконец установить } y \leftarrow x.\end{gathered} \tag{80}$$

В действительности идея поиска соответствующих масок неожиданно проста. Каждый бит, который мы хотим переместить в конечном итоге на $l = (l_{d-1} \ldots l_1 l_0)_2$ позиций вправо, должен перемещаться в 2^k-сдвиге, для которого $l_k = 1$.

Предположим, например, что $d = 3$ и $\chi = (10110010)_2$. (Мы должны считать, что $\chi \neq 0$.) Вспоминая, что в результат должно быть "вдвинуто" слева некоторое количество нулей, мы можем установить $\theta_0 = (00011001)_2$, $\theta_1 = (00000110)_2$, $\theta_2 = (11111000)_2$; тогда (80) выполняет отображение

$$(y_3x_3y_2y_1x_2x_1y_0x_0)_2 \mapsto (y_3x_3y_2y_2y_1x_1y_0y_0)_2 \mapsto (y_3x_3y_2y_2y_1y_2y_1y_0)_2 \mapsto (0000y_3y_2y_1y_0)_2.$$

В упр. 69 доказывается, что выделяемые биты никогда не взаимодействуют один с другим во время своих путешествий. Кроме того, имеется легкий способ вычисления подходящих масок θ_k динамически из χ за $O(d^2)$ шагов (см. упр. 70).

Операция "отделения агнцев от козлищ" ("sheep-and-goats"), или группировки, предназначенная для аппаратного обеспечения компьютера, расширяет (78), генерируя обобщенное неперетасованное слово

$$(x_{r-1}\ldots x_1x_0y_{s-1}\ldots y_1y_0)_2 = (z_{i_{r-1}}\ldots z_{i_1}z_{i_0}z_{j_{s-1}}\ldots z_{j_1}z_{j_0})_2; \tag{81}$$

здесь $i_{r-1} > \cdots > i_1 > i_0$ — индексы, для которых $\chi_i = 0$. Однако более полезной и простой в реализации оказывается операция под названием "сбор и переворот" ("gather-flip"), которая обращает порядок немаскированных битов и дает

$$(x_0x_1\ldots x_{r-1}y_{s-1}\ldots y_1y_0)_2 = (z_{i_0}z_{i_1}\ldots z_{i_{r-1}}z_{j_{s-1}}\ldots z_{j_1}z_{j_0})_2. \tag{81$'$}$$

Любая перестановка 2^d бит достижима при использовании любой из операций не более d раз (см. упр. 72 и 73).

Сдвиги также позволяют нам выйти за рамки перестановок и перейти к произвольным *отображениям* битов в регистре. Предположим, что мы хотим преобразовать

$$x = (x_{2^d-1}\ldots x_1x_0)_2 \quad\mapsto\quad x^\varphi = (x_{(2^d-1)\varphi}\ldots x_{1\varphi}x_{0\varphi})_2, \tag{82}$$

где φ является одной из $(2^d)^{2^d}$ функций, отображающих множество $\{0,1,\ldots,2^d-1\}$ в себя. Ц. М. Чжун (K. M. Chung) и Ч. К. Вонг (C. K. Wong) [*IEEE Transactions* **C-29** (1980), 1029–1032] предложили привлекательный способ выполнения такого преобразования за $O(d)$ шагов с использованием *циклических* δ-сдвигов, которые подобны (79), за исключением того, что мы устанавливаем

$$x \leftarrow x \oplus \bigl((x \oplus (x \gg \delta) \oplus (x \ll (2^d-\delta)))\ \&\ \theta\bigr). \tag{83}$$

Их идея заключается в том, чтобы начать с выяснения количества индексов j, таких, что $j\varphi = l$, для $0 \le l < 2^d$, и обозначить их как c_l. Затем они находят маски θ_0, θ_1, $\ldots$, θ_{d-1}, обладающие тем свойством, что циклический 2^k-сдвиг x с маской θ_k, выполняющийся последовательно для $0 \le k < d$, будет преобразовывать x в число x', которое содержит ровно c_l копий бита x_l для каждого l. Наконец для замены $x' \mapsto x^\varphi$ можно использовать обобщенную процедуру перестановки (71).

Предположим, например, что $d = 3$ и $x^\varphi = (x_3x_1x_1x_0x_3x_7x_5x_5)_2$. Тогда мы имеем $(c_0,c_1,c_2,c_3,c_4,c_5,c_6,c_7) = (1,2,0,2,0,2,0,1)$. Используя маски $\theta_0 = (00011100)_2$, $\theta_1 = (00001000)_2$ и $\theta_2 = (01100000)_2$, три циклических 2^k-сдвига отображают $x = (x_7x_6x_5x_4x_3x_2x_1x_0)_2 \mapsto (x_7x_6x_5x_5x_4x_3x_1x_0)_2 \mapsto (x_7x_6x_5x_5x_5x_3x_1x_0)_2 \mapsto (x_7x_3x_1x_5x_5x_3x_1x_0)_2 = x'$. Затем выполняем несколько δ-обменов: $x' \mapsto (x_3x_7x_5x_1x_3x_5x_1x_0)_2 \mapsto (x_3x_1x_5x_7x_3x_5x_1x_0)_2 \mapsto (x_3x_1x_1x_0x_3x_5x_5x_7)_2 \mapsto (x_3x_1x_1x_0x_3x_7x_5x_5)_2 = x^\varphi$; работа сделана! Конечно, любое 8-битовое отображение может быть достигнуто более быстро с помощью "грубой силы", по одному биту за раз. Метод Чжуна и Вонга становится более впечатляющим при 256-битовом регистре. Но он неплох даже в случае 64-битового регистра `MMIX`, требуя в худшем случае не более 96 тактов.

Чтобы найти θ_0, воспользуемся тем фактом, что $\sum c_l = 2^d$, и рассмотрим $\Sigma_{\text{четные}} = \sum c_{2l}$ и $\Sigma_{\text{нечетные}} = \sum c_{2l+1}$. Если $\Sigma_{\text{четные}} = \Sigma_{\text{нечетные}} = 2^{d-1}$, можно установить $\theta_0 = 0$ и пропустить циклический 1-сдвиг. Но если, скажем, $\Sigma_{\text{четные}} <$

$\Sigma_{\text{нечетные}}$, то мы найдем четное l, такое, что $c_l = 0$. Циклический сдвиг в биты l, $l+1$, $\ldots$, $l+t$ (по модулю 2^d) для некоторого t даст новые величины $(c'_0, \ldots, c'_{2^d-1})$, для которых $\Sigma'_{\text{четные}} = \Sigma'_{\text{нечетные}} = 2^{d-1}$; так что $\theta_0 = 2^l + \cdots + 2^{(l+t) \bmod 2^d}$. Тогда можно работать с битами в четных и нечетных позициях по отдельности, используя тот же метод, пока не дойдем до 1-битовых подслов. Детальное описание см. в упр. 74.

Работа с фрагментированными полями. Вместо выделения битов из различных частей слова и сбора их вместе часто можно просто работать с этими битами в их исходных позициях.

Предположим, например, что мы хотим пройти по всем подмножествам данного множества U, где, как обычно, множество определяется с помощью маски χ, такой, что $[k \in U] = (\chi \gg k) \mathbin{\&} 1$. Если $x \subseteq \chi$ и $x \neq \chi$, имеется простой способ вычисления следующего наибольшего подмножества U в лексикографическом порядке, а именно наименьшего целого числа $x' > x$, такого, что $x' \subseteq \chi$:

$$x' \;=\; (x - \chi) \mathbin{\&} \chi. \tag{84}$$

В частном случае, когда $x = 0$ и $\chi \neq 0$, мы уже видели в (37), что эта формула дает крайний справа бит χ, который соответствует лексикографически наименьшему непустому подмножеству U.

Почему работает формула (84)? Представим прибавление 1 к числу $x|\bar{\chi}$, которое содержит единицы там, где χ равно нулю. Перенос будет распространяться по всем этим единицам, пока не достигнет крайней справа позиции, где x имеет нуль, а χ — единицу; кроме того, все биты справа от этой позиции станут равными нулю. Следовательно, $x' = ((x \mid \bar{\chi}) + 1) \mathbin{\&} \chi$. Но мы имеем $(x \mid \bar{\chi}) + 1 = (x + \bar{\chi}) + 1 = x + (\bar{\chi} + 1) = x - \chi$ при $x \subseteq \chi$. QED.

Обратите также внимание, что $x' = 0$ тогда и только тогда, когда $x = \chi$. Так что мы будем знать, когда найдем наибольшее подмножество. В упр. 79 показано, как вернуться к x, зная x'.

Мы можем также захотеть пройти по всем элементам *подкуба*, например, чтобы найти все битовые шаблоны, которые отвечают спецификации наподобие $*10*1*01$, состоящие из нулей, единиц и безразличных значений. Такая спецификация может быть представлена кодами звездочек $a = (a_{n-1} \ldots a_0)_2$ и кодами битов $b = (b_{n-1} \ldots b_0)_2$, как в упр. 7.1.1–30; наш пример соответствует $a = (10010100)_2$, $b = (01001001)_2$. Задача перечисления всех подмножеств множества является частным случаем, где $a = \chi$ и $b = 0$. В более общей задаче подкуба преемником данного битового шаблона x является

$$x' \;=\; ((x - (a + b)) \mathbin{\&} a) + b. \tag{85}$$

Предположим, что биты числа $z = (z_{n-1} \ldots z_0)_2$ "сшиты" из двух подслов, $x = (x_{r-1} \ldots x_0)_2$ и $y = (y_{s-1} \ldots y_0)_2$, где $r + s = n$, с использованием произвольной маски χ, у которой $\nu\chi = s$, для управления "сшиванием". Например, $z = (y_2 x_4 x_3 y_1 x_2 y_0 x_1 x_0)_2$ при $n = 8$ и $\chi = (10010100)_2$. Мы можем рассматривать z как "рассеянный аккумулятор", в котором "чужие" биты x_i спрятаны среди "своих" битов y_j. С этой точки зрения задача поиска последовательных элементов подкуба, по сути, является задачей вычисления $y+1$ внутри рассеянного аккумулятора z без изменения значения x. Операция группировки (81) может разделить x и y; но она

дорогá, а (85) показывает, что мы можем решить задачу без нее. Действительно, мы можем вычислить $y + y'$, когда $y' = (y'_{s-1} \ldots y'_0)_2$ представляет собой *любое* значение внутри рассеянного аккумулятора z', если и y, и y' появляются в позициях, определенных χ. Рассмотрим $t = z \mathbin{\&} \chi$ и $t' = z' \mathbin{\&} \chi$. Если мы образуем сумму $(t \mid \bar{\chi}) + t'$, все переносы, образующиеся при обычном сложении $y + y'$, будут переданы через блоки единиц в $\bar{\chi}$ так же, как если бы рассеянные биты были смежными. Таким образом,

$$((z \mathbin{\&} \chi) + (z' \mid \bar{\chi})) \mathbin{\&} \chi \tag{86}$$

представляет собой сумму y и y' по модулю 2^s, рассеянную в соответствии с маской χ.

Работа с несколькими байтами одновременно. Вместо того чтобы сосредоточиваться на данных в одном поле в пределах слова, нам зачастую хотелось бы работать одновременно с двумя или более подсловами, выполняя вычисления с каждым из них параллельно. Например, многие приложения требуют обработки длинных последовательностей байтов, и мы можем ускорить работу, выполняя действия над восемью байтами одновременно; можно также использовать все 64 бит, предоставляемые нашей машиной. Обобщенные многобайтовые технологии были впервые предложены Лесли Лампортом (Leslie Lamport) в *CACM* **18** (1975), 471–475, и впоследствии развиты многими программистами.

Предположим сначала, что мы просто хотим взять две последовательности байтов и найти их сумму, рассматривая их как координаты векторов и для каждого байта используя арифметику по модулю 256. Говоря алгебраически, у нас имеются 8-байтовые векторы $x = (x_7 \ldots x_1 x_0)_{256}$ и $y = (y_7 \ldots y_1 y_0)_{256}$; мы хотим вычислить $z = (z_7 \ldots z_1 z_0)_{256}$, где $z_j = (x_j + y_j) \bmod 256$ для $0 \le j < 8$. Обычное сложение x с y не работает, поскольку нам нужно предотвратить распространение переносов между байтами. Так что мы выделяем биты высокого порядка и работаем с ними отдельно:

$$\begin{aligned} &z \leftarrow (x \oplus y) \mathbin{\&} h, \qquad \text{где } h = {}^{\#}\texttt{8080808080808080};\\ &z \leftarrow ((x \mathbin{\&} \bar{h}) + (y \mathbin{\&} \bar{h})) \oplus z. \end{aligned} \tag{87}$$

Общее время, затрачиваемое `MMIX` на выполнение этих действий, составляет 6υ плюс $3\mu + 3\upsilon$, если учесть время на загрузку x, загрузку y и сохранение z. Восемь же однобайтовых сложений (последовательность `LDBU`, `LDBU`, `ADDU` и `STBU`, повторенная восемь раз) стоили бы $8 \times (3\mu + 4\upsilon) = 24\mu + 32\upsilon$. Параллельное *вычитание* байтов почти такое же простое (см. упр. 88).

Мы можем также вычислить побайтовое *среднее* с $z_j = \lfloor (x_j + y_j)/2 \rfloor$ для каждого j:

$$\begin{aligned} &z \leftarrow ((x \oplus y) \mathbin{\&} \bar{l}) \gg 1, \qquad \text{где } l = {}^{\#}\texttt{0101010101010101};\\ &z \leftarrow (x \mathbin{\&} y) + z. \end{aligned} \tag{88}$$

Этот элегантный трюк, предложенный Г. Г. Дитцем (H. G. Dietz), основан на хорошо известной формуле

$$x + y \;=\; (x \oplus y) + ((x \mathbin{\&} y) \ll 1) \tag{89}$$

для двоичного сложения. (Мы можем реализовать (88) с помощью четырех, а не пяти команд `MMIX`, поскольку одна операция `MOR` меняет $x \oplus y$ на $((x \oplus y) \mathbin{\&} \bar{l}) \gg 1$.)

В упр. 88–93 и 100–104 эти идеи получают дальнейшее развитие, показывая, как работать с арифметикой в системе счисления со смешанным основанием, а также как выполнять такие действия, как сложение и вычитание векторов, компоненты которых рассматриваются по модулю m, когда m не обязательно является степенью 2.

По сути, мы можем рассматривать биты, байты или другие подполя регистра как если бы это были элементы массива независимых микропроцессоров, работающих независимо над собственными подзадачами, но при этом строго синхронизированными и сообщающимися один с другим посредством команд сдвига и битов переноса. Разработчики компьютеров многие годы развивали параллельные процессоры с так называемой SIMD-архитектурой, т. е. "Single Instruction stream with Multiple Data streams" (один поток команд с несколькими потоками данных); см., например, S. H. Unger, *Proc. IRE* **46** (1958), 1744–1750. Увеличившаяся доступность 64-битовых регистров означает, что программисты обычных последовательных компьютеров могут теперь ощутить всю прелесть SIMD-обработки. Действительно, такие вычисления, как (87)–(89), называются SWAR-методами (SIMD Within A Register, SIMD в пределах регистра) — имя, предложенное Р. Д. Фишером (R. J. Fisher) и Г. Г. Дитцем (H. G. Dietz) [см. *Lecture Notes in Computer Science* **1656** (1999), 290–305]. См. также R. B. Lee, *IEEE Micro* **16**, 4 (August 1996), 51–59.

Конечно, байты часто содержат алфавитные данные, а не только числа, и одной из наиболее важных задач программирования является поиск в длинной строке символов первого появления некоторого значения байта. Например, строки часто имеют представление в виде последовательности ненулевых байтов с завершающим нулем. Чтобы быстро найти конец строки, нужен быстрый способ определения, все ли восемь байтов заданного слова x ненулевые (поскольку обычно так и есть). Несколько неплохих решений этой задачи были найдены Лампортом (Lamport) и другими; но Алан Майкрофт (Alan Mycroft) в 1987 году обнаружил, что достаточно всего *трех* команд:

$$t \leftarrow h \,\&\, (x - l) \,\&\, \bar{x}, \tag{90}$$

где h и l указаны в (87) и (88). Если каждый байт x_j не нулевой, t будет равно нулю; $(x_j - 1) \,\&\, \bar{x}_j$ будет равно $2^{\rho x_j} - 1$, что всегда меньше $\#\mathtt{80} = 2^7$. Но если $x_j = 0$, в то время как его соседи справа $x_{j-1}, \ldots, x_0$ (если таковые имеются) ненулевые, вычитание $x - l$ даст $\#\mathtt{ff}$ в байте j и t будет ненулевым. В действительности ρt будет равно $8j + 7$.

Внимание: хотя вычисления в (90) выявляют *крайний справа* нулевой байт x, мы не можем выявить позицию *крайнего слева* нулевого байта на основании только лишь значения t. (См. упр. 94.) В этом отношении прямой порядок байтов оказывается предпочтительнее обратного. Приложение, которому требуется найти крайний слева нулевой байт, может использовать (90) для быстрого пропуска ненулевых слов, но после этого оно должно перейти к медленному поиску среди восьми байтов-финалистов. Следующая формула с четырьмя операциями дает совершенно точный результат проверки $t = (t_7 \ldots t_1 t_0)_{256}$, в котором $t_j = 128[x_j = 0]$ для каждого j:

$$t \leftarrow h \,\&\, {\sim}(x \mid ((x \mid h) - l)). \tag{91}$$

Крайний слева нулевой байт x — байт x_j, где $\lambda t = 8j + 7$.

Кстати, единственная команда MMIX ‘BDIF t,l,x’ немедленно решает задачу нулевого байта путем установки каждого байта t_j слова t равным $[x_j = 0]$, поскольку $1 \dot{-} x = [x = 0]$. Но здесь нас, в первую очередь, интересуют достаточно универсальные методы, не полагающиеся на экзотические аппаратные решения; особые возможности MMIX будут рассмотрены ниже.

Теперь, когда мы знаем быстрый способ поиска первого 0, мы можем использовать те же самые идеи для *любого* интересующего нас значения байта. Например, для проверки, является ли некоторый из байтов слова x символом новой строки (#a), мы просто ищем нулевой байт в $x \oplus$ #0a0a0a0a0a0a0a0a.

Эти методы открывают и многие другие двери. Предположим, например, что мы хотим вычислить $z = (z_7 \ldots z_1 z_0)_{256}$ из x и y, где $z_j = x_j$ при $x_j = y_j$, и $z_j =$ '*', когда $x_j \neq y_j$. (Таким образом если $x =$ 'beaching' и $y =$ 'belching', предполагается, что будет выполнена установка $z \leftarrow$ 'be*ching'.) Это просто:

$$\begin{aligned} &t \leftarrow h \mathbin{\&} ((x \oplus y) \mid (((x \oplus y) \mid h) - l)); \\ &m \leftarrow (t \ll 1) - (t \gg 7); \\ &z \leftarrow x \oplus ((x \oplus \texttt{'********'}) \mathbin{\&} m). \end{aligned} \tag{92}$$

На первом шаге используется вариант (91) для того, чтобы установить старшие биты каждого байта, где $x_j \neq y_j$. На следующем шаге создается маска для этих байтов: #00, если $x_j = y_j$, в противном случае #ff. А на последнем шаге, который можно также записать как $z \leftarrow (x \mathbin{\&} \overline{m}) \mid (\texttt{'********'} \mathbin{\&} m)$, устанавливается $z_j \leftarrow x_j$ или $z_j \leftarrow$ '*' в зависимости от маски.

Операции (90) и (91) изначально разрабатывались для проверки равенства байтов нулю; но более близкое знакомство показывает, что мы можем поступить более мудро и рассматривать их как проверку значений байтов, не меньше ли они 1. В действительности, если в любой формуле заменить l на $c \cdot l = (cccccccc)_{256}$, где c — любая положительная константа ≤ 128, то можно использовать (90) или (91) для того, чтобы узнать, имеются ли в x байты, меньшие, чем c. Кроме того, значение для сравнения c не обязано быть одним и тем же в каждой байтовой позиции; а ценой некоторого увеличения объема работы мы сможем также выполнять побайтовое сравнение и в случаях, когда $c > 128$. Вот 8-шаговая формула, которая устанавливает $t_j \leftarrow 128[x_j < y_j]$ для каждой позиции байта j в тестовом слове t:

$$t \;\leftarrow\; h \mathbin{\&} {\sim}\langle x \bar{y} z \rangle, \qquad \text{где } z = (x \mid h) - (y \mathbin{\&} \bar{h}). \tag{93}$$

(См. упр. 96.) Операция медианы в этой обобщенной формуле часто может быть упрощена; например, (93) сводится к (91) при $y = l$, поскольку $\langle x(-1)z \rangle = x \mid z$.

Найдя ненулевое t в (90), (91) или (93), мы можем захотеть вычислить ρt или λt, чтобы найти индекс j крайнего справа или слева байта, который был помечен. Задача вычисления ρ или λ сейчас проще, чем раньше, поскольку t может принимать только 256 различных значений. Действительно, для вычисления j достаточно операции

$$j \;\leftarrow\; table[((a \cdot t) \bmod 2^{64}) \gg 56], \quad \text{где } a = \frac{2^{56} - 1}{2^7 - 1}, \tag{94}$$

если иметь соответствующую 256-байтовую таблицу. А умножение здесь может зачастую быть выполнено быстрее при использовании вместо него трех операций “сдвиг и сложение”, “$t \leftarrow t + (t \ll 7)$, $t \leftarrow t + (t \ll 14)$, $t \leftarrow t + (t \ll 28)$”.

Вычисления с большими словами. К этому моменту мы познакомились более чем с десятком способов, которыми побитовые операции с высокой скоростью приводят к удивительным результатам. В приведенных ниже упражнениях содержится гораздо больше сюрпризов.

Элвин Берлекамп (Elwyn Berlekamp) заметил, что при разработке компьютерных микросхем с N триггерами продолжается наращивание значения N, хотя на практике в каждый момент реально задействовано только $O(\log N)$ из них. Удивительная эффективность битовых операций наводит на мысль о том, что компьютеры будущего сумеют воспользоваться этим нетронутым потенциалом с помощью расширенных модулей памяти, которые обеспечат эффективное выполнение n-битовых вычислений для достаточно больших значений n. Готовясь к этому времени, мы должны придумать хорошее название для концепции работы с "широкими словами". Лайл Рэмшоу (Lyle Ramshaw) предложил забавный термин "*broadword*" (дословно — "широкое слово"; здесь имеется определенная игра слов — broadword созвучно Broadway (Бродвей). Так что в русском языке можно воспользоваться не менее забавно звучащим термином "шлово". — *Примеч. пер.*), так что мы можем говорить о n-битовых величинах как о шловах шириной n.

Многие из рассматривавшихся методов *2-адические*, в том смысле что они корректно работают с бинарными числами произвольной (даже бесконечной) точности. Например, операция $x \mathbin{\&} -x$ всегда выделяет $2^{\rho x}$, наименьший единичный бит любого ненулевого 2-адического целого числа x. Другие же методы обладают, по сути, широкословной природой, такие как методы, использующие $O(d)$ шагов для вычисления контрольной суммы или перестановки битов 2^d-битового слова. Широкословное вычисление — это искусство работы с n-битовыми словами, когда n представляет собой не слишком малый параметр.

Одни из широкословных алгоритмов представляют собой только теоретический интерес, будучи эффективными лишь в асимптотическом смысле, когда n превышает размер Вселенной. Другие оказываются в высшей степени практичными даже при $n = 64$. В общем случае широкословный склад ума часто приводит к хорошим технологиям.

Одним очаровательным, но непрактичным фактом, касающимся широкословных операций, является открытие М. Л. Фредманом (M. L. Fredman) и Д. Э. Уиллардом (D. E. Willard) того, что достаточно $O(1)$ широкословных шагов для вычисления функции $\lambda x = \lfloor \lg x \rfloor$ для любого ненулевого n-битового числа x, причем величина n роли не играет. Вот эта замечательная схема при $n = g^2$ и g, являющемся степенью 2:

$$
\begin{aligned}
&t_1 \leftarrow h \mathbin{\&} (x \mid ((x \mid h) - l)), \quad \text{где } h = 2^{g-1} l \text{ и } l = (2^n - 1)/(2^g - 1);\\
&y \leftarrow (((a \cdot t_1) \bmod 2^n) \gg (n-g)) \cdot l, \quad \text{где } a = (2^{n-g} - 1)/(2^{g-1} - 1);\\
&t_2 \leftarrow h \mathbin{\&} (y \mid ((y \mid h) - b)), \quad \text{где } b = (2^{n+g} - 1)/(2^{g+1} - 1);\\
&m \leftarrow (t_2 \ll 1) - (t_2 \gg (g-1)),\ m \leftarrow m \oplus (m \gg g);\\
&z \leftarrow (((l \cdot (x \mathbin{\&} m)) \bmod 2^n) \gg (n-g)) \cdot l;\\
&t_3 \leftarrow h \mathbin{\&} (z \mid ((z \mid h) - b));\\
&\lambda \leftarrow ((l \cdot ((t_2 \gg (2g - \lg g - 1)) + (t_3 \gg (2g-1)))) \bmod 2^n) \gg (n-g).
\end{aligned}
\tag{95}
$$

(См. упр. 106.) Этот метод непрактичен, поскольку пять из приведенных 29 шагов являются умножениями, так что они в действительности не являются "битовыми"

операциями. Позже мы докажем, что умножение на константу требует как минимум $\Omega(\log n)$ битовых шагов.

Способ нахождения λx без умножений, только лишь с помощью $O(\log\log n)$ битовых широкословных операций, был открыт в 1997 году Гертом Бродалом (Gerth Brodal), метод которого даже более замечателен, чем (95). Он основан на формуле, аналогичной (49),

$$\lambda x \;=\; [\lambda x = \lambda(x \mathbin{\&} \bar\mu_0)] + 2[\lambda x = \lambda(x \mathbin{\&} \bar\mu_1)] + 4[\lambda x = \lambda(x \mathbin{\&} \bar\mu_2)] + \cdots, \tag{96}$$

и том факте, что соотношение $\lambda x = \lambda y$ легко проверяется (см. (58)).

Алгоритм B (*Бинарный логарифм*). Этот алгоритм использует n-битовые операции для вычисления $\lambda x = \lfloor \lg x \rfloor$, в предположении, что $0 < x < 2^n$ и $n = d \cdot 2^d$.

B1. [Уменьшение масштаба.] Установить $\lambda \leftarrow 0$. Затем установить $\lambda \leftarrow \lambda + 2^k$ и $x \leftarrow x \gg 2^k$, если $x \ge 2^{2^k}$, для $k = \lceil \lg n \rceil - 1$, $\lceil \lg n \rceil - 2$, $\ldots$, d.

B2. [Репликация.] (В этот момент $0 < x < 2^{2^d}$; остающаяся задача заключается в увеличении λ на $\lfloor \lg x \rfloor$. Мы заменяем x на d его копий, в 2^d-битовых полях.) Установить $x \leftarrow x \mid (x \ll 2^{d+k})$ для $0 \le k < \lceil \lg d \rceil$.

B3. [Изменение ведущих битов.] Установить $y \leftarrow x \mathbin{\&} {\sim}(\mu_{d,d-1} \ldots \mu_{d,1}\mu_{d,0})_{2^{2^d}}$. (См. (48).)

B4. [Сравнение всех полей.] Установить $t \leftarrow h \mathbin{\&} (y \mid ((y \mid h) - (x \oplus y)))$, где $h = (2^{2^d-1} \ldots 2^{2^d-1}2^{2^d-1})_{2^{2^d}}$.

B5. [Сжатие битов.] Установить $t \leftarrow (t + (t \ll (2^{d+k} - 2^k))) \bmod 2^n$ для $0 \le k < \lceil \lg d \rceil$.

B6. [Завершение.] Наконец установить $\lambda \leftarrow \lambda + (t \gg (n-d))$. ▮

Этот алгоритм практически конкурирует с (56) при $n = 64$ (см. упр. 107).

Еще один удивительно эффективный широкословный алгоритм был открыт в 2006 году М. С. Патерсоном (M. S. Paterson) и автором этой книги, которые рассматривали задачу идентификации всех появлений шаблона 01^r в заданной n-битовой бинарной строке. Эта задача, связанная с поиском r смежных свободных блоков при распределении памяти, эквивалентна вычислению

$$q \;=\; \bar x \mathbin{\&} (x \ll 1) \mathbin{\&} (x \ll 2) \mathbin{\&} (x \ll 3) \mathbin{\&} \cdots \mathbin{\&} (x \ll r) \tag{97}$$

для заданного $x = (x_{n-1} \ldots x_1 x_0)_2$. Так, если $n = 16$, $r = 3$ и $x = (1110111101100111)_2$, мы имеем $q = (0001000000001000)_2$. Интуитивно можно ожидать, что потребуется $\Omega(\log r)$ битовых операций. Но на самом деле для всех $n > r > 0$ работа выполняется с помощью следующего 20-шагового вычисления. Пусть $s = \lceil r/2 \rceil$, $l = \sum_{k\ge0} 2^{ks} \bmod 2^n$, $h = (2^{s-1}l) \bmod 2^n$ и $a = \bigl(\sum_{k\ge0}(-1)^{k+1}2^{2ks}\bigr) \bmod 2^n$.

$$\begin{aligned}
&y \leftarrow h \mathbin{\&} x \mathbin{\&} ((x \mathbin{\&} \bar h) + l);\\
&t \leftarrow (x + y) \mathbin{\&} \bar x \mathbin{\&} -2^r;\\
&u \leftarrow t \mathbin{\&} a,\;\; v \leftarrow t \mathbin{\&} \bar a;\\
&m \leftarrow (u - (u \gg r)) \mid (v - (v \gg r));\\
&q \leftarrow t \mathbin{\&} ((x \mathbin{\&} m) + ((t \gg r) \mathbin{\&} {\sim}(m \ll 1))).
\end{aligned} \tag{98}$$

В упр. 111 поясняется, почему этот интригующий набор операций корректно работает. Данный метод имеет слишком малую практическую ценность; имеется простой

способ вычисления (97) за $2\lceil \lg r \rceil + 2$ шагов, так что (98) невыгоден, пока значение r не превысит 512. Но (98) — еще один указатель неожиданной мощи широкословных методов.

***Нижние границы.** Существование такого большого количества трюков и методов естественным образом приводит к вопросу, не находимся ли мы в самом начале пути? Может, все сделанное до сих пор — только небольшая царапина на поверхности, и имеется множество невероятно быстрых методов, которые ожидают своего открытия? Известно несколько теоретических результатов, которые позволяют вывести некоторые ограничения на возможные методы. К сожалению, эти исследования все еще находятся в зачаточном состоянии.

Назовем *2-адической цепочкой* последовательность $(x_0, x_1, \ldots, x_r)$ 2-адических целых чисел, в которой каждый элемент x_i для $i > 0$ получается из своих предшественников с помощью битовых манипуляций. Точнее, мы хотим, чтобы шаги цепочки определялись бинарными операциями

$$x_i \;=\; x_{j(i)} \circ_i x_{k(i)} \quad \text{или} \quad c_i \circ_i x_{k(i)}, \quad \text{или} \quad x_{j(i)} \circ_i c_i, \tag{99}$$

где каждое $\circ_i$ представляет собой один из операторов $\{+, -, \&, |, \oplus, \equiv, \subset, \supset, \overline{\subset}, \overline{\supset}, \overline{\wedge}, \overline{\vee}, \ll, \gg\}$, а каждое c_i представляет собой константу. Кроме того, когда оператор $\circ_i$ является левым или правым сдвигом, величина сдвига должна быть положительной целой константой; операции, такие как $x_{j(i)} \ll x_{k(i)}$ или $c_i \gg x_{k(i)}$, *не* разрешены. (Без этого последнего ограничения мы не сможем вывести значимые нижние границы, поскольку *каждая* функция со значениями 0–1 от неотрицательного целого числа x может быть вычислена за два шага как "$(c \gg x)$ & 1" для некоторой константы c.)

Аналогично *широкословная цепочка* шириной n, также именуемая n-битовой широкословной цепочкой, представляет собой последовательность $(x_0, x_1, \ldots, x_r)$ n-битовых чисел с, по сути, теми же ограничениями, где n является параметром, а все операции выполняются по модулю 2^n. Во многом широкословные цепочки ведут себя, как 2-адические, но с некоторыми тонкими отличиями из-за потерь информации в левой части n-битовых вычислений (см. упр. 113).

Оба типа цепочек вычисляют функцию $f(x) = x_r$, когда мы начинаем их с некоторого заданного значения $x = x_0$. В упр. 114 показано, что mn-битовая широкословная цепочка в состоянии выполнять m, по сути, одновременных вычислений любой функции, которая вычислима n-битовой цепочкой. Наша цель заключается в изучении *кратчайших* цепочек, которые в состоянии вычислить данную функцию f.

Любая 2-адическая или широкословная цепочка $(x_0, x_1, \ldots, x_r)$ имеет последовательность "множеств сдвига" $(S_0, S_1, \ldots, S_r)$ и "границ" $(B_0, B_1, \ldots, B_r)$, определяемых следующим образом. Начнем с $S_0 = \{0\}$ и $B_0 = 1$; затем для $i \geq 1$ положим

$$S_i = \begin{cases} S_{j(i)} \cup S_{k(i)}, \\ S_{k(i)}, \\ S_{j(i)}, \\ S_{j(i)} + c_i, \\ S_{j(i)} - c_i, \end{cases} \quad \text{и} \quad B_i = \begin{cases} M_i B_{j(i)} B_{k(i)}, & \text{если } x_i = x_{j(i)} \circ_i x_{k(i)}, \\ M_i B_{k(i)}, & \text{если } x_i = c_i \circ_i x_{k(i)}, \\ M_i B_{j(i)}, & \text{если } x_i = x_{j(i)} \circ_i c_i, \\ B_{j(i)}, & \text{если } x_i = x_{j(i)} \gg c_i, \\ B_{j(i)}, & \text{если } x_i = x_{j(i)} \ll c_i, \end{cases} \tag{100}$$

где $M_i = 2$, если $\circ_i \in \{+, -\}$ и $M_i = 1$ в противном случае, и в этих формулах предполагается, что $\circ_i \notin \{\ll, \gg\}$. Например, рассмотрим следующую 7-шаговую

цепочку.

$$\begin{array}{lcc} \quad x_i & S_i & B_i \\ x_0 = x & \{0\} & 1 \\ x_1 = x_0 \mathbin{\&} -2 & \{0\} & 1 \\ x_2 = x_1 + 2 & \{0\} & 2 \\ x_3 = x_2 \gg 1 & \{1\} & 2 \\ x_4 = x_2 + x_3 & \{0,1\} & 8 \\ x_5 = x_4 \gg 4 & \{4,5\} & 8 \\ x_6 = x_4 + x_5 & \{0,1,4,5\} & 128 \\ x_7 = x_6 \gg 4 & \{4,5,8,9\} & 128 \end{array} \tag{101}$$

(Мы встречались с ней в упр. 4.4–9, где доказывалось, что эти операции дают $x_7 = \lfloor x/10 \rfloor$ для $0 \le x < 160$ при работе с 8-битовой арифметикой.)

Чтобы приступить к теории нижних границ, заметим для начала, что старшие биты $x = x_0$ не могут влиять на младшие биты иначе, кроме как при сдвиге вправо.

Лемма A. *Пусть в данной 2-адической или широкословной цепочке бинарным представлением x_i является $(\ldots x_{i2}x_{i1}x_{i0})_2$. Тогда бит x_{ip} может зависеть от бита x_{0q}, только если $q \le p + \max S_i$.*

Доказательство. По индукции по i можно показать, что, если $B_i = 1$, бит x_{ip} может зависеть от бита x_{0q}, только если $q - p \in S_i$. Сложение и вычитание, которые обеспечивают $B_i > 1$, позволяют любому конкретному биту их операндов влиять на все биты, лежащие в сумме или разности слева, но не на те, которые лежат справа. ▮

Следствие I. *Функция $x \dot{-} 1$ не может быть вычислена 2-адической цепочкой, как не может быть вычислена любая функция, как минимум один бит $f(x)$ которой зависит от неограниченного количества битов x.* ▮

Следствие W. *n-битовая функция $f(x)$ может быть вычислена n-битовой широкословной цепочкой без сдвигов тогда и только тогда, когда из $x \equiv y$* (по модулю 2^p) *вытекает $f(x) \equiv f(y)$* (по модулю 2^p) *для $0 \le p < n$.*

Доказательство. При отсутствии сдвигов мы имеем $S_i = \{0\}$ для всех i. Таким образом, бит x_{rp} не может зависеть от бита x_{0q}, если только не выполняется $q \le p$. Другими словами, мы должны иметь $x_r \equiv y_r$ (по модулю 2^p) при $x_0 \equiv y_0$ (по модулю 2^p).

И обратно, все такие функции достижимы с помощью достаточно длинной цепочки. В упр. 119 дана бессдвиговая n-битовая цепочка для функций

$$f_{py}(x) \;=\; 2^p[x \bmod 2^{p+1} = y] \qquad \text{при } 0 \le p < n \text{ и } 0 \le y < 2^{p+1}, \tag{102}$$

из которой все интересующие нас функции получаются путем сложения. [Г. С. Уоррен-мл. (H. S. Warren, Jr.) обобщил этот результат для функций от m переменных в *CACM* **20** (1977), 439–441.] ▮

Множества сдвигов S_i и границы B_i важны главным образом из-за фундаментальной леммы, которая является нашим основным инструментом для получения нижних границ.

Лемма B. *Пусть* $X_{pqr} = \{x_r \,\&\, \lfloor 2^p - 2^q \rfloor \mid x_0 \in V_{pqr}\}$ *в* n*-битовой широкословной цепочке, где*

$$V_{pqr} \;=\; \{x \mid x \,\&\, \lfloor 2^{p+s} - 2^{q+s} \rfloor = 0 \text{ для всех } s \in S_r\} \tag{103}$$

и $p > q$. *Тогда* $|X_{pqr}| \le B_r$. (Здесь p и q — целые числа, возможно, отрицательные.)

Эта лемма утверждает, что, когда определенные интервалы битов в x должны быть нулями, в $f(x)$ может иметься не более B_r различных битовых шаблонов $x_{r(p-1)} \ldots x_{rq}$.

Доказательство. Этот результат, определенно, выполняется при $r = 0$. В противном случае, если, например, $x_r = x_j + x_k$, по индукции мы знаем, что $|X_{pqj}| \le B_j$ и $|X_{pqk}| \le B_k$. Кроме того, $V_{pqr} = V_{pqj} \cap V_{pqk}$, поскольку $S_r = S_j \cup S_k$. Таким образом, возникает не более $B_j B_k$ вариантов для $(x_j + x_k) \,\&\, \lfloor 2^p - 2^q \rfloor$ при отсутствии переноса в позицию q и не более $B_j B_k$ при его наличии, что дает итоговое значение не более $B_r = 2B_j B_k$ вариантов. В упр. 122 рассматриваются другие случаи. ▮

Теперь мы можем доказать, что линеечная функция требует $\Omega(\log\log n)$ шагов.

Теорема R. *Если* $n = d \cdot 2^d$, *каждая* n*-битовая широкословная цепочка, вычисляющая* ρx *для* $0 < x < 2^n$, *имеет более* $\lg d$ *шагов без сдвигов.*

Доказательство. Если имеется l бессдвиговых шагов, мы имеем $|S_r| \le 2^l$ и $B_r \le 2^{2^l - 1}$. Применим лемму B с $p = d$ и $q = 0$ и предположим, что $|X_{d0r}| = 2^d - t$. Тогда имеется t значений $k < 2^d$, таких, что

$$\{2^k,\, 2^{k+2^d},\, 2^{k+2\cdot 2^d},\, \ldots,\, 2^{k+(d-1)2^d}\} \cap V_{d0r} \;=\; \emptyset.$$

Но V_{d0r} исключает не более $2^l d$ из n возможных степеней 2; так что $t \le 2^l$.

Если $l \le \lg d$, лемма B говорит, что $2^d - t \le B_r \le 2^{d-1}$; следовательно, $2^{d-1} \le t \le 2^l \le d$. Но это возможно только при $d \le 2$, когда теорема очевидно выполняется. ▮

То же доказательство работает и в случае бинарного логарифма.

Следствие L. *Если* $n = d \cdot 2^d > 2$, *каждая* n*-битовая широкословная цепочка, которая вычисляет* λx *для* $0 < x < 2^n$, *имеет более* $\lg d$ *шагов без сдвигов.* ▮

Используя лемму B с $q > 0$, мы можем вывести более сильную нижнюю границу $\Omega(\log n)$ для обращения битов, а следовательно и для перестановки битов в общем случае.

Теорема P. *Если* $2 \le g \le n$, *каждая* n*-битовая широкословная цепочка, которая вычисляет* g*-битовое обращение* x^R *для* $0 \le x < 2^g$, *имеет как минимум* $\lfloor \frac{1}{3} \lg g \rfloor$ *бессдвиговых шагов.*

Доказательство. Как и ранее, считаем, что имеется l бессдвиговых шагов. Положим $h = \lfloor \sqrt[3]{g} \rfloor$ и предположим, что $l < \lfloor \lg(h+1) \rfloor$. Тогда S_r является множеством не более $2^l \le \frac{1}{2}(h+1)$ величин сдвигов s. Применим лемму B с $p = q + h$, где $p \le g$ и $q \ge 0$, таким образом, всего в $g - h + 1$ случаях. Ключевым наблюдением является то, что $x^R \,\&\, \lfloor 2^p - 2^q \rfloor$ не зависит от $x \,\&\, \lfloor 2^{p+s} - 2^{q+s} \rfloor$, если не имеется таких индексов j и k, что $0 \le j, k < h$ и $g - 1 - q - j = q + s + k$. Количество "плохих" выборов q, для которых такие индексы существуют, не превышает $\frac{1}{2}(h+1)h^2 \le g - h$; следовательно, как минимум один "хороший" выбор q дает $|X_{pqr}| = 2^h$. Но тогда лемма B приводит к противоречию, поскольку мы, очевидно, не можем иметь $2^h \le B_r \le 2^{(h-1)/2}$. ▮

Следствие M. *Умножение на некоторую константу по модулю 2^n требует $\Omega(\log n)$ шагов в n-битовой широкословной цепочке.*

Доказательство. В хаке 167 классического меморандума HAKMEM (M.I.T. A.I. Laboratory, 1972) Ричард Шрёппель (Richard Schroeppel) заметил, что операции

$$t \leftarrow ((ax) \bmod 2^n) \mathbin{\&} b, \quad y \leftarrow ((ct) \bmod 2^n) \gg (n-g) \tag{104}$$

вычисляют $y = x^R$ для $n = g^2$ и $0 \le x < 2^g$ с использованием констант $a = (2^{n+g} - 1)/(2^{g+1}-1)$, $b = 2^{g-1}(2^n-1)/(2^g-1)$ и $c = (2^{n-g}-1)/(2^{g-1}-1)$. (См. упр. 123.) ▮

В этом месте читатель может задуматься: "Ну, хорошо, я согласен с тем, что широкословные цепочки иногда должны быть асимптотически длинными. Но программисты не обязаны ограничивать себя такими цепочками; мы можем использовать и другие методы, такие как условное ветвление или обращение к предвычисленным таблицам, что позволит нам преодолеть указанные ограничения".

Верно. Но нам везет, потому что широкословная теория может быть распространена и на другие, более общие модели вычислений. Рассмотрим, например, следующую идеализацию абстрактного RISC-компьютера, который называется компьютером с *базовой памятью* (*basic RAM*): машина имеет n-битовые регистры $r_1, \ldots, r_l$ и n-битовые слова памяти $\{M[0], \ldots, M[2^m-1]\}$. Она может выполнять команды

$$\begin{gathered} r_i \leftarrow r_j \pm r_k, \quad r_i \leftarrow r_j \circ r_k, \quad r_i \leftarrow r_j \gg r_k, \quad r_i \leftarrow c, \\ r_i \leftarrow M[r_j \bmod 2^m], \quad M[r_j \bmod 2^m] \leftarrow r_i, \end{gathered} \tag{105}$$

где $\circ$ — любой битовый булев оператор и где r_k в команде сдвига рассматривается как знаковое целое число в дополнительном двоичном коде. Машина также способна выполнять ветвление, если $r_i \le r_j$, рассматривая r_i и r_j как беззнаковые целые числа. Ее *состояние* представляет собой все содержимое регистров и памяти, а также "счетчика команд", который указывает на текущую команду. Программа начинается с предопределенного состояния, которое может включать предвычисленные таблицы в памяти, и с n-битовым входным значением x в регистре r_1. Это начальное состояние называется $Q(x,0)$, а $Q(x,t)$ обозначает состояние после t выполненных команд. После остановки машины r_1 будет содержать некоторое n-битовое значение $f(x)$. Для заданной функции $f(x)$ мы хотим найти нижнюю границу наименьшего значения t, при котором регистр r_1 равен $f(x)$ в состоянии $Q(x,t)$ для $0 \le x < 2^n$.

Теорема R′. *Пусть $\epsilon = 2^{-e}$. Базовая n-битовая память с параметром $m \le n^{1-\epsilon}$ требует как минимум $\lg\lg n - e$ шагов для вычисления линеечной функции ρx при $n \to \infty$.*

Доказательство. Пусть $n = 2^{2^{e+f}}$, так что $m \le 2^{2^{e+f}-2^f}$. В упр. 124 поясняется, как всезнающий наблюдатель может построить широкословную цепочку для определенного класса входных x таким образом, что каждое x приводит к одним и тем же ветвлениям с использованием одних и тех же величин сдвигов и с обращениями к одним и тем же ячейкам памяти. После этого можно применить наши рассматривавшиеся ранее методы для того, чтобы показать, что эта цепочка имеет длину $\ge f$. ▮

Скептически настроенный читатель может возразить, что теорема R′ не имеет практического значения, поскольку $\lg\lg n$ в реальности никогда не превысит 6. На это возражение ответить нечем. Но следующий результат несколько более существен.

Теорема P′. *Базовая n-битовая память требует как минимум $\frac{1}{3}\lg g$ шагов для вычисления g-битового обращения x^R при $0 \le x < 2^g$, если $g \le n$ и*

$$\max(m, 1+\lg n) \;<\; \frac{h+1}{2\lfloor \lg(h+1)\rfloor - 2}, \qquad h = \lfloor \sqrt[3]{g}\rfloor. \tag{106}$$

Доказательство. Рассуждения, подобные доказательству теоремы R′, приводятся в упр. 125. ▮

Лемма B и теоремы R, P, R′, P′ и их следствия получены А. Бродником (A. Brodnik), П. Б. Мильтерсеном (P. B. Miltersen) и Д. Я. Мунро (J. I. Munro), *Lecture Notes in Comp. Sci.* **1272** (1997), 426–439, основывавшимися на ранней работе Мильтерсена в *Lecture Notes in Comp. Sci.* **1099** (1996), 442–453.

Остается много нерешенных вопросов (см. упр. 126–130). Например, требует ли контрольное суммирование n-битовой широкословной цепочки с $\Omega(\log n)$ шагами? Можно ли вычислить функцию четности $(\nu x) \bmod 2$ или функцию мажоризации $[\nu x > n/2]$ существенно быстрее, чем само значение νx "пошловно"?

Приложение к ориентированным графам. Давайте теперь воспользуемся коечем из изученного, реализовав простой алгоритм. Для заданного ориентированного графа на множестве вершин V мы записываем $u \longrightarrow v$, когда имеется дуга от u до v. *Задача достижимости* заключается в поиске всех вершин, которые лежат на ориентированных путях, начинающихся в заданном множестве $Q \subseteq V$; другими словами, мы ищем множество

$$R \;=\; \{v \mid u \longrightarrow^* v \text{ для некоторого } u \in Q\}, \tag{107}$$

где $u \longrightarrow^* v$ означает существование последовательности t дуг

$$u = u_0 \longrightarrow u_1 \longrightarrow \cdots \longrightarrow u_t = v \qquad \text{для некоторого } t \ge 0. \tag{108}$$

Эта задача часто возникает на практике. Например, мы встречались с ней в разделе 2.3.5, когда помечали все элементы списков, не являющиеся "мусором".

Если количество вершин мало́, скажем, $|V| \le 64$, мы можем испытать подход к задаче достижимости, существенно отличающийся от использовавшегося ранее, работая непосредственно с подмножествами вершин. Пусть

$$S[u] \;=\; \{v \mid u \longrightarrow v\} \tag{109}$$

представляет собой множество преемников вершины u для всех $u \in V$. Тогда следующий алгоритм почти полностью отличается от алгоритма 2.3.5E, хотя и решает ту же абстрактную задачу.

Алгоритм R (*Достижимость*). Для заданного простого ориентированного графа, представленного множествами преемников $S[u]$ из (109), этот алгоритм вычисляет элементы R, достижимые из данного множества Q.

R1. [Инициализация.] Установить $R \leftarrow Q$ и $X \leftarrow \emptyset$. (В следующих шагах X представляет собой подмножество вершин $u \in R$, для которых мы уже выполнили просмотр $S[u]$.)

R2. [Выполнено?] Если $X = R$, завершить работу алгоритма.

R3. [Проверка другой вершины.] Пусть u является элементом $R \setminus X$. Установить $X \leftarrow X \cup u$, $R \leftarrow R \cup S[u]$ и вернуться к шагу R2. ▮

Этот алгоритм корректно работает, поскольку (i) каждый элемент, помещенный в R, достижим; (ii) каждый достижимый элемент u_j из (108) по индукции по j присутствует в R; и (iii) алгоритм в конечном счете завершает свою работу, поскольку шаг R3 всегда увеличивает $|X|$.

Для реализации алгоритма R будем считать, что $V = \{0, 1, \ldots, n-1\}$, где $n \le 64$. Множество X удобно представить целым числом $\sigma(X) = \sum\{2^u \mid u \in X\}$, и то же соглашение хорошо работает и для других множеств Q, R и $S[u]$. Заметим, что биты чисел $S[0]$, $S[1]$, $\ldots$, $S[n-1]$, по сути, представляют собой *матрицу смежности* данного ориентированного графа, как пояснялось в разделе 7, но с прямым порядком байтов: "диагональные" элементы, которые говорят нам о том, входит ли вершина в множество преемников, $u \in S[u]$, располагаются справа налево. Например, если $n = 3$ и дугами являются $\{0\to0, 0\to1, 1\to0, 2\to0\}$, мы имеем $S[0] = (011)_2$ и $S[1] = S[2] = (001)_2$, в то время как матрица смежности представляет собой $\left(\begin{smallmatrix}110\\100\\100\end{smallmatrix}\right)$.

Шаг R3 позволяет выбирать любой элемент из $R\setminus X$, так что мы используем для выбора наименьшего элемента линеечную функцию: $u \leftarrow \rho(\sigma(R) - \sigma(X))$. Битовые операции не требуют каких-либо иных трюков при адаптации данного алгоритма для машины MMIX.

Программа R (*Достижимость*). Входное множество Q задается в регистре q, а каждое множество преемников $S[u]$ находится в октете $\mathrm{M}_8[\mathtt{suc} + 8u]$. Выходное множество R будет располагаться в регистре r; другие регистры — s, t, tt, u и x — хранят промежуточные результаты.

01	1H SET	r,q	1	*R1. Инициализация.* r $\leftarrow \sigma(Q)$.
02	SET	x,0	1	x $\leftarrow \sigma(\emptyset)$.
03	JMP	2F	1	Переход к шагу R2.
04	3H SUBU	tt,t,1	$\lvert R\rvert$	*R3. Исследование другой вершины.* tt $\leftarrow$ t $-$ 1.
05	SADD	u,tt,t	$\lvert R\rvert$	$u \leftarrow \rho($t$)$ [см. (46)].
06	SLU	s,u,3	$\lvert R\rvert$	s $\leftarrow 8u$.
07	LDOU	s,suc,s	$\lvert R\rvert$	s $\leftarrow \sigma(S[u])$.
08	ANDN	tt,t,tt	$\lvert R\rvert$	tt $\leftarrow$ t & $\sim$tt $= 2^u$.
09	OR	x,x,tt	$\lvert R\rvert$	$X \leftarrow X \cup u$; т. е. x $\leftarrow$ x $\mid 2^u$, поскольку x $= \sigma(X)$.
10	OR	r,r,s	$\lvert R\rvert$	$R \leftarrow R \cup S[u]$; т. е. r $\leftarrow$ r $\mid$ s, поскольку r $= \sigma(R)$.
11	2H SUBU	t,r,x	$\lvert R\rvert + 1$	*R2. Выполнено?* t $\leftarrow$ r $-$ x $= \sigma(R \setminus X)$, так как $X \subseteq R$.
12	PBNZ	t,3B	$\lvert R\rvert + 1$	Переход к шагу R3, если $R \ne X$. ▮

Общее время работы равно $(\mu + 9\upsilon)|R| + 7\upsilon$. Для сравнения в упр. 131 алгоритм R реализован с применением связанных списков; в этом случае общее время работы увеличивается до $(3S+4|R|-2|Q|+1)\mu+(5S+12|R|-5|Q|+4)\upsilon$, где $S = \sum_{u\in R}|S[u]|$. (Но, конечно, зато такая программа в состоянии работать с графами с миллионами вершин.)

В упр. 132 представлен другой поучительный алгоритм, в котором битовые операции хорошо справляются с небольшими графами.

Применение для представления данных. Компьютеры бинарны, но (увы?) окружающий мир таковым не является. Нам часто требуется найти способ кодирования данных, не являющихся бинарными, с помощью нулей и единиц. Одной из наиболее распространенных задач такого вида является выбор эффективного представления элементов, которые могут находиться ровно в трех различных состояниях.

Предположим, мы знаем, что $x \in \{a, b, c\}$, и хотим представить x двумя битами $x_l x_r$. Мы можем, например, отобразить $a \mapsto 00$, $b \mapsto 01$ и $c \mapsto 10$. Но есть много других возможностей: в действительности 4 варианта для a, затем 3 варианта для b и 2 для c, т. е. всего 24 варианта. С одними из них может оказаться работать проще, чем с другими, в зависимости от того, что мы хотим делать с x.

Для двух заданных элементов $x, y \in \{a, b, c\}$ мы обычно хотим вычислить $z = x \circ y$ для некоторой бинарной операции $\circ$. Если $x = x_l x_r$ и $y = y_l y_r$, то $z = z_l z_r$, где

$$z_l = f_l(x_l, x_r, y_l, y_r) \qquad \text{и} \qquad z_r = f_r(x_l, x_r, y_l, y_r); \tag{110}$$

эти булевы функции f_l и f_r от четырех переменных зависят от $\circ$ и выбранного представления. Нас интересует представление, которое упрощает вычисление f_l и f_r.

Предположим, например, что $\{a, b, c\} = \{-1, 0, +1\}$ и что $\circ$ представляет собой умножение. Если мы решим использовать естественное отображение $x \mapsto x \bmod 3$, а именно

$$0 \mapsto 00, \qquad +1 \mapsto 01, \qquad -1 \mapsto 10, \tag{111}$$

так что $x = x_r - x_l$, то таблицы истинности для f_l и f_r будут иметь вид соответственно

$$f_l \leftrightarrow 000{*}001{*}010{*}{*}{*}{*}{*} \qquad \text{и} \qquad f_r \leftrightarrow 000{*}010{*}001{*}{*}{*}{*}{*}. \tag{112}$$

(Имеется семь безразличных значений для случаев $x_l x_r = 11$ и/или $y_l y_r = 11$.) Методы из раздела 7.1.2 говорят нам, как можно оптимально вычислить z_l и z_r, а именно

$$z_l = (x_l \oplus y_l) \wedge (x_r \oplus y_r), \qquad z_r = (x_l \oplus y_r) \wedge (x_r \oplus y_l); \tag{113}$$

к сожалению, функции f_l и f_r в (112) независимы в том смысле, что обе они не могут быть вычислены менее чем за $C(f_l) + C(f_r) = 6$ шагов.

С другой стороны, несколько менее естественная схема отображения

$$+1 \mapsto 00, \qquad 0 \mapsto 01, \qquad -1 \mapsto 10 \tag{114}$$

приводит к функциям преобразования

$$f_l \leftrightarrow 001{*}000{*}100{*}{*}{*}{*}{*} \qquad \text{и} \qquad f_r \leftrightarrow 010{*}111{*}010{*}{*}{*}{*}{*}, \tag{115}$$

и теперь для проведения вычислений достаточно трех операций:

$$z_r = x_r \vee y_r, \qquad z_l = (x_l \oplus y_l) \wedge \bar{z}_r. \tag{116}$$

Существует ли простой путь выявления таких усовершенствований? К счастью, нам не придется проверять все 24 возможных варианта, поскольку многие из них, по сути, одинаковы. Например, отображение $x \mapsto x_r x_l$ эквивалентно отображению

$x \mapsto x_l x_r$, поскольку новое представление $x'_l x'_r = x_r x_l$ получается путем обмена координат, делающего

$$f'_l(x'_l, x'_r, y'_l, y'_r) = z'_l = z_r = f_r(x_l, x_r, y_l, y_r);$$

новые функции преобразования f'_l и f'_r, определяемые как

$$f'_l(x_l, x_r, y_l, y_r) = f_r(x_r, x_l, y_r, y_l), \quad f'_r(x_l, x_r, y_l, y_r) = f_l(x_r, x_l, y_r, y_l), \tag{117}$$

имеют ту же сложность, что и f_l и f_r. Аналогично можно дополнять координату, полагая $x'_l x'_r = \bar{x}_l x_r$; тогда функциями преобразования оказываются

$$f'_l(x_l, x_r, y_l, y_r) = \bar{f}_l(\bar{x}_l, x_r, \bar{y}_l, y_r), \quad f'_r(x_l, x_r, y_l, y_r) = f_r(\bar{x}_l, x_r, \bar{y}_l, y_r), \tag{118}$$

и сложность, по сути, остается неизменной.

Многократное применение обмена и/или дополнения приводит к восьми отображениям, которые эквивалентны любому заданному. Так что 24 варианта приводятся только к трем, которые мы назовем классами I, II и III:

$$\begin{array}{lcccl}
 & \text{Класс I} & \text{Класс II} & \text{Класс III} & \\
a \mapsto & \overbrace{00\ 01\ 10\ 11\ 00\ 10\ 01\ 11} & \overbrace{00\ 01\ 10\ 11\ 00\ 10\ 01\ 11} & \overbrace{00\ 01\ 10\ 11\ 00\ 10\ 01\ 11} & ; \\
b \mapsto & 01\ 00\ 11\ 10\ 10\ 00\ 11\ 01 & 01\ 00\ 11\ 10\ 10\ 00\ 11\ 01 & 11\ 10\ 01\ 00\ 11\ 01\ 10\ 00 & ; \\
c \mapsto & 10\ 11\ 00\ 01\ 01\ 11\ 00\ 10 & 11\ 10\ 01\ 00\ 11\ 01\ 10\ 00 & 01\ 00\ 11\ 10\ 10\ 00\ 11\ 01 & .
\end{array} \tag{119}$$

Для выбора представления необходимо рассмотреть только по одному представителю каждого класса. Например, если $a = +1$, $b = 0$ и $c = -1$, представление (111) принадлежит классу II, а (114) относится к классу I. Как оказывается, класс III имеет стоимость 3, как и класс I. Так что представление (114), где z вычисляется как в (116), с точки зрения рассматриваемой задачи трехэлементного умножения столь же хорошее, как и любое другое.

Однако обратите внимание, что мы не обязаны отображать $\{a, b, c\}$ на *единственные* двухбитовые коды. Рассмотрим отображение "один ко многим"

$$+1 \mapsto 00, \qquad 0 \mapsto 01 \text{ или } 11, \qquad -1 \mapsto 10, \tag{120}$$

где и 01, и 11 разрешены как представления нуля. Таблицы истинности для f_l и f_r при этом достаточно сильно отличаются от (112) и (115), поскольку все входные данные разрешены, но некоторые выходы могут быть произвольными:

$$f_l \leftrightarrow 0{*}1{*}{*}{*}{*}1{*}0{*}{*}{*}{*} \qquad \text{и} \qquad f_r \leftrightarrow 0101111101011111. \tag{121}$$

При таком подходе фактически достаточно двух операций вместо трех в (116):

$$z_l = x_l \oplus y_l, \qquad z_r = x_r \vee y_r. \tag{122}$$

Небольшое размышление показывает, что эти операции действительно дают произведение $z = x \cdot y$, когда три элемента, $\{+1, 0, -1\}$, представлены так, как показано в (120).

Такое "неединственное" представление добавляет еще 36 возможных вариантов к имеющимся 24, которые мы рассматривали выше. И вновь они сводятся к небольшому количеству классов эквивалентности. Во-первых, имеются три класса, которые мы назовем IV_a, IV_b и IV_c в зависимости от того, какой из элементов имеет

неоднозначное представление:

$$\begin{array}{lcccccccccccccccccccccccccccccccc}
 & & \multicolumn{8}{c}{\text{Класс IV}_a} & & \multicolumn{8}{c}{\text{Класс IV}_b} & & \multicolumn{8}{c}{\text{Класс IV}_c} & \\
a & \mapsto & 0* & 0* & 1* & 1* & *0 & *0 & *1 & *1 & 11 & 10 & 01 & 00 & 11 & 01 & 10 & 00 & 10 & 11 & 00 & 01 & 01 & 11 & 00 & 10; \\
b & \mapsto & 10 & 11 & 00 & 01 & 01 & 11 & 00 & 10 & 0* & 0* & 1* & 1* & *0 & *0 & *1 & *1 & 11 & 10 & 01 & 00 & 11 & 01 & 10 & 00; \\
c & \mapsto & 11 & 10 & 01 & 00 & 11 & 01 & 10 & 00 & 10 & 11 & 00 & 01 & 01 & 11 & 00 & 10 & 0* & 0* & 1* & 1* & *0 & *0 & *1 & *1.
\end{array} \tag{123}$$

(Представление (120) принадлежит классу IV_b. Классы IV_a и IV_c в случае $z = x{\cdot}y$ работают плохо.) Во-вторых, имеются еще три класса с четырьмя отображениями в каждом:

$$\begin{array}{lcccccccccccccc}
 & & \multicolumn{4}{c}{\text{Класс V}_a} & & \multicolumn{4}{c}{\text{Класс V}_b} & & \multicolumn{4}{c}{\text{Класс V}_c} \\
a & \mapsto & tt & t\bar{t} & t\bar{t} & tt & 10 & 11 & 00 & 01 & 01 & 00 & 11 & 10; \\
b & \mapsto & 01 & 00 & 11 & 10 & tt & t\bar{t} & t\bar{t} & tt & 10 & 11 & 00 & 01; \\
c & \mapsto & 10 & 11 & 00 & 01 & 01 & 00 & 11 & 10 & tt & t\bar{t} & t\bar{t} & tt.
\end{array} \tag{124}$$

Эти классы представляют собой источник неприятностей, поскольку неопределенность в их таблицах истинности не может быть выражена просто в терминах безразличных значений, как это было сделано в (121). Например, если мы рассмотрим первое отображение класса V_a

$$+1 \mapsto 00 \text{ или } 11, \qquad 0 \mapsto 01, \qquad -1 \mapsto 10, \tag{125}$$

то нам понадобятся такие бинарные переменные $pqrst$, что

$$f_l \leftrightarrow p01q000010r1s01t \qquad \text{и} \qquad f_r \leftrightarrow p10q111101r0s10t. \tag{126}$$

Кроме того, отображения классов V_a, V_b и V_c почти никогда не оказываются лучше, чем отображения прочих шести классов (см. упр. 138). Тем не менее, чтобы быть уверенными в том, что найдено оптимальное отображение, следует проверить представителей всех девяти классов.

На практике мы часто хотим выполнять над переменными с тернарными значениями не единственную операцию наподобие умножения, а несколько различных операций. Например, мы можем захотеть вычислять как $\max(x, y)$, так и $x{\cdot}y$. В случае представления (120) лучшее, что мы можем сделать, — это $z_l = x_l \wedge y_l$, $z_r = (x_l \wedge y_r) \vee (x_r \wedge (y_l \vee y_r))$; но зато в этом случае выигрывает "естественное" отображение (111) с его $z_l = x_l \wedge y_l$, $z_r = x_r \vee y_r$. Класс III, как оказывается, имеет стоимость 4; другие классы ему проигрывают. Для выбора в этом случае между классами II, III и IV_b нам нужно знать относительные частоты $x{\cdot}y$ и $\max(x, y)$. Предположим, мы добавим в список операций еще и $\min(x, y)$. Классы II, III и IV_b вычисляют эту функцию со стоимостью 2, 5 и 5 соответственно; следовательно, лучшим выглядит представление (111).

Тернарные операции max и min возникают и в других контекстах, таких как трехзначная логика, разработанная в 1917 году Яном Лукашевичем (Jan Łukasiewicz). [См. его *Selected Works*, ed. by L. Borkowski (1970), 84–88, 153–178.] Рассмотрим логические значения "true" (истина), "false" (ложь) и "maybe" (может быть), обозначаемые соответственно как 1, 0 и $*$. Лукашевич определил три базовые опе-

рации конъюнкции, дизъюнкции и импликации на этих значениях с помощью таблиц

$$\begin{array}{c|ccc|} & \multicolumn{3}{c}{\overbrace{0\quad *\quad 1}^{y}} \\ \hline x\left\{\begin{array}{c}0*\\1\end{array}\right. & 0 & 0 & 0 \\ & 0 & * & * \\ & 0 & * & 1 \\ \hline \end{array}, \qquad \begin{array}{c|ccc|} & \multicolumn{3}{c}{\overbrace{0\quad *\quad 1}^{y}} \\ \hline x\left\{\begin{array}{c}0*\\1\end{array}\right. & 0 & * & 1 \\ & * & * & 1 \\ & 1 & 1 & 1 \\ \hline \end{array}, \qquad \begin{array}{c|ccc|} & \multicolumn{3}{c}{\overbrace{0\quad *\quad 1}^{y}} \\ \hline x\left\{\begin{array}{c}0*\\1\end{array}\right. & 1 & 1 & 1 \\ & * & 1 & 1 \\ & 0 & * & 1 \\ \hline \end{array}. \tag{127}$$

$$x \wedge y \qquad\qquad x \vee y \qquad\qquad x \Rightarrow y$$

Для этих операций представленные выше методы показывают, что бинарное представление

$$0 \mapsto 00, \qquad * \mapsto 01, \qquad 1 \mapsto 11 \tag{128}$$

работает отлично, поскольку мы можем вычислить логические операции следующим образом:

$$\begin{aligned} x_l x_r \wedge y_l y_r &= (x_l \wedge y_l)(x_r \wedge y_r), \qquad x_l x_r \vee y_l y_r = (x_l \vee y_l)(x_r \vee y_r), \\ x_l x_r \Rightarrow y_l y_r &= ((\bar{x}_l \vee y_l) \wedge (\bar{x}_r \vee y_r))\,(\bar{x}_l \vee y_r). \end{aligned} \tag{129}$$

Конечно, в этом обсуждении x не обязано быть изолированным тернарным значением; нам часто приходится работать с тернарными *векторами* $x = x_1 x_2 \ldots x_n$, где каждое x_j является либо a, либо b, либо c. Такие тернарные векторы удобно представлять двумя бинарными векторами

$$x_l = x_{1l} x_{2l} \ldots x_{nl} \qquad \text{и} \qquad x_r = x_{1r} x_{2r} \ldots x_{nr}, \tag{130}$$

где $x_j \mapsto x_{jl} x_{jr}$, как и ранее. Можно также упаковать тернарные значения в двухбитовые поля одного вектора,

$$x = x_{1l} x_{1r} x_{2l} x_{2r} \ldots x_{nl} x_{nr}, \tag{131}$$

который будет неплохо работать при, скажем, операторах $\wedge$ и $\vee$ логики Лукашевича (Łukasiewicz), но не в случае $\Rightarrow$. Однако обычно двухвекторный подход (130) оказывается лучшим решением, поскольку позволяет выполнять битовые вычисления без сдвигов и масок.

Приложения к структурам данных. Битовые операции предлагают множество эффективных способов представления элементов данных и отношений между ними. Например, в программах для игры в шахматы часто используется “битовая доска” для представления позиций фигур (см. упр. 143).

В главе 8 мы рассмотрим важные структуры данных, разработанные Петером ван Эмде Боасом (Peter van Emde Boas) для представления динамически изменяющихся подмножеств целых чисел между 0 и N. Вставки, удаления и другие операции, такие как “найти наибольший элемент, меньший, чем x”, могут быть выполнены его методами за $O(\log\log N)$ шагов; общая идея заключается в рекурсивной организации всей структуры как $\sqrt{N}$ подструктур для подмножеств интервалов размером $\sqrt{N}$ вместе со вспомогательной структурой, которая говорит о том, какие из этих интервалов заняты. [См. *Information Processing Letters* **6** (1977), 80–82; а также P. van Emde Boas, R. Kaas и E. Zijlstra, *Math. Systems Theory* **10** (1977), 99–127.] Битовые операции делают эти операции очень быстрыми.

Иерархические данные иногда могут быть организованы таким образом, что связи между элементами оказываются неявными. Например, в разделе 5.2.3 мы изучали “пирамиды”, где n элементов последовательного массива неявно обладают структурой бинарного дерева наподобие

(132)

когда, скажем, $n = 10$. (Здесь номера узлов приведены как в десятичной, так и в двоичной записях.) Нет необходимости хранить в памяти указатели для связи узла j пирамиды с его родительским узлом (который представляет собой узел $j \gg 1$, если $j \neq 1$) или с его братом (который представляет собой узел $j \oplus 1$, если $j \neq 1$), или с его дочерними узлами (которыми являются узлы $j \ll 1$ и $(j \ll 1) + 1$, если эти числа не превосходят n), поскольку простые вычисления ведут непосредственно от j к любому требующемуся соседу по пирамиде.

Аналогично *скошенная пирамида* предоставляет неявные связи для другого полезного семейства n-узловых структур бинарного дерева, типичным представителем которого при $n = 10$ является

(133)

(Иногда нам приходится выходить за пределы n при переходе от узла к его родительскому узлу, как на показанном здесь пути от 10 через 12 к 8.) И пирамиды, и скошенные пирамиды могут рассматриваться как узлы с первого по n-й *бесконечной* структуры бинарного дерева: пирамида с $n = \infty$ имеет корень в узле 1 и не имеет листьев; скошенная же пирамида с $n = \infty$ имеет бесконечно много листьев 1, 3, 5, ..., но не имеет корня(!).

Листьями скошенной пирамиды являются нечетные числа, а их родителями — нечетные числа, умноженные на 2. Их родители (“деды” листьев), в свою очередь, представляют собой нечетные числа, умноженные на 4, и т. д. Таким образом, линеечная функция ρj говорит о том, как высоко над уровнем листьев находится узел j.

Как легко видеть, родителем узла j в бесконечной скошенной пирамиде является узел

$$(j - k) \mid (k \ll 1), \qquad \text{где } k = j \,\&\, -j; \tag{134}$$

эта формула округляет j до ближайшего нечетного числа, умноженного на $2^{1+\rho j}$. Дочерними узлами являются

$$j - (k \gg 1) \qquad \text{и} \qquad j + (k \gg 1) \tag{135}$$

при четном j. В общем случае потомки узла j образуют отрезок

$$[j - 2^{\rho j} + 1 \,..\, j + 2^{\rho j} - 1], \tag{136}$$

организованный в виде полного бинарного дерева с $2^{1+\rho j}-1$ узлами. (Эти потомки "включительные", т. е. в их число входит и сам узел j.) Предком узла j на высоте h является узел

$$(j \mid (1 \ll h)) \mathbin{\&} -(1 \ll h) = ((j \gg h) \mid 1) \ll h \tag{137}$$

при $h \geq \rho j$. Обратите внимание, что симметричный порядок обхода узлов дает натуральные числа 1, 2, 3, ... в их естественном порядке.

Дов Харел (Dov Harel) указал эти свойства в своей диссертации (U. of California, Irvine, 1980) и заметил, что *ближайший* общий предок двух узлов скошенной пирамиды также легко вычисляется. Действительно, если узел l является ближайшим общим предком узлов i и j, где $i \leq j$, то выполняется замечательное тождество

$$\rho l \;=\; \max\{\rho x \mid i \leq x \leq j\} \;=\; \lambda(j \mathbin{\&} -i), \tag{138}$$

которое связывает функции ρ и λ. (См. упр. 146.) Следовательно, для вычисления l мы можем использовать формулу (137) с $h = \lambda(j \mathbin{\&} -i)$.

Хитроумные расширения этого подхода приводят к асимптотически эффективному алгоритму поиска ближайшего общего предка в *любом* ориентированном лесу, дуги которого растут динамически [D. Harel and R. E. Tarjan, *SICOMP* **13** (1984), 338–355]. Барух Шибер (Baruch Schieber) и Узи Вишкин (Uzi Vishkin) [*SICOMP* **17** (1988), 1253–1262] впоследствии открыли гораздо более простой способ вычисления ближайших общих предков в произвольном (но фиксированном) ориентированном лесу с применением привлекательной и поучительной смеси битовых и алгоритмических методов, которые мы рассмотрим далее.

Вспомним, что ориентированным лесом с m деревьями и n вершинами является ациклический ориентированный граф с $n-m$ дугами. Из каждой вершины выходит не более одной дуги; вершины с нулевой выходящей степенью являются корнями деревьев. Мы говорим, что v является *родителем* u, когда $u \longrightarrow v$, и что v является (включительным) *предком* u при $u \longrightarrow^* v$. Две вершины имеют общий предок тогда и только тогда, когда они принадлежат одному и тому же дереву. Вершина w называется ближайшим общим предком u и v, когда мы имеем

$$u \longrightarrow^* z \text{ и } v \longrightarrow^* z \quad \text{тогда и только тогда, когда} \quad w \longrightarrow^* z. \tag{139}$$

Шибер и Вишкин предварительно обрабатывали лес, отображая его вершины в скошенную пирамиду S размером n путем вычисления трех величин для каждой вершины v:

πv, ранг v в прямом порядке обхода ($1 \leq \pi v \leq n$);
βv, узел скошенной пирамиды S ($1 \leq \beta v \leq n$);
αv, $(1+\lambda n)$-битовый код маршрута ($1 \leq \alpha v < 2^{1+\lambda n}$).

Если $u \longrightarrow v$, мы имеем $\pi u > \pi v$ по определению прямого порядка обхода. Узел βv определяется как ближайший общий предок всех узлов скошенной пирамиды πu, такой, что v является потомком вершины u (всегда подразумевается *включительный* предок). Определим также

$$\alpha v \;=\; \sum \{2^{\rho\beta w} \mid v \longrightarrow^* w\}. \tag{140}$$

Например, ниже приведен ориентированный лес с десятью вершинами и двумя деревьями.

(141)

Каждый узел помечен рангом в прямом порядке обхода, откуда можно вычислить коды β и α.

v =	A	B	C	D	E	F	G	H	I	J
πv =	0001	1000	0010	0100	1001	0011	0101	0111	1010	0110
βv =	0100	1000	0010	0100	1010	0011	0110	0111	1010	0110
αv =	0100	1000	0110	0100	1010	0111	0110	0101	1010	0110

Заметим, что, например, $\beta A = 4 = 0100$, поскольку для потомков A ранги в прямом порядке обхода представляют собой $\{1, 2, 3, 4, 5, 6, 7\}$. А $\alpha H = 0101$, поскольку предки H имеют β-коды $\{\beta H, \beta D, \beta A\} = \{0111, 0100\}$. Можно без труда доказать, что отображение $v \mapsto \beta v$ удовлетворяет следующим ключевым свойствам.

i) Если в лесу $u \longrightarrow v$, то βu является потомком βv в S.

ii) Если несколько вершин имеют одно и то же значение βv, они образуют путь в лесу.

Свойство (ii) выполняется потому, что ровно один дочерний узел u узла v имеет $\beta u = \beta v$ при $\beta v \neq \pi v$.

Теперь представим размещение каждой вершины v леса в узле βv скошенной пирамиды S.

(142)

Если k вершин отображаются в узел j, их можно расположить в виде пути

$$v_0 \longrightarrow v_1 \longrightarrow \cdots \longrightarrow v_{k-1} \longrightarrow v_k, \qquad \text{где } \beta v_0 = \beta v_1 = \cdots = \beta v_{k-1} = j. \tag{143}$$

Эти пути показаны в (142); например, $J \longrightarrow G \longrightarrow D$ представляет собой путь в (141), а '$_{J \to G \to D}$' появляется при узле $0110 = \beta J = \beta G$.

Алгоритм предварительной обработки вычисляет также таблицу τj для всех узлов j скошенной пирамиды S, содержащую указатели на вершины v_k в концах путей (143).

j =	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010
τj =	Λ	A	C	Λ	Λ	D	D	Λ	Λ	B

В упр. 149 показано, что все четыре таблицы, πv, βv, αv и τj, могут быть подготовлены за $O(n)$ шагов. А когда эти таблицы готовы, они содержат достаточно информации для быстрого определения ближайшего общего предка любых двух заданных вершин.

Алгоритм V (*Ближайший общий предок*). Предположим, что нам известны πv, βv, αv и τj для всех n вершин v ориентированного леса для $1 \le j \le n$. Предполагается также наличие фиктивной вершины Λ, у которой $\pi\Lambda = \beta\Lambda = \alpha\Lambda = 0$. Данный алгоритм вычисляет ближайший общий предок z для любых заданных вершин x и y, возвращая $z = \Lambda$, если x и y принадлежат различным деревьям. Мы считаем, что предварительно вычислены значения $\lambda j = \lfloor \lg j \rfloor$ для $1 \le j \le n$ и что $\lambda 0 = \lambda n$.

V1. [Поиск общей высоты.] Если $\beta x \le \beta y$, установить $h \leftarrow \lambda(\beta y \mathbin{\&} -\beta x)$; в противном случае установить $h \leftarrow \lambda(\beta x \mathbin{\&} -\beta y)$. (См. (138).)

V2. [Поиск истинной высоты.] Установить $k \leftarrow \alpha x \mathbin{\&} \alpha y \mathbin{\&} -(1 \ll h)$, затем $h \leftarrow \lambda(k \mathbin{\&} -k)$.

V3. [Поиск βz.] Установить $j \leftarrow ((\beta x \gg h) \mid 1) \ll h$. (Теперь $j = \beta z$, если $z \ne \Lambda$.)

V4. [Поиск $\hat{x}$ и $\hat{y}$.] (Сейчас мы ищем наинизший предок x и y в узле j.) Если $j = \beta x$, установить $\hat{x} \leftarrow x$; в противном случае установить $l \leftarrow \lambda(\alpha x \mathbin{\&} ((1 \ll h) - 1))$ и $\hat{x} \leftarrow \tau(((\beta x \gg l) \mid 1) \ll l)$. Аналогично, если $j = \beta y$, установить $\hat{y} \leftarrow y$; в противном случае установить $l \leftarrow \lambda(\alpha y \mathbin{\&} ((1 \ll h) - 1))$ и $\hat{y} \leftarrow \tau(((\beta y \gg l) \mid 1) \ll l)$.

V5. [Поиск z.] Установить $z \leftarrow \hat{x}$, если $\pi\hat{x} \le \pi\hat{y}$, в противном случае $z \leftarrow \hat{y}$. ▮

Очевидно, что этот хитрый алгоритм использует (137); в упр. 152 поясняется, почему он корректно работает.

Скошенные пирамиды могут также использоваться для реализации интересного типа очереди с приоритетами, которую Ю. Катаяйнен (J. Katajainen) и Ф. Витале (F. Vitale) назвали навигационной стопкой (navigation pile), показанной здесь для $n = 10$.

(144)

Элементы данных располагаются в позициях листьев 1, 3, ..., $2n - 1$ скошенной пирамиды; они могут иметь многобитовую ширину и располагаться в любом порядке. Позиции же ветвления 2, 4, 6, ..., напротив, содержат указатели на наибольший потомок. Изюминкой является то, что эти указатели почти не занимают дополнительного пространства — менее двух битов на элемент данных в среднем, — поскольку для указателей 2, 6, 10, ... требуется только один бит, только два бита нужны для указателей 4, 12, 20, ... и только ρj бит — для указателя j в общем случае. (См. упр. 153.) Таким образом, навигационная стопка требует очень малого количества памяти и обладает хорошим поведением по отношению к производительности кэша типичного компьютера.

***Ячейки на гиперболической плоскости.** Гиперболическая геометрия приводит к поучительной неявной структуре данных существенно иной разновидности. *Гиперболическая плоскость* представляет собой очаровательный пример неевклидовой геометрии, который удобно рассматривать путем проецирования ее точек на

Рис. 13. Два изображения пяти прямых на гиперболической плоскости.

внутренность круга. Ее прямые линии при этом становятся круговыми дугами, пересекающимися с границами круга под прямыми углами. Например, прямые PP', QQ' и RR' на рис. 13 пересекаются в точках O, A, B, и эти точки образуют треугольник. Прямые SQ' и QQ' *параллельны*: они не имеют общих точек, но точки этих прямых становятся все ближе и ближе одна к другой. Прямая QT также параллельна прямой QQ'.

Мы получаем различные изображения в зависимости от того, какая точка оказывается центральной. Например, второе изображение на рис. 13 помещает O прямо в центр. Обратите внимание, что прямые, проходящие через центр, остаются прямыми и после проекции; такие диаметральные хорды представляют собой частный случай "круговых дуг" с бесконечным радиусом.

Большинство аксиом Евклида для планиметрии остаются верными и на гиперболической плоскости. Например, через любые две различные точки можно провести одну и только одну прямую; а если точка A лежит на прямой PP', то существует ровно одна прямая QQ', такая, что угол PAQ имеет заданное значение $0 < \theta < 180°$. Но знаменитый пятый постулат Евклида *не* выполняется: если точка C *не* лежит на прямой QQ', всегда имеется ровно *две* прямые, проходящие через C и параллельные QQ'. Кроме того, имеется много пар прямых, наподобие RR' и SQ' на рис. 13, которые совершенно не пересекаются, являясь *расходящимися* (*ultraparallel*) в том смысле, что их точки никогда не становятся сколь угодно близкими. [Эти свойства гиперболической плоскости были открыты Д. Саккери (G. Saccheri) в начале 1700-х годов и строго доказаны Н. И. Лобачевским, Я. Бойяи (J. Bolyai) и К. Ф. Гауссом (C. F. Gauss) столетием позже.]

Говоря количественно, когда точки проецируются на единичный круг $|z| < 1$, дуга, пересекающая его границу в точках $e^{i\theta}$ и $e^{-i\theta}$, имеет центр $\sec\theta$ и радиус $\tan\theta$. Реальное расстояние между двумя точками, проекциями которых являются z и z', равно $d(z, z') = \ln(|1 - \bar{z}z'| + |z - z'|) - \ln(|1 - \bar{z}z'| - |z - z'|)$. Таким образом, объекты при удалении от центра и приближении к границе круга выглядят сильно уменьшенными.

Сумма углов гиперболического треугольника всегда *меньше* 180°. Например, углы в точках O, A и B на рис. 13 равны соответственно 90°, 45° и 36°. Десять таких 36°-45°-90°-треугольников можно разместить вместе, создав таким образом правильный пятиугольник, все углы которого — прямые (90°). А четыре таких пятиугольника легко соединяются углами в одной точке, позволяя замостить всю гиперболическую плоскость правильными пятиугольниками (см. рис. 14). Стороны таких пятиугольников образуют интересное семейство прямых, каждые две из которых являются либо расходящимися, либо перпендикулярными; так что мы имеем сетчатую структуру, аналогичную единичным квадратам на обычной плоскости.

Рис. 14. Пентагрид, в котором гиперболическая плоскость замощена идентичными пятиугольниками.

Круговая регулярная мозаика, окруженная со всех сторон бесконечно малыми фигурами, действительно чудесна.

— М. К. ЭШЕР (M. C. ESCHER), письмо Д. Эшеру (G. Escher) (9 ноября 1958 года)

Назовем ее *пентагридом** (*pentagrid*), поскольку каждая ячейка теперь содержит пять соседей вместо четырех.

Имеется красивый способ навигации по пентагриду с использованием чисел Фибоначчи, основанный на идеях Мориса Маргенштерна (Maurice Margenstern) [см. F. Herrmann and M. Margenstern, *Theoretical Comp. Sci.* **296** (2003), 345–351]. Однако вместо обычной последовательности Фибоначчи $\langle F_n \rangle$ мы будем использовать числа Фибоначчи с *отрицательными индексами*, или "негафибоначчиевы числа" $\langle F_{-n} \rangle$, а именно

$$F_{-1} = 1,\ F_{-2} = -1,\ F_{-3} = 2,\ F_{-4} = -3,\ \ldots,\ F_{-n} = (-1)^{n-1} F_n. \tag{145}$$

В упр. 1.2.8–34 вводится система счисления Фибоначчи, в которой каждое неотрицательное число x может быть единственным образом записано в виде

$$x = F_{k_1} + F_{k_2} + \cdots + F_{k_r}, \qquad \text{где } k_1 \gg k_2 \gg \cdots \gg k_r \gg 0; \tag{146}$$

здесь '$j \gg k$' означает '$j \geq k + 2$'. Но имеется также и *система счисления, основанная на числах Фибоначчи с отрицательными индексами*, которая лучше подходит для наших целей: *каждое целое число x независимо от того, положительное оно, отрицательное или равное нулю, может быть записано единственным*

* Следует заметить, что этот термин в русскоязычной технической литературе имеет и иное значение: синоним *гептода*, названия семиэлектродной электронной лампы с пятью сетками. — *Примеч. пер.*

способом в виде

$$x = F_{k_1} + F_{k_2} + \cdots + F_{k_r}, \qquad \text{где } k_1 \prec\!\!\prec k_2 \prec\!\!\prec \cdots \prec\!\!\prec k_r \prec\!\!\prec 1. \tag{147}$$

Например, $4 = 5 - 1 = F_{-5} + F_{-2}$ и $-2 = -3 + 1 = F_{-4} + F_{-1}$. Это представление можно удобно выразить в виде бинарного кода $\alpha = \ldots a_3 a_2 a_1$, означающего $N(\alpha) = \sum_k a_k F_{-k}$, причем в строке не имеется двух последовательных единиц. Например, вот коды такого представления для всех целых чисел между -14 и $+15$.

$-14 = 10010100$	$-8 = 100000$	$-2 = 1001$	$4 = 10010$	$10 = 1001000$
$-13 = 10010101$	$-7 = 100001$	$-1 = 10$	$5 = 10000$	$11 = 1001001$
$-12 = 101010$	$-6 = 100100$	$0 = 0$	$6 = 10001$	$12 = 1000010$
$-11 = 101000$	$-5 = 100101$	$1 = 1$	$7 = 10100$	$13 = 1000000$
$-10 = 101001$	$-4 = 1010$	$2 = 100$	$8 = 10101$	$14 = 1000001$
$-9 = 100010$	$-3 = 1000$	$3 = 101$	$9 = 1001010$	$15 = 1000100$

Как и в “негадесятичной системе счисления” (см. 4.1–(6) и (7)), можно говорить о том, является ли x отрицательным или нет по тому, четно или нечетно количество цифр в его представлении.

Предшественник $\alpha-$ и преемник $\alpha+$ любого негафибоначчиева бинарного кода α может быть вычислен рекурсивно с использованием правил

$$(\alpha 01)- = \alpha 00, \quad (\alpha 000)- = \alpha 010, \quad (\alpha 100)- = \alpha 001, \quad (\alpha 10)- = (\alpha-)01,$$
$$(\alpha 10)+ = \alpha 00, \quad (\alpha 00)+ = \alpha 01, \quad (\alpha 1)+ = (\alpha-)0. \tag{148}$$

(См. упр. 157.) Однако десять элегантных 2-адических шагов выполняют это вычисление непосредственно:

$$\begin{aligned} &y \leftarrow x \oplus \bar{\mu}_0,\ z \leftarrow y \oplus (y \pm 1),\ \text{где } x = (\alpha)_2; \\ &z \leftarrow z \mid (x \,\&\, (z \ll 1)); \\ &w \leftarrow x \oplus z \oplus ((z+1) \gg 2);\ \text{тогда } w = (\alpha\pm)_2. \end{aligned} \tag{149}$$

Мы просто используем в первой строке $y - 1$ для получения предшественника и $y + 1$ — для преемника.

А теперь перейдем к тому, зачем все это делалось. Каждой ячейке пентагрида можно назначить негафибоначчиев код таким образом, чтобы можно было очень легко вычислить коды ее пяти соседей. Обозначим соседей буквами n, s, e, w и o, означающими “north” (север), “south” (юг), “east” (восток), “west” (запад) и “other” (иное). Если α представляет собой код, назначенный данной ячейке, определим

$$\alpha_n = \alpha \gg 2, \quad \alpha_s = \alpha \ll 2, \quad \alpha_e = \alpha_s+, \quad \alpha_w = \alpha_s-; \tag{150}$$

таким образом, $\alpha_{sn} = \alpha$, а также $\alpha_{en} = (\alpha 01)_n = \alpha$. “Иное” направление немного сложнее:

$$\alpha_o = \begin{cases} \alpha_n+, & \text{если } \alpha \,\&\, 1 = 1; \\ \alpha_w-, & \text{если } \alpha \,\&\, 1 = 0. \end{cases} \tag{151}$$

Например, $1000_o = 101001$ и $101001_o = 1000$. Этот таинственный нарушитель лежит между севером и востоком, когда α заканчивается на 1, и между севером и западом, когда α заканчивается на 0.

Если мы выберем любую ячейку и пометим ее кодом 0 и если мы также выберем ориентацию таким образом, чтобы ее соседями по часовой стрелке были n, e, s, w

и o, то правила (150) и (151) будут согласованным образом назначать метки каждой ячейке пентагрида. (См. упр. 160.) Например, окрестности ячейки с меткой 1000 будут выглядеть следующим образом.

(152)

Однако метки с кодами не идентифицируют ячейки единственным образом, поскольку бесконечно много ячеек могут получить одну и ту же метку. (Действительно, мы очевидным образом получаем $0_n = 0_s = 0$ и $1_w = 1_o = 1$.) Для получения уникального идентификатора добавим вторую координату, так что полное имя каждой ячейки имеет вид (α, y), где y представляет собой целое число. Когда y является константой, а α проходит все негафибоначчиевы коды, ячейки (α, y) образуют более или менее крюковидную полосу, края которой поворачиваются на 90° по соседству с ячейкой $(0, y)$. В общем случае пятью соседями (α, y) являются $(\alpha, y)_n = (\alpha_n, y + \delta_n(\alpha))$, $(\alpha, y)_s = (\alpha_s, y + \delta_s(\alpha))$, $(\alpha, y)_e = (\alpha_e, y + \delta_e(\alpha))$, $(\alpha, y)_w = (\alpha_w, y + \delta_w(\alpha))$ и $(\alpha, y)_o = (\alpha_o, y + \delta_o(\alpha))$, где

$$\delta_n(\alpha) = [\alpha = 0], \quad \delta_s(\alpha) = -[\alpha = 0], \quad \delta_e(\alpha) = 0, \quad \delta_w(\alpha) = -[\alpha = 1];$$

$$\delta_o(\alpha) = \begin{cases} \operatorname{sign}(\alpha_o - \alpha_n)[\alpha_o \mathbin{\&} \alpha_n = 0], & \text{если } \alpha \mathbin{\&} 1 = 1; \\ \operatorname{sign}(\alpha_o - \alpha_w)[\alpha_o \mathbin{\&} \alpha_w = 0], & \text{если } \alpha \mathbin{\&} 1 = 0. \end{cases} \tag{153}$$

(См. рисунок ниже.) Теперь битовые операции позволяют нам легко обойти всю гиперболическую плоскость. С другой стороны, мы можем также игнорировать при движении значения координаты y, тем самым обвивая “гиперболический цилиндр” из пятиугольников; α-координаты определяют интересный мультиграф на множестве всех негафибоначчиевых кодов, в котором каждая вершина имеет степень 5.

(154)

Растровая графика. Очень интересно писать программы, работающие с различными рисунками и формами, поскольку при этом требуется включение одновременно и левого, и правого полушарий мозга. При работе с изображениями результаты могут оказаться увлекательными даже при наличии ошибок в коде.

Книга, которую вы сейчас читаете, была набрана с помощью программного обеспечения, которое рассматривает каждую страницу как гигантскую матрицу из нулей и единиц, именуемую растром или битовой картой (bitmap), содержащей миллионы квадратных элементов рисунка, так называемых пикселей (pixel). Растры передаются в печатные машины, и маленькие точки типографской краски размещаются там, где в матрице находится 1. Физические свойства типографской краски и бумаги приводят к тому, что маленькие кластеры точек выглядят как гладкие кривые; но "квадратность" основы каждого пикселя станет очевидной, если десятикратно увеличить изображения, как в букве 'A', показанной на рис. 15, (а).

С помощью битовых операций можно достичь специальных эффектов наподобие "кастеризации", при которой окруженные со всех сторон черные точки исчезают (рис. 15, (б)).

Рис. 15. Буква 'A' до и после кастеризации.

Эта операция, введенная Р. Э. Киршем (R. A. Kirsch), Л. Каном (L. Cahn), Ч. Рэем (C. Ray) и Ж. Х. Урбаном (G. H. Urban) [*Proc. Eastern Joint Computer Conf.* **12** (1957), 221–229], может быть выражена как

$$\mathrm{custer}(X) \;=\; X \,\&\, {\sim}\big((X \mathbin{\vee\!\!\!\vee} 1) \,\&\, (X \gg 1) \,\&\, (X \ll 1) \,\&\, (X \mathbin{\wedge\!\!\!\wedge} 1)\big), \tag{155}$$

где '$X \vee\!\!\!\vee 1$' и '$X \wedge\!\!\!\wedge 1$' обозначают соответственно результат сдвига растра X вниз или вверх на одну строку. Обозначим однопиксельные сдвиги растра X как

$$X_{\rm N} = X \vee\!\!\!\vee 1, \quad X_{\rm W} = X \gg 1, \quad X_{\rm E} = X \ll 1, \quad X_{\rm S} = X \wedge\!\!\!\wedge 1. \tag{156}$$

Тогда, например, символическое выражение '$X_{\rm N}$ & $(X_{\rm S} \mid \overline{X}_{\rm E})$' вычисляет 1 в тех позициях пикселей, в которых северный сосед черный и имеется или черный сосед к югу, или белый к востоку. При использовании этих обозначений (155) принимает вид

$$\text{custer}(X) \;=\; X \,\&\, {\sim}(X_{\rm N} \,\&\, X_{\rm W} \,\&\, X_{\rm E} \,\&\, X_{\rm S}), \tag{157}$$

что можно также записать как $X \,\&\, (\overline{X}_{\rm N} \mid \overline{X}_{\rm W} \mid \overline{X}_{\rm E} \mid \overline{X}_{\rm S})$.

У каждого пикселя есть четыре "соседа по ходу ладьи", с которыми он имеет общие границы сверху, снизу, слева и справа. У него также восемь "соседей по ходу короля", с которыми у него имеется как минимум одна общая угловая точка. Например, "королевские" соседи, лежащие к северо-востоку от всех пикселей растра X, могут быть обозначены как $X_{\rm NE}$, что в пиксельной алгебре эквивалентно обозначению $(X_{\rm N})_{\rm E}$. Заметим также, что $X_{\rm NE} = (X_{\rm E})_{\rm N}$.

Клеточный автомат размером 3×3 представляет собой массив пикселей, динамически изменяющихся путем последовательности одновременно выполняющихся локальных преобразований: состояние каждого пикселя в момент $t+1$ полностью зависит от его состояния в момент t и состояния в этот же момент его соседей по ходу короля. Таким образом, автомат определяет последовательность растров $X^{(0)}$, $X^{(1)}$, $X^{(2)}$, ..., которая начинается с любого заданного начального состояния $X^{(0)}$, где

$$X^{(t+1)} \;=\; f(X^{(t)}_{\rm NW}, X^{(t)}_{\rm N}, X^{(t)}_{\rm NE}, X^{(t)}_{\rm W}, X^{(t)}, X^{(t)}_{\rm E}, X^{(t)}_{\rm SW}, X^{(t)}_{\rm S}, X^{(t)}_{\rm SE}) \tag{158}$$

а f — любая битовая булева функция от девяти переменных. Таким образом часто получаются очаровательные узоры. Например, после того как Мартин Гарднер (Martin Gardner) рассказал в 1970 году всему миру об игре Джона Конвея (John Conway) "Жизнь" ("Life"), вероятно, в последующие несколько лет на изучение ее следствий было затрачено больше компьютерного времени, чем на любую другую вычислительную задачу, хотя в те годы люди, играющие на компьютере, встречались очень редко!* (См. упр. 167.)

Существует 2^{512} булевых функций от девяти переменных, так что имеется 2^{512} различных клеточных автоматов 3×3. Многие из них тривиальны, но большинство, вероятно, имеет настолько сложное поведение, что его понимание находится за пределами человеческих возможностей. К счастью, имеется также множество полезных на практике случаев, интерес к которым с экономической точки зрения оправдать проще, чем интерес к обычным играм.

Например, в алгоритмах для распознавания букв, отпечатков пальцев или иных подобных рисунков часто используется процесс "утончения", который удаляет лишние черные пиксели и приводит каждый компонент изображения к некоторому лежащему в его основе скелету, который относительно просто анализировать. Ряд авторов предлагали для решения этой задачи клеточные автоматы, начиная с Д. Рутовица (D. Rutovitz) [*J. Royal Stat. Society* **A129** (1966), 512–513], который пред-

* Если вам захочется немного поразвлечься, найдите одну из наилучших реализаций по адресу `http://golly.sourceforge.net/`. — *Примеч. пер.*

Рис. 16. Пример результатов работы автомата 3×3 Гуо и Холла для утончения компонентов растрового изображения. ("Пустые" пиксели изначально были черными.)

ложил схему 4×4. Но параллельные алгоритмы, как известно, — дело тонкое, и после того как различные методы были опубликованы, стали обнаруживаться их недостатки. Например, в компоненте наподобие ▩ должны быть удалены один, два или три черных пикселя, но симметричная схема ошибочно удаляет все четыре.

Удовлетворительное решение задачи утончения в конце концов было найдено З. Гуо (Z. Guo) и Р. У. Холлом (R. W. Hall) [*CACM* **32** (1989), 359–373, 759] с использованием клеточного автомата 3×3, который применяет чередующиеся правила на четных и нечетных шагах. Рассмотрим функцию

$$f(x_{\rm NW}, x_{\rm N}, x_{\rm NE}, x_{\rm W}, x, x_{\rm E}, x_{\rm SW}, x_{\rm S}, x_{\rm SE}) = x \wedge \neg g(x_{\rm NW}, \ldots, x_{\rm W}, x_{\rm E}, \ldots, x_{\rm SE}), \tag{159}$$

где $g = 1$ только в следующих 37 конфигурациях окружения черного пикселя.

Тогда на четных шагах мы используем (158), но с заменой $f(x_{\rm NW}, x_{\rm N}, x_{\rm NE}, x_{\rm W}, x, x_{\rm E}, x_{\rm SW}, x_{\rm S}, x_{\rm SE})$ на его 180°-ный поворот $f(x_{\rm SE}, x_{\rm S}, x_{\rm SW}, x_{\rm E}, x, x_{\rm W}, x_{\rm NE}, x_{\rm N}, x_{\rm NW})$. Процесс завершается, когда два последовательных цикла не вносят изменений.

Гуо и Холл доказали, что при использовании указанных правил клеточный автомат размером 3×3 сохраняет структуру связности изображения в строгом смысле, рассматриваемом ниже. Кроме того, их алгоритм очевидным образом оставляет изображение нетронутым, если оно уже настолько тонкое, что не содержит трех пикселей, являющихся "королевскими" соседями друг друга. С другой стороны, он обычно успешно "удаляет мясо с костей" каждого черного компонента, как показано на рис. 16. Немного более разреженное утончение получается при добавлении четырех дополнительных конфигураций,

, (160)

к 37 перечисленным выше. В любом случае функция g может быть вычислена с помощью булевой цепочки длиной 25. (См. упр. 170–172.)

В общем случае черные пиксели изображения могут быть сгруппированы в сегменты или компоненты, *связанные ходом короля*, в том смысле, что любой черный пиксель может быть достигнут из любого другого пикселя своего компонента с помощью последовательности ходов королем по черным пикселям. Белые пиксели также образуют компоненты, *связанные ходом ладьи*: любые две белые ячейки компонента взаимно достижимы ходами ладьи, не проходящими по черным ячейкам. Лучше всего использовать для белых и черных ячеек разные виды связности, чтобы обеспечить сохранность топологических понятий "внутри" и "вне", знакомые

Рис. 17. Сжатие Чеширского кота путем

из континуальной геометрии [см. A. Rosenfeld, *JACM* **17** (1970), 146–160]. Если представить, что угловые точки растра черные, то бесконечно тонкая черная кривая может пройти между пикселями в углу, в то время как для белой кривой это невозможно. (Мы могли бы также представить белые угловые точки, что привело бы к "ладейной" связности для черного цвета и "королевской" — для белого.)

Занимательный алгоритм сжатия рисунка при сохранении его связности, за исключением исчезновения изолированных белых и черных точек, был представлен С. Левиальди (S. Levialdi) в *CACM* **15** (1972), 7–10; эквивалентный алгоритм, но с обращением белого и черного, был также разработан в диссертации Т. Бейера (T. Beyer) (M.I.T., 1969). Идея заключается в использовании клеточного автомата с простой функцией перехода

$$f(x_{\rm NW}, x_{\rm N}, x_{\rm NE}, x_{\rm W}, x, x_{\rm E}, x_{\rm SW}, x_{\rm S}, x_{\rm SE}) = (x \wedge (x_{\rm W} \vee x_{\rm SW} \vee x_{\rm S})) \vee (x_{\rm W} \wedge x_{\rm S}) \qquad (161)$$

на каждом шаге. Эта формула в действительности представляет собой правило 2×2, но нам все равно требуется окно размером 3×3, если мы хотим отслеживать случаи исчезновения однопиксельных компонентов.

Например, на рис. 17, (а) размером 25×30, изображающем Чеширского кота, имеется семь "королевских" черных компонентов: контур головы, две ушные раковины, два глаза, нос и улыбка. Результат после однократного применения (161) показан на рис. 17, (б): пока что остались все семь компонентов, но в одном ухе имеется одна изолированная точка, а второе ухо станет таким на следующем шаге. На рис. 17, (в) осталось только пять компонентов. После шести шагов кот теряет нос, и даже улыбка исчезает после 14 шагов. Как это ни прискорбно, на шаге 46 исчезает последний бит кота.

Не более $M + N - 1$ переходов сотрут любое изображение размером $M \times N$, поскольку нижняя граница видимости, представляющая собой диагональную прямую, идущую с северо-запада на юго-восток, с каждым шагом настойчиво поднимается вверх. В упр. 176 и 177 доказывается, что разные компоненты никогда не сливаются вместе и не влияют друг на друга.

Конечно, такой клеточный метод с кубическим временем работы — не самый быстрый способ подсчета или идентификации компонентов изображения. В действительности мы можем выполнять эту работу в "оперативном режиме", при просмотре большого изображения по одной строке за раз, не беспокоясь о сохранении всех ранее просмотренных строк в памяти, если не хотим обращаться к ним снова.

Время – 5 Время – 10 Время – 20

(г) (д) (е)

многократного применения преобразования Левиальди.

В процессе анализа компонентов мы можем также записывать отношения между ними. Будем считать, что имеется только конечное количество черных пикселей. Тогда в наличии имеется бесконечный компонент из белых пикселей, называющийся *фоном*. Черные компоненты, смежные с фоном, представляют собой главные *объекты* изображения. А эти объекты могут, в свою очередь, иметь *дыры*, которые могут служить фоном для объектов другого уровня, и т. д. Таким образом, связные компоненты любого конечного изображения образуют иерархию — ориентированное дерево, корнем которого является фон. Черные компоненты находятся на нечетных уровнях этого дерева, а белые — на четных, с чередующейся "королевской" и "ладейной" связностями. Каждый компонент, за исключением фона, *окружен* родительским компонентом. Компоненты, у которых нет потомков, называются *просто связными*.

Например, вот компоненты Чеширского кота, помеченные цифрами для белых пикселей и буквами для черных, и соответствующее ориентированное дерево.

```
0000000000000000000000000A000000
0000AAA0000000000000000000AA000000
0000A1AA000000000000000A11A000000
0000A111AA000AAAAAAAA00A11A000000
0000A1B11AAAA111111AAA111A00000
0000A1B11A111111111111A1C1A00000
0000A1B11A111111111111A1C1A00000
0000A111111111111111111A111A00000
0000A111111111111111111A11A00000
00000A11111111111111111111A00000
000000A11DDD1111111EEE11111A0000
000000A1D222D11111E333E11111A0000
AA00A11DDD2D11111EEE3E11111A000
00AAA11DDDDD111111EEE111111A000
0000AA111111F1F111111111111A000
AAAA111111111F1F111111111AAAA
0000AAAA11111F11F111111AAAA1A000
0000A11111111FF11111111111111A000
0000A111GG1111111111111AAAAAAAAA
000000A111GGGGG11111111111111A0000
0000000AA111GGGGGGGGGGG111A00000
00000000AA11GGGGGGGG1111AA000000
000000000A111111111AAA00000000
0000000000000AAAAAAAAAAA0000000000
```

(162)

В процессе сжатия, показанного на рис. 17, компоненты исчезают в порядке C, {B, 2, 3} (все в момент времени 3), F, E, D, G, 1, A.

Предположим, что мы хотим проанализировать компоненты такого изображения по одной строке за раз. После того как будут просмотрены четыре строки,

результат окажется следующим:

```
0000000000000000000000A000000
00000BBB00000000000000AA000000
0000B1BB00000000000000A22A000000
000B111BB000CCCCCCC00A22A00000
```

, (163)

и мы будем готовы сканировать пятую строку. Сравнение четвертой и пятой строк покажет, что B и C должны сливаться с A, но должны быть начаты новые компоненты B и 3. В упр. 179 содержится детальное описание поучительного алгоритма, который корректно обновляет текущее дерево при просмотре новой строки. "На лету" может вычисляться и дополнительная информация; например, мы можем определить площадь каждого компонента, положение его первого и последнего пикселей, наименьший охватывающий прямоугольник и/или его центр тяжести.

***Заполнение.** Давайте завершим наш быстрый тур по растровой графике обсуждением того, как заполнять области, ограниченные прямыми линиями и/или простыми кривыми. Особенно эффективный алгоритм имеется в том случае, когда кривые построены из "конических сечений" — окружностей, эллипсов, парабол или гипербол, как в классической геометрии.

Отдавая дань геометрическим традициям, в данном обсуждении мы будем использовать декартовы координаты (x, y) вместо того, чтобы говорить о строках и столбцах пикселей: увеличение x будет означать перемещение вправо, а увеличение y — перемещение вверх. Что более важно, мы сосредоточимся на *границах* между квадратными пикселями, а не на самих пикселях. Граница проходит между целыми точками плоскости (x, y) и (x', y'), когда $|x - x'| + |y - y'| = 1$. Каждый пиксель имеет четыре границы: (x, y) — $(x{-}1, y)$ — $(x{-}1, y{-}1)$ — $(x, y{-}1)$ — (x, y). Опыт показывает, что алгоритмы для заполнения контуров становятся проще и быстрее, если мы концентрируемся на границах между белым и черным цветом, а не на черных пикселях кастеризованной линии раздела. (См., например, рассмотрение вопроса в B. D. Ackland и N. Weste, *IEEE Trans.* **C-30** (1981), 41–48.)

Рассмотрим непрерывную кривую $z(t) = \bigl(x(t), y(t)\bigr)$, вычерчиваемую при пробегании t значений от 0 до 1. Мы считаем, что кривая не должна иметь самопересечений при $0 \le t < 1$ и что $z(0) = z(1)$. Знаменитая теорема Джордан о кривых [C. Jordan, *Cours d'analyse* **3** (1887), 587–594; O. Veblen, *Trans. Amer. Math. Soc.* **6** (1905), 83–98] указывает, что каждая такая кривая делит плоскость на две области, внутреннюю и внешнюю. Мы можем "оцифровать" $z(t)$, заставляя проходить ее по границам между пикселями; тогда мы получим приближение, в котором внутренние пиксели являются черными, а внешние — белыми. Этот процесс оцифровки, по сути, заменяет исходную кривую последовательностью целочисленных точек

$$\operatorname{round}(z(t)) \;=\; \bigl(\lfloor x(t) + \tfrac{1}{2}\rfloor, \lfloor y(t) + \tfrac{1}{2}\rfloor\bigr) \qquad \text{для } 0 \le t \le 1. \tag{164}$$

При необходимости кривая может быть слегка смещена так, чтобы $z(t)$ никогда не проходила по центру пикселя. Тогда оцифрованная кривая с ростом t дискретно проходит по границам пикселей; а пиксель считается лежащим внутри оцифрованной кривой тогда и только тогда, когда его центр лежит внутри исходной непрерывной кривой $\{z(t) \mid 0 \le t \le 1\}$.

Например, уравнения $x(t) = 20\cos 2\pi t$ и $y(t) = 10\sin 2\pi t$ определяют эллипс. Его оцифровка, $\operatorname{round}(z(t))$, начинается в точке $(20, 0)$ при $t = 0$, затем переходит

к $(20,1)$ при $t \approx .008$, а $10\sin 2\pi t = 0.5$. Затем, когда t растет и принимает значения $0.024, 0.036, 0.040, 0.057, 0.062, \ldots, 0.976, 0.992$, оцифровка дает точки $(20,2)$, $(19,2)$, $(19,3)$, $(19,4)$, $(18,4)$, $\ldots$, $(20,-1)$, $(20,0)$.

(165)

Горизонтальные границы такого контура удобно представить в виде битовых векторов $H(y)$ для каждого y; например, $H(10) = \ldots 0000001111111111111000000\ldots$ и $H(9) = \ldots 0111110000000000000111110\ldots$ в (165). Если эллипс заполнен черным цветом для получения растра B, векторы H отмечают переходы между черным и белым цветами, так что мы имеем соотношение

$$H \;=\; B \oplus (B \mathbin{\hat{\gg}} 1). \tag{166}$$

И наоборот, легко получить B по заданным векторам H:

$$\begin{aligned} B(y) &= H(y_{\max}) \oplus H(y_{\max-1}) \oplus \cdots \oplus H(y+1) \\ &= H(y_{\min}) \oplus H(y_{\min+1}) \oplus \cdots \oplus H(y). \end{aligned} \tag{167}$$

Обратите внимание, что $H(y_{\min}) \oplus H(y_{\min+1}) \oplus \cdots \oplus H(y_{\max})$ представляет собой нулевой вектор, поскольку каждое растровое изображение является белым как сверху, так и снизу. Заметим также, что аналогичные векторы для *вертикальных* границ $V(x)$ являются излишними: они удовлетворяют формулам $V = B \oplus (B \ll 1)$ и $B = V^{\oplus}$ (см. упр. 36), но нам не требуется беспокоиться об их отслеживании.

Работать с коническими сечениями проще, чем с большинством других кривых, поскольку при этом можно легко избавиться от параметра t. Например, эллипс, приводящий к (165), может быть задан уравнением $(x/20)^2 + (y/10)^2 = 1$, а не синусами и косинусами. Следовательно, пиксель (x,y) должен быть черным тогда и только тогда, когда его центральная точка $(x-\frac{1}{2}, y-\frac{1}{2})$ лежит внутри эллипса, тогда и только тогда, когда $(x-\frac{1}{2})^2/400 + (y-\frac{1}{2})^2/100 - 1 < 0$.

В общем случае каждое коническое сечение представляет собой множество точек, для которых $F(x,y) = 0$, где F — соответствующая квадратичная форма. Следовательно, имеется квадратичная форма

$$Q(x,y) \;=\; F(x-\tfrac{1}{2}, y-\tfrac{1}{2}) \;=\; ax^2 + bxy + cy^2 + dx + ey + f, \tag{168}$$

которая отрицательна в целочисленной точке (x,y) тогда и только тогда, когда пиксель (x,y) лежит с заданной стороны оцифрованной кривой.

Для практических целей можно считать, что коэффициенты $(a,b,\ldots,f)$ квадратичной формы Q представляют собой не слишком большие целые числа. Тогда нам повезло, поскольку вычислить точное значение $Q(x,y)$ легко. Фактически, как указал М. Л. В. Питтвей (M. L. V. Pitteway) [*Comp. J.* **10** (1967), 282–289], имеется красивый "трехрегистровый алгоритм", с помощью которого можно быстро отслеживать граничные точки. Пусть x и y — целые числа, и предположим, что

в трех регистрах (Q, Q_x, Q_y) у нас имеются значения $Q(x,y)$, $Q_x(x,y)$ и $Q_y(x,y)$, где

$$Q_x(x,y) = 2ax + by + d \qquad \text{и} \qquad Q_y(x,y) = bx + 2cy + e \tag{169}$$

представляют собой $\frac{\partial}{\partial x}Q$ и $\frac{\partial}{\partial y}Q$. Тогда мы можем перейти в любую соседнюю целочисленную точку, поскольку

$$\begin{aligned}
&Q(x\pm1,y) = Q(x,y)\pm Q_x(x,y)+a, && Q(x,y\pm1) = Q(x,y)\pm Q_y(x,y)+c,\\
&Q_x(x\pm1,y) = Q_x(x,y)\pm 2a, && Q_x(x,y\pm1) = Q_x(x,y)\pm b,\\
&Q_y(x\pm1,y) = Q_y(x,y)\pm b; && Q_y(x,y\pm1) = Q_y(x,y)\pm 2c.
\end{aligned} \tag{170}$$

Кроме того, мы можем разделить контур на отдельные части, в каждой из которых и $x(t)$, и $y(t)$ монотонны. Например, когда эллипс (165) идет от $(20,0)$ к $(0,10)$, значение x уменьшается, в то время как значение y увеличивается; таким образом, нам требуется переход только от (x,y) к $(x{-}1,y)$ или к $(x,y{+}1)$. Если в регистрах (Q,R,S) хранятся соответственно $(Q, Q_x - a, Q_y + c)$, то переход к $(x{-}1,y)$ просто устанавливает $Q \leftarrow Q - R$, $R \leftarrow R - 2a$ и $S \leftarrow S - b$; переход к $(x,y{+}1)$ выполняется так же быстро. При аккуратной реализации эта идея приводит к очень быстрому способу корректной оцифровки почти любой конической кривой.

Например, квадратичная форма $Q(x,y)$ для эллипса (165) имеет вид $4x^2+16y^2-(4x+16y+1595)$ при приведении коэффициентов к целочисленным значениям. Мы имеем $Q(20,0) = F(19.5,-0.5) = -75$ и $Q(21,0) = +85$; следовательно, пиксель $(20,0)$, центр которого находится в $(19.5,-0.5)$, располагается внутри эллипса, в то время как пиксель $(21,0)$ — нет. Давайте посмотрим на рисунок в большем масштабе.

(171)

Контур можно получить без изучения Q в очень большом количестве точек. Действительно, нам не нужно исследовать $Q(21,0)$, поскольку мы знаем, что все границы от $(20,0)$ до $(0,10)$ должны вести только либо вверх, либо влево. Сначала мы проверяем $Q(20,1)$ и обнаруживаем, что значение в этой точке отрицательное (-75); так что мы движемся вверх. Отрицательное значение (-43) получается и в точке $Q(20,2)$, так что мы продолжаем движение вверх. Затем мы проверяем $Q(20,3)$ и обнаруживаем, что значение в этой точке положительно (21); так что мы перемещаемся влево, и т. д. При корректном применении трехрегистрового метода фактические проверяемые значения Q представляют собой -75, -43, 21, -131, -35, 93, -51, ...

Алгоритм Т (*Трехрегистровый алгоритм для конических сечений*). Для двух заданных целочисленных точек (x, y) и (x', y') и целочисленной квадратичной формы Q, как в (168), этот алгоритм указывает, как оцифровывать часть конического сечения, определяемого уравнением $F(x, y) = 0$, где $F(x, y) = Q(x + \frac{1}{2}, y + \frac{1}{2})$. Он создает $|x' - x|$ горизонтальных и $|y' - y|$ вертикальных границ, которые образуют путь от (x, y) до (x', y'). Мы считаем, что

i) существуют точки (ξ, η) и (ξ', η') с действительными значениями, такие, что $F(\xi, \eta) = F(\xi', \eta') = 0$;

ii) кривая проходит от (ξ, η) к (ξ', η') монотонно по обеим координатам;

iii) $x = \lfloor \xi + \frac{1}{2} \rfloor$, $y = \lfloor \eta + \frac{1}{2} \rfloor$, $x' = \lfloor \xi' + \frac{1}{2} \rfloor$ и $y' = \lfloor \eta' + \frac{1}{2} \rfloor$;

iv) при движении по кривой от (ξ, η) до (ξ', η') мы видим значения $F < 0$ слева;

v) никакая граница целочисленной решетки не содержит двух корней Q (см. упр. 183).

T1. [Инициализация.] Если $x = x'$, перейти к шагу T11; если $y = y'$, перейти к шагу T10. Если $x < x'$ и $y < y'$, установить $Q \leftarrow Q(x+1, y+1)$, $R \leftarrow Q_x(x+1, y+1) + a$, $S \leftarrow Q_y(x+1, y+1) + c$ и перейти к шагу T2. Если $x < x'$ и $y > y'$, установить $Q \leftarrow Q(x+1, y)$, $R \leftarrow Q_x(x+1, y) + a$, $S \leftarrow Q_y(x+1, y) - c$ и перейти к шагу T3. Если $x > x'$ и $y < y'$, установить $Q \leftarrow Q(x, y+1)$, $R \leftarrow Q_x(x, y+1) - a$, $S \leftarrow Q_y(x, y+1) + c$ и перейти к шагу T4. Если $x > x'$ и $y > y'$, установить $Q \leftarrow Q(x, y)$, $R \leftarrow Q_x(x, y) - a$, $S \leftarrow Q_y(x, y) - c$ и перейти к шагу T5.

T2. [Вправо или вверх.] Если $Q < 0$, выполнить шаг T9; в противном случае выполнить шаг T6. Повторять до прерывания.

T3. [Вниз или вправо.] Если $Q < 0$, выполнить шаг T7; в противном случае выполнить шаг T9. Повторять до прерывания.

T4. [Вверх или влево.] Если $Q < 0$, выполнить шаг T6; в противном случае выполнить шаг T8. Повторять до прерывания.

T5. [Влево или вниз.] Если $Q < 0$, выполнить шаг T8; в противном случае выполнить шаг T7. Повторять до прерывания.

T6. [Перемещение вверх.] Создать границу (x, y) — $(x, y+1)$, затем установить $y \leftarrow y+1$. Прерывание к шагу T10, если $y = y'$; в противном случае установить $Q \leftarrow Q + S$, $R \leftarrow R + b$, $S \leftarrow S + 2c$.

T7. [Перемещение вниз.] Создать границу (x, y) — $(x, y-1)$, затем установить $y \leftarrow y-1$. Прерывание к шагу T10, если $y = y'$; в противном случае установить $Q \leftarrow Q - S$, $R \leftarrow R - b$, $S \leftarrow S - 2c$.

T8. [Перемещение влево.] Создать границу (x, y) — $(x-1, y)$, затем установить $x \leftarrow x-1$. Прерывание к шагу T11, если $x = x'$; в противном случае установить $Q \leftarrow Q - R$, $R \leftarrow R - 2a$, $S \leftarrow S - b$.

T9. [Перемещение вправо.] Создать границу (x, y) — $(x+1, y)$, затем установить $x \leftarrow x+1$. Прерывание к шагу T11, если $x = x'$; в противном случае установить $Q \leftarrow Q + R$, $R \leftarrow R + 2a$, $S \leftarrow S + b$.

T10. [Завершение горизонтали.] Пока $x < x'$, создавать границы (x, y) — $(x+1, y)$ и устанавливать $x \leftarrow x + 1$. Пока $x > x'$, создавать границы (x, y) — $(x-1, y)$ и устанавливать $x \leftarrow x - 1$. Завершить работу алгоритма.

T11. [Завершение вертикали.] Пока $y < y'$, создавать границы (x, y) — $(x, y+1)$ и устанавливать $y \leftarrow y + 1$. Пока $y > y'$, создавать границы (x, y) — $(x, y-1)$ и устанавливать $y \leftarrow y - 1$. Завершить работу алгоритма. ▮

Например, при выполнении для $(x, y) = (20, 0)$, $(x', y') = (0, 10)$ и $Q(x, y) = 4x^2 + 16y^2 - 4x - 16y - 1595$ этот алгоритм создаст границы $(20, 0)$ — $(20, 1)$ — $(20, 2)$ — $(19, 2)$ — $(19, 3)$ — $(19, 4)$ — $(18, 4)$ — $(18, 5)$ — $(17, 5)$ — $(17, 6)$ — $\cdots$ — $(6, 9)$ — $(6, 10)$, а затем прямую линию до $(0, 10)$. (См.~(165) и (171).) В упр. 182 поясняется, почему этот алгоритм корректно работает.

Передвижение влево на шаге T8 удобно реализовать путем установки $H(y) \leftarrow H(y) \oplus (1 \ll (x_{\max} - x))$, используя векторы H из (166) и (167). Передвижение вправо реализуется аналогично, но с предварительной установкой $x \leftarrow x + 1$. Шаг T10 мог бы устанавливать

$$H(y) \leftarrow H(y) \oplus ((1 \ll (x_{\max} - \min(x, x'))) - (1 \ll (x_{\max} - \max(x, x')))); \qquad (172)$$

однако выполнение по одному шагу за раз может оказаться столь же эффективным, поскольку значение $|x' - x|$ часто достаточно мало́. Движение вверх и вниз не требует никаких действий, поскольку вертикальные границы излишни.

Заметим, что алгоритм работает несколько быстрее в частном случае, когда $b = 0$; окружности всегда относятся к этому частному случаю. Еще более частным случаем являются прямые линии, когда $a = b = c = 0$ (и, конечно же, еще более быстрым); для этого случая есть простой *однорегистровый* алгоритм (см. упр. 185).

Рис. 18. Пиксели изменяются с белых на черные и обратно на границах оцифрованных окружностей.

При заполнении большого числа контуров в одном и том же изображении с использованием векторов H значения пикселей изменяются между черным и белым при пересечении нечетного количества границ. На рис. 18 показано замощение гиперболической плоскости равносторонними треугольниками 45°-45°-45°, получаемыми наложением результатов нескольких сотен применений алгоритма T.

Алгоритм T применим только к коническим кривым. Но на практике это не такое уж сильное ограничение, поскольку почти любую фигуру, которую нужно начертить, можно приближенно представить как "кусочно коническую", т. е. состоящую из частей конических сечений, именуемых квадратичными сплайнами Безье, или *сквайнами* (*squine*). Например, на рис. 19 показана типичная кривая, состоящая из сквайнов, проходящая через 40 точек $(z_0, z_1, \ldots, z_{39}, z_{40})$, где $z_{40} = z_0$. Точки с четными номерами $(z_0, z_2, \ldots, z_{40})$ лежат на этой кривой; другие точки, $(z_1, z_3, \ldots, z_{39})$, называются управляющими, поскольку предназначены для управления локальными изгибами. Каждый сегмент $S(z_{2j}, z_{2j+1}, z_{2j+2})$ начинается в точке z_{2j} и имеет в этой точке направление $z_{2j+1} - z_{2j}$. Заканчивается он в точке z_{2j+2} с направлением $z_{2j+2} - z_{2j+1}$. Таким образом, если z_{2j} лежит на прямой линии,

Рис. 19. Сквайны, определяющие внешний контур символа '**S**'.

проходящей через точки z_{2j-1} и z_{2j+1}, то сквайн проходит точку z_{2j}, не изменяя направления.

В упр. 186 дается точное определение $S(z_{2j}, z_{2j+1}, z_{2j+2})$, а в упр. 187 поясняется, как оцифровать любую кривую из сквайнов с помощью алгоритма T. Область внутри оцифрованного контура может быть залита черными пикселями.

Кстати, задача *черчения* прямых и кривых линий на растре оказывается гораздо более сложной, чем задача *заполнения* оцифрованного контура, поскольку требуется, чтобы диагональные отрезки выглядели имеющими ту же ширину, что и горизонтальные и вертикальные. Отличное решение этой задачи было найдено Джоном Д. Хобби (John D. Hobby), *JACM* **36** (1989), 209–229.

***Вычисление без ветвлений.** Общая тенденция современных компьютеров — замедление вычислений при наличии в программе команд условного ветвления, поскольку неопределенный поток управления может влиять на схемы предиктивного предпросмотра. Поэтому мы использовали команды условной установки типа `CSNZ` в `MMIX`-программах наподобие (56). Действительно, четыре такие команды, как '`ADD z,y,1; SR t,u,2; CSNZ x,q,z; CSNZ v,q,t`', вероятно, окажутся быстрее, чем их трехкомандный аналог

```
BZ q,@+12;  ADD x,y,1;  SR v,u,2                    (173)
```

при измерении реального времени работы на высококонвейеризованных машинах, несмотря на то, что эмпирически вычисленная стоимость (173) составляет всего лишь $3v$, в соответствии с табл. 1.3.1′–1.

Битовые операции могут помочь уменьшить необходимость в дорогостоящем ветвлении. Например, если бы `MMIX` не имел команды `CSNZ`, мы могли бы записать

```
NEGU m,q;  OR m,m,q;  SR m,m,63;
ADD t,y,1;  XOR t,t,x;  AND t,t,m;  XOR x,x,t;      (174)
SR t,u,2;  XOR t,t,v;  AND t,t,m;  XOR v,v,t;
```

здесь первая строка создает маску $\mathtt{m} = -[q \neq 0]$. На некоторых компьютерах эти одиннадцать команд без ветвления все равно окажутся быстрее трех команд из (173).

Внутренний цикл алгоритма сортировки слиянием является поучительным примером. Предположим, что мы хотим многократно выполнять следующие операции.

Если $x_i < y_j$, установить $z_k \leftarrow x_i$, $i \leftarrow i + 1$
и перейти к *x_done*, если $i = i_{\max}$.
В противном случае установить $z_k \leftarrow y_j$, $j \leftarrow j + 1$
и перейти к *y_done*, если $j = j_{\max}$.
Затем установить $k \leftarrow k + 1$
и перейти к *z_done*, если $k = k_{\max}$.

Если реализовать эти действия "очевидным" способом, будут использованы четыре команды условного ветвления, три из которых активны на каждом пути в цикле:

```
1H CMP  t,xi,yj; BNN t,2F     Ветвление при x_i ≥ y_j.
   STO  xi,zbase,kk           z_k ← x_i.
   ADD  ii,ii,8               i ← i + 1.
   BZ   ii,X_Done             Переход к x_done при i = i_max.
   LDO  xi,xbase,ii           Загрузка x_i в регистр xi.
   JMP  3F                    Объединение с другой ветвью.
2H STO  yj,zbase,kk           z_k ← y_j.
   ADD  jj,jj,8               j ← j + 1.
   BZ   jj,Y_Done             Переход к y_done при j = j_max.
   LDO  yj,ybase,jj           Загрузка y_j в регистр yj.
3H ADD  kk,kk,8               k ← k + 1.
   PBNZ kk,1B                 Повтор, если k ≠ k_max.
   JMP  Z_Done                Переход к z_done. ▮
```

(Здесь $\mathtt{ii} = 8(i - i_{\max})$, $\mathtt{jj} = 8(j - j_{\max})$ и $\mathtt{kk} = 8(k - k_{\max})$; множитель 8 необходим из-за того, что x_i, y_j и z_k представляют собой октеты.) Эти четыре ветвления можно свести всего лишь к одному.

```
1H CMP  t,xi,yj               t ← sign(x_i − y_j).
   CSN  yj,t,xi               yj ← min(x_i, y_j).
   STO  yj,zbase,kk           z_k ← yj.
   AND  t,t,8                 t ← 8[x_i < y_j].
   ADD  ii,ii,t               i ← i + [x_i < y_j].
   LDO  xi,xbase,ii           Загрузка x_i в регистр xi.
   XOR  t,t,8                 t ← t ⊕ 8.
   ADD  jj,jj,t               j ← j + [x_i ≥ y_j].
   LDO  yj,ybase,jj           Загрузка y_j в регистр yj.
   ADD  kk,kk,8               k ← k + 1.
   AND  u,ii,jj; AND u,u,kk   u ← ii & jj & kk.
   PBN  u,1B                  Повтор, если i<i_max, j<j_max и k<k_max. ▮
```

Когда в этой версии завершается цикл, мы можем легко решить, следует ли продолжать с *x_done*, *y_done* или *z_done*. Эти команды каждый раз загружают из памяти как x_i, так и y_j, но излишнее значение уже находится в кэше.

***Другие применения MOR и MXOR.** Давайте завершим наше изучение работы с битами рассмотрением двух операций, разработанных специально для 64-битовой

работы. Команды MMIX MOR и MXOR, которые, по сути, выполняют матричное умножение булевых матриц размером 8×8, оказываются в высшей степени гибкими и мощными как сами по себе, так и в комбинации с другими битовыми операциями.

Если $x = (x_7 \ldots x_1 x_0)_{256}$ представляет собой октет, а $a = (a_7 \ldots a_1 a_0)_2$ является отдельным байтом, то команда MOR t,x,a выполняет установку $t \leftarrow a_7 x_7 \mid \cdots \mid a_1 x_1 \mid a_0 x_0$, в то время как команда MXOR t,x,a устанавливает $t \leftarrow a_7 x_7 \oplus \cdots \oplus a_1 x_1 \oplus a_0 x_0$. Например, и MOR t,x,2, и MXOR t,x,2 устанавливают $t \leftarrow x_1$; MOR t,x,3 устанавливает $t \leftarrow x_1 \mid x_0$; а MXOR t,x,3 устанавливает $t \leftarrow x_1 \oplus x_0$.

В общем случае, конечно, MOR и MXOR представляют собой функции от октетов. Если $y = (y_7 \ldots y_1 y_0)_{256}$ — обобщенный октет, команда MOR t,x,y дает октет t, j-й байт которого t_j является результатом применения MOR к x и y_j.

Предположим, что $x = -1 = {}^{\#}\texttt{ffffffffffffffff}$. Тогда MOR t,x,y вычисляет маску t, в которой байт t_j равен ${}^{\#}\texttt{ff}$, когда $y_j \neq 0$, а при $y_j = 0$ байт t_j равен нулю. Этот простой частный случай достаточно полезен, поскольку он выполняет всего за одну операцию то, для чего ранее в ситуации наподобие (92) требовалось семь операций.

В (66) мы заметили, что двух MOR будет достаточно для обращения порядка битов любого 64-битового слова, и многие другие перестановки битов также упрощаются при наличии MOR в списке команд компьютера. Предположим, что π представляет собой перестановку $\{0, 1, \ldots, 7\}$, которая отображает $0 \mapsto 0\pi$, $1 \mapsto 1\pi$, $\ldots$, $7 \mapsto 7\pi$. Тогда октет $p = (2^{7\pi} \ldots 2^{1\pi} 2^{0\pi})_{256}$ соответствует матрице перестановки, которая заставляет MOR сделать красивый трюк: MOR t,x,p приводит к *перестановке байтов* x, выполняя присваивания $t_j \leftarrow x_{j\pi}$. Кроме того, MOR u,p,y выполняет *перестановку битов* каждого байта y в соответствии с *обратной* перестановкой; команда устанавливает $u_j \leftarrow (a_7 \ldots a_1 a_0)_2$ при $y_j = (a_{7\pi} \ldots a_{1\pi} a_{0\pi})_2$.

С помощью еще одного небольшого мошенничества мы можем быстро выполнять другие перестановки, такие как идеальное тасование (76), которое преобразовывает заданный октет $z = 2^{32}x + y = (x_{31} \ldots x_1 x_0 y_{31} \ldots y_1 y_0)_2$ в "застегнутый" октет

$$w \;=\; x \ddagger y \;=\; (x_{31} y_{31} \ldots x_1 y_1 x_0 y_0)_2. \tag{175}$$

С помощью соответствующих матриц перестановок p, q и r промежуточные результаты

$$t = (x_{31} x_{27} x_{30} x_{26} x_{29} x_{25} x_{28} x_{24} y_{31} y_{27} y_{30} y_{26} y_{29} y_{25} y_{28} y_{24} \cdots \\ x_7 x_3 x_6 x_2 x_5 x_1 x_4 x_0 y_7 y_3 y_6 y_2 y_5 y_1 y_4 y_0)_2, \tag{176}$$

$$u = (y_{27} y_{31} y_{26} y_{30} y_{25} y_{29} y_{24} y_{28} x_{27} x_{31} x_{26} x_{30} x_{25} x_{29} x_{24} x_{28} \cdots \\ y_3 y_7 y_2 y_6 y_1 y_5 y_0 y_4 x_3 x_7 x_2 x_6 x_1 x_5 x_0 x_4)_2 \tag{177}$$

могут быть быстро вычислены с помощью четырех команд

$$\texttt{MOR t,z,p;}\;\; \texttt{MOR t,q,t;}\;\; \texttt{MOR u,t,r;}\;\; \texttt{MOR u,r,u;} \tag{178}$$

см. упр. 204. Так что имеется маска m, для которой 'PUT rM,m; MUX w,t,u' завершает идеальное тасование всего лишь за шесть тактов. Традиционный же метод в упр. 53 требует 30 тактов (пяти δ-обменов).

Аналогичная команда MXOR особенно полезна в бинарной линейной алгебре. Например, в упр. 1.3.1′–37 показано, что XOR и MXOR непосредственно реализуют сложение и умножение в конечном поле из 2^k элементов для $k \leq 8$.

Задача *циклической избыточной проверки* представляет поучительный пример другого случая, где проявляется великолепие MXOR. Потоки данных часто сопровождаются "байтами CRC" для обнаружения распространенных типов ошибок передачи [см. W. W. Peterson and D. T. Brown, *Proc. IRE* **49** (1961), 228–235]. Один популярный метод, используемый, например, в MP3-аудиофайлах, состоит в рассмотрении каждого байта $\alpha = (a_7 \ldots a_1 a_0)_2$ как если бы это был полином

$$\alpha(x) \;=\; (a_7 \ldots a_1 a_0)_x \;=\; a_7 x^7 + \cdots + a_1 x + a_0. \tag{179}$$

При передаче n байт $\alpha_{n-1} \ldots \alpha_1 \alpha_0$ мы вычисляем остаток

$$\beta = \bigl(\alpha_{n-1}(x) x^{8(n-1)} + \cdots + \alpha_1(x) x^8 + \alpha_0(x)\bigr) x^{16} \bmod p(x), \tag{180}$$

где $p(x) = x^{16} + x^{15} + x^2 + 1$, с применением полиномиальной арифметики по модулю 2 и добавляем коэффициенты β в качестве 16-битовой избыточной проверки.

Обычный способ вычисления β заключается в обработке по одному байту за раз в соответствии с классическими методами наподобие алгоритма 4.6.1D. Основная идея состоит в определении частичного результата $\beta_m = \bigl(\alpha_{n-1}(x) x^{8(n-1-m)} + \cdots + \alpha_{m+1}(x) x^8 + \alpha_m(x)\bigr) x^{16} \bmod p(x)$, так что $\beta_n = 0$, а затем в применении рекурсии

$$\beta_m = ((\beta_{m+1} \ll 8) \mathbin{\&} \#\texttt{ff00}) \oplus \mathit{crc_table}[(\beta_{m+1} \gg 8) \oplus \alpha_m] \tag{181}$$

для уменьшения m на 1, пока не будет достигнуто $m = 0$. Здесь $\mathit{crc_table}[\alpha]$ представляет собой 16-битовую запись таблицы, которая хранит остаток $\alpha(x) x^{16}$ по модулю $p(x)$ и по модулю 2 для $0 \le \alpha < 256$. [См. A. Perez, *IEEE Micro* **3**, 3 (June 1983), 40–50.]

Но, конечно, мы бы предпочли обрабатывать за один раз 64 бит вместо 8. Решение заключается в поиске матриц A и B размером 8×8, таких, что

$$\alpha(x) x^{64} \equiv (\alpha A)(x) + (\alpha B)(x) x^{-8} \quad (\text{по модулю } p(x) \text{ и } 2), \tag{182}$$

для произвольных байтов α, рассматривая α как вектор битов 1×8. Тогда мы можем дополнить заданные байты данных $\alpha_{n-1} \ldots \alpha_1 \alpha_0$ ведущими нулями так, чтобы n было кратно 8, и использовать следующий эффективный метод приведения.

$$\begin{aligned} &\text{Начать с } c \leftarrow 0,\ n \leftarrow n - 8 \text{ и } t \leftarrow (\alpha_{n+7} \ldots \alpha_n)_{256}. \\ &\text{Пока } n > 0, \text{ устанавливать } u \leftarrow t \cdot A,\ v \leftarrow t \cdot B,\ n \leftarrow n - 8, \\ &\qquad t \leftarrow (\alpha_{n+7} \ldots \alpha_n)_{256} \oplus u \oplus (v \gg 8) \oplus (c \ll 56) \text{ и } c \leftarrow v \mathbin{\&} \#\texttt{ff}. \end{aligned} \tag{183}$$

Здесь $t \cdot A$ и $t \cdot B$ обозначают матричное умножение, выполняемое с помощью команды MXOR. Искомые CRC-байты $(tx^{16} + cx^8) \bmod p(x)$ при этом легко получить из 64-битовой величины t и 8-битовой величины c. Детальное описание имеется в упр. 213; общее время работы для n байт оказывается равным всего лишь $(\mu + 10\upsilon) n/8 + O(1)$.

В приведенных ниже упражнениях содержится гораздо больше примеров значительной экономии при использовании команд MOR и MXOR. Несомненно, своего открытия ждут и новые трюки.

Для дальнейшего чтения. В книге *Hacker's Delight* Генри С. Уоррена-мл. (Henry S. Warren, Jr.) (Addison-Wesley, 2002)* глубоко рассматриваются битовые операции

* Имеется перевод на русский язык: Генри С. Уоррен. *Алгоритмические трюки для программистов.* М.: Издательский дом "Вильямс", 2003. — *Примеч. пер.*

и особое значение придается разнообразным возможностям, доступным в реальных компьютерах, не столь идеальных, как MMIX.

Упражнения

▶ **1.** [*15*] Каким оказывается результат присваиваний $x \leftarrow x \oplus y$, $y \leftarrow y \oplus (x \mathbin{\&} m)$, $x \leftarrow x \oplus y$?

2. [*16*] (Г. С. Уоррен-мл. (H. S. Warren, Jr.).) Справедливы ли следующие соотношения для всех целых чисел x и y? (i) $x \oplus y \le x \mid y$; (ii) $x \mathbin{\&} y \le x \mid y$; (iii) $|x - y| \le x \oplus y$.

3. [*M20*] Пусть $x = (x_{n-1} \ldots x_1 x_0)_2$ и пусть $x_{n-1} = 1$. Обозначим $x^M = (\bar{x}_{n-1} \ldots \bar{x}_1 \bar{x}_0)_2$. Таким образом, мы имеем 0^M, 1^M, 2^M, 3^M, $\ldots = -1$, 0, 1, 0, 3, 2, 1, 0, 7, 6, $\ldots$, если считать, что $0^M = -1$. Докажите, что $(x \oplus y)^M < |x - y| \le x \oplus y$ для всех $x, y \ge 0$.

▶ **4.** [*M16*] Пусть $x^C = \bar{x}$, $x^N = -x$, $x^S = x + 1$ и $x^P = x - 1$ обозначают дополнение, отрицание, следующее и предшествующее числа для целого числа x с бесконечной точностью. Тогда мы имеем $x^{CC} = x^{NN} = x^{SP} = x^{PS} = x$. Что собой представляют x^{CN} и x^{NC}?

5. [*M21*] Докажите или опровергните следующие гипотезы относительно бинарных сдвигов:

а) $(x \ll j) \ll k = x \ll (j + k)$;

б) $(x \gg j) \mathbin{\&} (y \ll k) = ((x \gg (j + k)) \mathbin{\&} y) \ll k = (x \mathbin{\&} (y \ll (j + k))) \gg j$.

6. [*M22*] Найдите все целые числа x и y, такие, что (а) $x \gg y = y \gg x$; (б) $x \ll y = y \ll x$.

7. [*M22*] (Р. Шрёппель (R. Schroeppel), 1972.) Найдите быстрый способ преобразования бинарного числа $x = (\ldots x_2 x_1 x_0)_2$ в его негабинарный (по основанию -2) двойник $x = (\ldots x'_2 x'_1 x'_0)_{-2}$, и наоборот. *Указание*: достаточно только двух битовых операций!

▶ **8.** [*M22*] Для данного конечного множества S неотрицательных целых чисел "минимальный эксклудант" S определяется как

$$\operatorname{mex}(S) = \min\{\, k \mid k \ge 0 \text{ и } k \notin S \,\}.$$

Обозначим через $x \oplus S$ множество $\{x \oplus y \mid y \in S\}$, а через $S \oplus y$ — множество $\{x \oplus y \mid x \in S\}$. Докажите, что если $x = \operatorname{mex}(S)$ и $y = \operatorname{mex}(T)$, то $x \oplus y = \operatorname{mex}\bigl((S \oplus y) \cup (x \oplus T)\bigr)$.

9. [*M26*] (*Ним.*) Два игрока играют в игру с k кучками камней, где в кучке j имеется a_j камней. Если, когда наступает ход очередного игрока, оказывается, что $a_1 = \cdots = a_k = 0$, этот игрок проигрывает; в противном случае игрок уменьшает одну из кучек на любое количество камней, отбрасывая удаленные камни, и ход переходит к другому игроку. Докажите, что игрок может обеспечить себе победу тогда и только тогда, когда $a_1 \oplus \cdots \oplus a_k \ne 0$. [*Указание*: воспользуйтесь результатами упр. 8.]

10. [*HM40*] (*Нимберы*, или *поле Конвея*.) Продолжая выполнение упр. 8, рекурсивно определим операцию "ним-умножения" $x \otimes y$ с помощью формулы

$$x \otimes y = \operatorname{mex}\bigl\{(x \otimes j) \oplus (i \otimes y) \oplus (i \otimes j) \mid 0 \le i < x, 0 \le j < y\bigr\}.$$

Докажите, что $\oplus$ и $\otimes$ определяют *поле* на множестве всех неотрицательных целых чисел. Кроме того, докажите что если $0 \le x, y < 2^{2^n}$, то $x \otimes y < 2^{2^n}$, а $2^{2^n} \otimes y = 2^{2^n} y$. (В частности, это поле содержит подполя размером 2^{2^n} для всех $n \ge 0$.) Поясните, как эффективно вычислить $x \otimes y$.

▶ **11.** [*M26*] (Х. В. Ленстра (H. W. Lenstra), 1978.) Найдите простой способ охарактеризовать все пары положительных целых чисел (m, n), для которых в поле Конвея $m \otimes n = mn$.

12. [*M26*] Разработайте алгоритм *деления* нимберов. *Указание*: если $x < 2^{2^{n+1}}$, то мы имеем $x \otimes (x \oplus (x \gg 2^n)) < 2^{2^n}$.

13. [*M32*] (*Ним второго порядка.*) Расширим игру из упр. 9, разрешив два вида ходов: либо, как и ранее, уменьшается a_j для некоторого j; либо уменьшается a_j, и a_i для некоторого $i < j$ заменяется произвольным неотрицательным целым числом. Докажите, что игрок, чей ход текущий, может обеспечить свою победу тогда и только тогда, когда размеры кучек камней удовлетворяют либо условию $a_2 \neq a_3 \oplus \cdots \oplus a_k$, либо $a_1 \neq a_3 \oplus (2 \otimes a_4) \oplus \cdots \oplus ((k-2) \otimes a_k)$. Например, когда $k = 4$ и $(a_1, a_2, a_3, a_4) = (7, 5, 0, 5)$, единственным выигрышным ходом является $(7, 5, 6, 3)$.

14. [*M30*] Предположим, что каждый узел полного бесконечного бинарного дерева помечен 0 или 1. Такие метки удобно представить в виде последовательности $T = (t, t_0, t_1, t_{00}, t_{01}, t_{10}, t_{11}, t_{000}, \ldots)$ с одним битом t_α для каждой бинарной строки α; корень имеет метку t, метками левого поддерева являются $T_0 = (t_0, t_{00}, t_{01}, t_{000}, \ldots)$, а метками правого поддерева — $T_1 = (t_1, t_{10}, t_{11}, t_{100}, \ldots)$. Любая такая маркировка может использоваться для преобразования 2-адического целого числа $x = (\ldots x_2 x_1 x_0)_2$ в 2-адическое целое число $y = (\ldots y_2 y_1 y_0)_2 = T(x)$ путем установок $y_0 = t$, $y_1 = t_{x_0}$, $y_2 = t_{x_0 x_1}$ и других, так, чтобы $T(x) = 2T_{x_0}(\lfloor x/2 \rfloor) + t$. (Другими словами, x определяет бесконечный путь в бинарном дереве, а y соответствует меткам на этом пути, справа налево в битовой строке при перемещении по дереву сверху вниз.)

Функция ветвления (*branching function*) представляет собой отображение $x^T = x \oplus T(x)$, определяемое такой маркировкой. Например, если $t_{01} = 1$ и все прочие t_α равны 0, мы имеем $x^T = x \oplus 4[x \bmod 4 = 2]$.

а) Докажите, что каждая функция ветвления представляет собой перестановку 2-адических чисел.

б) Для каких целых чисел k выражение $x \oplus (x \ll k)$ является функцией ветвления?

в) Пусть $x \mapsto x^T$ является отображением 2-адических целых чисел в 2-адические целые числа. Докажите, что x^T является функцией ветвления тогда и только тогда, когда $\rho(x \oplus y) = \rho(x^T \oplus y^T)$ для всех 2-адических x и y.

г) Докажите, что композиции и обращения функций ветвления являются функциями ветвления. (Таким образом, множество $\mathcal{B}$ всех функций ветвления представляет собой группу перестановок.)

д) Функция ветвления является *сбалансированной*, если метки удовлетворяют условию $t_\alpha = t_{\alpha 0} \oplus t_{\alpha 1}$ для всех α. Покажите, что множество всех сбалансированных функций ветвления представляет собой подгруппу $\mathcal{B}$.

▶ **15.** [*M26*] Студент Шустрый заметил, что $((x + 2) \oplus 3) - 2 = ((x - 2) \oplus 3) + 2$ для всех x. Найдите все константы a и b, такие, что $((x + a) \oplus b) - a = ((x - a) \oplus b) + a$ является тождеством.

16. [*M31*] Функция от x называется *анимирующей* (*animating*), если она может быть записана в виде

$$((\ldots((((x + a_1) \oplus b_1) + a_2) \oplus b_2) + \cdots) + a_m) \oplus b_m$$

для некоторых целочисленных констант a_1, b_1, a_2, b_2, $\ldots$, a_m, b_m, где $m > 0$.

а) Докажите, что каждая анимирующая функция является функцией ветвления (см. упр. 14).

б) Кроме того, докажите, что она является сбалансированной тогда и только тогда, когда $b_1 \oplus b_2 \oplus \cdots \oplus b_m = 0$. *Указание*: какая маркировка бинарного дерева соответствует анимирующей функции $((x \oplus c) - 1) \oplus c$?

в) Пусть $\lfloor x \rceil = x \oplus (x - 1) = 2^{\rho(x)+1} - 1$. Покажите, что каждая сбалансированная анимирующая функция может быть записана в виде

$$x \oplus \lfloor x \oplus p_1 \rceil \oplus \lfloor x \oplus p_2 \rceil \oplus \cdots \oplus \lfloor x \oplus p_l \rceil, \qquad p_1 < p_2 < \cdots < p_l,$$

для некоторых целых чисел $\{p_1, p_2, \ldots, p_l\}$, где $l \geq 0$, и такое представление является единственным.

г) И обратно, покажите, что каждое такое выражение определяет сбалансированную анимирующую функцию.

17. [*HM36*] Результаты упр. 16 делают возможным выяснение, являются ли две заданные анимирующие функции равными. Существует ли алгоритм, выясняющий, тождественно ли *любое* заданное выражение нулю, в случае, когда это выражение построено из конечного числа целочисленных переменных и констант с применением только лишь бинарных операций $+$ и $\oplus$? Что если мы также разрешим операцию &?

18. [*M25*] Показанный здесь забавный узор в строке x и столбце y имеет значение $(x^2 y \gg 11)\,\&\,1$, где $1 \leq x, y \leq 256$. Существует ли простой способ пояснить некоторые основные его характеристики математически?

▶ **19.** [*M37*] (*Теорема о переупорядочении Пейли.*) Пусть для трех заданных векторов, $A = (a_0, \ldots, a_{2^n-1})$, $B = (b_0, \ldots, b_{2^n-1})$ и $C = (c_0, \ldots, c_{2^n-1})$, состоящих из неотрицательных чисел,

$$f(A, B, C) \;=\; \sum_{j \oplus k \oplus l = 0} a_j b_k c_l.$$

Например, если $n = 2$, мы имеем $f(A, B, C) = a_0b_0c_0 + a_0b_1c_1 + a_0b_2c_2 + a_0b_3c_3 + a_1b_0c_1 + a_1b_1c_0 + a_1b_2c_3 + \cdots + a_3b_3c_0$; в общем случае имеется 2^{2n} членов, по одному для каждого выбора j и k. Наша цель заключается в доказательстве того, что $f(A, B, C) \leq f(A^*, B^*, C^*)$, где A^* означает вектор A, отсортированный в неувеличивающемся порядке: $a_0^* \geq a_1^* \geq \cdots \geq a_{2^n-1}^*$.

а) Докажите этот результат для случая, когда все элементы A, B и C являются нулями и единицами.

б) Покажите, что, следовательно, это справедливо и в общем случае.

в) Аналогично $f(A, B, C, D) = \sum_{j \oplus k \oplus l \oplus m = 0} a_j b_k c_l d_m \leq f(A^*, B^*, C^*, D^*)$.

▶ **20.** [*21*] (*Хак Госпера.*) Следующие семь операций представляют собой полезную функцию y от x, где x — положительное целое число. Поясните, что́ собой представляет эта функция и чем она так полезна.

$$u \leftarrow x \,\&\, -x; \qquad v \leftarrow x + u; \qquad y \leftarrow v + (((v \oplus x)/u) \gg 2).$$

21. [*22*] *Обратите* хак Госпера — покажите, как вычислить x, зная y.

22. [*21*] Эффективно реализуйте хак Госпера на машине `MMIX` в предположении, что $x < 2^{64}$, и без использования деления.

▶ **23.** [*27*] Последовательность вложенных скобок можно представить в виде бинарного числа, помещая 1 в позиции каждой правой скобки. Например, '`(())()`' при этом соответствует $(001101)_2$, числу 13. Назовем такое число *скобочным следом* (*parenthesis trace*).

а) Каковы наименьший и наибольший скобочные следы, имеющие ровно m единиц?

б) Предположим, что x представляет собой скобочный след, а y — следующий по величине скобочный след с тем же количеством единиц. Покажите, что y можно вычислить из x с помощью короткой цепочки операций, аналогичной хаку Госпера.

в) Реализуйте свой метод на машине `MMIX` в предположении, что $\nu x \leq 32$.

▶ **24.** [*M30*] Программа 1.3.2′P указывает `MMIX`, как построить таблицу первых пяти сотен простых чисел с использованием пробного деления для выяснения простоты. Напишите

MMIX-программу, использующую "решето Эратосфена" (упр. 4.5.4–8) для построения таблицы всех нечетных простых чисел, меньших N, упакованную в октеты $Q_0, Q_1, \ldots, Q_{N/128-1}$, как в (27). Считаем, что $N \le 2^{32}$ и что оно кратно 128. Чему равно время работы программы для $N = 3584$?

▶ **25.** [*15*] На книжной полке стоят рядом четыре тома. В каждом из них ровно 500 страниц, напечатанных на 250 листах бумаги толщиной 0.1 мм; каждая книга имеет переднюю и заднюю обложки толщиной по 1 мм. Книжный червь ползет от страницы 1 тома 1 до страницы 500 тома 4. Какой длины путешествие он при этом совершит?

26. [*22*] Предположим, что нам требуется обеспечить произвольный доступ к таблице с 12 миллионами 5-битовых элементов данных. Можно упаковать 12 таких элементов в одно 64-битовое слово, тем самым разместив таблицу в 8 Мбайт памяти. Но произвольный доступ при этом потребует деления на 12, что достаточно медленно, так что мы можем предпочесть затратить 12 Мбайт памяти, отводя каждому элементу целый байт.

Покажите, что существует эффективный с точки зрения использования памяти подход, не требующий деления.

27. [*21*] Как бы вы вычисляли (использованы обозначения из (32)–(43)) (а) $(\alpha 10^a 01^b)_2$? (б) $(\alpha 10^a 11^b)_2$? (в) $(\alpha 00^a 01^b)_2$? (г) $(0^\infty 11^a 00^b)_2$? (д) $(0^\infty 01^a 00^b)_2$? (е) $(0^\infty 11^a 11^b)_2$?

28. [*16*] Каков результат операции $(x+1)\ \&\ \bar{x}$?

29. [*20*] (В. Р. Пратт (V. R. Pratt).) Выразите магическую маску μ_k из (47) через μ_{k+1}.

30. [*20*] Если $x = 0$, команды MMIX (46) выполняют установку $\rho \leftarrow 64$ (что является достаточным приближением к ∞). Какие изменения в (50) и (51) дадут тот же результат?

▶ **31.** [*20*] Студент Шустрый решил вычислять линеечную функцию с помощью простого цикла следующим образом: "установить $\rho \leftarrow 0$; затем, пока $x\ \&\ 1 = 0$, устанавливать $\rho \leftarrow \rho + 1$ и $x \leftarrow x \gg 1$". Он поясняет, что, когда x представляет собой случайное целое число, среднее количество правых сдвигов равно среднему значению ρ, которое равно 1; а стандартное отклонение составляет всего лишь $\sqrt{2}$, так что цикл почти всегда будет быстро завершаться. Раскритикуйте это решение.

32. [*20*] Чему равно время вычисления ρx, если запрограммировать (52) для машины MMIX?

▶ **33.** [*26*] (Лейзерсон (Leiserson), Прокоп (Prokop) и Рэндолл (Randall), 1998.) Покажите, что если в (52) заменить '58' на '49', то этот метод можно использовать для быстрого распознавания *обоих* битов числа $y = 2^j + 2^k$ при $64 > j > k \ge 0$. (Хотя $\binom{64}{2} = 2016$ случаев должно быть выделено.)

34. [*M23*] Пусть x и y являются 2-адическими целыми числами. Истинны или ложны утверждения: (а) $\rho(x\ \&\ y) = \max(\rho x, \rho y)$; (б) $\rho(x \mid y) = \min(\rho x, \rho y)$; (в) $\rho x = \rho y$ тогда и только тогда, когда $x \oplus y = (x-1) \oplus (y-1)$.

▶ **35.** [*M26*] Согласно теореме Райтвизнера (Reitwiesner), упр. 4.1–34, каждое целое число n имеет единственное представление $n = n^+ - n^-$, такое, что $n^+ \& n^- = (n^+ \mid n^-) \& ((n^+ \mid n^-) \gg 1) = 0$. Покажите, что n^+ и n^- можно быстро вычислить с помощью битовых операций. *Указание*: докажите тождество $(x \oplus 3x)\ \&\ ((x \oplus 3x) \gg 1) = 0$.

36. [*20*] Для заданного $x = (x_{63} \ldots x_1 x_0)_2$ предложите эффективный способ вычисления величин

i) $x^\oplus = (x^\oplus_{63} \ldots x^\oplus_1 x^\oplus_0)_2$, где $x^\oplus_k = x_k \oplus \cdots \oplus x_1 \oplus x_0$ для $0 \le k < 64$;
ii) $x^\& = (x^\&_{63} \ldots x^\&_1 x^\&_0)_2$, где $x^\&_k = x_k \wedge \cdots \wedge x_1 \wedge x_0$ для $0 \le k < 64$.

37. [*16*] Какие изменения в (55) и (56) сделают $\lambda 0$ дающей результат -1?

38. [*17*] Сколько времени выполняется процедура выделения крайнего слева бита (57) при ее реализации на машине MMIX?

▶ **39.** [*20*] Формула (43) показывает, как удалить крайнюю справа серию единичных битов из заданного числа x. А как удалить крайнюю слева серию единичных битов?

▶ **40.** [*21*] Докажите (58) и найдите простой способ выяснить, справедливо ли соотношение $\lambda x < \lambda y$ для заданных x и $y \geq 0$.

41. [*M22*] Каковы производящие функции целочисленных последовательностей (а) ρn, (б) λn и (в) νn?

42. [*M21*] Пусть $n = 2^{e_1} + \cdots + 2^{e_r}$, с $e_1 > \cdots > e_r \geq 0$. Выразите сумму $\sum_{k=0}^{n-1} \nu k$ через показатели степеней $e_1, \ldots, e_r$.

▶ **43.** [*20*] Насколько разреженным должно быть x, чтобы реализация (63) на машине `MMIX` была быстрее реализации (62)?

▶ **44.** [*23*] (Э. Фрид (E. Freed), 1983.) Предложите быстрый способ вычисления *взвешенной* суммы битов $\sum jx_j$.

▶ **45.** [*20*] (Т. Рокицки (T. Rokicki), 1999.) Поясните, как проверить выполнение соотношения $x^R < y^R$ без обращения порядка битов в x и y.

46. [*22*] Метод (68) использует шесть операций для обмена двух битов $x_i \leftrightarrow x_j$ регистра. Покажите, что в действительности этот обмен можно выполнить с помощью только *трех* команд `MMIX`.

47. [*10*] Может ли обобщенный δ-обмен (69) быть выполнен методом наподобие (67)?

48. [*M21*] Сколько различных δ-обменов возможны в n-битовом регистре? (Когда $n = 4$, δ-обмен может преобразовать 1234 в 1234, 1243, 1324, 1432, 2134, 2143, 3214, 3412, 4231.)

▶ **49.** [*M30*] Пусть $s(n)$ обозначает наименьшие δ-обмены, которых достаточно для обращения порядка битов n-битового числа.

а) Докажите, что $s(n) \geq \lceil \log_3 n \rceil$ при нечетном n, $s(n) \geq \lceil \log_3 3n/2 \rceil$ при четном n.
б) Вычислите $s(n)$ при $n = 3^m$, $2 \cdot 3^m$, $(3^m + 1)/2$ и $(3^m - 1)/2$.
в) Чему равны $s(32)$ и $s(64)$? *Указание*: покажите, что $s(5n + 2) \leq s(n) + 2$.

50. [*M37*] Продолжая упр. 49, докажите, что $s(n) = \log_3 n + O(\log \log n)$.

51. [*23*] Пусть c представляет собой константу, $0 \leq c < 2^d$. Найдите все последовательности масок $(\theta_0, \theta_1, \ldots, \theta_{d-1}, \hat{\theta}_{d-2}, \ldots, \hat{\theta}_1, \hat{\theta}_0)$, такие, что общая схема перестановок (71) отображает $x \mapsto x^\pi$, где битовая перестановка π определяется либо как (а) $j\pi = j \oplus c$, либо как (б) $j\pi = (j + c) \bmod 2^d$. [Маски должны удовлетворять соотношениям $\theta_k \subseteq \mu_{d,k}$ и $\hat{\theta}_k \subseteq \mu_{d,k}$, так что (71) соответствует рис. 12; см. (48). Заметим, что обращение порядка битов $x^\pi = x^R$ представляет собой частный случай $c = 2^d - 1$ п. (а), в то время как п. (б) соответствует циклическому сдвигу вправо $x^\pi = (x \gg c) + (x \ll (2^d - c))$.]

52. [*22*] Найдите шестнадцатеричные константы $(\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \hat{\theta}_4, \hat{\theta}_3, \hat{\theta}_2, \hat{\theta}_1, \hat{\theta}_0)$, которые заставляют (71) выполнять следующие важные 64-битовые перестановки, основанные на бинарном представлении $j = (j_5 j_4 j_3 j_2 j_1 j_0)_2$: (а) $j\pi = (j_0 j_5 j_4 j_3 j_2 j_1)_2$; (б) $j\pi = (j_2 j_1 j_0 j_5 j_4 j_3)_2$; (в) $j\pi = (j_1 j_0 j_5 j_4 j_3 j_2)_2$; (г) $j\pi = (j_0 j_1 j_2 j_3 j_4 j_5)_2$. [Случай (а) представляет собой "идеальное тасование" (175), которое превращает $(x_{63} \ldots x_{33} x_{32} x_{31} \ldots x_1 x_0)_2$ в $(x_{63} x_{31} \ldots x_{33} x_1 x_{32} x_0)_2$; случай (б) транспонирует матрицу битов размером 8×8; случай (в), аналогично, транспонирует матрицу 4×16; а случай (г) возникает в связи с "быстрым преобразованием Фурье", см. упр. 4.6.4–14.]

▶ **53.** [*M25*] Перестановки в упр. 52, как говорится, "вызваны перестановкой индексных цифр", поскольку мы получаем $j\pi$ путем перестановки бинарных цифр j. Предположим, что $j\pi = (j_{(d-1)\psi} \ldots j_{1\psi} j_{0\psi})_2$, где ψ — перестановка $\{0, 1, \ldots, d-1\}$. Докажите, что если ψ имеет t циклов, то 2^d-битовая перестановка $x \mapsto x^\pi$ может быть получена путем только $d - t$ обменов. В частности, покажите, что это наблюдение ускоряет все четыре случая из упр. 52.

54. [*22*] (Р. В. Госпер (R. W. Gosper), 1985.) Покажите, как транспонировать битовую матрицу размером $m \times m$, хранящуюся в m^2 крайних справа битах регистра, путем $(2^k(m-1))$-обменов для $0 \le k < \lceil \lg m \rceil$. Детально опишите метод при $m = 7$.

▶ **55.** [*26*] Предположим, что битовая матрица размером $n \times n$ хранится в крайних справа n^2 битах n^3-битового регистра. Докажите, что для умножения двух таких матриц при $n = 2^d$ достаточно $18d + 2$ битовых операций; умножение матриц может быть как булевым (наподобие `MOR`), так и по модулю 2 (наподобие `MXOR`).

56. [*24*] Предложите способ транспонирования битовой матрицы размером 7×9 в 64-битовом регистре.

57. [*22*] Сеть $P(2^d)$ на рис. 12 содержит $(2d-1)2^{d-1}$ перекрещиваний. Докажите, что любая перестановка 2^d элементов может быть осуществлена путем некоторой настройки, в которой активны не более $d2^{d-1}$ из них.

▶ **58.** [*M32*] Первые d столбцов модулей перекрещивания в перестановочной сети $P(2^d)$ выполняют 1-обмен, затем 2-обмен, ... и наконец 2^{d-1}-обмен, когда провода сети превращаются в горизонтальные линии, как показано здесь для $d = 3$. Пусть $N = 2^d$. Эти N линий, вместе с $Nd/2$ перекрещиваниями, образуют так называемый "омега-маршрутизатор" или "обратную бабочку". Цель данного упражнения заключается в изучении множества Ω всех перестановок φ, таких, что мы можем получить $(0\varphi, 1\varphi, \ldots, (N-1)\varphi)$ в качестве выходов справа от омега-маршрутизатора, когда входами слева являются $(0, 1, \ldots, N-1)$.

а) Докажите, что $|\Omega| = 2^{Nd/2}$. (Таким образом, $\lg|\Omega| = Nd/2 \sim \frac{1}{2}\lg N!$.)

б) Докажите, что перестановка φ множества $\{0, 1, \ldots, N-1\}$ принадлежит Ω тогда и только тогда, когда

$$\text{из} \quad i \bmod 2^k = j \bmod 2^k \quad \text{и} \quad i\varphi \gg k = j\varphi \gg k \quad \text{вытекает} \quad i\varphi = j\varphi \qquad (*)$$

для всех $0 \le i, j < N$ и всех $0 \le k \le d$.

в) Упростите условие $(*)$ до следующего для всех $0 \le i, j < N$:

$$\text{из} \quad \lambda(i\varphi \oplus j\varphi) < \rho(i \oplus j) \quad \text{вытекает} \quad i = j.$$

г) Пусть T представляет собой множество всех перестановок τ множества $\{0, 1, \ldots, N-1\}$, таких, что $\rho(i \oplus j) = \rho(i\tau \oplus j\tau)$ для всех i и j. (Это множество функций ветвления, рассматривавшихся в упр. 14, по модулю 2^d; так что в нем имеется 2^{N-1} членов, $2^{N/2+d-1}$ из которых являются анимирующими функциями по модулю 2^d.) Докажите, что $\varphi \in \Omega$ тогда и только тогда, когда $\tau\varphi \in \Omega$ для всех $\tau \in T$.

д) Предположим, что φ и ψ являются перестановками Ω, влияющими на разные элементы; т. е. из $j\varphi \ne j$ вытекает $j\psi = j$ для $0 \le j < N$. Докажите, что $\varphi\psi \in \Omega$.

е) Докажите, что перестановка $0\varphi \ldots (N-1)\varphi$ является омега-маршрутизируемой тогда и только тогда, когда она сортируема битонической сортирующей сетью Батчера порядка N. (См. раздел 5.3.4.)

59. [*M30*] Сколько имеется омега-маршрутизируемых перестановок, работающих только на отрезке $[a\,..\,b]$, для заданных $0 \le a < b < N = 2^d$? (Таким образом, мы хотим подсчитать количество $\varphi \in \Omega$, таких, что из $j\varphi \ne j$ вытекает $a \le j \le b$. Упр. 58, (а) представляет собой частный случай $a = 0$, $b = N - 1$.)

60. [*HM28*] Для заданной случайной перестановки множества $\{0, 1, \ldots, 2n-1\}$ обозначим через p_{nk} вероятность того, что имеется 2^k способов установки перекрещиваний в первом и последнем столбцах перестановочной сети $P(2n)$ при осуществлении этой перестановки. Другими словами, p_{nk} представляет собой вероятность того, что связанный граф имеет

k циклов (см. (75)). Что собой представляет производящая функция $\sum_{k\ge 0} p_{nk} z^k$? Чему равны среднее значение и дисперсия 2^k?

61. [*46*] Является ли выяснение, осуществляется ли заданная перестановка при использовании рекурсивного метода с рис. 12, при реализации (71) как минимум с одной маской $\theta_j = 0$, NP-сложной задачей?

▶ **62.** [*22*] Пусть $N = 2^d$. Можно очевидным образом представить перестановку π множества $\{0, 1, \ldots, N-1\}$ с помощью таблицы из N чисел по d бит каждое. При таком представлении мы получаем моментальный доступ к $y = x\pi$ для данного x; но нам требуется $\Omega(N)$ шагов для поиска $x = y\pi^-$ для заданного y.

Покажите, что при использовании того же количества памяти можно представить произвольную перестановку таким образом, что и $x\pi$, и $y\pi^-$ будут вычислимы за $O(d)$ шагов.

63. [*19*] Для каких целых чисел w, x, y и z функция застежки удовлетворяет соотношениям (i) $x \ddagger y = y \ddagger x$? (ii) $(x \ddagger y) \gg z = (x \gg \lceil z/2 \rceil) \ddagger (y \gg \lfloor z/2 \rfloor)$? (iii) $(w \ddagger x) \& (y \ddagger z) = (w \& y) \ddagger (x \& z)$?

64. [*22*] Найдите "простое" выражение для zip-операции над суммами $(x + x') \ddagger (y + y')$ как функцию от $z = x \ddagger y$ и $z' = x' \ddagger y'$.

65. [*M16*] Бинарный полином $u(x) = u_0 + u_1 x + \cdots + u_{n-1} x^{n-1} \pmod 2$ можно представить целым числом $u = (u_{n-1} \ldots u_1 u_0)_2$. Если $u(x)$ и $v(x)$ соответствуют, таким образом, целым числам u и v, то какой полином соответствует $u \ddagger v$?

▶ **66.** [*M26*] Предположим, что полином $u(x)$ был представлен как n-битовое целое число u, как в упр. 65, и что $v = u \oplus (u \ll \delta) \oplus (u \ll 2\delta) \oplus (u \ll 3\delta) \oplus \cdots$ для некоторого целого числа δ.

а) Каким простым образом описать полином $v(x)$?

б) Предположим, что n велико и биты u упакованы в 64-битовые слова. Как бы вы вычисляли v при $\delta = 1$ с использованием булевых операций с 64-битовыми регистрами?

в) Ответьте на вопрос п. (б), но при $\delta = 64$.

г) Ответьте на вопрос п. (б), но при $\delta = 3$.

д) Ответьте на вопрос п. (б), но при $\delta = 67$.

67. [*M31*] Пусть $u(x)$ представляет собой полином степени $< n$, представленный, как в упр. 65. Обсудите вычисление $v(x) = u(x)^2 \bmod (x^n + x^m + 1)$, когда $0 < m < n$ и как m, так и n нечетны. *Указание*: эта задача имеет интересное отношение к идеальному тасованию.

68. [*20*] Какие три команды `MMIX` реализуют операцию δ-сдвига (79)?

69. [*25*] Докажите, что метод (80) всегда выделяет должные биты при надлежащей установке масок θ_k: мы никогда не затрем ни один из важных битов y_j.

▶ **70.** [*31*] (Гай Л. Стил-мл. (Guy L. Steele Jr.), 1994.) Подскажите хороший способ вычисления масок θ_0, θ_1, …, θ_{d-1}, необходимых в обобщенной процедуре сжатия (80), для заданного $\chi \ne 0$?

71. [*20*] Поясните, как *обратить* процедуру (80), переходя от сжатого значения $y = (y_{r-1} \ldots y_1 y_0)_2$ к числу $z = (z_{63} \ldots z_1 z_0)_2$, у которого $z_{j_i} = y_i$ для $0 \le i < r$.

72. [*25*] (Е. Хилевиц (Y. Hilewitz) и Р. Б. Ли (R. B. Lee).) Докажите, что операция "сбор и переворот" ($81'$) является омега-маршрутизируемой в смысле упр. 58.

73. [*22*] Докажите, что d тщательно отобранных шагов (а) операции "отделения агнцев от козлищ" (sheep-and-goats) или (б) операции сбора с переворотом ($81'$) реализуют любую требуемую 2^d-битовую перестановку.

74. [*22*] Для заданных количеств $(c_0, c_1, \ldots, c_{2^d-1})$ для процедуры Чжуна–Вонга поясните, почему подходящий циклический 1-сдвиг всегда может дать новые количества $(c'_0, c'_1, \ldots, c'_{2^d-1})$, для которых $\sum c'_{2l} = \sum c'_{2l+1}$, тем самым позволив применить рекурсию.

▶ **75.** [*32*] Метод Чжуна–Вонга реплицирует бит l регистра ровно c_l раз, но дает результаты в "зашифрованном" порядке. Например, показанный в разделе случай $(c_0, \ldots, c_7) = (1, 2, 0, 2, 0, 2, 0, 1)$ дает $(x_7x_3x_1x_5x_5x_3x_1x_0)_2$. В некоторых приложениях это может оказаться неудобным; мы можем предпочесть, чтобы биты сохраняли свой исходный порядок, а именно $(x_7x_5x_5x_3x_3x_1x_1x_0)_2$ для данного примера.

Докажите, что перестановочная сеть $P(2^d)$ на рис. 12 может быть модифицирована для достижения этой цели для любой заданной последовательности количеств $(c_0, c_1, \ldots, c_{2^d-1})$, если заменить $d \cdot 2^{d-1}$ модулей перекрещивания в правой половине обобщенными *модулями отображения* 2×2. (Модуль перекрещивания со входами (a, b) дает в качестве выхода либо (a, b), либо (b, a); модуль отображения может, кроме того, давать (a, a) или (b, b).)

76. [*47*] *Отображающая сеть* аналогична сети сортировки или перестановки, но использует модули отображения 2×2 вместо компараторов или перекрещиваний; предполагается, что такая сеть способна выводить все n^n возможных отображений n входов. В упр. 75, в сочетании с рис. 12, показано, что для $n = 2^d$ существует отображающая сеть со всего лишь $4d - 2$ уровнями задержки и с $n/2$ модулями на каждом уровне; кроме того, такая конструкция требует применения модулей отображения 2×2 (вместо простых перекрещиваний) только в d из этих уровней.

Каково с точностью $O(n)$ наименьшее количество $G(n)$ модулей, достаточное для реализации обобщенной n-элементной отображающей сети?

77. [*26*] (Р. В. Флойд (R. W. Floyd) и В. Р. Пратт (V. R. Pratt).) Разработайте алгоритм, который проверяет, является ли данная стандартная n-сеть сортирующей, как определено в упр. раздела 5.3.4. Если заданная сеть имеет r компараторов, ваш алгоритм должен использовать $O(r)$ битовых операций со словами длиной 2^n.

78. [*M27*] (*Проверка непересекаемости.*) Предположим, что каждое бинарное число x_1, x_2, ..., x_m представляет множество в генеральной совокупности из $n - k$ элементов, так что каждое x_j меньше 2^{n-k}. Студент Шустрый решил проверять множества на непересекаемость путем проверки выполнения условия

$$x_1 \mid x_2 \mid \cdots \mid x_m \;=\; (x_1 + x_2 + \cdots + x_m) \bmod 2^n.$$

Докажите или опровергните следующее утверждение: проверка Шустрого работает тогда и только тогда, когда $k \geq \lg(m - 1)$.

▶ **79.** [*20*] Пусть $x \neq 0$ и $x \subseteq \chi$. Каким образом можно легко найти наибольшее целое число $x_\prime < x$, такое, что $x_\prime \subseteq \chi$? (Таким образом, $(x_\prime)' = (x')_\prime = x$ в связи с (84).)

80. [*20*] Предложите быстрый способ поиска всех максимальных собственных подмножеств множества. Говоря точнее, для данного χ с $\nu\chi = m$ мы хотим найти все $x \subseteq \chi$, такие, что $\nu x = m - 1$.

81. [*21*] Найдите формулу для "рассеянной разности", парной к "рассеянной сумме" (86).

82. [*21*] Легко ли сдвинуть влево на 1 рассеянный аккумулятор (например, заменить $(y_2x_4x_3y_1x_2y_0x_1x_0)_2$ на $(y_1x_4x_3y_0x_20x_1x_0)_2$)?

▶ **83.** [*33*] Продолжая выполнять упр. 82, найдите способ сдвига рассеянного 2^d-битового аккумулятора *вправо* на 1 для данных z и χ за $O(d)$ шагов.

84. [*25*] Поясните, как для заданных n-битовых чисел $z = (z_{n-1} \ldots z_1z_0)_2$ и $\chi = (\chi_{n-1} \ldots \chi_1\chi_0)_2$ вычислить "растянутые" величины $z \leftharpoonup \chi = (z_{(n-1)\leftharpoonup\chi} \ldots z_{1\leftharpoonup\chi}z_{0\leftharpoonup\chi})_2$ и $z \rightharpoonup \chi = (z_{(n-1)\rightharpoonup\chi} \ldots z_{1\rightharpoonup\chi}z_{0\rightharpoonup\chi})_2$, где

$$j \leftharpoonup \chi = \max\{k \mid k \leq j \text{ и } \chi_k = 1\}, \qquad j \rightharpoonup \chi = \min\{k \mid k \geq j \text{ и } \chi_k = 1\};$$

мы полагаем $z_{j\leftarrow\chi} = 0$, если $\chi_k = 0$ для $0 \le k \le j$, и $z_{j\rightarrow\chi} = 0$, если $\chi_k = 0$ для $n > k \ge j$. Например, если $n = 11$ и $\chi = (01101110010)_2$, то $z \leftarrow \chi = (z_9 z_9 z_8 z_6 z_6 z_5 z_4 z_1 z_1 z_1 0)_2$ и $z \rightarrow \chi = (0 z_9 z_8 z_8 z_6 z_5 z_4 z_4 z_4 z_1 z_1)_2$.

85. [*22*] (К. Д. Точер (K. D. Tocher), 1954.) Представим, что у вас имеется древняя ЭВМ 1950-х годов с барабанной памятью для хранения данных и вам нужно выполнить некоторые вычисления с массивом $a[i, j, k]$ размером $32 \times 32 \times 32$, индексы которого представляют собой 5-битовые целые числа в диапазоне $0 \le i, j, k < 32$. К сожалению, ваша машина имеет очень небольшую высокоскоростную память: в любой момент времени вы можете обращаться только к 128 последовательным элементам массива в быстрой памяти. Поскольку ваше приложение обычно выполняет перемещения из $a[i, j, k]$ в соседнюю позицию $a[i', j', k']$, где $|i - i'| + |j - j'| + |k - k'| = 1$, вы должны разместить массив в памяти таким образом, чтобы, если $i = (i_4 i_3 i_2 i_1 i_0)_2$, $j = (j_4 j_3 j_2 j_1 j_0)_2$ и $k = (k_4 k_3 k_2 k_1 k_0)_2$, элемент массива $a[i, j, k]$ хранился на барабане в позиции $(k_4 j_4 i_4 k_3 j_3 i_3 k_2 j_2 i_2 k_1 j_1 i_1 k_0 j_0 i_0)_2$. При таком чередовании битов небольшие изменения i, j или k будут приводить к небольшим изменениям адреса.

Обсудите реализацию этой функции адресации: (а) как изменяются ее значения при изменении i, j или k на ± 1? (б) как обеспечить произвольный доступ к $a[i, j, k]$ для данных i, j и k? (в) как обнаруживать ошибку отсутствия страницы (т. е. когда необходимо загрузить в быструю память с барабана новый сегмент со 128 элементами)?

86. [*M27*] Массив из $2^p \times 2^q \times 2^r$ элементов размещен таким образом, что элемент $a[i, j, k]$ находится в позиции, биты которой представляют собой $p + q + r$ бит (i, j, k), некоторым образом переставленных. Кроме того, этот массив хранится во внешней памяти с использованием страниц размером 2^s. (В упр. 85 рассмотрен случай $p = q = r = 5$ и $s = 7$.) Какая стратегия распределения такого рода минимизирует количество случаев, когда $a[i, j, k]$ находится на странице, отличной от страницы $a[i', j', k']$, при суммировании по всем i, j, k, i', j' и k', таким, что $|i - i'| + |j - j'| + |k - k'| = 1$?

▶ **87.** [*20*] Предположим, что каждый байт 64-битового слова x содержит код ASCII, который представляет букву, цифру либо пробел. Какие три битовые операции конвертируют все буквы из нижнего регистра в верхний?

88. [*20*] Для заданных $x = (x_7 \ldots x_0)_{256}$ и $y = (y_7 \ldots y_0)_{256}$ вычислите $z = (z_7 \ldots z_0)_{256}$, где $z_j = (x_j - y_j) \bmod 256$ для $0 \le j < 8$. (См. операцию сложения в (87).)

89. [*23*] Для заданных $x = (x_{31} \ldots x_1 x_0)_4$ и $y = (y_{31} \ldots y_1 y_0)_4$ вычислите $z = (z_{31} \ldots z_1 z_0)_4$, где $z_j = \lfloor x_j / y_j \rfloor$ для $0 \le j < 32$, в предположении, что ни одно значение y_j не равно нулю.

90. [*20*] Правило битового усреднения (88) всегда выполняет округление вниз при нечетном $x_j + y_j$. Сделайте его менее смещенным путем округления в таких случаях к ближайшему нечетному целому числу.

▶ **91.** [*26*] (*Альфа-каналы.*) Метод (88) является хорошим способом вычисления битового среднего, но приложения компьютерной графики часто требуют более общего смешивания 8-битовых значений. Покажите, как для заданных трех октетов $x = (x_7 \ldots x_0)_{256}$, $y = (y_7 \ldots y_0)_{256}$ и $\alpha = (a_7 \ldots a_0)_{256}$ битовые операции позволяют вычислить $z = (z_7 \ldots z_0)_{256}$, где каждый байт z_j представляет собой хорошее приближение к $((255 - a_j)x_j + a_j y_j)/255$ *без* привлечения умножений. Реализуйте свой метод на машине `MMIX`.

▶ **92.** [*21*] Что будет, если заменить вторую строку (88) строкой '$z \leftarrow (x \mid y) - z$'?

93. [*18*] Какова основная формула для вычитания аналогична формуле (89) для сложения?

94. [*21*] Пусть в (90) $x = (x_7 \ldots x_1 x_0)_{256}$ и $t = (t_7 \ldots t_1 t_0)_{256}$. Может ли t_j быть ненулевым при ненулевом x_j? Может ли t_j быть нулевым при нулевом x_j?

95. [*22*] Как с помощью битовых операций выяснить, все ли байты $x = (x_7 \ldots x_1 x_0)_{256}$ различны?

96. [*21*] Поясните (93) и найдите подобную формулу, которая устанавливает тестовые флаги $t_j \leftarrow 128[x_j \le y_j]$.

97. [*23*] В статье Лесли Лампорта (Leslie Lamport), вышедшей в свет в 1975 году, представлена следующая "задача, взятая из реального алгоритма оптимизации компилятора": для заданных октетов $x = (x_7 \ldots x_0)_{256}$ и $y = (y_7 \ldots y_0)_{256}$ вычислить $t = (t_7 \ldots t_0)_{256}$ и $z = (z_7 \ldots z_0)_{256}$ так, чтобы $t_j \ne 0$ тогда и только тогда, когда $x_j \ne 0$, $x_j \ne$ `'*'` и $x_j \ne y_j$; а $z_j = (x_j = 0?\ y_j\colon (x_j \ne$ `'*'` $\wedge\, x_j \ne y_j?$ `'*'`$\colon x_j))$.

98. [*20*] Для заданных $x = (x_7 \ldots x_0)_{256}$ и $y = (y_7 \ldots y_0)_{256}$ вычислить $z = (z_7 \ldots z_0)_{256}$ и $w = (w_7 \ldots w_0)_{256}$, где $z_j = \max(x_j, y_j)$ и $w_j = \min(x_j, y_j)$ для $0 \le j < 8$.

▶ **99.** [*28*] Найдите шестнадцатеричные константы a, b, c, d и e, такие, что шесть битовых операций

$$y \leftarrow x \oplus a, \quad t \leftarrow ((((y \,\&\, b) + c) \mid y) \oplus d) \,\&\, e$$

будут для любых байтов $x = (x_7 \ldots x_1 x_0)_{256}$ вычислять флаги $t = (f_7 \ldots f_1 f_0)_{256} \ll 7$, где

$$f_0 = [x_0 = \texttt{'!'}],\ f_1 = [x_1 \ne \texttt{'*'}],\ f_2 = [x_2 < \texttt{'A'}],\ f_3 = [x_3 > \texttt{'z'}],\ f_4 = [x_4 \ge \texttt{'a'}],$$
$$f_5 = [x_5 \in \{\texttt{'h'}, \texttt{'i'}, \ldots, \texttt{'9'}\}],\ f_6 = [x_6 \le 168],\ f_7 = [x_7 \in \{\texttt{'<'}, \texttt{'='}, \texttt{'>'}, \texttt{'?'}\}].$$

100. [*25*] Предположим, что $x = (x_{15} \ldots x_1 x_0)_{16}$ и $y = (y_{15} \ldots y_1 y_0)_{16}$ являются *двоично-десятичными* числами, где $0 \le x_j, y_j < 10$ для каждого j. Поясните, как вычислить их сумму $u = (u_{15} \ldots u_1 u_0)_{16}$ и разность $v = (v_{15} \ldots v_1 v_0)_{16}$, где $0 \le u_j, v_j < 10$ и

$$(u_{15} \ldots u_1 u_0)_{10} = ((x_{15} \ldots x_1 x_0)_{10} + (y_{15} \ldots y_1 y_0)_{10}) \bmod 10^{16},$$
$$(v_{15} \ldots v_1 v_0)_{10} = ((x_{15} \ldots x_1 x_0)_{10} - (y_{15} \ldots y_1 y_0)_{10}) \bmod 10^{16},$$

без попыток выполнить преобразования между системами счисления.

▶ **101.** [*22*] Два октета, x и y, содержат отрезки времени, представленные пятью полями, которые соответственно означают дни (3 байт), часы (1 байт), минуты (1 байт), секунды (1 байт) и миллисекунды (2 байт). Можете ли вы быстро складывать и вычитать эти величины, не преобразовывая их из этого представления в системе счисления со смешанным основанием в бинарную и обратно?

102. [*25*] Обсудите подпрограммы для сложения и вычитания полиномов по модулю 5, когда в 64-битовое слово упаковано (а) 16 4-битовых коэффициентов или (б) 21 3-битовый коэффициент.

▶ **103.** [*22*] Иногда удобно представлять небольшие числа в *унарной* записи, так что 0, 1, 2, 3, ..., k выглядят в компьютере соответственно как $(0)_2$, $(1)_2$, $(11)_2$, $(111)_2$, ..., $2^k - 1$. Тогда max и min легко реализовать как $\mid$ и &.

Предположим, что байтами $x = (x_7 \ldots x_0)_{256}$ являются такие унарные числа, в то время как байты $y = (y_7 \ldots y_0)_{256}$ представляют собой либо 0, либо 1. Поясните, как "сложить" y с x или "вычесть" y из x, получая $u = (u_7 \ldots u_0)_{256}$ и $v = (v_7 \ldots v_0)_{256}$, где

$$u_j = 2^{\min(8, \lg(x_j+1)+y_j)} - 1 \qquad \text{и} \qquad v_j = 2^{\max(0, \lg(x_j+1)-y_j)} - 1.$$

104. [*22*] Используйте битовые операции для проверки корректности даты, представленной полями "год-месяц-день" (y, m, d), как в (22). Вам нужно вычислить значение t, которое равно нулю тогда и только тогда, когда $1900 < y < 2100$, $1 \le m \le 12$ и $1 \le d \le max_day(m)$, где месяц m имеет не более $max_day(m)$ дней. Можно ли сделать это менее чем за 20 операций?

105. [*30*] Обсудите битовые операции, позволяющие *отсортировать* заданные значения $x = (x_7 \ldots x_0)_{256}$ и $y = (y_7 \ldots y_0)_{256}$, так что в результате получится $x_0 \le y_0 \le \cdots \le x_7 \le y_7$.

106. [*27*] Поясните процедуру Фредмана–Уилларда (95). Покажите также, что простая модификация их метода будет вычислять $2^{\lambda x}$ без выполнения левых сдвигов.

▶ **107.** [*22*] Реализуйте алгоритм B на машине `MMIX` при $d = 4$ и сравните свою реализацию с (56).

108. [*26*] Адаптируйте алгоритм B для случаев, когда n не имеет вида $d \cdot 2^d$.

109. [*20*] Вычислите ρx для n-битовых чисел x за $O(\log\log n)$ широкословных шагов.

▶ **110.** [*30*] Предположим, что $n = 2^{2^e}$ и $0 \le x < n$. Покажите, как вычислить $1 \ll x$ за $O(e)$ широкословных шагов, используя команды сдвига только на константные величины. (Таким образом, вместе с алгоритмом B мы можем выделить старший бит n-битового числа за $O(\log\log n)$ таких шагов.)

111. [*23*] Поясните, как работает распознавание шаблона 01^r в (98).

112. [*46*] Можно ли идентифицировать все появления шаблона 1^r0 за $O(1)$ широкословных шагов?

113. [*23*] *Сильная широкословная цепочка* представляет собой широкословную цепочку определенной ширины n, которая является также 2-адической цепочкой для всех n-битовых выборов x_0. Например, 2-битовая широкословная цепочка (x_0, x_1) с $x_1 = x_0 + 1$ не является сильной, поскольку $x_0 = (11)_2$ делает $x_1 = (00)_2$. Но $(x_0, x_1, \ldots, x_4)$ является сильной широкословной цепочкой, которая вычисляет $(x_0 + 1) \bmod 4$ для всех $0 \le x_0 < 4$, если мы установим $x_1 = x_0 \oplus 1$, $x_2 = x_0 \mathbin{\&} 1$, $x_3 = x_2 \ll 1$ и $x_4 = x_1 \oplus x_3$.

Для заданной широкословной цепочки $(x_0, x_1, \ldots, x_r)$ шириной n постройте сильную широкословную цепочку $(x'_0, x'_1, \ldots, x'_{r'})$ той же ширины, такую, что $r' = O(r)$, а $(x_0, x_1, \ldots, x_r)$ является подпоследовательностью $(x'_0, x'_1, \ldots, x'_{r'})$.

114. [*16*] Предположим, что $(x_0, x_1, \ldots, x_r)$ представляет собой сильную широкословную цепочку шириной n, которая вычисляет значение $f(x) = x_r$ по заданному n-битовому числу $x = x_0$. Постройте широкословную цепочку $(X_0, X_1, \ldots, X_r)$ шириной mn, которая вычисляет $X_r = (f(\xi_1) \ldots f(\xi_m))_{2^n}$ для любого заданного mn-битового значения $X_0 = (\xi_1 \ldots \xi_m)_{2^n}$, где $0 \le \xi_1, \ldots, \xi_m < 2^n$.

▶ **115.** [*24*] Для заданного 2-адического целого числа $x = (\ldots x_2 x_1 x_0)_2$ мы хотим вычислить $y = (\ldots y_2 y_1 y_0)_2 = f(x)$ из x путем обнуления всех блоков из последовательных единиц, которые обладают тем свойством, что (а) непосредственно за блоком не следуют два нуля; или (б) за блоком следует нечетное количество нулей, за которым начинается новый блок из единиц, или (в) блок содержит нечетное количество единиц. Например, если x имеет вид $(\ldots 01110111001101000110)_2$, то y представляет собой (а) $(\ldots 00000111000001000110)_2$; (б) $(\ldots 00000111000000000110)_2$; (в) $(\ldots 00000000001100000110)_2$. (Считается, что справа от x_0 имеется бесконечно много нулей. Таким образом, в случае (а) мы имеем

$$y_j = x_j \wedge ((\bar{x}_{j-1} \wedge \bar{x}_{j-2}) \vee (x_{j-1} \wedge \bar{x}_{j-2} \wedge \bar{x}_{j-3}) \vee (x_{j-1} \wedge x_{j-2} \wedge \bar{x}_{j-3} \wedge \bar{x}_{j-4}) \vee \cdots)$$

для всех j, где $x_k = 0$ для $k < 0$.) Найдите 2-адические цепочки для y в каждом случае.

116. [*HM30*] Предположим, что $x = (\ldots x_2 x_1 x_0)_2$ и $y = (\ldots y_2 y_1 y_0)_2 = f(x)$, где y вычисляется с помощью 2-адической цепочки без операций сдвига. Пусть L — множество всех бинарных строк, таких, что $y_j = [x_j \ldots x_1 x_0 \in L]$, и будем считать, что все константы, использованные в цепочке, являются рациональными 2-адическими числами. Докажите, что L является регулярным языком. Какие языки L соответствуют функциям в упр. 115, п. (а) и (б)?

117. [*HM46*] Продолжая выполнение упр. 116, ответьте, существует ли простой способ охарактеризовать регулярные языки L, возникающие в 2-адических цепочках без сдвигов? (Язык $L = 0^*(10^*10^*)^*$, кажется, не соответствует ни одной такой цепочке.)

118. [*30*] Согласно лемме A мы не можем вычислить функцию $x \gg 1$ для всех n-битовых чисел x, используя только сложения, вычитания и битовые булевы операции (без сдвигов и ветвлений). Покажите, однако, что $O(n)$ таких операций будет необходимо и достаточно, если добавить в репертуар оператор "монус" $y \dot{-} z$.

119. [*20*] Вычислите функцию $f_{py}(x)$ в (102) с помощью четырех широкословных шагов.

▶ **120.** [*M25*] Имеется $2^{n2^{mn}}$ функций, которые отображают n-битовые числа $(x_1, \ldots, x_m)$ в n-битовое число $f(x_1, \ldots, x_m)$. Сколько из них можно реализовать с помощью сложений, вычитаний, умножений и несдвиговых битовых булевых операций (по модулю 2^n)?

▶ **121.** [*M25*] Согласно упр. 3.1–6, функция, отображающая $[0\,..\,2^n)$ в себя, является в конечном итоге периодической.

а) Докажите, что если f является любой n-битовой широкословной функцией, которая может быть реализована без команд сдвига, то длины периодов таких функций всегда представляют собой степени 2.

б) Однако для каждого p между 1 и n существует n-битовая широкословная цепочка длиной 3, имеющая период p.

122. [*M22*] Завершите доказательство леммы B.

123. [*M23*] Пусть a_q представляет собой константу $1 + 2^q + 2^{2q} + \cdots + 2^{(q-1)q} = (2^{q^2} - 1)/(2^q - 1)$. Используя (104), покажите, что существует бесконечно много q, таких, что операция умножения на a_q, по модулю 2^{q^2} требует $\Omega(\log q)$ шагов в любой n-битовой широкословной цепочке с $n \geq q^2$.

124. [*M38*] Завершите доказательство теоремы R′ путем определения n-битовой широкословной цепочки $(x_0, x_1, \ldots, x_f)$ и установки $(U_0, U_1, \ldots, U_f)$, так, что для $0 \leq t \leq f$, все входные данные $x \in U_t$ приводят к, по сути, подобному состоянию $Q(x,t)$ в следующем смысле: (i) текущая команда в $Q(x,t)$ не зависит от x; (ii) если регистр r_j имеет известное значение в $Q(x,t)$, он хранит $x_{j'}$ для некоторого определенного индекса $j' \leq t$; (iii) если ячейка памяти $M[z]$ была изменена, она содержит $x_{z''}$ для некоторого определенного индекса $z'' \leq t$. (Значения j' и z'' зависят от j, z и t, но не от x.) Кроме того, $|U_t| \geq n/2^{2^t-1}$, и программа не может гарантировать, что $r_1 = \rho x$ при $t < f$. *Указание*: из леммы B следует, что при малом t требуется рассмотреть ограниченное количество величин сдвигов и адресов памяти.

125. [*M33*] Докажите теорему P′. *Указание*: лемма B остается справедливой, если заменить в (103) '$= 0$' на '$= \alpha_s$', для любых значений α_s.

126. [*M46*] Требует ли операция извлечения старшего бита, $2^{\lambda x}$, $\Omega(\log\log n)$ шагов на n-битовой машине с базовой памятью? (См. упр. 110.)

127. [*HM40*] Докажите, что необходимо как минимум $\Omega(\log n/\log\log n)$ широкословных шагов для вычисления функции четности, $(\nu x) \bmod 2$, с использованием теории сложности схем. [*Указание*: каждая широкословная операция принадлежит классу сложности AC_0.]

128. [*M46*] Можно ли вычислить $(\nu x) \bmod 2$ за $O(\log n/\log\log n)$ широкословных шагов?

129. [*M46*] Требует ли контрольное сложение $\Omega(\log n)$ широкословных шагов?

130. [*M46*] Существует ли n-битовая константа a, такая, что функция $(a \ll x) \bmod 2^n$ требует $\Omega(\log n)$ n-битовых широкословных шагов?

▶ **131.** [*23*] Напишите `MMIX`-программу для алгоритма R, когда граф представлен списками дуг. Узлы вершин имеют как минимум два поля, именуемые `LINK` и `ARCS`, а узлы дуг

имеют поля TIP и NEXT, как поясняется в разделе 7. Изначально все поля LINK нулевые, за исключением заданного множества вершин Q, которое представлено циклическим списком. Ваша программа должна изменить этот циклический список так, чтобы он представлял множество R всех достижимых вершин.

▶ **132.** [*M27*] *Клика* в графе представляет собой множество взаимно смежных вершин; клика является *максимальной*, если она не содержится ни в какой другой. Цель данного упражнения заключается в рассмотрении алгоритма Д. К. М. Муди (J. K. M. Moody) и Д. Холлиса (J. Hollis), которые предложили удобный способ поиска каждой максимальной клики в не слишком большом графе с использованием битовых операций.

Предположим, что G представляет собой граф с n вершинами $V = \{0, 1, \ldots, n-1\}$. Пусть $\rho_v = \sum\{2^u \mid u \text{ — } v \text{ или } u = v\}$ — строка v рефлексивной матрицы смежности G и пусть $\delta_v = \sum\{2^u \mid u \neq v\} = 2^n - 1 - 2^v$. Каждое подмножество $U \subseteq V$ представимо в виде n-битового целого числа $\sigma(U) = \sum_{u \in U} 2^u$; например, $\delta_v = \sigma(V \setminus v)$. Мы также определим битовое пересечение

$$\tau(U) \;=\; \mathop{\&}_{0 \le u < n} (u \in U?\ \rho_u\colon \delta_u).$$

Например, если $n = 5$, мы имеем $\tau(\{0, 2\}) = \rho_0 \mathbin{\&} \delta_1 \mathbin{\&} \rho_2 \mathbin{\&} \delta_3 \mathbin{\&} \delta_4$.

а) Докажите, что U представляет собой клику тогда и только тогда, когда $\tau(U) = \sigma(U)$.

б) Покажите, что если $\tau(U) = \sigma(T)$, то T является кликой.

в) Для $1 \le k \le n$ рассмотрим 2^k битовых пересечений

$$C_k \;=\; \left\{ \mathop{\&}_{0 \le u < k} (u \in U?\ \rho_u\colon \delta_u) \;\middle|\; U \subseteq \{0, 1, \ldots, k-1\} \right\},$$

и пусть C_k^+ — максимальный элемент C_k. Докажите, что U является максимальной кликой тогда и только тогда, когда $\sigma(U) \in C_n^+$.

г) Поясните, как вычислить C_k^+ из C_{k-1}^+, начиная с $C_0^+ = \{2^n - 1\}$.

▶ **133.** [*20*] Как можно применить алгоритм из упр. 132 к заданному графу G, чтобы найти (а) все максимальные независимые множества вершин? (б) все минимальные вершинные покрытия (множества, соприкасающиеся со всеми ребрами)?

134. [*15*] В (119), (123) и (124) встречаются девять классов отображений тернарных значений. К какому классу принадлежит представление (128), если $a = 0$, $b = *$, $c = 1$?

135. [*22*] В свою трехзначную логику Лукашевич (Łukasiewicz), помимо (127), включил несколько других операций: $\neg x$ (отрицание) обменивает 0 с 1, но оставляет неизменным значение $*$; $\diamond x$ (возможность) определена как $\neg x \Rightarrow x$; $\square x$ (необходимость) определена как $\neg\diamond\neg x$; и $x \Leftrightarrow y$ (эквивалентность) определена как $(x \Rightarrow y) \wedge (y \Rightarrow x)$. Поясните, как выполнять эти операции с использованием представления (128).

136. [*29*] Предложите двухбитовое кодирование для бинарных операций на множестве $\{a, b, c\}$, которые определены следующими "таблицами умножения":

$$\text{(а)}\ \begin{pmatrix} a & b & c \\ b & c & c \\ c & c & c \end{pmatrix}; \qquad \text{(б)}\ \begin{pmatrix} a & c & b \\ c & b & a \\ b & a & c \end{pmatrix}; \qquad \text{(в)}\ \begin{pmatrix} a & b & a \\ a & a & c \\ a & b & c \end{pmatrix}.$$

137. [*21*] Покажите, что операция из упр. 136, (в) проще с упакованными векторами наподобие (131), чем с неупакованными векторами (130).

138. [*24*] Найдите пример двухбитовой кодировки трех состояний, для которой класс V_a является наилучшим.

139. [*25*] Если x и y представляют собой знаковые биты 0, +1 или −1, то какое двухбитовое кодирование подходит для вычисления их суммы $(z_1 z_2)_3 = x + y$, где z_1 и z_2 также являются знаковыми битами? (Это "полусумматор" для сбалансированных тернарных чисел.)

140. [*27*] Разработайте экономичный *полный сумматор* для сбалансированных тернарных чисел: покажите, как вычислять знаковые биты u и v, такие, что $3u + v = x + y + z$ при $x, y, z \in \{0, +1, -1\}$.

▶ **141.** [*30*] *Числа Улама* $\langle U_1, U_2, \ldots\rangle = \langle 1, 2, 3, 4, 6, 8, 11, 13, 16, 18, 26, \ldots\rangle$ для $n \geq 3$ определяются следующим образом: U_n представляет собой наименьшее целое число $> U_{n-1}$, которое имеет *единственное представление* $U_n = U_j + U_k$ для $0 < j < k < n$. Покажите, что с помощью битовых методов можно быстро вычислить миллион чисел Улама.

▶ **142.** [*33*] Подкуб, такой как $*10*1*01$, может быть представлен с помощью кодов звездочек 10010100 и кодов битов 01001001, как в (85); но возможны и многие другие методы. Какая схема представления подкубов работает лучше всего в случае поиска простых импликант с помощью алгоритма на основе консенсуса из упр. 7.1.1–31?

143. [*20*] Пусть x является 64-битовым числом, которое представляет шахматную доску размером 8×8, где бит 1 в любой позиции означает наличие в ней коня. Найдите формулу для 64-битового числа $f(x)$, которое имеет 1 в каждой позиции, достижимой за один ход конем из x. Например, белые кони в начале игры соответствуют $x = {}^{\#}\texttt{42}$; тогда $f(x) = {}^{\#}\texttt{a51800}$.

144. [*16*] Какой узел является братом узла j в скошенной пирамиде? (См. (134).)

145. [*17*] Поясните (137), когда h *меньше* высоты j.

▶ **146.** [*M20*] Докажите соотношение (138), связывающее функции ρ и λ.

▶ **147.** [*M20*] Какие значения πv, βv, αv и τj получаются в алгоритме V, когда лес представляет собой

а) пустой ориентированный граф с вершинами $\{v_1, \ldots, v_n\}$ и без дуг?

б) ориентированный путь $v_n \longrightarrow \cdots \longrightarrow v_2 \longrightarrow v_1$?

148. [*M21*] Может ли при предвычислениях для алгоритма V в S иметься $\beta x_3 \longrightarrow^* \beta y_2 \longrightarrow^* \beta x_2 \longrightarrow^* \beta y_1 \longrightarrow^* \beta x_1$, если в лесу $x_3 \longrightarrow x_2 \longrightarrow x_1 \longrightarrow \Lambda$ и $y_2 \longrightarrow y_1 \longrightarrow \Lambda$? (Если может, то два различных дерева в S "переплетены".)

▶ **149.** [*23*] Разработайте процедуру предвычислений для алгоритма V.

▶ **150.** [*25*] Для заданного массива элементов $A_1, \ldots, A_n$ задача поиска *минимума на отрезке* заключается в определении $k(i, j)$, такого, что $A_{k(i,j)} = \min(A_i, \ldots, A_j)$ для любых заданных индексов i и j с $1 \leq i \leq j \leq n$. Докажите, что алгоритм V будет решать эту задачу после выполнения $O(n)$ шагов предвычислений, которые на основе массива A подготовят необходимые таблицы $(\pi, \beta, \alpha, \tau)$. *Указание*: рассмотрите бинарное дерево поиска, построенное из последовательности ключей $(p(1), p(2), \ldots, p(n))$, где p представляет собой перестановку $\{1, 2, \ldots, n\}$, такую, что $A_{p(1)} \leq A_{p(2)} \leq \cdots \leq A_{p(n)}$.

151. [*22*] И обратно, покажите, что любой алгоритм для поиска минимума на отрезке можно использовать для поиска ближайших общих предков, по сути, с той же эффективностью.

152. [*M21*] Докажите корректность алгоритма V.

▶ **153.** [*M20*] Указатели в навигационной стопке наподобие (144) могут быть упакованы в бинарную строку, такую, как

0	1 0	0	1 0 0	0	0 0	1	0 1 0 0	0	0 0	0	0 0 0
2	4	6	8	10	12	14	16	18	20	22	24

.

В какой битовой позиции (слева) заканчивается указатель для узла j?

154. [*20*] Серые линии на рис. 14 показывают, как каждый пятиугольник составлен из десяти треугольников. Какое разложение гиперболической плоскости определяется одними этими серыми линиями, без черных ребер пятиугольников?

▶ **155.** [*M21*] Докажите, что если α представляет собой негафибоначчиев код x, то справедливо соотношение $(x\phi) \bmod 1 = (\alpha 0)_{1/\phi}$.

156. [*21*] Разработайте алгоритм (а) для преобразования данного целого числа x в его негафибоначчиев код α и (б) для преобразования данного негафибоначчиева кода α в $x = N(\alpha)$.

157. [*M21*] Поясните рекурсию (148) для негафибоначчиевых предшественника и преемника.

158. [*M26*] Пусть $\alpha = a_n \ldots a_1$ представляет собой бинарный код $F(\alpha 0) = a_n F_{n+1} + \cdots + a_1 F_2$ в стандартной фибоначчиевой системе счисления (146). Разработайте методы, аналогичные (148) и (149), для увеличения и уменьшения таких кодовых слов.

159. [*M34*] В упр. 7 показано, что преобразования между бинарной и негабинарной системами счисления выполняются весьма просто. Рассмотрите преобразования между негафибоначчиевыми кодовыми словами и обычными фибоначчиевыми кодами из упр. 158.

160. [*M29*] Докажите, что (150) и (151) дают согласованные метки кодов для пентагрида.

161. [*20*] Клетки шахматной доски можно раскрасить черным и белым цветами таким образом, чтобы соседние клетки имели разные цвета. Обладает ли этим свойством пентагрид?

▶ **162.** [*HM37*] Поясните, как начертить пентагрид (рис. 14). Какие окружности в нем присутствуют?

163. [*HM41*] Разработайте способ навигации по треугольникам в мозаике на рис. 18.

164. [*23*] Исходное определение кастеризации в 1957 году имело вид не (157), а

$$\mathrm{custer}'(X) = X \mathbin{\&} {\sim}(X_{\mathrm{NW}} \mathbin{\&} X_{\mathrm{N}} \mathbin{\&} X_{\mathrm{NE}} \mathbin{\&} X_{\mathrm{W}} \mathbin{\&} X_{\mathrm{E}} \mathbin{\&} X_{\mathrm{SW}} \mathbin{\&} X_{\mathrm{S}} \mathbin{\&} X_{\mathrm{SE}}).$$

Почему определение (157) предпочтительнее?

165. [*21*] (Р. Э. Кирш (R. A. Kirsch).) Обсудите вычисление клеточного автомата размером 3×3, у которого

$$X^{(t+1)} = \mathrm{custer}(\overline{X}^{(t)}) = {\sim}X^{(t)} \mathbin{\&} (X_{\mathrm{N}}^{(t)} \mid X_{\mathrm{W}}^{(t)} \mid X_{\mathrm{E}}^{(t)} \mid X_{\mathrm{S}}^{(t)}).$$

166. [*M23*] Пусть $f(M, N)$ представляет собой максимальное число черных пикселей в растре X размером $M \times N$, для которого $X = \mathrm{custer}(X)$. Докажите, что $f(M, N) = \frac{4}{5}MN + O(M + N)$.

167. [*24*] (*Жизнь.*) Если растр X представляет массив ячеек, которые либо мертвы (0), либо живы (1), булева функция

$$f(x_{\mathrm{NW}}, \ldots, x, \ldots, x_{\mathrm{SE}}) = [\,2 < x_{\mathrm{NW}} + x_{\mathrm{N}} + x_{\mathrm{NE}} + x_{\mathrm{W}} + \tfrac{1}{2}x + x_{\mathrm{E}} + x_{\mathrm{SW}} + x_{\mathrm{S}} + x_{\mathrm{SE}} < 4\,]$$

может привести к изумительным жизненным историям под управлением клеточного автомата, как в (158).

а) Найдите способ вычисления f с помощью булевой цепочки длиной не более 26 шагов.

б) Пусть $X_j^{(t)}$ обозначает строку j растра X в момент t. Покажите, что $X_j^{(t+1)}$ можно вычислить не более чем за 23 широкословных шага, как функцию трех строк — $X_{j-1}^{(t)}$, $X_j^{(t)}$ и $X_{j+1}^{(t)}$.

▶ **168.** [*23*] Чтобы изображение оставалось конечным, мы можем потребовать, чтобы клеточный автомат размером 3×3 рассматривал растр размером $M \times N$ как *тор*, со "сшитыми" границами: верхней с нижней, а левой с правой. Задача эффективного моделирования действий такого автомата оказывается несколько сложнее: нам необходимо минимизировать обращения к памяти, но каждое новое значение пикселя зависит от старых значений

пикселей со всех сторон. Кроме того, сдвиг битов между соседними словами обычно неудобен, будучи ограниченным емкостью регистра.

Покажите, что эти трудности можно преодолеть, поддерживая массив n-битовых слов A_{jk} для $0 \le j \le M$ и $0 \le k \le N' = \lceil N/(n-2) \rceil$. Если $j \ne M$ и $k \ne 0$, слово A_{jk} должно содержать пиксели строки j и столбцов с $(k-1)(n-2)$ по $k(n-2)+1$ включительно; другие слова A_{Mk} и A_{j0} обеспечивают вспомогательное буферное пространство. (Обратите внимание, что некоторые биты растра встречаются дважды.)

169. [*22*] Продолжая выполнение двух предыдущих упражнений, ответьте, что произойдет с Чеширским котом на рис. 17, (а), если применить к нему правила игры "Жизнь" на торе размером 26×31?

▶ **170.** [*21*] Что получится в результате применения утончающего автомата Гуо–Холла к черному прямоугольнику с M строками и N столбцами? Сколько времени это займет?

171. [*24*] Найдите булеву цепочку длиной ≤ 25 для вычисления локально утончающей функции $g(x_{\rm NW}, x_{\rm N}, x_{\rm NE}, x_{\rm W}, x_{\rm E}, x_{\rm SW}, x_{\rm S}, x_{\rm SE})$ из (159), с или без дополнительных ситуаций из (160).

172. [*M29*] Докажите или опровергните следующее утверждение: если шаблон содержит три черных пикселя, являющихся соседями один другого по ходу короля, то процедура Гуо–Холла, расширенная (160), будет уменьшать его до тех пор, пока ни один из указанных пикселей не сможет быть удален без уничтожения связности.

▶ **173.** [*M30*] Растровые изображения часто требуют очистки от шумов. Например, случайные черные или белые точки могут существенно повлиять на работу утончающего алгоритма для оптического распознавания символов.

Будем говорить, что растр X *закрыт*, если каждый белый пиксель является частью квадрата размером 2×2, состоящего из белых пикселей, и *открыт*, если каждый черный пиксель является частью квадрата размером 2×2, состоящего из черных пикселей. Пусть

$$X^D = \mathop{\&}\{Y \mid Y \supseteq X \text{ и } Y \text{ закрыт}\}; \quad X^L = \mathop{|}\{Y \mid Y \subseteq X \text{ и } Y \text{ открыт}\}.$$

Растр называется *чистым*, если он равен X^{DL} для некоторого X. Мы можем, например, иметь

$X =$; $X^D =$; $X^{DL} =$.

В общем случае X^D "темнее", чем X, в то время как X^L "светлее": $X^D \supseteq X \supseteq X^L$.

а) Докажите, что $(X^{DL})^{DL} = X^{DL}$. *Указание*: из $X \subseteq Y$ вытекает, что $X^D \subseteq Y^D$ и $X^L \subseteq Y^L$.

б) Покажите, что X^D может быть вычислен за один шаг клеточным автоматом 3×3.

174. [*M46*] (М. Мински (M. Minsky) и С. Пейперт (S. Papert).) Существует ли трехмерный сжимающий алгоритм, аналогичный (161)?

175. [*15*] Сколько черных компонентов, связанных ходом *ладьи*, имеется у Чеширского кота?

176. [*M24*] Пусть G представляет собой граф, вершины которого являются черными пикселями заданного растра X, с u — v, когда u и v связаны ходом короля. Пусть G' — соответствующий граф после применения сжимающего преобразования (161). Цель данного упражнения — показать, что количество связных компонентов G' равно количеству компонентов G минус количество изолированных вершин G.

Пусть $N_{(i,j)} = \{(i,j), (i-1,j), (i-1,j+1), (i,j+1)\}$ является пикселем (i,j) вместе с его северным и/или восточным соседями. Пусть $S(v) = \{v' \in G' \mid v' \in N_v\}$ для каждой $v \in G$.

а) Докажите, что $S(v)$ пусто тогда и только тогда, когда v представляет собой изолированную вершину в G.

б) Докажите, что $u' \text{—}^* v'$ в G' (т. е. они находятся в одном и том же компоненте), если $u \text{—} v$ в G, $u' \in S(u)$ и $v' \in S(v)$.

в) Для каждой $v' \in G'$ введем $S'(v') = \{v \in G \mid v' \in N_v\}$. Всегда ли множество $S'(v')$ пустое?

г) Докажите, что $u \text{—}^* v$ в G, если $u' \text{—} v'$ в G', $u \in S'(u')$ и $v \in S'(v')$.

д) Следовательно, существует взаимно однозначное соответствие между нетривиальными компонентами G и компонентами G'.

177. [*M22*] Продолжая выполнять упр. 176, докажите аналогичный результат для белых пикселей.

178. [*20*] Если X представляет собой растр размером $M \times N$, обозначим $M \times (2N+1)$-растр $X \ddagger (X \mid (X \ll 1))$ как X^*. Покажите, что компоненты X^*, связанные ходом короля, связаны также и ходом ладьи, и что растр X^* имеет то же "дерево окруженности" (162), что и X.

▶ **179.** [*34*] Разработайте алгоритм, который строит дерево окруженности данного растра размером $M \times N$, сканируя изображение построчно, как говорилось в разделе. (См. (162) и (163).)

▶ **180.** [*M24*] Вручную оцифруйте гиперболу $y^2 = x^2 + 13$ для $0 < y \le 7$.

181. [*HM20*] Поясните, как подразделить обобщенное коническое сечение (168) с рациональными коэффициентами на монотонные части, чтобы можно было применить алгоритм T.

182. [*M31*] Почему корректно работает трехрегистровый метод (алгоритм T)?

▶ **183.** [*M29*] (Г. Роте (G. Rote).) Поясните, почему алгоритм T может некорректно работать, если ложно условие (v).

▶ **184.** [*M22*] Найдите квадратичную форму $Q'(x, y)$, такую, чтобы при применении алгоритма T к (x', y'), (x, y) и Q' он давал ту же самую границу, что и для (x, y), (x', y') и Q, но в обратном порядке. *Указание*: имеется простой ответ.

▶ **185.** [*23*] Путем упрощения алгоритма T разработайте алгоритм, корректно оцифровывающий прямую линию от (ξ, η) до (ξ', η'), когда ξ, η, ξ' и η' являются рациональными числами.

186. [*HM22*] Для трех заданных комплексных чисел (z_0, z_1, z_2) рассмотрим кривую, вычерчиваемую следующим образом:

$$B(t) \;=\; (1-t)^2 z_0 + 2(1-t)t z_1 + t^2 z_2 \qquad \text{для } 0 \le t \le 1.$$

а) Как себя ведет $B(t)$ при t, близком к 0 или 1?

б) Пусть $S(z_0, z_1, z_2) = \{B(t) \mid 0 \le t \le 1\}$. Докажите, что все точки $S(z_0, z_1, z_2)$ лежат на треугольнике с вершинами z_0, z_1 и z_2 или внутри него.

в) Истинно или ложно следующее утверждение: $S(w + \zeta z_0, w + \zeta z_1, w + \zeta z_2) = w + \zeta S(z_0, z_1, z_2)$?

г) Докажите, что $S(z_0, z_1, z_2)$ является частью прямой линии тогда и только тогда, когда z_0, z_1 и z_2 коллинеарны; в противном случае это часть параболы.

д) Докажите, что если $0 \le \theta \le 1$, то мы имеем рекуррентное соотношение

$$S(z_0, z_1, z_2) \;=\; S\big(z_0, (1-\theta)z_0 + \theta z_1, B(\theta)\big) \cup S\big(B(\theta), (1-\theta)z_1 + \theta z_2, z_2\big).$$

187. [*M29*] Продолжая выполнять упр. 186, покажите, как оцифровать $S(z_0, z_1, z_2)$ с помощью трехрегистрового метода (алгоритм T). Для получения наилучших результатов

оцифровка $S(z_2, z_1, z_0)$ и $S(z_0, z_1, z_2)$ должна давать одинаковые границы, но в противоположных направлениях.

▶ **188.** [*25*] Растровые изображения часто удобно рассматривать с использованием пикселей с *оттенками серого цвета*, а не просто черных или белых. Такие "уровни серого" обычно представляют собой 8-битовые значения в диапазоне от 0 (черный) до 255 (белый); отметим, что соглашение "черный/белый" традиционно *обратное* используемому в 1-битовом случае. Растр размером $m \times n$ с разрешением 600 dpi (точек на дюйм) прекрасно соответствует изображению $(m/8) \times (n/8)$ в оттенках серого с разрешением 75 ppi (пикселей на дюйм), которое получается путем отображения каждого подмассива 8×8 однобитовых пикселей в серый цвет уровня $\lfloor 255(1 - k/64)^{1/\gamma} + \frac{1}{2} \rfloor$, где $\gamma = 1.3$, а k представляет собой число единиц в подмассиве.

Напишите MMIX-программу для преобразования заданного массива $m \times n$ BITMAP в соответствующее изображение $(m/8) \times (n/8)$ GRAYMAP в предположении, что $m = 8m'$ и $n = 64n'$.

189. [*25*] Как эффективно (а) транспонировать и (б) повернуть на 90° против часовой стрелки растр размером 64×64, используя операции с 64-битовыми числами?

190. [*23*] *Шаблон четности* длиной m и шириной n представляет собой матрицу размером $m \times n$, состоящую из нулей и единиц, обладающую тем свойством, что каждый элемент представляет собой сумму "ладейных" соседей, взятую по модулю 2. Например,

$$\begin{matrix} 1\,1 \\ 0\,0 \\ 1\,1 \end{matrix}, \quad \begin{matrix} 0\,0\,1\,1 \\ 0\,1\,0\,0 \\ 1\,1\,0\,1 \\ 0\,1\,0\,1 \end{matrix}, \quad \begin{matrix} 0\,1\,0\,1\,0 \\ 1\,1\,0\,1\,1 \\ 0\,1\,0\,1\,0 \end{matrix}, \quad \begin{matrix} 1\,0\,0 \\ 1\,1\,0 \\ 1\,0\,1 \\ 0\,1\,1 \\ 0\,0\,1 \end{matrix} \quad \text{и} \quad \begin{matrix} 0\,1\,1\,1\,0 \\ 1\,0\,1\,0\,1 \\ 1\,1\,0\,1\,1 \\ 1\,0\,1\,0\,1 \\ 0\,1\,1\,1\,0 \end{matrix}$$

являются шаблонами четности размеров 3×2, 4×4, 3×5, 5×3 и 5×5.

а) Покажите, что если бинарные векторы $\alpha_1, \alpha_2, \ldots, \alpha_m$ являются строками шаблона четности, то $\alpha_2, \ldots, \alpha_m$ могут быть вычислены из верхней строки α_1 с помощью битовых операций. Таким образом, с любого заданного вектора может начинаться не более одного шаблона четности размером $m \times n$.

б) Истинно или ложно утверждение: сумма (по модулю 2) двух шаблонов четности размером $m \times n$ представляет собой шаблон четности.

в) Шаблон четности называется *идеальным*, если он не содержит строки или столбца, полностью состоящего из нулей. Например, три из приведенных выше матриц идеальны, но примеры размером 3×2 и 3×5 таковыми не являются. Покажите, что каждый шаблон четности размером $m \times n$ содержит идеальный шаблон четности в качестве подматрицы. Кроме того, все такие подматрицы имеют один и тот же размер, $m' \times n'$, где $m' + 1$ является делителем $m + 1$, а $n' + 1$ — делителем $n + 1$.

г) Имеется идеальный шаблон четности, первая строка которого имеет вид 0011, но не существует такого шаблона, начинающегося с 01010. Есть ли простой способ определить, является ли данный бинарный вектор верхней строкой идеального шаблона четности?

д) Докажите, что имеется единственный идеальный шаблон четности, начинающийся с $1\overbrace{0\ldots0}^{n-1}$.

191. [*M30*] *Завернутый шаблон четности* аналогичен шаблону четности из упр. 190, за исключением того, что крайний слева и крайний справа элементы каждой строки также являются соседями.

а) Найдите простое соотношение между шаблоном четности шириной n, который начинается с α, и завернутым шаблоном четности шириной $2n + 2$, который начинается с $0\alpha0\alpha^R$.

б) Полиномы Фибоначчи $F_j(x)$ определены рекуррентным соотношением

$$F_0(x) = 0, \qquad F_1(x) = 1, \qquad \text{и} \qquad F_{j+1}(x) = xF_j(x) + F_{j-1}(x) \quad \text{для } j \geq 1.$$

Покажите, что имеется простое соотношение между завернутыми шаблонами четности, которые начинаются с $10\ldots0$ ($N-1$ нулей), и полиномами Фибоначчи по модулю x^N+1. *Указание*: рассмотрите $F_j(x^{-1}+1+x)$ и выполняйте арифметические действия как по модулю 2, так и по модулю x^N+1.

в) Если α является бинарной строкой $a_1\ldots a_n$, введем обозначение $f_\alpha(x) = a_1x + \cdots + a_nx^n$. Покажите, что

$$f_{(\alpha_j 0 \alpha_j^R)}(x) \;=\; (f_\alpha(x) + f_\alpha(x^{-1}))F_j(x^{-1}+1+x) \bmod (x^N+1) \text{ и } \bmod 2,$$

когда $N = 2n+2$, а α_j является строкой j шаблона четности шириной n, начинающегося с α.

г) Следовательно, можно вычислить α_j из α всего лишь за $O(n^2 \log j)$ шагов. *Указания*: см. упр. 4.6.3–26; воспользуйтесь также тождеством $F_{m+n}(x) = F_m(x)F_{n+1}(x) + F_{m-1}(x)F_n(x)$, которое обобщает формулу 1.2.8–(6).

192. [*HM38*] Кратчайший шаблон четности, начинающийся с заданной строки, может оказаться достаточно длинным; например, оказывается, что идеальный шаблон шириной 120, первая строка которого — $10\ldots0$, имеет длину 36 028 797 018 963 966(!). Цель данного упражнения — рассмотреть, как вычислить следующую интересную функцию:

$$c(q) = 1 + \max\{\, m \mid \text{существует идеальный шаблон четности длиной } m \text{ и шириной } q-1\},$$

начальные значения которой $(1, 3, 4, 6, 5, 24, 9, 12, 28)$ для $1 \leq q \leq 9$ легко вычисляются вручную.

а) Охарактеризуйте $c(q)$ алгебраически, используя полиномы Фибоначчи из упр. 191.

б) Поясните, как вычислить $c(q)$, если мы знаем число M, такое, что $c(q)$ делит M, и если мы также знаем простые делители M.

в) Докажите, что $c(2^e) = 3 \cdot 2^{e-1}$ при $e > 0$. *Указание*: $F_{2^e}(y)$ имеет простой вид по модулю 2.

г) Докажите, что, когда q нечетно и не кратно 3, $c(q)$ является делителем $2^{2e}-1$, где e — степень 2 по модулю q. *Указание*: $F_{2^e-1}(y)$ имеет простой вид по модулю 2.

д) Что будет, когда q будет представлять собой нечетное число, кратное 3?

е) Наконец, поясните, как справиться со случаем четного q.

▶ **193.** [*M21*] Покажите, что если существует идеальный шаблон четности $m \times n$ при нечетных m и n, то существует также идеальный шаблон четности $(2m+1) \times (2n+1)$. (Многократное применение этого наблюдения приводит к замысловатым фракталам; например, шаблон размером 5×5 из упр. 190 приводит к рис. 20.)

194. [*M24*] Найдите все $n \leq 383$, для которых существует идеальный шаблон четности размера $n \times n$ с восьмикратной симметрией, как в примере на рис. 20. *Указание*: диагональные элементы всех таких шаблонов должны быть нулевыми.

Рис. 20. Идеальный 383×383-шаблон четности.

▶ **195.** [*HM25*] Пусть A представляет собой бинарную матрицу со строками $\alpha_1, \ldots, \alpha_m$ длиной n. Поясните, как воспользоваться битовыми операциями для вычисления ранга $m-r$ матрицы A над бинарным полем $\{0,1\}$ и как найти линейно независимые бинарные векторы $\theta_1, \ldots, \theta_r$ длиной m, такие, что $\theta_j A = 0\ldots0$ для $1 \leq j \leq r$. *Указание*: см. алгоритм триангуляризации 4.6.2N.

196. [*21*] (К. Томпсон (K. Thompson), 1992.) Целые числа из диапазона $0 \le x < 2^{31}$ можно закодировать в виде строки $\alpha(x) = \alpha_1 \ldots \alpha_l$ длиной до 6 байт следующим образом: если $x < 2^7$, установить $l \leftarrow 1$ и $\alpha_1 \leftarrow x$. В противном случае пусть $x = (x_5 \ldots x_1 x_0)_{64}$; установить $l \leftarrow \lceil(\lambda x)/5\rceil$, $\alpha_1 \leftarrow 2^8 - 2^{8-l} + x_{l-1}$ и $\alpha_j \leftarrow 2^7 + x_{l-j}$ для $2 \le j \le l$. Обратите внимание, что $\alpha(x)$ содержит нулевой байт тогда и только тогда, когда $x = 0$.

а) Что собой представляют коды для #a, #3a3, #7b97 и #1d141?

б) Докажите, что если $x \le x'$, то $\alpha(x) \le \alpha(x')$ в лексикографическом порядке.

в) Предположим, что последовательность значений $x^{(1)}x^{(2)}\ldots x^{(n)}$ закодирована в виде байтовой строки $\alpha(x^{(1)})\alpha(x^{(2)})\ldots\alpha(x^{(n)})$, и пусть α_k представляет собой k-й байт этой строки. Покажите, что легко найти значение $x^{(i)}$, из которого получено α_k, путем просмотра при необходимости нескольких соседних байтов.

197. [*22*] Универсальный набор символов (Universal Character Set — UCS), известный также как Unicode, представляет собой стандартное отображение символов на целочисленные кодовые точки x в диапазоне $0 \le x < 2^{20} + 2^{16}$. Кодировка, известная как UTF-16, представляет такие целые числа как один или два двухбайтовых слова ("дуэта") $\beta(x) = \beta_1$ или $\beta(x) = \beta_1\beta_2$ следующим образом: если $x < 2^{16}$, то $\beta(x) = x$; в противном случае

$$\beta_1 = {}^{\#}\texttt{d800} + \lfloor y/2^{10}\rfloor \text{ и } \beta_2 = {}^{\#}\texttt{dc00} + (y \bmod 2^{10}), \text{ где } y = x - 2^{16}.$$

Ответьте на вопросы (а)–(в) упр. 196 для данной кодировки.

▶ **198.** [*21*] Символы Unicode часто представляются в виде строк байтов с использованием схемы под названием "UTF-8", которая представляет собой кодировку из упр. 196, ограниченную целыми числами из диапазона $0 \le x < 2^{20} + 2^{16}$. Обратите внимание, что UTF-8 сохраняет нетронутым стандартный набор символов ASCII (кодовые точки с $x < 2^7$) и тем самым существенно отличается от UTF-16.

Пусть α_1 является первым байтом строки UTF-8 $\alpha(x)$. Покажите, что существуют достаточно небольшие целочисленные константы a, b и c, такие, что достаточно только четырех битовых операций

$$(a \gg ((\alpha_1 \gg b)\ \&\ c))\ \&\ 3$$

для определения количества $l - 1$ байт между α_1 и концом $\alpha(x)$.

▶ **199.** [*23*] Можно попытаться закодировать в UTF-8 #a как #c08a или как #e0808a, или даже как #f080808a, поскольку очевидный алгоритм декодирования даст в каждом случае один и тот же результат. Но такой излишне длинный код некорректен, поскольку может привести к "дырам" в системе безопасности.

Предположим, что α_1 и α_2 представляют собой байты, такие, что $\alpha_1 \ge {}^{\#}\texttt{80}$ и ${}^{\#}\texttt{80} \le \alpha_2 < {}^{\#}\texttt{c0}$. Найдите не использующий ветвлений способ определить, являются ли α_1 и α_2 первыми двумя байтами как минимум одной корректной строки UTF-8 $\alpha(x)$.

200. [*20*] Поясните, что будет находиться в регистре \$3 после выполнения следующих трех команд MMIX: `MOR $1,$0,#94; MXOR $2,$0,#94; SUBU $3,$1,$2`.

201. [*20*] Предположим, что $x = (x_{15} \ldots x_1 x_0)_{16}$ состоит из шестнадцати шестнадцатеричных цифр. Какая единственная команда MMIX заменит каждую ненулевую цифру на f, оставляя нули нетронутыми?

202. [*20*] Какие две команды заменят ненулевые двухбайтовые дуэты октета на #ffff?

203. [*22*] Предположим, что мы хотим преобразовать тетрабайт $x = (x_7 \ldots x_1 x_0)_{16}$ в октабайт $y = (y_7 \ldots y_1 y_0)_{256}$, где y_j представляет собой ASCII-код шестнадцатеричной цифры x_j. Например, если $x = {}^{\#}\texttt{1234abcd}$, y должно представлять 8-символьную строку '1234abcd'. Какой очевидный выбор констант a, b, c, d и e заставит приведенные далее

команды MMIX выполнить требуемую работу?

```
MOR t,x,a;  SLU s,t,4;  XOR t,s,t;  AND t,t,b
ADD t,t,c;  MOR s,d,t;  ADD t,t,e;  ADD y,t,s
```

▶ **204.** [*22*] Каковы поразительные константы p, q, r и m, которые позволяют выполнить идеальное тасование с помощью всего лишь шести команд MMIX? (См. (175)–(178).)

▶ **205.** [*22*] Как выполнить операцию, *обратную* идеальному тасованию, на MMIX, возвращаясь от w из (175) назад к z?

206. [*20*] Идеальное тасование (175) иногда называют внешним (outshuffle), в противоположность внутреннему (inshuffle), которое выполняет отображение $z \mapsto y \ddagger x = (y_{31}x_{31} \ldots y_1x_1y_0x_0)_2$; внешнее тасование сохраняет крайние слева и справа биты z, но внутреннее тасование не имеет фиксированных точек. Можно ли выполнить внутреннее тасование столь же эффективно, как и внешнее?

207. [*22*] Используйте MOR для выполнения тройного идеального тасования, которое отображает $(x_{63} \ldots x_0)_2$ в $(x_{21}x_{42}x_{63}x_{20} \ldots x_2x_{23}x_{44}x_1x_{22}x_{43}x_0)_2$, а также обратного к нему преобразования.

▶ **208.** [*23*] Как на MMIX быстро выполнить транспонирование булевой матрицы размером 8×8?

▶ **209.** [*21*] Легко ли выполнить операцию вычисления суффиксной четности $x^{\oplus}$ из упр. 36 с помощью MXOR?

210. [*22*] Головоломка: регистр x содержит число $8j + k$, где $0 \le j, k < 8$. Регистры a и b содержат произвольные октеты $(a_7 \ldots a_1a_0)_{256}$ и $(b_7 \ldots b_1b_0)_{256}$. Найдите последовательность четырех команд MMIX, которые поместят $a_j \mathbin{\&} b_k$ в регистр x.

▶ **211.** [*M25*] Таблица истинности булевой функции $f(x_1, \ldots, x_6)$, по сути, представляет собой 64-битовое число $f = \bigl(f(0,0,0,0,0,0) \ldots f(1,1,1,1,1,0)f(1,1,1,1,1,1)\bigr)_2$. Покажите, что две команды MOR преобразуют f в таблицу истинности наименьшей монотонной булевой функции, $\hat{f}$, которая не меньше f в каждой точке.

212. [*M32*] Предположим, что $a = (a_{63} \ldots a_1a_0)_2$ представляет полином

$$a(x) = (a_{63} \ldots a_1a_0)_x = a_{63}x^{63} + \cdots + a_1x + a_0.$$

Рассмотрите применение MXOR для вычисления произведения $c(x) = a(x)b(x)$ по модулю x^{64} и модулю 2.

▶ **213.** [*HM26*] Реализуйте процедуру CRC (183) на машине MMIX.

▶ **214.** [*HM28*] (Р. В. Госпер (R. W. Gosper).) Найдите короткий, без ветвлений, способ обращения на машине MMIX любой заданной матрицы размером 8×8, состоящей из нулей и единиц, по модулю 2, если ее определитель $\det X$ нечетен.

▶ **215.** [*21*] Как на машине MMIX быстро проверить, делится ли заданное 64-битовое число на 3?

▶ **216.** [*M26*] Для заданных n-битовых целых чисел $x_1, \ldots, x_m \ge 0$, $n \ge \lambda m$, за $O(m)$ шагов вычислите наименьшее $y > 0$, такое, что $y \notin \{a_1x_1 + \cdots + a_mx_m \mid a_1, \ldots, a_m \in \{0,1\}\}$, если λx требует единичного времени.

217. [*40*] Исследуйте упаковку длинных строк текста: представьте 64 последовательных символа как последовательность 8 октабайт $w_0 \ldots w_7$, где w_k содержит все 64 их k-х бита.

▶ **218.** [*M30*] (Ганс Петтер Селаски (Hans Petter Selasky), 2009.) Разработайте для фиксированного $d \ge 3$ алгоритм вычисления $a \cdot x^y \bmod 2^d$ для заданных целых a, x и y, где x нечетно, использующий $O(d)$ сложений и битовых операций, а также единственное умножение на y.

В распространенном использовании термин **БДР** *почти всегда означает*
"приведенная упорядоченная бинарная диаграмма решений"
(в литературе это название употребляется, когда необходимо подчеркнуть ее упорядоченность и сжатость).

— Wikipedia, *The Free Encyclopedia* (7 июля 2007 года)

7.1.4. Бинарные диаграммы решений

Обратимся теперь к важному семейству структур данных, которые быстро стали распространенным средством представления и работы с булевыми функциями в компьютерах. Основная идея заключается в схеме "разделяй и властвуй", чем-то похожей на бинарные лучи из раздела 6.3, но с рядом новых особенностей.

На рис. 21 показана бинарная диаграмма решений для простой булевой функции от трех переменных, а именно для медианы $\langle x_1x_2x_3\rangle$ из 7.1.1–(43). Ее можно понимать следующим образом. Верхний узел называется *корнем*. Каждый внутренний узел (j), именуемый также *узлом ветвления*, помечен именем или индексом $j = V((j))$, который обозначает переменную; например, корневой узел (1) на рис. 21 обозначает x_1. Узлы ветвления имеют по два наследника, указываемые нисходящими линиями. Один из наследников изображается с помощью пунктирной линии и называется LO ("нижний", "младший"); другой изображается с помощью сплошной линии и называется HI ("верхний", "старший"). Эти узлы ветвления определяют путь в диаграмме для любых значений булевых переменных, если начать с корня и двигаться из узла (j) по пути LO при $x_j = 0$ и по пути HI при $x_j = 1$. В конечном итоге этот путь заканчивается в *узле стока*, который представляет собой либо $\boxed{\bot}$ (обозначение для FALSE (ЛОЖЬ)), либо $\boxed{\top}$ (обозначение для TRUE (ИСТИНА)). Легко убедиться, что на рис. 21 этот процесс дает значение FALSE, когда как минимум две из переменных $\{x_1, x_2, x_3\}$ равны 0, в противном случае он дает TRUE.

Рис. 21. Бинарная диаграмма решений (БДР) для функции мажоризации $\langle x_1x_2x_3\rangle$.

Многие авторы для обозначения узлов стока используют $\boxed{0}$ и $\boxed{1}$. Мы применяем $\boxed{\bot}$ и $\boxed{\top}$, чтобы избежать малейшей возможности спутать их с узлами ветвления (0) и (1).

В компьютере рис. 21 может быть представлен как множество из четырех узлов в произвольных ячейках памяти, где каждый узел имеет три поля: $\boxed{\texttt{V} \mid \texttt{LO} \mid \texttt{HI}}$. Поле V хранит индекс переменной, а каждое из полей LO и HI указывает на другой узел или сток.

(1)

В случае 64-битовых слов можно, например, использовать 8 бит для V и по 28 бит для LO и HI.

Такая структура называется бинарной диаграммой решений или, для краткости, БДР. Небольшие БДР легко начертить на листе бумаги или на экране компьютера, как обычные диаграммы. Но по сути каждая БДР является абстрактным множеством связанных узлов, которое, пожалуй, более правильно было бы называть бинарным ациклическим ориентированным графом решений — бинарным деревом с общими поддеревьями, ациклическим ориентированным графом, в котором из каждого нестокового узла выходят ровно две дуги.

Мы будем считать, что каждая БДР подчиняется двум важным ограничениям. Во-первых, она *упорядочена*: когда дуга LO или HI идет из узла ветвления (i) в узел ветвления (j), то должно выполняться $i < j$. Таким образом, в частности, никакая переменная x_j не будет запрошена дважды при вычислении функции. Во-вторых, БДР должна быть *приведенной* в том смысле, что она не расходует память понапрасну. Это означает, что указатели узла ветвления LO и HI не могут быть равны и что никакие два узла не могут иметь одинаковые тройки значений (V, LO, HI). Каждый узел должен также быть доступным из корня. Например, диаграммы

(2)

не являются БДР, поскольку первая из них не является упорядоченной, а вторая не является приведенной.

Были разработаны многие другие разновидности диаграмм решений, и в компьютерной литературе можно встретить массу разнообразных аббревиатур наподобие EVBDD, FBDD, IBDD, OBDD, OFDD, OKFDD, PBDD, ..., ZDD. В этой книге, если явно не оговорено иное, мы всегда будем понимать под БДР упорядоченную и приведенную, как описано выше, бинарную диаграмму решений и использовать простую и незатейливую аббревиатуру “БДР” (“BDD”), так же как используем слово “дерево” для обозначения упорядоченного (плоского) дерева, поскольку такие БДР и такие деревья наиболее распространены на практике.

Вспомним из раздела 7.1.1, что каждая булева функция $f(x_1, \ldots, x_n)$ соответствует *таблице истинности*, которая представляет собой 2^n-битовую бинарную строку, которая начинается со значения функции $f(0, \ldots, 0)$ и продолжается значениями функции $f(0, \ldots, 0, 1)$, $f(0, \ldots, 0, 1, 0)$, $f(0, \ldots, 0, 1, 1)$, ..., $f(1, \ldots, 1, 1, 1)$. Например, таблица истинности функции медианы $\langle x_1 x_2 x_3 \rangle$ равна 00010111. Обратите внимание, что эта таблица совпадает с последовательностью листьев в неприведенном дереве решений (2) с $0 \mapsto \boxed{\bot}$ и $1 \mapsto \boxed{\top}$. На самом деле это не совпадение, а важное соотношение между таблицами истинности и БДР, которое проще всего понять в терминах класса бинарных строк, именуемых “бусинами” (“beads”).

Таблица истинности порядка n представляет собой бинарную строку длиной 2^n. *Бусиной* порядка n является таблица истинности β порядка n, которая не является квадратом, т. е. β не имеет вид $\alpha\alpha$ для некоторой строки α длиной 2^{n-1}. (Математики бы сказали, что бусина является “примитивной строкой длиной 2^n”.) Имеется

две бусины порядка 0, а именно 0 и 1; и две бусины порядка 1, а именно 01 и 10. В общем случае имеется $2^{2^n}-2^{2^{n-1}}$ бусин порядка n при $n>0$, поскольку существует 2^{2^n} бинарных строк длиной 2^n и $2^{2^{n-1}}$ из них являются квадратами. $16-4=12$ бусинами порядка 2 являются

$$0001, 0010, 0011, 0100, 0110, 0111, 1000, 1001, 1011, 1100, 1101, 1110; \tag{3}$$

это также таблицы истинности всех функций $f(x_1,x_2)$, которые зависят от x_1 в том смысле, что функция $f(0,x_2)$ отлична от функции $f(1,x_2)$.

Каждая таблица истинности τ является степенью единственной бусины, именуемой ее корнем. Если τ имеет длину 2^n и не является бусиной, то она представляет собой квадрат другой таблицы истинности τ'; а по индукции по длине τ мы должны иметь $\tau'=\beta^k$ для некоторого корня β. Следовательно, $\tau=\beta^{2k}$, и β является корнем как τ, так и τ'. (Конечно же, k является степенью 2.)

Таблица истинности τ порядка $n>0$ всегда имеет вид $\tau_0\tau_1$, где τ_0 и τ_1 являются таблицами истинности порядка $n-1$. Ясно, что τ представляет функцию $f(x_1,x_2,\ldots,x_n)$ тогда и только тогда, когда τ_0 представляет $f(0,x_2,\ldots,x_n)$, а τ_1 представляет $f(1,x_2,\ldots,x_n)$. Эти функции $f(0,x_2,\ldots,x_n)$ и $f(1,x_2,\ldots,x_n)$ называются *подфункциями* f; а их таблицы истинности, τ_0 и τ_1, называются *подтаблицами* τ.

Подтаблицы подтаблиц также рассматриваются как подтаблицы, в том числе таблица рассматривается как подтаблица самой себя. Таким образом, в общем случае таблица истинности порядка n имеет 2^k подтаблиц порядка $n-k$ для $0\le k\le n$, соответствующих 2^k возможным установкам первых k переменных $(x_1,\ldots,x_k)$. Многие из этих подтаблиц часто оказываются идентичными; в таких случаях можно представить τ в сжатом виде.

Бусами булевой функции являются подтаблицы ее таблицы истинности, которые оказываются бусинами. В качестве примера вновь рассмотрим функцию медианы $\langle x_1x_2x_3\rangle$ с ее таблицей истинности 00010111. Различными подтаблицами этой таблицы истинности являются $\{00010111, 0001, 0111, 00, 01, 11, 0, 1\}$; и все они, за исключением 00 и 11, представляют собой бусины. Следовательно, бусами $\langle x_1x_2x_3\rangle$ являются

$$\{00010111, 0001, 0111, 01, 0, 1\}. \tag{4}$$

А теперь перейдем к сути дела: *узлы БДР булевой функции взаимно однозначно соответствуют ее бусам.* Например, можно перерисовать рис. 21, помещая в каждый узел соответствующую бусину:

. (5)

В общем случае таблицы истинности порядка $n+1-k$ функции соответствуют ее подфункциям $f(c_1,\ldots,c_{k-1},x_k,\ldots,x_n)$ указанного порядка; так что бусы функции порядка $n+1-k$ соответствуют тем подфункциям, которые зависят от их первой переменной, x_k. Следовательно, каждая такая бусина соответствует узлу ветвления (k) в БДР. А если (k) представляет собой узел ветвления, соответствующий

таблице истинности $\tau' = \tau'_0\tau'_1$, его ветви LO и HI указывают на узлы, которые соответствуют корням τ'_0 и τ'_1.

Это соответствие между бусами и узлами доказывает, что *каждая булева функция имеет одно и только одно представление в виде БДР.* Отдельные узлы такой БДР могут, конечно, располагаться в разных местах в памяти компьютера.

Если f является некоторой булевой функцией, обозначим через $B(f)$ количество бусин, которые она имеет. Это значение представляет собой размер ее БДР — общее количество узлов, включая стоки. Например, $B(f) = 6$ в случае, когда f представляет собой функцию медианы трех элементов, поскольку (5) имеет размер 6.

Для определенности давайте разберем другой пример, "более или менее случайную" функцию из 7.1.1–(22) и 7.1.2–(6). Ее таблица истинности, 1100100100001111, является бусиной, и то же относится к ее подтаблицам 11001001 и 00001111. Таким образом, мы знаем, что корнем ее БДР будет ветвь (1) и что оба узла, LO и HI, под корнем являются (2). Подтаблицы длиной 4 имеют вид $\{1100, 1001, 0000, 1111\}$; первые две из них являются бусинами, а вторые две — квадратами. Чтобы перейти к следующему уровню, мы разбиваем бусины пополам и переносим квадратные корни не бусин, идентифицируя дубли; это дает $\{11, 00, 10, 01\}$. И вновь у нас имеются две бусины, и последний шаг приводит к искомой БДР:

(6)

(На этой диаграмме и на прочих диаграммах, приведенных ниже, оказывается удобным повторять стоковые узлы $\boxed{\bot}$ и $\boxed{\top}$, чтобы избежать слишком длинных соединяющих линий. В действительности же в БДР присутствуют только один узел $\boxed{\bot}$ и один узел $\boxed{\top}$; так что размер (6) равен 9, а не 13.)

Внимательный читатель в этом месте может задуматься: "Все это хорошо, но что если БДР окажется огромной?" Действительно, можно легко построить функции, БДР которых будут до невозможности огромными; позже мы изучим такие случаи. Но удивительно то, что очень много важных с практической точки зрения булевых функций имеют достаточно небольшие значения $B(f)$. Так что мы должны сосредоточиться на этой хорошей новости и оставить плохие до тех пор, пока не поймем, почему БДР столь популярны.

Достоинства БДР. Если $f(x) = f(x_1, \ldots, x_n)$ представляет собой булеву функцию, БДР которой достаточно мала, мы можем выполнять множество действий быстро и просто. Например, так.

• Мы можем *вычислить* $f(x)$ не более чем за n шагов для любого заданного входного вектора $x = x_1 \ldots x_n$, просто начав с корня и выполняя ветвления, пока не достигнем стока.

• Мы можем *найти лексикографически наименьшее* x, такое, что $f(x) = 1$, начав с корня и многократно поворачивая на ветвь LO, если только она не ведет непосредственно в $\boxed{\bot}$. Решение имеет $x_j = 1$, только когда в (j) необходимо свернуть

на ветвь HI. Например, эта процедура дает $x_1x_2x_3 = 011$ в случае БДР, показанной на рис. 21, и $x_1x_2x_3x_4 = 0000$ в случае (6). (Метод определяет значение x, соответствующее крайней слева 1 в таблице истинности f.) Для поиска требуется всего лишь n шагов, поскольку каждый узел ветвления соответствует ненулевой бусине; мы всегда можем найти нисходящий путь к $\boxed{\top}$ без возврата. Конечно, этот метод не работает, если корень сам по себе является $\boxed{\bot}$. Но это случается только тогда, когда f тождественно равна нулю.

• Мы можем *подсчитать количество решений* уравнения $f(x) = 1$, используя приведенный ниже алгоритм C. Этот алгоритм выполняет $B(f)$ операций с n-битовыми числами; так что время его работы в худшем случае составляет $O(nB(f))$.

• После работы алгоритма C можно быстро *сгенерировать случайные решения* уравнения $f(x) = 1$ таким образом, что каждое решение окажется равновероятным.

• Можно также *сгенерировать все решения* x уравнения $f(x) = 1$. Алгоритм из упр. 16 делает это за $O(nN)$ шагов при наличии N решений.

• Мы можем *решить задачу линейного булева программирования*: найти x, такое, что

$$w_1x_1 + \cdots + w_nx_n \text{ максимально, при условии } f(x_1, \ldots, x_n) = 1, \tag{7}$$

для заданных констант $(w_1, \ldots, w_n)$. Приведенный ниже алгоритм B решает эту задачу за $O(n + B(f))$ шагов.

• Мы можем *вычислить производящую функцию* $a_0 + a_1z + \cdots + a_nz^n$, где a_j — количество решений $f(x_1, \ldots, x_n) = 1$, таких, что $x_1 + \cdots + x_n = j$. (См. упр. 25.)

• Мы можем *вычислить полином надежности* $F(p_1, \ldots, p_n)$, который представляет собой вероятность того, что $f(x_1, \ldots, x_n) = 1$, когда каждое x_j независимо устанавливается равным 1 с заданной вероятностью p_j. В упр. 26 это делается за $O(B(f))$ шагов.

Кроме того, мы увидим, что бинарные диаграммы решений могут эффективно объединяться и модифицироваться. Например, несложно образовать бинарные диаграммы решений для $f(x_1, \ldots, x_n) \wedge g(x_1, \ldots, x_n)$ и $f(x_1, \ldots, x_{j-1}, g(x_1, \ldots, x_n), x_{j+1}, \ldots, x_n)$ из бинарных диаграмм решений для f и g.

Алгоритмы для решения фундаментальных задач с использованием бинарных диаграмм решений часто описываются наиболее просто, если задать бинарную диаграмму решений как последовательный список инструкций ветвления $I_{s-1}, I_{s-2}, \ldots, I_1, I_0$, где каждое I_k имеет вид $(\bar{v}_k ?\, l_k : h_k)$. Например, (6) можно представить как список из $s = 9$ инструкций,

$$\begin{array}{lll} I_8 = (\bar{1}?\, 7{:}\,6), & I_5 = (\bar{3}?\, 1{:}\,0), & I_2 = (\bar{4}?\, 0{:}\,1), \\ I_7 = (\bar{2}?\, 5{:}\,4), & I_4 = (\bar{3}?\, 3{:}\,2), & I_1 = (\bar{5}?\, 1{:}\,1), \\ I_6 = (\bar{2}?\, 0{:}\,1), & I_3 = (\bar{4}?\, 1{:}\,0), & I_0 = (\bar{5}?\, 0{:}\,0), \end{array} \tag{8}$$

с $v_8 = 1$, $l_8 = 7$, $h_8 = 6$, $v_7 = 2$, $l_7 = 5$, $h_7 = 4$, $\ldots$, $v_0 = 5$, $l_0 = h_0 = 0$. В общем случае инструкция '$(\bar{v}?\, l{:}\, h)$' означает "если $x_v = 0$, перейти к I_l, в противном случае перейти к I_h", за исключением особенных последних инструкций I_1 и I_0. Мы требуем, чтобы LO- и HI-ветви l_k и h_k удовлетворяли условиям

$$l_k < k, \qquad h_k < k, \qquad v_{l_k} > v_k \quad \text{и} \quad v_{h_k} > v_k \qquad \text{для } s > k \geq 2; \tag{9}$$

другими словами, все ветви идут вниз, к переменным с бóльшими индексами. Но стоковые узлы $\boxed{\top}$ и $\boxed{\bot}$ представлены фиктивными инструкциями I_1 и I_0, в которых $l_k = h_k = k$, а "индекс переменной" v_k имеет невозможное значение $n+1$.

Эти инструкции могут быть пронумерованы любым способом, который не нарушает топологическое упорядочение бинарной диаграммы решений, что требуется в (9). Корневой узел должен соответствовать I_{s-1}, а стоковые узлы должны соответствовать I_1 и I_0, но другие номера индексов не являются предписанными так строго. Например, (6) можно выразить и как

$$\begin{aligned} I'_8 &= (\bar{1}?\,7{:}\,2), & I'_5 &= (\bar{4}?\,0{:}\,1), & I'_2 &= (\bar{2}?\,0{:}\,1),\\ I'_7 &= (\bar{2}?\,4{:}\,6), & I'_4 &= (\bar{3}?\,1{:}\,0), & I'_1 &= (\bar{5}?\,1{:}\,1),\\ I'_6 &= (\bar{3}?\,3{:}\,5), & I'_3 &= (\bar{4}?\,1{:}\,0), & I'_0 &= (\bar{5}?\,0{:}\,0) \end{aligned} \tag{10}$$

и еще 46 другими изоморфными способами. В компьютере бинарная диаграмма решений не обязана располагаться в последовательных ячейках памяти; мы можем легко обходить узлы любого ациклического ориентированного графа в топологическом порядке, когда узлы связаны так, как в (1). Но мы будем представлять, что они расположены последовательно, как в (8), чтобы проще было понимать работу различных алгоритмов.

Стоит отметить одну формальность: если $f(x) = 1$ для всех x, так что бинарная диаграмма решений представляет собой просто стоковый узел $\boxed{\top}$, в таком последовательном представлении мы полагаем $s = 2$. В противном случае s равно размеру бинарной диаграммы решений. Тогда корень всегда представлен как I_{s-1}.

Алгоритм C (*Подсчет решений*). Для заданной бинарной диаграммы решений булевой функции $f(x) = f(x_1, \ldots, x_n)$, представленной в виде последовательности $I_{s-1}, \ldots, I_0$, как описано выше, этот алгоритм определяет $|f|$, количество бинарных векторов $x = x_1 \ldots x_n$, таких, что $f(x) = 1$. Он также вычисляет таблицу $c_0, c_1, \ldots, c_{s-1}$, где c_k представляет собой количество единиц в бусине, соответствующей I_k.

C1. [Цикл по k.] Установить $c_0 \leftarrow 0$, $c_1 \leftarrow 1$ и выполнить шаг C2 для $k = 2, 3, \ldots, s-1$. Затем вернуть ответ $2^{v_{s-1}-1} c_{s-1}$.

C2. [Вычисление c_k.] Установить $l \leftarrow l_k$, $h \leftarrow h_k$ и $c_k \leftarrow 2^{v_l - v_k - 1} c_l + 2^{v_h - v_k - 1} c_h$. ▮

Например, для представления (8) этот алгоритм вычисляет

$$c_2 \leftarrow 1,\ c_3 \leftarrow 1,\ c_4 \leftarrow 2,\ c_5 \leftarrow 2,\ c_6 \leftarrow 4,\ c_7 \leftarrow 4,\ c_8 \leftarrow 8;$$

общее количество решений $f(x_1, x_2, x_3, x_4) = 1$ равно 8.

Целые числа c_k в алгоритме C удовлетворяют условию

$$0 \le c_k < 2^{n+1-v_k} \qquad \text{для } 2 \le k < s, \tag{11}$$

и эта верхняя граница — наилучшая возможная. Следовательно, при бóльших n может потребоваться арифметика с многократной точностью. Если дополнительная память для высокой точности проблематична, можно использовать вместо нее модулярную арифметику, выполняя алгоритм несколько раз и вычисляя $c_k \bmod p$ для разных простых чисел p с однократной точностью; затем окончательный ответ можно вывести с помощью алгоритма, основанного на китайской теореме об остатках (4.3.2–(24)). С другой стороны, на практике обычно достаточно арифметики с плавающей точкой.

Давайте взглянем на несколько примеров, более интересных, чем (6). Бинарные диаграммы решений

Независимые множества Ядра (12)

представляют функции от шести переменных, которые соответствуют подмножествам вершин в циклическом графе C_6. В этом случае такой вектор, как $x_1 \ldots x_6 = 100110$, обозначает подмножество $\{1, 4, 5\}$; вектор 000000 обозначает пустое подмножество; и т. д. Слева показана БДР, для которой мы имеем $f(x) = 1$, когда x *независимы* в C_6; справа показана БДР для *максимальных* независимых подмножеств, именуемых также *ядрами* C_6 (см. упр. 12). В общем случае независимые подмножества C_n соответствуют расположениям нулей и единиц в окружности длиной n, в которых нет двух рядом стоящих единиц; ядра соответствуют таким размещениям, в которых, кроме того, нет трех последовательных нулей.

Алгоритм C "украшает" БДР значениями c_k снизу вверх, где c_k представляет собой число способов перехода от узла k к $\boxed{\top}$ путем выбора значений $x_l \ldots x_n$, где l представляет собой метку узла k. При применении этого алгоритма к БДР в (12) мы получим

; (13)

следовательно, C_6 имеет 18 независимых множеств и 5 ядер.

Эти величины делают генерацию равномерно распределенных *случайных* решений очень простой задачей. Например, чтобы получить случайный вектор независимого множества $x_1 \ldots x_6$, воспользуемся знанием о том, что 13 решений находятся в левой части БДР и имеют $x_1 = 0$, а прочие 5 имеют $x_1 = 1$. Так что мы устанавливаем $x_1 \leftarrow 0$ с вероятностью 13/18 и движемся по ветви LO; в противном случае мы устанавливаем $x_1 \leftarrow 1$ и движемся по ветви HI. В последнем случае $x_1 = 1$ приводит к $x_2 \leftarrow 0$, но затем x_3 может быть любым.

Предположим, что мы выбрали установки $x_1 \leftarrow 1$, $x_2 \leftarrow 0$, $x_3 \leftarrow 0$ и $x_4 \leftarrow 0$; эта ситуация возникает с вероятностью $\frac{5}{18} \cdot \frac{5}{5} \cdot \frac{3}{5} \cdot \frac{2}{3} = \frac{2}{18}$. Затем имеется ветвь от (4) к (6), так что мы можем подбросить монетку и установить x_5 совершенно случайным образом. В общем случае ветвь от (i) в (j) означает, что $j - i - 1$ промежуточных битов $x_{i+1} \ldots x_{j-1}$ должны независимо устанавливаться равными 0 или 1 с равной вероятностью. Аналогично ветвь от (i) к $\boxed{\top}$ должна назначать случайные значения для $x_{i+1} \ldots x_n$.

Конечно, имеются и более простые способы случайного выбора из 18 решений комбинаторной задачи. Более того, правая бинарная диаграмма решений в (13) представляет собой весьма сложный способ представления пяти ядер C_6: ведь можно просто перечислить их, 001001, 010010, 010101, 100100, 101010! Но суть заключается в том, что один и тот же метод дает как независимые множества, так и ядра C_n при гораздо бóльших n. Например, цикл C_{100} длиной 100 имеет 1 630 580 875 002 ядра, но при этом описывающая их БДР имеет только 855 узлов. Так что совершенно случайное ядро из этой неимоверной коллекции получается всего за 100 шагов.

Булево программирование и связанные задачи. Восходящий алгоритм, аналогичный алгоритму C, способен также найти наилучшие *взвешенные* решения (7) булева уравнения $f(x) = 1$. Основная идея состоит в том, что легко вывести оптимальное решение для любой бусины f, если известны оптимальные решения для бусин LO и HI, лежащих непосредственно под рассматриваемой.

Алгоритм B (*Решения наибольшего веса*). Пусть $I_{s-1}, \ldots, I_0$ представляет собой последовательность инструкций ветвления, которая представляет БДР для булевой функции f, как в алгоритме C, и пусть $(w_1, \ldots, w_n)$ представляет собой произвольную последовательность целочисленных весов. Данный алгоритм находит бинарный вектор $x = x_1 \ldots x_n$, такой, что значение $w_1x_1 + \cdots + w_nx_n$ максимально среди всех x, обладающих тем свойством, что $f(x) = 1$. Мы считаем, что $s > 1$; в противном случае функция $f(x)$ тождественно равна 0. Вспомогательные целочисленные векторы $m_1 \ldots m_{s-1}$ и $W_1 \ldots W_{n+1}$ используются для вычислений, как и вспомогательный битовый вектор $t_2 \ldots t_{s-1}$.

B1. [Инициализация.] Установить $W_{n+1} \leftarrow 0$ и $W_j \leftarrow W_{j+1} + \max(w_j, 0)$ для $n \geq j \geq 1$.

B2. [Цикл по k.] Установить $m_1 \leftarrow 0$ и выполнить шаг B3 для $2 \leq k < s$. Затем выполнить шаг B4.

B3. [Обработка I_k.] Установить $v \leftarrow v_k$, $l \leftarrow l_k$, $h \leftarrow h_k$, $t_k \leftarrow 0$. Если $l \neq 0$, установить $m_k \leftarrow m_l + W_{v+1} - W_{v_l}$. Затем, если $h \neq 0$, выполнить следующее: вычислить $m \leftarrow m_h + W_{v+1} - W_{v_h} + w_v$; и если $l = 0$ или $m > m_k$, установить $m_k \leftarrow m$ и $t_k \leftarrow 1$.

B4. [Вычисление x.] Установить $j \leftarrow 0$, $k \leftarrow s-1$ и выполнять следующие операции до тех пор, пока не будет достигнуто равенство $j = n$: пока $j < v_k - 1$, устанавливать $j \leftarrow j + 1$ и $x_j \leftarrow [w_j > 0]$; если $k > 1$, установить $j \leftarrow j + 1$ и $x_j \leftarrow t_k$, и $k \leftarrow (t_k = 0?\ l_k\colon h_k)$. ▮

Простой случай этого алгоритма разработан в упр. 18. На шаге B3 выполняются технические действия, которые могут выглядеть пугающе, но их конечный эффект

заключается просто в вычислении

$$m_k \leftarrow \max(m_l + W_{v+1} - W_{v_l},\, m_h + W_{v+1} - W_{v_h} + w_v) \tag{14}$$

и в записи в t_k, что лучше — l или h. В действительности обычно и v_l, и v_h равны $v+1$; тогда вычисления просто устанавливают $m_k \leftarrow \max(m_l, m_h + w_v)$, соответствующие случаям $x_v = 0$ и $x_v = 1$. Все это связано с нашим желанием избежать выборки значения m_0, равного $-\infty$, и с тем, что v_l или v_h может превысить $v+1$.

С этим алгоритмом мы можем, например, быстро найти оптимальное множество вершин ядра в n-цикле C_n, используя веса, основанные на последовательности "Тью–Морзе",

$$w_j \;=\; (-1)^{\nu j}; \tag{15}$$

здесь νj обозначает контрольное суммирование 7.1.3–(59). Другими словами, w_j равно -1 или $+1$, в зависимости от того, имеет ли j, выраженное в виде бинарного числа, нечетную или четную четность. Максимум $w_1x_1 + \cdots + w_nx_n$ достигается, когда в ядре вершины с четной четностью 3, 5, 6, 9, 10, 12, 15, ... наиболее сильно превышают по численности вершины с нечетной четностью 1, 2, 4, 7, 8, 11, 13, ... Оказывается, что в этом смысле оптимальным ядром при $n = 100$ оказывается

$$\{1,3,6,9,12,15,18,20,23,25,27,30,33,36,39,41,43,46,48,\\ 51,54,57,60,63,66,68,71,73,75,78,80,83,86,89,92,95,97,99\}. \tag{16}$$

В это множество из 38 вершин, удовлетворяющих условиям ядра, необходимо включить только пять вершин с нечетной четностью, а именно $\{1,25,41,73,97\}$, следовательно, $\max(w_1x_1 + \cdots + w_{100}x_{100}) = 28$. Благодаря алгоритму B нескольких тысяч компьютерных инструкций достаточно для выбора (16) из более чем триллиона возможных ядер, поскольку БДР для всех этих ядер малы.

Математически исходные задачи, связанные с комбинаторными объектами наподобие ядер циклов, могут также эффективно решаться с помощью более традиционных методов, основанных на рекуррентности и индукции. Но красота методов бинарных диаграмм решений в том, что они применимы и к реальным задачам, не имеющим такой элегантной структуры. Например, рассмотрим граф из 49 штатов из 7–(17) и 7–(61). Булева функция, представляющая все максимальные независимые множества этого графа (все ядра), имеет БДР размером 780, которая начинается следующим образом.

(17)

Алгоритм B быстро находит следующие ядра наибольшего и наименьшего весов, в случае, когда каждая вершина, представляющая штат, имеет вес, равный сумме

букв в его почтовом коде ($w_{\mathtt{CA}} = 3 + 1$, $w_{\mathtt{DC}} = 4 + 3$, ..., $w_{\mathtt{WY}} = 23 + 25$).

Наименьший вес — 155

Наибольший вес — 492 (18)

Этот граф имеет 266 137 ядер; но в случае применения алгоритма B не нужно генерировать их все. Фактически правый пример в (18) можно получить с помощью меньшей БДР размером 428, которая характеризует *независимые множества*, поскольку все веса в этом случае положительны. (В этих случаях ядро максимального веса представляет собой то же, что и независимое множество максимального веса.) У данного графа имеется 211 954 906 независимых множеств, намного больше, чем ядер; но мы можем найти независимое множество максимального веса быстрее, чем ядро максимального веса, поскольку в этом случае БДР меньше по размеру.

Рис. 22. Сеть $P_3 \,\square\, P_3$ и БДР для ее связанных подграфов.

Совершенно иной вид БДР, связанных с графами, приведен на рис. 22. Показанная здесь БДР основана на решетке $P_3 \square P_3$ размером 3×3; она характеризует множества ребер, связывающие вместе все вершины решетки. Таким образом, это функция $f(x_{12}, x_{13}, \ldots, x_{89})$ от двенадцати ребер $1 — 2$, $1 — 3$, ..., $8 — 9$, а не от девяти вершин $\{1, \ldots, 9\}$. В упр. 55 описан один из способов ее построения. Алгоритм C после применения к этой БДР говорит о том, что из $2^{12} = 4096$ остовных подграфов $P_3 \,\square\, P_3$ ровно 431 является связным.

Простое расширение алгоритма C (см. упр. 25) улучшает полученный результат и вычисляет *производящую функцию* для этих решений, а именно

$$G(z) \;=\; \sum_x z^{\nu x} f(x) \;=\; 192z^8 + 164z^9 + 62z^{10} + 12z^{11} + z^{12}. \tag{19}$$

Таким образом, $P_3 \,\square\, P_3$ имеет 192 остовных дерева, плюс 164 остовных подграфа, являющихся связными и имеющими девять ребер, и т. д. В упр. 7.2.1.6–106, (a) дана формула количества остовных деревьев в $P_m \,\square\, P_n$ для произвольных m и n; но полная производящая функция $G(z)$ содержит значительно больше информации и, вероятно, не имеет простой формулы, если только значение $\min(m, n)$ не является достаточно малым.

Предположим, что каждое ребро u — v присутствует с вероятностью p_{uv}, независимо от всех других ребер $P_3 \,\square\, P_3$. Какова вероятность того, что полученный в результате подграф связный? Ответ дает *полином надежности*, который имеет также множество других названий в связи с тем, что появляется во многих других приложениях. В общем случае, как описывается в упр. 7.1.1–12, каждая булева функция $f(x_1, \ldots, x_n)$ имеет единственное представление в виде полинома $F(x_1, \ldots, x_n)$, обладающего теми свойствами, что

i) $F(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$, когда каждое x_j представляет собой 0 или 1;

ii) $F(x_1, \ldots, x_n)$ является мультилинейным: его степень x_j имеет значение ≤ 1 для всех j.

Этот полином F имеет целочисленные коэффициенты и удовлетворяет базисному рекуррентному соотношению

$$F(x_1, \ldots, x_n) \;=\; (1 - x_1)F_0(x_2, \ldots, x_n) + x_1 F_1(x_2, \ldots, x_n), \tag{20}$$

где F_0 и F_1 — целочисленные мультилинейные представления $f(0, x_2, \ldots, x_n)$ и $f(1, x_2, \ldots, x_n)$. В действительности (20) — это ни что иное, как “закон разложения” Джорджа Буля (George Boole).

Из рекуррентного соотношения (20) следуют две важные вещи. Во-первых, F представляет собой в точности ни что иное, как упоминавшийся ранее полином надежности $F(p_1, \ldots, p_n)$, поскольку последний очевидным образом удовлетворяет тому же рекуррентному соотношению. Во-вторых, F легко вычисляется из бинарной диаграммы решений для функции f работой в восходящем направлении и применением (20) для вычисления надежности каждой бусины. (См. упр. 26.)

Функция связности для решетки $P_8 \,\square\, P_8$ размером 8×8, конечно, гораздо более сложная, чем таковая для решетки $P_3 \,\square\, P_3$; она представляет собой булеву функцию от 112 переменных, а ее бинарная диаграмма решений имеет 43 790 узлов (сравните со всего лишь 37 узлами на рис. 22). Тем не менее вычисления с помощью этой бинарной диаграммы решений достаточно практичны, и за секунду-две можно вычислить

$$\begin{aligned} G(z) = {} & 1262313229124985396825948 16z^{63} \\ & + 1006611140035411062600761344z^{64} \\ & + \cdots + 6212z^{110} + 112z^{111} + z^{112}, \end{aligned}$$

а также вероятность $F(p)$ связности и ее производную $F'(p)$, когда каждое ребро присутствует с вероятностью p (см. упр. 29):

$F(p)$:

; $F'(p)$:

. (21)

***Широкое обобщение.** Алгоритмы B и C и алгоритмы для восходящего сканирования БДР в действительности представляют собой частные случаи гораздо более общей схемы, которая может применяться многими дополнительными способами.

Рассмотрим абстрактную алгебру с двумя ассоциативными бинарными операторами $\circ$ и $\bullet$, удовлетворяющими законам дистрибутивности

$$\alpha \bullet (\beta \circ \gamma) = (\alpha \bullet \beta) \circ (\alpha \bullet \gamma) \qquad \text{и} \qquad (\beta \circ \gamma) \bullet \alpha = (\beta \bullet \alpha) \circ (\gamma \bullet \alpha). \tag{22}$$

Каждая булева функция $f(x_1, \ldots, x_n)$ соответствует *полностью детализированной таблице истинности*, включающей символы $\circ$, $\bullet$, $\bot$ и $\top$ наряду с $\bar{x}_j$ и x_j для $1 \le j \le n$ способом, который проще всего понять при рассмотрении небольшого примера: при $n = 2$ и обычной таблице истинности f, имеющей вид 0010, полностью детализированная таблица истинности представляет собой

$$(\bar{x}_1 \bullet \bar{x}_2 \bullet \bot) \circ (\bar{x}_1 \bullet x_2 \bullet \bot) \circ (x_1 \bullet \bar{x}_2 \bullet \top) \circ (x_1 \bullet x_2 \bullet \bot). \tag{23}$$

Смысл этого выражения зависит от значений, вкладываемых в символы $\circ$, $\bullet$, $\bot$, $\top$ и в литералы $\bar{x}_j$ и x_j; но что бы ни означало это выражение, его можно вычислить непосредственно из бинарной диаграммы решений для f.

Например, вернемся к рис. 21 — БДР для $\langle x_1 x_2 x_3 \rangle$. Развитием узлов $\boxed{\bot}$ и $\boxed{\top}$ являются $\alpha_\bot = \bot$ и $\alpha_\top = \top$ соответственно. Тогда развитием $\textcircled{3}$ является $\alpha_3 = (\bar{x}_3 \bullet \alpha_\bot) \circ (x_3 \bullet \alpha_\top)$; развитием узлов, помеченных $\textcircled{2}$, являются $\alpha_2^l = (\bar{x}_2 \bullet (\bar{x}_3 \circ x_3) \bullet \alpha_\bot) \circ (x_2 \bullet \alpha_3)$ слева и $\alpha_2^r = (\bar{x}_2 \bullet \alpha_3) \circ (x_2 \bullet (\bar{x}_3 \circ x_3) \bullet \alpha_\top)$ справа; а развитием узла $\textcircled{1}$ является $\alpha_1 = (\bar{x}_1 \bullet \alpha_2^l) \circ (x_1 \bullet \alpha_2^r)$. (Общая процедура рассматривается в упр. 31.) Раскрытие этих формул с использованием законов дистрибутивности (22) приводит к полному виду с $2^n = 8$ "членами":

$$\begin{aligned}\alpha_1 = {} & (\bar{x}_1 \bullet \bar{x}_2 \bullet \bar{x}_3 \bullet \bot) \circ (\bar{x}_1 \bullet \bar{x}_2 \bullet x_3 \bullet \bot) \circ (\bar{x}_1 \bullet x_2 \bullet \bar{x}_3 \bullet \bot) \circ (\bar{x}_1 \bullet x_2 \bullet x_3 \bullet \top) \\ & \circ (x_1 \bullet \bar{x}_2 \bullet \bar{x}_3 \bullet \bot) \circ (x_1 \bullet \bar{x}_2 \bullet x_3 \bullet \top) \circ (x_1 \bullet x_2 \bullet \bar{x}_3 \bullet \top) \circ (x_1 \bullet x_2 \bullet x_3 \bullet \top).\end{aligned} \tag{24}$$

Алгоритм C представляет собой частный случай, когда '$\circ$' является сложением, '$\bullet$' является умножением, '$\bot$' представляет собой 0, '$\top$' представляет собой 1, и как '$\bar{x}_j$', так и 'x_j' являются единицами. Алгоритм B получается, когда '$\circ$' представляет собой *оператор максимума*, а '$\bullet$' — сложение; законы дистрибутивности

$$\alpha + \max(\beta, \gamma) = \max(\alpha + \beta, \alpha + \gamma) \qquad \text{и} \qquad \max(\beta, \gamma) + \alpha = \max(\beta + \alpha, \gamma + \alpha) \tag{25}$$

легко проверяемы. Мы интерпретируем '$\bot$' как $-\infty$, '$\top$' как 0, '$\bar{x}_j$' как 0, а 'x_j' как w_j. Тогда, например, (24) превращается в

$$\max(-\infty, -\infty, -\infty, w_2 + w_3, -\infty, w_1 + w_3, w_1 + w_2, w_1 + w_2 + w_3);$$

и в целом полное преобразование при такой интерпретации дает выражение

$$\max\{w_1 x_1 + \cdots + w_n x_n \mid f(x_1, \ldots, x_n) = 1\}.$$

Дружественные функции. Известны многие семейства функций, имеющие БДР довольно скромного размера. Если f представляет собой, например, симметричную функцию от n переменных, то легко видеть, что $B(f) = O(n^2)$. Действительно, при

$n = 5$ можно начать с треугольного шаблона

(26)

и устанавливать листья как $\boxed{\bot}$ или $\boxed{\top}$ в зависимости от соответствующих значений f, когда значение $\nu x = x_1 + \cdots + x_5$ равно 0, 1, 2, 3, 4 или 5. Затем можно удалить избыточные или эквивалентные узлы, всегда получая БДР размером не более $\binom{n+1}{2} + 2$.

Предположим, что мы берем произвольную функцию $f(x_1, \ldots, x_n)$ и делаем равными две соседние переменные:

$$g(x_1, \ldots, x_n) \;=\; f(x_1, \ldots, x_{k-1}, x_k, x_k, x_{k+2}, \ldots, x_n). \tag{27}$$

В упр. 40 доказывается, что $B(g) \le B(f)$. Повторяя этот процесс сжатия, мы обнаружим, что функция наподобие $f(x_1, x_1, x_3, x_3, x_3, x_6)$ имеет малую БДР, если мало́ $B(f)$. В частности, пороговая функция $[2x_1 + 3x_3 + x_6 \ge t]$ должна иметь малую БДР для любого значения t, поскольку это сжатая версия симметричной функции $f(x_1, \ldots, x_6) = [x_1 + \cdots + x_6 \ge t]$. Эти рассуждения показывают, что *любая* пороговая функция с неотрицательными целыми весами,

$$f(x_1, x_2, \ldots, x_n) \;=\; [w_1x_1 + w_2x_2 + \cdots + w_nx_n \ge t], \tag{28}$$

может быть получена сжатием симметричной функции от $w_1 + w_2 + \cdots + w_n$ переменных, так что размер ее БДР составляет $O(w_1 + w_2 + \cdots + w_n)^2$.

Пороговые функции часто оказываются простыми даже при экспоненциальном росте весов. Например, пусть $t = (t_1t_2 \ldots t_n)_2$, и рассмотрим

$$f_t(x_1, x_2, \ldots, x_n) = [2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + x_n \ge t]. \tag{29}$$

Эта функция истинна тогда и только тогда, когда бинарная строка $x_1x_2 \ldots x_n$ лексикографически больше или равна $t_1t_2 \ldots t_n$, а ее БДР всегда имеет ровно $n + 2$ узлов при $t_n = 1$. (См. упр. 170.)

Другой вид функции с малой БДР — это 2^m-канальный мультиплексор из уравнения 7.1.2–(31), функция от $n = m + 2^m$ переменных:

$$M_m(x_1, \ldots, x_m; x_{m+1}, \ldots, x_n) \;=\; x_{m+1+(x_1\ldots x_m)_2}. \tag{30}$$

Ее БДР начинается с 2^{k-1} узлов ветвления $\textcircled{k}$ для $1 \le k \le m$. Но ниже этого полного бинарного дерева имеется только по одному $\textcircled{k}$ для каждого x_k из основного блока переменных, такого, что $m < k \le n$. Следовательно, $B(M_m) = 1 + 2 + \cdots + 2^{m-1} + 2^m + 2 = 2^{m+1} + 1 < 2n$.

Линейная сетевая модель вычислений, показанная на рис. 23, помогает прояснить ситуации, когда БДР особенно эффективна. Рассмотрим размещение вычис-

лительных модулей M_1, M_2, ..., M_n, при котором булева переменная x_k является входом для модуля M_k; между соседними модулями имеются также провода, проводящие булевы сигналы, с a_k проводами, идущими от модуля M_k к M_{k+1}, и b_k проводами от модуля M_{k+1} к M_k для $1 \le k \le n$. Отдельный провод, выходящий из модуля M_n, содержит вывод функции, $f(x_1, \ldots, x_n)$. Мы определяем $a_0 = b_0 = b_n = 0$ и $a_n = 1$, так что модуль M_k имеет ровно $c_k = 1 + a_{k-1} + b_k$ входных портов и ровно $d_k = a_k + b_{k-1}$ выходных портов для каждого k. Он вычисляет d_k булевых функций от своих c_k входов.

Индивидуальные функции, вычисляемые каждым модулем, могут быть произвольно сложными, но они должны быть *вполне определенными* в том смысле, что их объединенные значения полностью определяются значениями x: каждый выбор $(x_1, \ldots, x_n)$ должен вести к единственному способу установки сигналов во всех проводах, согласующемуся со всеми данными функциями.

Теорема М. *Если f может быть вычислена такой сетью, то* $B(f) \le \sum_{k=0}^{n} 2^{a_k 2^{b_k}}$.

Доказательство. Мы покажем, что БДР для f имеет не более $2^{a_{k-1}2^{b_{k-1}}}$ узлов ветвления $\textcircled{k}$, для $1 \le k \le n$. Это очевидно, если $b_{k-1} = 0$, поскольку для любых заданных значений от x_1 до x_{k-1} возможны не более $2^{a_{k-1}}$ подфункций. Так что мы покажем, что любая сеть, которая имеет a_{k-1} прямых и b_{k-1} обратных проводов между M_{k-1} и M_k, может быть заменена эквивалентной сетью, которая имеет $a_{k-1}2^{b_{k-1}}$ прямых проводов и ни одного обратного.

Для удобства рассмотрим случай $k = 4$ на рис. 23 с $a_3 = 4$ и $b_3 = 2$; мы хотим заменить эти 6 проводов на 16, которые идут только вперед. Предположим, что Алиса отвечает за модуль M_3, а Боб — за модуль M_4. Алиса отправляет 4-битовый сигнал a Бобу, в то время как он отправляет 2-битовый сигнал b Алисе. Точнее говоря, для любого фиксированного значения $(x_1, \ldots, x_n)$ Алиса вычисляет некоторую функцию A, а Боб вычисляет функцию B, где

$$A(b) = a \qquad \text{и} \qquad B(a) = b. \tag{31}$$

Функция Алисы A зависит от (x_1, x_2, x_3), так что Боб не знает ее; функция Боба B точно так же неизвестна Алисе, поскольку она зависит от $(x_4, \ldots, x_n)$. Но эти неизвестные функции обладают тем ключевым свойством, что для каждого выбора $(x_1, \ldots, x_n)$ имеется ровно одно решение (a, b) уравнений (31).

Рис. 23. Обобщенная сеть булевых модулей, для которой справедлива теорема М.

Так что Алиса изменяет поведение модуля M_3: она посылает Бобу *четыре* 4-битовых значения, $A(00)$, $A(01)$, $A(10)$ и $A(11)$, тем самым раскрывая свою функ-

цию A. А Боб изменяет поведение M_4: вместо того чтобы отослать ответ, он смотрит на полученные сообщения вместе со своими прочими входными данными (а именно с битами x_4 и b_4, полученными от M_5) и находит уникальные a и b, решающие (31). Его новый модуль использует это значение a для вычисления битов a_4, которые он отправляет модулю M_5. ▌

Теорема М гласит, что размер бинарной диаграммы решений будет достаточно невелик, если мы можем построить такую сеть с небольшими значениями a_k и b_k. Действительно, $B(f)$ будет равно $O(n)$, если a и b ограничены, хотя входящая в $O(n)$ константа может быть огромной. Давайте рассмотрим функцию "*три единицы подряд*",

$$f(x_1,\ldots,x_n) = x_1x_2x_3 \lor x_2x_3x_4 \lor \cdots \lor x_{n-2}x_{n-1}x_n \lor x_{n-1}x_nx_1 \lor x_nx_1x_2, \tag{32}$$

которая истинна тогда и только тогда, когда циклическое "ожерелье", состоящее из битов x_1, ..., x_n, содержит три последовательные единицы. Один из способов ее реализации с помощью булевых модулей заключается в передаче M_k трех входных значений (u_k, v_k, w_k) от M_{k-1} и двух входных значений (y_k, z_k) от M_{k+1}, где

$$\begin{gathered} u_k = x_{k-1}, \quad v_k = x_{k-2}x_{k-1}, \quad w_k = x_{n-1}x_nx_1 \lor \cdots \lor x_{k-3}x_{k-2}x_{k-1}; \\ y_k = x_n, \quad z_k = x_{n-1}x_n. \end{gathered} \tag{33}$$

Здесь индексы рассматриваются по модулю n, и при $k = 1$ или $k \ge n-1$ слева и справа вносятся соответствующие изменения. Тогда M_k вычисляет функции

$$u_{k+1} = x_k, \quad v_{k+1} = u_kx_k, \quad w_{k+1} = w_k \lor v_kx_k, \quad y_{k-1} = y_k, \quad z_{k-1} = z_k \tag{34}$$

почти для всех значений k; детали вы найдете в упр. 45. При таком построении мы имеем $a_k \le 3$ и $b_k \le 2$ для всех k; следовательно, согласно теореме М, $B(f) \le 2^{12}n = 4096n$. Действительность существенно приятнее: $B(f)$ на самом деле $< 9n$ (см. упр. 46).

Совместно используемые БДР. Зачастую нам приходится работать одновременно с несколькими функциями, и такие связанные функции часто имеют общие подфункции. В таких случаях мы можем работать с "базой БДР" (BDD base) для $\{f_1(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n)\}$, представляющей собой ориентированный ациклический граф, который содержит по одному узлу для каждой бусины, встречающейся в таблицах истинности любой из функций. База БДР также имеет m "корневых указателей" F_j, по одному для каждой функции f_j; БДР для f_j при этом представляет собой множество всех узлов, достижимых из узла F_j. Обратите внимание, что сам узел F_j достижим из узла F_i тогда и только тогда, когда f_j представляет собой подфункцию f_i.

Рассмотрим, например, задачу вычисления $n+1$ бит суммы двух n-битовых чисел,

$$(f_{n+1}f_nf_{n-1}\ldots f_1)_2 \;=\; (x_1x_3\ldots x_{2n-1})_2 + (x_2x_4\ldots x_{2n})_2. \tag{35}$$

База БДР для этих $n+1$ бит при $n=4$ имеет следующий вид.

$$\begin{array}{r} x_1x_3x_5x_7 \\ +\ x_2x_4x_6x_8 \\ \hline f_5f_4f_3f_2f_1 \end{array} \tag{36}$$

Способ нумерации x в (35) в данном случае важен (см. упр. 51). В общем случае имеется ровно $B(f_1,\ldots,f_{n+1}) = 9n-5$ узлов при $n>1$. Узел, находящийся непосредственно слева от F_j, для $1\le j\le n$, представляет подфункцию для *переноса* c_j из j-й битовой позиции справа; узел, находящийся непосредственно справа от F_j, представляет дополнение этого переноса, $\bar c_j$; а узел F_{n+1} представляет последний перенос c_n.

Операции над БДР. Мы рассмотрели множество вещей, которые можно сделать с помощью уже имеющейся БДР. Но пока что остается открытым вопрос, как же нам сперва получить эту БДР в компьютере?

Один из способов заключается в том, чтобы начать с упорядоченной бинарной диаграммы решений, такой, как (26) или изображенная в правой части примера (2), и выполнить ее приведение, с тем чтобы она превратилась в настоящую БДР. Приведенный далее алгоритм, основанный на идеях Д. Зилинга (D. Sieling) и И. Вегенера (I. Wegener) [*Information Processing Letters* **48** (1993), 139–144], показывает, как произвольная N-узловая бинарная диаграмма решений с правильно упорядоченными ветвями может быть приведена к БДР за $O(N+n)$ шагов при наличии n переменных.

Конечно, нам требуется некоторая дополнительная память для того, чтобы при выполнении приведения решить, эквивалентны ли два узла. Только лишь трех описанных в (1) полей (`V`, `LO`, `HI`) в каждом узле недостаточно для маневра. К счастью, достаточно только одного дополнительного поля с размером указателя, которое мы назовем `AUX`, а также двух дополнительных битов состояния. Для удобства будем считать, что биты состояния неявно присутствуют в *знаках* полей `LO` и `AUX`, так что алгоритм должен работать только с четырьмя полями: (`V`, `LO`, `HI`, `AUX`). Резервирование знака означает, что 28-битовое поле `LO` будет принимать не более 2^{27} узлов (около 134 миллионов) вместо 2^{28}. (На компьютерах типа `MMIX` можно предпочесть иной путь — считать, что все адреса узлов четны, и добавлять к полю 1, а не выполнять дополнение, как делается здесь.)

Алгоритм R (*Приведение к БДР*). Этот алгоритм трансформирует заданную бинарную диаграмму решений, которая является упорядоченной, но необязательно приведенной, в корректную БДР путем удаления излишних узлов и перенаправления соответствующим образом всех указателей. Каждый узел считается имеющим четыре поля (`V`, `LO`, `HI`, `AUX`), как описано выше, а `ROOT` указывает на верхний узел диаграммы. Начальные значения полей `AUX` несущественны, но должны быть неотрицательными; по завершении процесса они вновь будут неотрицательными. Все

удаленные узлы вносятся в стек, адресуемый указателем AVAIL, связываясь вместе с помощью полей HI их узлов. (Поля LO этих узлов будут отрицательными; их дополнения указывают на эквивалентные узлы, которые *не* были удалены.)

Мы считаем, что поля V узлов ветвления пробегают от V(ROOT) до $v_{\max}$ в возрастающем порядке сверху вниз в данном ациклическом ориентированном графе. Узлы стока $\boxed{\bot}$ и $\boxed{\top}$ рассматриваются как узлы 0 и 1 соответственно, с неотрицательными полями LO и HI. Они никогда не удаляются; фактически они остаются нетронутыми, если не считать их полей AUX. Вспомогательный массив указателей HEAD[v] для V(ROOT) $\le v \le v_{\max}$ используется для создания временных списков всех узлов, имеющих данное значение V.

R1. [Инициализация.] Завершить работу алгоритма немедленно, если ROOT ≤ 1. В противном случае установить AUX(0) $\leftarrow$ AUX(1) $\leftarrow$ AUX(ROOT) $\leftarrow -1$ и для V(ROOT) $\le v \le v_{\max}$ установить HEAD[v] $\leftarrow -1$. (Мы используем тот факт, что $-1 = \sim 0$ представляет собой битовое дополнение 0.) Затем установить $s \leftarrow$ ROOT и выполнять следующие операции, пока $s \ne 0$.

Установить $p \leftarrow s$, $s \leftarrow \sim$AUX(p), AUX(p) $\leftarrow$ HEAD[V(p)], HEAD[V(p)] $\leftarrow \sim p$.
Если AUX(LO(p)) ≥ 0, установить AUX(LO(p)) $\leftarrow \sim s$ и $s \leftarrow$ LO(p).
Если AUX(HI(p)) ≥ 0, установить AUX(HI(p)) $\leftarrow \sim s$ и $s \leftarrow$ HI(p).

(По сути, мы выполняем поиск в глубину в ациклическом ориентированном графе, временно помечая все узлы, достижимые из ROOT, путем установки отрицательных значений их полей AUX.)

R2. [Цикл по v.] Установить AUX(0) $\leftarrow$ AUX(1) $\leftarrow 0$ и $v \leftarrow v_{\max}$.

R3. [Карманная сортировка.] (В этот момент все остающиеся узлы, поле V которых превышает v, были корректно приведены, а их поля AUX неотрицательны.) Установить $p \leftarrow \sim$HEAD[v], $s \leftarrow 0$, и выполнять следующие шаги, пока $p \ne 0$.

Установить $p' \leftarrow \sim$AUX(p).
Установить $q \leftarrow$ HI(p); если LO(q) < 0, установить HI(p) $\leftarrow \sim$LO(q).
Установить $q \leftarrow$ LO(p);
если LO(q) < 0, установить LO(p) $\leftarrow \sim$LO(q) и $q \leftarrow$ LO(p).
Если $q =$ HI(p), установить LO(p) $\leftarrow \sim q$,
HI(p) $\leftarrow$ AVAIL, AUX(p) $\leftarrow 0$, AVAIL $\leftarrow p$;
в противном случае при AUX(q) ≥ 0
установить AUX(p) $\leftarrow s$, $s \leftarrow \sim q$ и AUX(q) $\leftarrow \sim p$;
в противном случае установить
AUX(p) $\leftarrow$ AUX($\sim$AUX(q)) и AUX($\sim$AUX(q)) $\leftarrow p$.
Затем установить $p \leftarrow p'$.

R4. [Очистка.] (Узлы с LO $= x \ne$ HI были связаны вместе с помощью их полей AUX, начиная с $\sim$AUX(x).) Установить $r \leftarrow \sim s$, $s \leftarrow 0$, и выполнять следующие действия, пока $r \ge 0$.

Установить $q \leftarrow \sim$AUX(r) и AUX(r) $\leftarrow 0$.
Если $s = 0$, установить $s \leftarrow q$; в противном случае установить AUX(p) $\leftarrow q$.
Установить $p \leftarrow q$; затем, пока AUX(p) > 0, устанавливать $p \leftarrow$ AUX(p).
Установить $r \leftarrow \sim$AUX(p).

R5. [Цикл по p.] Установить $p \leftarrow s$. Перейти к шагу R9, если $p = 0$. В противном случае установить $q \leftarrow p$.

R6. [Проверка кармана.] Установить $s \leftarrow \texttt{LO}(p)$. (В этот момент $p = q$.)

R7. [Удаление дублей.] Установить $r \leftarrow \texttt{HI}(q)$. Если $\texttt{AUX}(r) \geq 0$, установить $\texttt{AUX}(r) \leftarrow \sim q$; в противном случае установить $\texttt{LO}(q) \leftarrow \texttt{AUX}(r)$, $\texttt{HI}(q) \leftarrow \texttt{AVAIL}$ и $\texttt{AVAIL} \leftarrow q$. Затем установить $q \leftarrow \texttt{AUX}(q)$. Если $q \neq 0$ и $\texttt{LO}(q) = s$, повторить шаг R7.

R8. [Еще одна очистка.] Если $\texttt{LO}(p) \geq 0$, установить $\texttt{AUX}(\texttt{HI}(p)) \leftarrow 0$. Затем установить $p \leftarrow \texttt{AUX}(p)$ и повторять шаг R8, пока не будет выполнено условие $p = q$.

R9. [Выполнено?] Если $p \neq 0$, вернуться к шагу R6. В противном случае при $v > \texttt{V}(\texttt{ROOT})$ установить $v \leftarrow v - 1$ и вернуться к шагу R3. В противном случае при $\texttt{LO}(\texttt{ROOT}) < 0$ установить $\texttt{ROOT} \leftarrow \sim\texttt{LO}(\texttt{ROOT})$. ▮

Запутанные манипуляции связями в алгоритме R проще запрограммировать, чем объяснить, но они очень поучительны и на самом деле не так уж и сложны. Настоятельно советуем читателю проработать пример из упр. 53.

Алгоритм R можно также использовать для вычисления БДР для любого *ограничения* заданной функции, а именно для любой функции, получающейся "прошивкой" константных значений вместо одной или нескольких переменных. Идея заключается в выполнении небольшой дополнительной работы между шагами R1 и R2, устанавливающей $\texttt{HI}(p) \leftarrow \texttt{LO}(p)$, если переменная $\texttt{V}(p)$ рассматривается как фиксированный 0, или $\texttt{LO}(p) \leftarrow \texttt{HI}(p)$, если $\texttt{V}(p)$ рассматривается как фиксированная 1. Нам также нужно удалить все узлы, которые становятся недоступными после выполнения такого ограничения. Детали этого процесса освещаются в упр. 57.

Синтез БДР. Теперь мы готовы приступить к наиболее важному алгоритму, связанному с БДР, который берет БДР для одной функции, f, и комбинирует ее с БДР для другой функции, g, получая в результате БДР для других функций, таких как $f \wedge g$ или $f \oplus g$. Операции синтеза такого рода являются основным способом построения БДР для сложных функций, и тот факт, что это может быть сделано эффективно, и является основной причиной популярности структур данных БДР. Мы рассмотрим несколько подходов к проблеме синтеза, начиная с простого метода, а затем ускоряя его различными способами.

Базовым понятием, лежащим в основе синтеза, является операция произведения структур БДР, которую мы назовем *слиянием* (*melding*). Предположим, что $\alpha = (v, l, h)$ и $\alpha' = (v', l', h')$ представляют собой узлы БДР, каждый из которых содержит индекс переменной вместе с указателями LO и HI. "Смесь" α и α' (записывается как $\alpha \diamond \alpha'$) определяется следующим образом (в предположении, что α и α' не являются одновременно узлами стока):

$$\alpha \diamond \alpha' = \begin{cases} (v,\ l \diamond l',\ h \diamond h'), & \text{если } v = v'; \\ (v,\ l \diamond \alpha',\ h \diamond \alpha'), & \text{если } v < v'; \\ (v',\ \alpha \diamond l',\ \alpha \diamond h'), & \text{если } v > v'. \end{cases} \tag{37}$$

Например, на рис. 24 показано, как выполняется слияние двух небольших, но типичных БДР. БДР слева, с узлами ветвления $(\alpha, \beta, \gamma, \delta)$, представляет $f(x_1, x_2, x_3, x_4) = (x_1 \vee x_2) \wedge (x_3 \vee x_4)$; БДР в средине, с узлами ветвления $(\omega, \psi, \chi, \varphi, \upsilon, \tau)$, представляет

$g(x_1, x_2, x_3, x_4) = (x_1 \oplus x_2) \vee (x_3 \oplus x_4)$. Узлы δ и τ — это, по сути, один и тот же узел, так что мы бы имели $\delta = \tau$, если бы f и g были частью одной базы БДР; но слияние может быть применено и к БДР, не имеющим общих узлов. Справа на рис. 24 $\alpha \diamond \omega$ представляет собой корень диаграммы, которая имеет одиннадцать узлов ветвления и, по сути, представляет *упорядоченную пару* (f, g).

Рис. 24. Две БДР могут быть слиты с помощью операции $\diamond$ (37).

Упорядоченная пара двух булевых функций может быть визуализирована путем размещения таблицы истинности одной из них над таблицей истинности другой. При такой интерпретации $\alpha \diamond \omega$ означает упорядоченную пару $\begin{smallmatrix}0000011101110111\\0110111111110110\end{smallmatrix}$, а $\beta \diamond \chi$ означает $\begin{smallmatrix}00000111\\01101111\end{smallmatrix}$, и т. д. Слитые БДР на рис. 24 соответствуют диаграмме

, (38)

которая аналогична (5) с тем отличием, что каждый узел обозначает упорядоченную пару функций, а не единственную функцию. Бусины и подтаблицы определяются на упорядоченных парах так же, как и ранее. Но теперь у нас имеется четыре возможных стока вместо двух, а именно

$$\bot \diamond \bot, \qquad \bot \diamond \top, \qquad \top \diamond \bot \qquad \text{и} \qquad \top \diamond \top, \tag{39}$$

соответствующие упорядоченным парам $\begin{smallmatrix}0\\0\end{smallmatrix}$, $\begin{smallmatrix}0\\1\end{smallmatrix}$, $\begin{smallmatrix}1\\0\end{smallmatrix}$ и $\begin{smallmatrix}1\\1\end{smallmatrix}$.

Для вычисления конъюнкции $f \wedge g$ мы выполняем операцию И с таблицами истинности функций f и g. Эта операция соответствует замене $\begin{smallmatrix}0\\0\end{smallmatrix}$, $\begin{smallmatrix}0\\1\end{smallmatrix}$, $\begin{smallmatrix}1\\0\end{smallmatrix}$ и $\begin{smallmatrix}1\\1\end{smallmatrix}$ на 0, 0, 0 и 1 соответственно; так что мы получаем БДР для $f \wedge g$ из $f \diamond g$ путем замены соответствующих стоковых узлов (39) на $\boxed{\bot}$, $\boxed{\bot}$, $\boxed{\bot}$ и $\boxed{\top}$ с последующим приведением результата. Аналогично БДР для $f \oplus g$ получается, если мы заменим стоки (39) на $\boxed{\bot}$, $\boxed{\top}$, $\boxed{\top}$ и $\boxed{\bot}$. (В этом частном случае $f \oplus g$ оказывается симметричной функцией $S_{1,4}(x_1, x_2, x_3, x_4)$, показанной на рис. 9 в разделе 7.1.2.) Полученная слиянием диаграмма $f \diamond g$ содержит всю информацию, необходимую для

вычисления *любой* булевой комбинации f и g; и БДР для каждой такой комбинации имеет не более $B(f \diamond g)$ узлов.

Ясно, что $B(f \diamond g) \le B(f)B(g)$, поскольку каждый узел $f \diamond g$ соответствует узлу f и узлу g. Следовательно, слияние небольших БДР не может дать очень большую диаграмму. Фактически обычно процедура слияния дает результат, который значительно меньше этой верхней границы для наихудшего случая с чем-то наподобие $B(f)+B(g)$ узлов вместо $B(f)B(g)$. В упр. 60 обсуждаются более строгие границы, которые проливают свет на то, почему слияние часто оказывается небольшим. Но в упр. 59, (б) и 63 имеются интересные примеры, в которых наблюдается квадратичный рост.

Слияние предлагает простой алгоритм для синтеза: мы можем образовать массив из $B(f)B(g)$ узлов с узлом $\alpha \diamond \alpha'$ в строке α и столбце α' для каждого α в БДР для f и каждого α' в БДР для g. Затем мы можем преобразовать четыре стоковых узла (39) в $\boxed{\bot}$ или $\boxed{\top}$ и применить алгоритм R к корневому узлу $f \diamond g$. Вуаля! Мы получили БДР для $f \wedge g$ или $f \oplus g$, или $f \vee \bar{g}$, или для чего нам хочется.

Ясно, что время работы этого алгоритма имеет порядок $B(f)B(g)$. Мы можем снизить его до порядка $B(f \diamond g)$, поскольку не требуется заполнять все записи матрицы $\alpha \diamond \alpha'$; играют роль только те узлы, которые достижимы из $f \diamond g$, и мы можем при необходимости генерировать их "на лету". Но даже с таким усовершенствованием простой алгоритм неудовлетворителен из-за размещения $B(f)B(g)$ узлов в памяти. При работе с БДР время дешево, а память дорогá: попытки решать большие задачи чаще проваливаются из-за нехватки памяти, а не времени. Вот почему алгоритм R выполнял свои "махинации", используя только одну вспомогательную связь на узел.

Приведенный далее алгоритм решает задачу синтеза с требованием порядка $B(f \diamond g)$ рабочего пространства; фактически он требует только около 16 байт на элемент БДР для $f \diamond g$. Этот алгоритм разработан для использования в качестве основного процессора "калькулятора булевых функций", который представляет функции в виде БДР в сжатом виде в последовательном стеке. Этот стек поддерживается на нижнем конце большого массива, именуемого *пулом*. Каждая БДР в стеке является последовательностью *узлов*, каждый из которых имеет три поля: (V, LO, HI). Остаток пула доступен для хранения временных результатов, именуемых *шаблонами* (*templates*), каждый из которых, в свою очередь, имеет четыре поля: (L, H, LEFT, RIGHT). Узел обычно занимает один октет памяти, в то время как шаблон занимает два октета.

Целью алгоритма S является изучение двух верхних булевых функций в стеке, f и g, и замена их булевой комбинацией $f \circ g$, где $\circ$ представляет собой один из 16 возможных бинарных операторов. Этот оператор идентифицируется своей 4-битовой таблицей истинности *op*. Например, алгоритм S образует БДР для $f \oplus g$, когда *op* представляет собой $(0110)_2 = 6$, и для $f \wedge g$, когда $op = 1$.

Когда алгоритм начинает работу, операнд f находится в позициях $[f_0 \mathinner{..} g_0)$ пула, а операнд g — в позициях $[g_0 \mathinner{..} \mathtt{NTOP})$. Все более высокие позиции [NTOP . . POOLSIZE) доступны для хранения шаблонов, в которых нуждается алгоритм. Эти шаблоны располагаются в позициях [TBOT . . POOLSIZE) в верхнем конце пула; маркеры границ NTOP и TBOT динамически изменяются в процессе работы алгоритма. Получающаяся в результате бинарная диаграмма решений для $f \circ g$ в конечном итоге будет помещена в позиции $[f_0 \mathinner{..} \mathtt{NTOP})$, занимая пространство, ранее занятое f и g. Мы счита-

ем, что шаблон занимает пространство двух узлов. Таким образом, присваивания "$t \leftarrow \texttt{TBOT} - 2$, $\texttt{TBOT} \leftarrow t$" выделяют память для нового шаблона, на который указывает t; присваивания "$p \leftarrow \texttt{NTOP}$, $\texttt{NTOP} \leftarrow p + 1$" распределяют новый узел p. Для простоты описания алгоритм S не проверяет выполнимости условия $\texttt{NTOP} \le \texttt{TBOT}$ в процессе работы; но, конечно, такие проверки существенны на практике. В упр. 69 эта оплошность исправлена.

Входные функции f и g передаются алгоритму S как последовательности инструкций $(I_{s-1}, \ldots, I_1, I_0)$ и $(I'_{s'-1}, \ldots, I'_1, I'_0)$, как в алгоритмах B и C выше. Длины этих последовательностей равны $s = B^+(f)$ и $s' = B^+(g)$, где

$$B^+(f) \;=\; B(f) + [f \text{ тождественна } 1] \tag{40}$$

представляет собой число узлов бинарной диаграммы решений при вынужденном наличии стока $\boxed{\perp}$. Например, две бинарные диаграммы решений слева на рис. 24 могут быть определены с помощью инструкций

$$\begin{array}{ll} I_5 = (\bar{1}?\ 4{:}\ 3), & I_3 = (\bar{3}?\ 2{:}\ 1), \\ I_4 = (\bar{2}?\ 0{:}\ 3), & I_2 = (\bar{4}?\ 0{:}\ 1); \end{array} \qquad \begin{array}{ll} I'_7 = (\bar{1}?\ 5{:}\ 6), & I'_4 = (\bar{3}?\ 2{:}\ 3), \\ I'_6 = (\bar{2}?\ 1{:}\ 4), & I'_3 = (\bar{4}?\ 1{:}\ 0), \\ I'_5 = (\bar{2}?\ 4{:}\ 1), & I'_2 = (\bar{4}?\ 0{:}\ 1); \end{array} \tag{41}$$

как обычно, I_1, I_0, I'_1 и I'_0 являются стоками. Эти инструкции упакованы в узлы, так что если $I_k = (\bar{v}_k?\ l_k{:}\ h_k)$, то, когда алгоритм S начинает работу, мы имеем $\texttt{V}(f_0 + k) = v_k$, $\texttt{LO}(f_0 + k) = l_k$ и $\texttt{HI}(f_0 + k) = h_k$ для $2 \le k < s$. Аналогичные соглашения применимы к инструкциям I'_k, которые определяют g. Кроме того,

$$\texttt{V}(f_0) = \texttt{V}(f_0 + 1) = \texttt{V}(g_0) = \texttt{V}(g_0 + 1) = v_{\max} + 1, \tag{42}$$

где мы считаем, что f и g зависят только от переменных x_v для $1 \le v \le v_{\max}$.

Подобно простому, но требующему большого количества памяти алгоритму, описанному ранее, алгоритм S работает в два этапа: сначала он строит бинарную диаграмму решений для $f \diamond g$, конструируя шаблоны так, чтобы каждое важное слияние $\alpha \diamond \alpha'$ было представлено в виде шаблона t, для которого

$$\texttt{LEFT}(t) = \alpha,\ \texttt{RIGHT}(t) = \alpha',\ \ \texttt{L}(t) = \mathrm{LO}(\alpha \diamond \alpha'),\ \texttt{H}(t) = \mathrm{HI}(\alpha \diamond \alpha'). \tag{43}$$

(Поля L и H указывают на шаблоны, а не на узлы.) Затем на втором этапе выполняется приведение этих шаблонов с использованием процедуры, подобной алгоритму R; она заменяет шаблон t из (43) на

$$\begin{gathered} \texttt{LEFT}(t) = {\sim}\kappa(t),\ \ \texttt{RIGHT}(t) = \tau(t), \\ \texttt{L}(t) = \tau(\mathrm{LO}(\alpha \diamond \alpha')),\ \texttt{H}(t) = \tau(\mathrm{HI}(\alpha \diamond \alpha')), \end{gathered} \tag{44}$$

где $\tau(t)$ — уникальный шаблон, к которому был приведен t, и где $\kappa(t)$ является "клоном" t, если $\tau(t) = t$. Каждый приведенный шаблон t соответствует узлу инструкции в БДР $f \circ g$, а $\kappa(t)$ представляет собой индекс этого узла по отношению к позиции f_0 в стеке. (Установка $\texttt{LEFT}(t)$ равным ${\sim}\kappa(t)$ вместо $\kappa(t)$ представляет собой хитрый трюк, который заставляет шаги S7–S10 выполняться быстрее.) Для стоков в *нижней* части пула постоянно зарезервированы специальные перекрывающиеся шаблоны, так что мы всегда имеем

$$\texttt{LEFT}(0) = {\sim}0,\ \ \texttt{RIGHT}(0) = 0,\ \ \texttt{LEFT}(1) = {\sim}1,\ \ \texttt{RIGHT}(1) = 1 \tag{45}$$

в соответствии с соглашениями (42) и (44).

Нам не требуется делать шаблон для $\alpha \diamond \alpha'$, когда значение $\alpha \circ \alpha'$ является очевидной константой. Например, если мы вычисляем $f \wedge g$, то знаем, что $\alpha \diamond \alpha'$ в конечном счете будет приведено к $\boxed{\perp}$, если $\alpha = 0$ или $\alpha' = 0$. Такие упрощения выявляются подпрограммой под названием *find_level*(f, g), которая возвращает положительное целое число j, когда корень $f \diamond g$ начинается с ветви $\textcircled{j}$, если только $f \circ g$ не является очевидной константой; в последнем случае *find_level*(f, g) возвращает значение $-(f \circ g)$, которое равно 0 или -1. Процедура носит немного формальный характер, но проста и использует глобальную таблицу истинности *op*.

Подпрограмма *find_level*(f, g) использует локальную переменную t:
Если $f \le 1$ и $g \le 1$, вернуть $-((op \gg (3 - 2f - g)) \,\&\, 1)$, равное $-(f \circ g)$.
Если $f \le 1$ и $g > 1$, установить $t \leftarrow (f?\ op \,\&\, 3\colon\ op \gg 2)$;
вернуть 0, если $t = 0$, и -1, если $t = 3$. (46)
Если $f > 1$ и $g \le 1$, установить $t \leftarrow (g?\ op\colon\ op \gg 1) \,\&\, 5$;
вернуть 0, если $t = 0$, и -1, если $t = 5$.
Иначе вернуть $\min(\mathtt{V}(f_0 + f), \mathtt{V}(g_0 + g))$.

Основная трудность, возникающая перед нами при генерации шаблона для потомка $\alpha \diamond \alpha'$ согласно (37), заключается в том, чтобы решить, существует ли уже такой шаблон, и, если существует, то осуществить с ним связь. Наилучший способ решения таких задач обычно состоит в применении хеш-таблицы; но тогда нам придется также решать, где поместить эту таблицу и сколько дополнительной памяти выделить для нее. Такие альтернативы, как бинарные деревья поиска, гораздо легче адаптировать для наших целей, но они добавляют ко времени работы нежелательный множитель $\log B(f \diamond g)$. Проблема синтеза в действительности может быть решена с использованием $O(B(f \diamond g))$ памяти за такое $(O(B(f \diamond g)))$ время путем применения метода карманной сортировки, аналогичного алгоритму R (см. упр. 72); но такое решение оказывается более сложным, и его труднее реализовать.

К счастью, имеется красивый способ разрешения этой дилеммы, почти не требующий дополнительной памяти и с умеренно сложным кодом, если генерировать за один раз шаблоны на одном уровне. Перед генерацией шаблонов для уровня l мы будем знать количество N_l шаблонов, запрашиваемых на этом уровне. Так что мы можем временно выделить память на вершине текущей свободной области для 2^b шаблонов, где $b = \lceil \lg N_l \rceil$, и поместить туда новые шаблоны, выполняя хеширование в той же области. Идея заключается в использовании цепочек с отдельными списками, как на рис. 38 в разделе 6.4; поля H и L наших шаблонов и потенциальных шаблонов играют роли головных узлов и связей на этом рисунке, в то время как ключи находятся в (LEFT, RIGHT). Вот более детальное описание используемой логики.

Подпрограмма *make_template*(f, g), использует локальную переменную t.
Установить $h \leftarrow \mathtt{HBASE} + 2(((314159257f + 271828171g) \bmod 2^d) \gg (d - b))$, где d представляет собой удобную верхнюю границу размера указателя (обычно $d = 32$). Затем установить $t \leftarrow \mathtt{H}(h)$. Пока $t \ne \Lambda$ и либо $\mathtt{LEFT}(t) \ne f$, либо $\mathtt{RIGHT}(t) \ne g$, устанавливать $t \leftarrow \mathtt{L}(t)$. Если $t = \Lambda$, установить $t \leftarrow \mathtt{TBOT} - 2$, $\mathtt{TBOT} \leftarrow t$, $\mathtt{LEFT}(t) \leftarrow f$, $\mathtt{RIGHT}(t) \leftarrow g$, $\mathtt{L}(t) \leftarrow \mathtt{H}(h)$ и $\mathtt{H}(h) \leftarrow t$. Наконец вернуть значение t. (47)

Вызывающая программа в шагах S4 и S5 обеспечивает выполнение условия $\texttt{NTOP} \le \texttt{HBASE} \le \texttt{TBOT}$.

Эта стратегия "синтеза в ширину", по одному уровню за раз, имеет дополнительное преимущество, заключающееся в поддержке "локальности ссылок": обращения к памяти в этом случае имеют тенденцию сосредоточиваться в близких позициях, которые недавно были просмотрены, так что количество промахов кэша и ошибок отсутствия страницы оказывается существенно сниженным. Кроме того, узлы БДР, помещаемые в конечном итоге в стек, также будут упорядочены, так что все ветвления для одной и той же переменной будут располагаться последовательно.

Алгоритм S (*Синтез БДР в ширину*). Этот алгоритм вычисляет БДР для $f \circ g$, как описано выше, с использованием подпрограмм (46) и (47). Используются вспомогательные массивы $\texttt{LSTART}[l]$, $\texttt{LCOUNT}[l]$, $\texttt{LLIST}[l]$ и $\texttt{HLIST}[l]$ для $0 \le l \le v_{\max}$.

S1. [Инициализация.] Установить $f \leftarrow g_0 - 1 - f_0$, $g \leftarrow \texttt{NTOP} - 1 - g_0$ и $l \leftarrow \mathit{find_level}(f, g)$. См. упр. 66, если $l \le 0$. В противном случае установить $\texttt{LSTART}[l-1] \leftarrow \texttt{POOLSIZE}$ и $\texttt{LLIST}[k] \leftarrow \texttt{HLIST}[k] \leftarrow \Lambda$, $\texttt{LCOUNT}[k] \leftarrow 0$ для $l < k \le v_{\max}$. Установить $\texttt{TBOT} \leftarrow \texttt{POOLSIZE} - 2$, $\texttt{LEFT}(\texttt{TBOT}) \leftarrow f$ и $\texttt{RIGHT}(\texttt{TBOT}) \leftarrow g$.

S2. [Сканирование шаблонов уровня l.] Выполнить установки $\texttt{LSTART}[l] \leftarrow \texttt{TBOT}$ и $t \leftarrow \texttt{LSTART}[l-1]$. Пока $t > \texttt{TBOT}$, составлять запросы для будущих уровней, выполняя следующие действия.

Установить $t \leftarrow t-2$, $f \leftarrow \texttt{LEFT}(t)$, $g \leftarrow \texttt{RIGHT}(t)$, $vf \leftarrow \texttt{V}(f_0+f)$, $vg \leftarrow \texttt{V}(g_0+g)$,

$ll \leftarrow \mathit{find_level}((vf \le vg?\ \texttt{LO}(f_0+f)\colon f), (vf \ge vg?\ \texttt{LO}(g_0+g)\colon g))$,
$lh \leftarrow \mathit{find_level}((vf \le vg?\ \texttt{HI}(f_0+f)\colon f), (vf \ge vg?\ \texttt{HI}(g_0+g)\colon g))$.

Если $ll \le 0$, установить $\texttt{L}(t) \leftarrow -ll$; в противном случае установить $\texttt{L}(t) \leftarrow \texttt{LLIST}[ll]$, $\texttt{LLIST}[ll] \leftarrow t$, $\texttt{LCOUNT}[ll] \leftarrow \texttt{LCOUNT}[ll] + 1$.
Если $lh \le 0$, установить $\texttt{H}(t) \leftarrow -lh$; в противном случае установить $\texttt{H}(t) \leftarrow \texttt{HLIST}[lh]$, $\texttt{HLIST}[lh] \leftarrow t$, $\texttt{LCOUNT}[lh] \leftarrow \texttt{LCOUNT}[lh] + 1$.

S3. [Выполнен ли первый этап?] Перейти к шагу S6, если $l = v_{\max}$. В противном случае установить $l \leftarrow l + 1$ и вернуться к шагу S2, если $\texttt{LCOUNT}[l] = 0$.

S4. [Инициализация для хеширования.] Установить $b \leftarrow \lceil \lg \texttt{LCOUNT}[l] \rceil$, $\texttt{HBASE} \leftarrow \texttt{TBOT} - 2^{b+1}$ и $\texttt{H}(\texttt{HBASE} + 2k) \leftarrow \Lambda$ для $0 \le k < 2^b$.

S5. [Создание шаблонов для уровня l.] Установить $t \leftarrow \texttt{LLIST}[l]$. Пока $t \ne \Lambda$, устанавливать $s \leftarrow \texttt{L}(t)$, $f \leftarrow \texttt{LEFT}(t)$, $g \leftarrow \texttt{RIGHT}(t)$, $vf \leftarrow \texttt{V}(f_0+f)$, $vg \leftarrow \texttt{V}(g_0+g)$, $\texttt{L}(t) \leftarrow \mathit{make_template}((vf \le vg?\ \texttt{LO}(f_0+f)\colon f), (vf \ge vg?\ \texttt{LO}(g_0+g)\colon g))$, $t \leftarrow s$. (Половина сделана.) Затем установить $t \leftarrow \texttt{HLIST}[l]$. Пока $t \ne \Lambda$, устанавливать $s \leftarrow \texttt{H}(t)$, $f \leftarrow \texttt{LEFT}(t)$, $g \leftarrow \texttt{RIGHT}(t)$, $vf \leftarrow \texttt{V}(f_0+f)$, $vg \leftarrow \texttt{V}(g_0+g)$, $\texttt{H}(t) \leftarrow \mathit{make_template}((vf \le vg?\ \texttt{HI}(f_0+f)\colon f), (vf \ge vg?\ \texttt{HI}(g_0+g)\colon g))$, $t \leftarrow s$. (Теперь сделана вторая половина.) Вернуться к шагу S2.

S6. [Подготовка ко второму этапу.] (В этот момент можно безопасно стереть узлы f и g, поскольку мы уже построили все шаблоны (43). Теперь мы будем приводить их к виду (44). Заметим, что $\texttt{V}(f_0) = \texttt{V}(f_0+1) = v_{\max} + 1$.) Установить $\texttt{NTOP} \leftarrow f_0 + 2$.

S7. [Карманная сортировка.] Установить $t \leftarrow \texttt{LSTART}[l-1]$. Пока $t > \texttt{LSTART}[l]$, выполнять следующие действия.

> Установить $t \leftarrow t-2$, $\texttt{L}(t) \leftarrow \texttt{RIGHT}(\texttt{L}(t))$ и $\texttt{H}(t) \leftarrow \texttt{RIGHT}(\texttt{H}(t))$.
> Если $\texttt{L}(t) = \texttt{H}(t)$, установить $\texttt{RIGHT}(t) \leftarrow \texttt{L}(t)$. (Это ветвление избыточно.)
> В противном случае установить $\texttt{RIGHT}(t) \leftarrow -1$, $\texttt{LEFT}(t) \leftarrow \texttt{LEFT}(\texttt{L}(t))$, $\texttt{LEFT}(\texttt{L}(t)) \leftarrow t$.

S8. [Восстановление адресов клонов.] Если $t = \texttt{LSTART}[l-1]$, установить $t \leftarrow \texttt{LSTART}[l] - 2$ и перейти к шагу S9. В противном случае, если $\texttt{LEFT}(t) < 0$, установить $\texttt{LEFT}(\texttt{L}(t)) \leftarrow \texttt{LEFT}(t)$. Установить $t \leftarrow t+2$ и повторить шаг S8.

S9. [Уровень завершен?] Установить $t \leftarrow t+2$. Если $t = \texttt{LSTART}[l-1]$, перейти к шагу S12. В противном случае, если $\texttt{RIGHT}(t) \geq 0$, повторить шаг S9.

S10. [Проверка кармана.] (Допустим, что $\texttt{L}(t_1) = \texttt{L}(t_2) = \texttt{L}(t_3)$, где $t_1 > t_2 > t_3 = t$ и на уровне l нет иных шаблонов с данным значением $\texttt{L}$. Тогда в этот момент мы имеем $\texttt{LEFT}(t_3) = t_2$, $\texttt{LEFT}(t_2) = t_1$, $\texttt{LEFT}(t_1) < 0$ и $\texttt{RIGHT}(t_1) = \texttt{RIGHT}(t_2) = \texttt{RIGHT}(t_3) = -1$.) Установить $s \leftarrow t$. Пока $s > 0$, выполнять следующие действия. Установить $r \leftarrow \texttt{H}(s)$, $\texttt{RIGHT}(s) \leftarrow \texttt{LEFT}(r)$; если $\texttt{LEFT}(r) < 0$, установить $\texttt{LEFT}(r) \leftarrow s$ и установить $s \leftarrow \texttt{LEFT}(s)$. Наконец вновь установить $s \leftarrow t$.

S11. [Клонирование.] Если $s < 0$, вернуться к шагу S9. В противном случае при $\texttt{RIGHT}(s) \geq 0$ установить $s \leftarrow \texttt{LEFT}(s)$. В противном случае установить $r \leftarrow \texttt{LEFT}(s)$, $\texttt{LEFT}(\texttt{H}(s)) \leftarrow \texttt{RIGHT}(s)$, $\texttt{RIGHT}(s) \leftarrow s$, $q \leftarrow \texttt{NTOP}$, $\texttt{NTOP} \leftarrow q+1$, $\texttt{LEFT}(s) \leftarrow \sim(q - f_0)$, $\texttt{LO}(q) \leftarrow \sim\texttt{LEFT}(\texttt{L}(s))$, $\texttt{HI}(q) \leftarrow \sim\texttt{LEFT}(\texttt{H}(s))$, $\texttt{V}(q) \leftarrow l$, $s \leftarrow r$. Повторить шаг S11.

S12. [Цикл по l.] Установить $l \leftarrow l-1$. Вернуться к шагу S7, если $\texttt{LSTART}[l] < \texttt{POOLSIZE}$. В противном случае при $\texttt{RIGHT}(\texttt{POOLSIZE}-2) = 0$ установить $\texttt{NTOP} \leftarrow \texttt{NTOP} - 1$ (поскольку $f \circ g$ тождественно равно 0). ▮

Как обычно, наилучший способ понять алгоритм наподобие данного — это выполнить его трассировку на небольшом примере. В упр. 67 рассматриваются действия алгоритма S при вычислении $f \wedge g$ для бинарных диаграмм решений, заданных в (41).

Алгоритм S может использоваться, например, для построения бинарных диаграмм решений для таких интересных функций, как "функция монотонной функции" $\mu_n(x_1, \ldots, x_{2^n})$, которая истинна тогда и только тогда, когда $x_1 \ldots x_{2^n}$ является таблицей истинности монотонной функции:

$$\mu_n(x_1, \ldots, x_{2^n}) = \bigwedge_{0 \le i \subseteq j < 2^n} [x_{i+1} \le x_{j+1}]. \tag{48}$$

Начиная с $\mu_0(x_1) = 1$ эта функция удовлетворяет рекурсивному соотношению

$$\mu_n(x_1, \ldots, x_{2^n}) = \mu_{n-1}(x_1, x_3, \ldots, x_{2^n-1}) \wedge \mu_{n-1}(x_2, x_4, \ldots, x_{2^n}) \wedge G_{2^n}(x_1, \ldots, x_{2^n}), \tag{49}$$

где $G_{2^n}(x_1, \ldots, x_{2^n}) = [x_1 \le x_2] \wedge [x_3 \le x_4] \wedge \cdots \wedge [x_{2^n-1} \le x_{2^n}]$. Так что ее бинарную диаграмму решений легко получить с помощью калькулятора БДР наподобие алгоритма S: БДР для $\mu_{n-1}(x_1, x_3, \ldots, x_{2^n-1})$ и $\mu_{n-1}(x_2, x_4, \ldots, x_{2^n})$ являются простыми вариантами БДР для $\mu_{n-1}(x_1, x_2, \ldots, x_{2^{n-1}})$, а G_{2^n} имеет исключительно простую БДР (рис. 25).

Рис. 25. $\mu_2(x_1,x_3,x_5,x_7)\wedge\mu_2(x_2,x_4,x_6,x_8)\wedge G_8(x_1,\ldots,x_8) = \mu_3(x_1,\ldots,x_8)$; вычисление с помощью алгоритма S.

Повторение этого процесса шесть раз даст БДР для μ_6, которая имеет 103 924 узла. Имеется в точности 7 828 354 монотонные булевы функции от шести переменных (см. упр. 5.3.4–31); эта БДР отлично характеризует их все, и для ее вычисления с помощью алгоритма S необходимо около 4.8 миллиона обращений к памяти. Кроме того, 6.7 миллиарда обращений к памяти будет достаточно для вычисления БДР для μ_7, которая имеет 155 207 320 узлов и характеризует 2 414 682 040 998 монотонных функций.

На этом нам следует остановиться, поскольку $B(\mu_8)$ имеет колоссальное значение 69 258 301 585 604 (см. упр. 77).

Синтез в базе БДР. Другой подход примени́м, когда мы работаем со многими функциями одновременно, вместо вычисления одной БДР "на лету". Функции базы БДР часто совместно используют общие подфункции, как в (36). Алгоритм S разработан так, чтобы, взяв две непересекающиеся БДР, эффективно их объединить, с последующим уничтожением оригиналов; но во многих случаях желательно образовывать комбинации функций, БДР которых перекрываются. Кроме того, после образования новой функции, скажем, $f\wedge g$, мы можем захотеть сохранить f и g для будущего использования; более того, новая функция может также совместно использовать узлы вместе с функцией f или g или с обеими.

Поэтому давайте рассмотрим разработку инструментария общего назначения, предназначенного для работы с коллекцией булевых функций. Для этой цели БДР в особенности привлекательны, поскольку большинство необходимых операций имеют простую рекурсивную формулировку. Мы знаем, что каждая неконстантная булева функция может быть записана как

$$f(x_1,x_2,\ldots,x_n) = (\bar{x}_v?\ f_l\colon f_h), \tag{50}$$

где $v=f_v$ индексирует первую переменную, от которой зависит f, и где мы имеем

$$f_l = f(0,\ldots,0,x_{v+1},\ldots,x_n);\quad f_h = f(1,\ldots,1,x_{v+1},\ldots,x_n). \tag{51}$$

Это правило соответствует узлу ветвления $\textcircled{v}$ на вершине БДР для f; а остаток БДР получается рекурсивным использованием (50) и (51), пока мы не достигнем константных функций, соответствующих $\boxed{\bot}$ или $\boxed{\top}$. Подобная рекурсия определяет любые комбинации функций $f \circ g$: если f и g не являются одновременно константами, то мы имеем

$$f(x_1, \ldots, x_n) = (\bar{x}_v ?\ f_l \colon f_h) \quad \text{и} \quad g(x_1, \ldots, x_n) = (\bar{x}_v ?\ g_l \colon g_h), \tag{52}$$

где $v = \min(f_v, g_v)$ и где f_l, f_h, g_l, g_h задаются формулами (51). Тогда тут же получается

$$f \circ g \ = \ (\bar{x}_v ?\ f_l \circ g_l \colon f_h \circ g_h). \tag{53}$$

Эта важная формула представляет собой другой способ записи правила (37), определяющего слияние.

Предостережение. Следует правильно понимать записанное выше: подфункции f_l и f_h в (50) могут не быть теми же, что и f_l и f_h в (52). Предположим, например, что $f = x_2 \vee x_3$, в то время как $g = x_1 \oplus x_3$. Тогда уравнение (50) выполняется с $f_v = 2$ и $f = (\bar{x}_2 ?\ f_l \colon f_h)$, где $f_l = x_3$ и $f_h = 1$. Мы также имеем $g_v = 1$ и $g = (\bar{x}_1 ?\ x_3 \colon \bar{x}_3)$. Но в (52) мы используем одну и ту же переменную ветвления x_v для обеих функций, и в нашем примере $v = \min(f_v, g_v) = 1$; так что уравнения (52) выполняются с $f = (\bar{x}_1 ?\ f_l \colon f_h)$ и $f_l = f_h = x_2 \vee x_3$.

Каждый узел базы БДР представляет булеву функцию. Кроме того, база БДР является приведенной; следовательно, две из ее функций или подфункций равны тогда и только тогда, когда они соответствуют в точности одному и тому же узлу. (Это удобное свойство уникальности *не* выполнялось в алгоритме S.)

Формулы (51)–(53) непосредственно предлагают рекурсивный способ вычисления $f \wedge g$:

$$\text{И}(f, g) = \begin{cases} \text{Если } f \wedge g \text{ имеет очевидное значение, вернуть его.} \\ \text{В противном случае представить } f \text{ и } g\text{, как в (52);} \\ \text{вычислить } r_l \leftarrow \text{И}(f_l, g_l) \text{ и } r_h \leftarrow \text{И}(f_h, g_h); \\ \text{вернуть функцию } (\bar{x}_v ?\ r_l \colon r_h). \end{cases} \tag{54}$$

(Рекурсии всегда должны завершаться при достижении достаточно простых ситуаций. “Очевидные” значения в первой строке соответствуют терминальным случаям $f \wedge 1 = f$, $1 \wedge g = g$, $f \wedge 0 = 0 \wedge g = 0$ и $f \wedge g = f$ при $f = g$.) Когда f и g являются функциями из нашего примера выше, (54) сводит $f \wedge g$ к вычислению $(x_2 \vee x_3) \wedge x_3$ и $(x_2 \vee x_3) \wedge \bar{x}_3$. Затем $(x_2 \vee x_3) \wedge x_3$ сводится к $x_3 \wedge x_3$ и $1 \wedge x_3$; и т. д.

Однако при попытке непосредственной реализации (54) “как есть” мы столкнемся с проблемами, поскольку каждый нетерминальный шаг приводит к появлению двух экземпляров рекурсии. Вычисления просто взрываются, приводя на глубине k-го уровня к 2^k экземплярам И!

К счастью, имеется хороший способ избежать этого взрыва. Поскольку f имеет только $B(f)$ различных подфункций, может осуществиться не более $B(f)B(g)$ явно различных вызовов И. Чтобы ограничить количество выполняемых вычислений, следует просто запоминать, что мы уже делали раньше, *занося в память* тот факт, что возвращаемое значение r получается в результате вычисления $f \wedge g = r$. В таком случае, когда та же самая подзадача возникнет позже, можно будет прочесть запись и сказать: “Да ведь мы это уже делали раньше!” Таким образом, ранее сохраненные

вычисления превращаются в терминалы, и только отличающиеся от уже решенных подзадачи могут требовать новых вычислений. (Технология вычислений с запоминанием (memoization) будет детально рассмотрена в главе 8.)

Алгоритм в (54) решает и другую задачу: не так просто "вернуть функцию $(\bar{x}_v?\ r_l\colon\ r_h)$", поскольку мы должны поддерживать базу БДР приведенной. Если $r_l = r_h$, мы должны вернуть узел r_l; а если $r_l \neq r_h$, нам нужно выяснить, не существует ли уже узел ветвления $(\bar{x}_v?\ r_l\colon\ r_h)$, прежде чем создавать новый.

Таким образом, необходимо поддерживать дополнительную информацию, помимо самих узлов БДР. Нам нужны записи об уже решенных задачах; нам также нужно уметь находить узел по его содержимому, а не по адресу. Здесь на помощь приходят алгоритмы из главы 6, указывающие, как выполнить оба эти поиска, например, с помощью хеширования. Для записи информации о том, что $f \wedge g = r$, можно хешировать ключ '$(f, \wedge, g)$' и связать его со значением r; для записи существования имеющегося узла $(\mathrm{V}, \mathrm{LO}, \mathrm{HI})$ можно хешировать ключ '$(\mathrm{V}, \mathrm{LO}, \mathrm{HI})$' и связать его с адресом узла в памяти.

Словарь всех существующих узлов $(\mathrm{V}, \mathrm{LO}, \mathrm{HI})$ в базе БДР традиционно именуется *таблицей уникальности* (*unique table*), поскольку мы используем ее для обеспечения крайне важного критерия уникальности, запрещающего дублирование. Однако оказывается, что вместо помещения всей информации в один гигантский словарь лучше поддерживать коллекцию меньших таблиц уникальности, по одной для каждой переменной V. При наличии таких отдельных таблиц можно эффективно находить все узлы, выполняющие переходы по конкретной переменной.

Вычисления с записью удобны, но не так критичны, как записи таблицы уникальности. Если так случится, что мы забудем записать отдельный факт, что $f \wedge g = r$, то мы всегда сможем выполнить вычисления заново. Экспоненциальный взрыв не должен нас беспокоить, если ответы к подзадачам $f_l \wedge g_l$ и $f_h \wedge g_h$ записываются с достаточно высокой вероятностью. Следовательно, можно использовать менее дорогостоящие методы запоминания, разработанные для выполнения "хорошей, но не идеальной" работы по выборке данных: после хеширования ключа '$(f, \wedge, g)$' в позицию таблицы p нужно просмотреть записи только в этой одной позиции, не беспокоясь о возможных коллизиях с другими ключами. Если у нескольких ключей оказывается один и тот же хеш-адрес, позиция p будет содержать только последнюю запись. Такая упрощенная схема остается вполне адекватной на практике, если хеш-таблица достаточно велика. Назовем такую близкую к идеальной таблицу *кэшем записей*, поскольку она аналогична аппаратным кэшам, с помощью которых компьютер пытается запомнить важные значения, для получения которых приходится работать с относительно медленными устройствами хранения.

Итак, давайте конкретизируем алгоритм (54), явно указав, как он взаимодействует с таблицами уникальности и кэшем записей.

$$\text{И}(f,g) = \begin{cases} \text{Если } f \wedge g \text{ имеет очевидное значение, вернуть его.} \\ \text{В противном случае: если } f \wedge g = r \text{ находится в кэше} \\ \text{записей, вернуть } r. \text{ Иначе представить } f \text{ и } g, \text{ как в (52);} \\ \text{вычислить } r_l \leftarrow \text{И}(f_l, g_l) \text{ и } r_h \leftarrow \text{И}(f_h, g_h); \\ \text{установить } r \leftarrow \text{UNIQUE}(v, r_l, r_h), \text{ используя алгоритм U;} \\ \text{поместить '} f \wedge g = r \text{' в кэш записей и вернуть } r. \end{cases} \tag{55}$$

Алгоритм U (*Поиск в таблице уникальности*). Для заданного (v, p, q), где v представляет собой целое число, а p и q указывают на узлы базы БДР с рангом переменных $> v$, этот алгоритм возвращает указатель на узел $\mathrm{UNIQUE}(v, p, q)$, который представляет функцию $(\bar{x}_v ?\ p\colon q)$. Если этой функции еще нет в наличии, в базу добавляется новый узел.

U1. [Простой случай?] Если $p = q$, вернуть p.

U2. [Проверка таблицы.] Выполнить поиск ключа (p, q) в таблице уникальности переменной x_v. Если поиск успешен и находит значение r, вернуть r.

U3. [Создание узла.] Выделить память для нового узла r и установить $\mathtt{V}(r) \leftarrow v$, $\mathtt{LO}(r) \leftarrow p$, $\mathtt{HI}(r) \leftarrow q$. Поместить r в таблицу уникальности x_v с использованием ключа (p, q). Вернуть r. ▮

Обратите внимание, что обнулять кэш записей после окончания вычисления верхнего уровня $\text{И}(f, g)$ не нужно. Каждая запись, которую мы создали, указывает на соотношение между узлами в структуре; эти факты остаются корректными и могут оказаться полезными позднее, когда мы захотим вычислить $\text{И}(f, g)$ для новых функций f и g.

Небольшое усовершенствование (55) приведет к дальнейшему улучшению этого метода. Это обмен $f \leftrightarrow g$ в случае, когда мы выясняем, что $f > g$ при неочевидном значении $f \wedge g$. Мы не хотим терять время на вычисление $f \wedge g$, если уже вычислено $g \wedge f$.

Внеся небольшие изменения в (55), можно легко вычислять другие бинарные операторы, такие как $\text{ИЛИ}(f, g)$, $\text{ЛИБО}(f, g)$, $\text{НО-НЕ}(f, g)$, $\text{НЕ-ИЛИ}(f, g)$, ...; см. упр. 81.

Комбинация (55) и алгоритма U выглядит значительно проще алгоритма S. Таким образом, можно спросить, зачем вообще трудиться и изучать другой метод? Подход с поиском в ширину выглядит существенно сложнее вычислений "в глубину" в рекурсивной структуре (55); тем более что алгоритм S способен работать только с непересекающимися бинарными диаграммами решений, в то время как алгоритм U и рекурсии наподобие (55) применимы к любой базе БДР.

Однако внешность может быть обманчивой: алгоритм S был описан на низком уровне, с явным описанием всех изменений каждого элемента структур данных. Напротив, высокоуровневое описание в (55) и алгоритма U предполагает наличие "за сценой" серьезной инфраструктуры. Кэш записей и таблицы уникальности следует настроить, их размеры при росте или сокращении базы БДР должны аккуратно изменяться. Когда все это будет сделано, общая длина программы, корректно реализующей алгоритмы (55) и U "с нуля", будет грубо на порядок большей, чем аналогичная программа для алгоритма S.

На самом деле поддержка базы БДР влечет за собой интересные вопросы динамического распределения памяти, поскольку мы хотим освобождать память, когда узлы становятся более недостижимыми. Алгоритм S решает эту проблему способом "последним вошел, первым вышел", просто содержа узлы и шаблоны в последовательных стеках и работая с единственной небольшой хеш-таблицей, которая может быть легко интегрирована с другими данными. Однако обобщенная база БДР требует более сложной системы.

Пожалуй, наилучший способ поддержки динамической базы БДР заключается в использовании *счетчиков ссылок*, рассматривавшихся в разделе 2.3.5, поскольку бинарные диаграммы решений ацикличны по определению. Таким образом, будем считать, что каждый узел БДР в дополнение к полям V, LO и HI имеет поле REF. Это поле REF говорит нам о том, сколько имеется ссылок на этот узел — из указателей LO или HI других узлов или из внешних корневых указателей F_j, как в (36). Например, поля REF для узлов, помеченных 3 в (36) равны соответственно 4, 1 и 2; а все узлы, помеченные 2, 4 или 6 в этом же примере, имеют REF = 1. В упр. 82 рассматривается такой непростой вопрос, как корректное увеличение и уменьшение счетчиков REF в процессе рекурсивных вычислений.

Узел становится *мертвым*, когда его счетчик ссылок становится равным нулю. Когда это происходит, нужно уменьшить поля REF двух узлов ниже его; они также могут "умереть" тем же образом, рекурсивно распространяя эпидемию уничтожения узлов.

Однако мертвый узел не требует немедленного удаления из памяти. Он все еще представляет потенциально полезную булеву функцию, и может оказаться, что эта функция вновь потребуется нам при проведении вычислений. Например, на шаге U2 можно обнаружить мертвый узел, поскольку указатели из таблицы уникальности не подсчитываются, как ссылки. Аналогично в (55) мы можем случайно обнаружить, что кэш записей говорит нам о том, что $f \wedge g = r$, в то время как r в настоящее время мертв. В таких случаях узел r возвращается к жизни. (И мы должны увеличить счетчики REF его потомков LO и HI, возможно, заставив их при этом рекурсивно выполнить те же действия по отношению к своим потомкам.)

Периодически, однако, хотелось бы очищать память, удаляя мертвецов. В таких случаях следует выполнять две вещи: убирать из кэша все записи, в которых f, g либо r мертвы, а также удалять все мертвые узлы из памяти и из их таблиц уникальности. В упр. 84 рассматриваются типичные эвристические стратегии, с помощью которых автоматизированные системы могут решать, когда следует прибегать к такой очистке и когда надо динамически изменять размеры таблиц.

Из-за дополнительных механизмов, необходимых для поддержки базы БДР, нельзя ожидать, что эффективность алгоритма U и нисходящих рекурсий типа (55) сравняется с эффективностью алгоритма S в таких единичных примерах, как функция монотонной функции μ_n из (49). Время работы при применении к этому примеру более общего подхода оказывается примерно учетверенным, а требования к памяти вырастают примерно в 2.4 раза.

База БДР начинает реально проявлять себя в ряде других приложений. Предположим, например, что нам нужны формулы для каждого бита произведения двух бинарных чисел,

$$(z_1 \ldots z_{m+n})_2 \;=\; (x_1 \ldots x_m)_2 \times (y_1 \ldots y_n)_2. \tag{56}$$

Ясно, что $z_1 \ldots z_m = 0 \ldots 0$ при $n = 0$ и что простое рекуррентное соотношение

$$(x_1 \ldots x_m)_2 \times (y_1 \ldots y_n y_{n+1})_2 \;=\; (z_1 \ldots z_{m+n} 0)_2 + (x_1 \ldots x_m)_2 y_{n+1} \tag{57}$$

позволяет увеличить n на 1. Это рекуррентное соотношение легко закодировать для базы БДР. Далее показано, что́ мы получаем при $m = n = 3$ с индексами, выбранными таким образом, чтобы соответствовать аналогичной диаграмме для

бинарного сложения из (36).

$$\begin{array}{r} x_1x_3x_5 \\ \times\ x_2x_4x_6 \\ \hline \cdot\ \cdot\ \cdot \\ \cdot\ \cdot\ \cdot\ \\ \cdot\ \cdot\ \cdot\ \ \\ \hline f_6f_5f_4f_3f_2f_1 \end{array}$$

(58)

Ясно, что битовое умножение гораздо сложнее битового сложения. (Действительно, если бы это было не так, то разложение на множители не было бы таким сложным.) Соответствующая база БДР для бинарного умножения при $m = n = 16$ оказывается огромной, с $B(f_1, \ldots, f_{32}) = 136\,398\,751$ узлом. Ее можно получить после выполнения порядка 56 миллиардов обращений к памяти при вычислениях с помощью алгоритма U, при этом требуется 6.3 Гбайт памяти и выполняется около 1.9 миллиарда вызовов рекурсивных процедур с сотнями динамических изменений размеров таблиц уникальности и кэша записей, а также десятки длительных сборок мусора. Аналогичные вычисления с использованием алгоритма S практически невообразимы, хотя отдельные функции в этом конкретном примере не так часто совместно используют общие подфункции: оказывается, что $B(f_1) + \cdots + B(f_{32}) = 168\,640\,131$, с максимумом в "среднем бите", $B(f_{16}) = 38\,174\,143$.

***Тернарные операции.** Если даны три булевы функции, $f = f(x_1, \ldots, x_n)$, $g = g(x_1, \ldots, x_n)$ и $h = h(x_1, \ldots, x_n)$, из которых не все являются константными, можно обобщить (52) до

$$f = (\bar{x}_v ?\ f_l \colon f_h) \quad \text{и} \quad g = (\bar{x}_v ?\ g_l \colon g_h) \quad \text{и} \quad h = (\bar{x}_v ?\ h_l \colon h_h), \tag{59}$$

беря $v = \min(f_v, g_v, h_v)$. Тогда, например, (53) обобщается до

$$\langle fgh \rangle \;=\; \big(\bar{x}_v ?\ \langle f_l g_l h_l \rangle \colon \langle f_h g_h h_h \rangle\big); \tag{60}$$

аналогичные формулы выполняются для *любых* тернарных операций над f, g и h, включая

$$(\bar{f} ?\ g \colon h) \;=\; \big(\bar{x}_v ?\ (\bar{f}_l ?\ g_l \colon h_l) \colon (\bar{f}_h ?\ g_h \colon h_h)\big). \tag{61}$$

(Читателя этих формул просим простить два разных значения 'h' в 'h_h'.)

Теперь легко обобщить (55) на тернарные комбинации наподобие мультиплексирования.

$$\mathrm{MUX}(f, g, h) = \begin{cases} \text{Если } (\bar{f} ?\ g \colon h) \text{ имеет очевидное значение, вернуть его.} \\ \text{В противном случае если } (\bar{f} ?\ g \colon h) = r \text{ находится} \\ \text{в кэше записей, вернуть } r. \\ \text{В противном случае представить } f,\ g \text{ и } h \text{ как в (59);} \\ \text{вычислить } r_l \leftarrow \mathrm{MUX}(f_l, g_l, h_l) \text{ и } r_h \leftarrow \mathrm{MUX}(f_h, g_h, h_h); \\ \text{установить } r \leftarrow \mathrm{UNIQUE}(v, r_l, r_h), \text{ используя алгоритм U;} \\ \text{поместить '}(\bar{f} ?\ g \colon h) = r\text{' в кэш записей и вернуть } r. \end{cases} \tag{62}$$

(См. упр. 86 и 87.) Время работы составляет $O\bigl(B(f)B(g)B(h)\bigr)$. Кэш записей теперь должен работать с более сложными ключами, чем ранее, содержащими *три* указателя (f,g,h) вместо двух, вместе с кодом соответствующей операции. Но каждая запись (op,f,g,h,r) все еще может быть удобно представлена, скажем, двумя октабайтами, если количество различных адресов указателей не превышает 2^{31}.

Интересным частным случаем является тернарная операция $f\wedge g\wedge h$. Ее можно вычислить с помощью двух вызовов (55), либо как $\text{И}(f,\text{И}(g,h))$, либо как $\text{И}(g,\text{И}(h,f))$, либо как $\text{И}(h,\text{И}(f,g))$; либо можно использовать тернарную подпрограмму $\text{И-И}(f,g,h)$, аналогичную (62). Эта тернарная подпрограмма сначала сортирует операнды таким образом, чтобы указатели удовлетворяли соотношению $f\le g\le h$. Тогда, если $f=0$, она возвращает 0; если $f=1$ или $f=g$, она возвращает $\text{И}(g,h)$; если $g=h$, она возвращает $\text{И}(f,g)$; в противном случае $1<f<g<h$ и операция на данном уровне рекурсии остается тернарной.

Предположим, например, что $f=\mu_5(x_1,x_3,\ldots,x_{63})$, $g=\mu_5(x_2,x_4,\ldots,x_{64})$ и $h=G_{64}(x_1,\ldots,x_{64})$, как в (49). Вычисление $\text{И}(f,\text{И}(g,h))$ в экспериментальной реализации автора имеет стоимость $0.2+6.8=7.0$ миллионов обращений к памяти; $\text{И}(g,\text{И}(h,f))$ имеет стоимость $0.1+7.0=7.1$; $\text{И}(h,\text{И}(f,g))$ имеет стоимость $24.4+5.6=30.0$ (!); а $\text{И-И}(f,g,h)$ имеет стоимость 7.5. Так что в данном случае побеждает бинарный подход, если только не выбрать плохой порядок вычислений. Но иногда тернарная операция И-И выигрывает у всех трех бинарных соперников (см. упр. 88).

***Кванторы.** Если $f=f(x_1,\ldots,x_n)$ представляет собой булеву функцию и $1\le j\le n$, логики традиционно определяют *квантификацию существования и всеобщности* с помощью формул

$$\exists x_j\, f(x_1,\ldots,x_n)=f_0\vee f_1 \quad \text{и} \quad \forall x_j\, f(x_1,\ldots,x_n)=f_0\wedge f_1, \tag{63}$$

где $f_c=f(x_1,\ldots,x_{j-1},c,x_{j+1},\ldots,x_n)$. Таким образом, квантор '$\exists x_j$' (произносится как "существует x_j, такое, что") заменяет f функцией от остальных переменных $(x_1,\ldots,x_{j-1},x_{j+1},\ldots,x_n)$, которая истинна тогда и только тогда, когда как минимум одно значение x_j удовлетворяет $f(x_1,\ldots,x_n)$; квантор '$\forall x_j$' (произносится как "для всех x_j") заменяет f функцией, которая истинна тогда и только тогда, когда, когда *оба* значения x_j удовлетворяют f.

Зачастую одновременно применяется несколько кванторов. Например, формула $\exists x_2\,\exists x_3\,\exists x_6\, f(x_1,\ldots,x_n)$ означает ИЛИ восьми членов, представленных восемью функциями от переменных $(x_1,x_4,x_5,x_7,\ldots,x_n)$, которые получаются, когда мы подставляем значения 0 или 1 вместо переменных x_2, x_3 и x_6 всеми возможными способами. Аналогично $\forall x_2\,\forall x_3\,\forall x_6\, f(x_1,\ldots,x_n)$ означает И с теми же восемью членами.

Одно из распространенных приложений — когда функция $f(i_1,\ldots,i_l;j_1,\ldots,j_m)$ представляет собой значение на пересечении строки $(i_1\ldots i_l)_2$ и столбца $(j_1\ldots j_m)_2$ булевой матрицы F размером $2^l\times 2^m$. Тогда функция $h(i_1,\ldots,i_l;k_1,\ldots,k_n)$, задаваемая как

$$\exists j_1\ldots\exists j_m\bigl(f(i_1,\ldots,i_l;j_1,\ldots,j_m)\wedge g(j_1,\ldots,j_m;k_1,\ldots,k_n)\bigr) \tag{64}$$

представляет матрицу H, являющуюся булевым произведением $F\,G$.

Удобный способ реализовать множественную квантификацию в базе БДР был предложен Р. Л. Руделлом (R. L. Rudell). Пусть $g=x_{j_1}\wedge\cdots\wedge x_{j_m}$ представ-

ляет собой конъюнкцию положительных литералов. Тогда мы можем рассматривать $\exists x_{j_1} \ldots \exists x_{j_m} f$ как бинарную операцию $f \,\mathrm{E}\, g$, реализуемую следующим вариантом (55).

$$\mathrm{EXISTS}(f,g) = \begin{cases} \text{Если } f \,\mathrm{E}\, g \text{ имеет очевидное значение, вернуть его.} \\ \text{В противном случае представить } f \text{ и } g \text{ как в (52);} \\ \text{если } v \neq f_v, \text{ вернуть } \mathrm{EXISTS}(f, g_h). \\ \text{В противном случае если } f \,\mathrm{E}\, g = r \text{ находится в кэше записей,} \\ \text{вернуть } r. \\ \text{В противном случае, } r_l \leftarrow \mathrm{EXISTS}(f_l, g_h) \text{ и } r_h \leftarrow \mathrm{EXISTS}(f_h, g_h); \\ \text{если } v \neq g_v, \text{ установить } r \leftarrow \mathrm{UNIQUE}(v, r_l, r_h) \\ \text{с использованием алгоритма U,} \\ \text{в противном случае вычислить } r \leftarrow \text{ИЛИ}(r_l, r_h); \\ \text{поместить '} f \,\mathrm{E}\, g = r \text{' в кэш записей и вернуть } r. \end{cases} \tag{65}$$

(См. упр. 94.) Операция E не определена, когда g *не* имеет указанного вида. Обратите внимание, как хорошо кэш записей восстанавливает вычисления существования, выполнявшиеся ранее.

Время работы (65) очень изменчиво (в отличие от (55), где мы знали, что наихудшим возможным случаем является $O(B(f)B(g))$), поскольку, когда g определяет m-кратную квантификацию, выполняется m операций ИЛИ. Теперь наихудший случай может быть порядка $B(f)^{2^m}$, если вся квантификация осуществляется вблизи корня БДР для f; это значение составляет всего лишь $O(B(f)^2)$ при $m = 1$, но с ростом m может стать невероятно огромным. С другой стороны, если все квантификации осуществляются вблизи стоков, то время работы превращается в просто $O(B(f))$, независимо от размера m. (См. упр. 97.)

Сто́ит упомянуть и несколько других столь же простых кванторов, хотя они и не так известны, как $\exists$ и $\forall$. *Булева разность* и *кванторы да/нет* определяются формулами, аналогичными (63):

$$\mathsf{D} x_j\, f = f_0 \oplus f_1; \qquad \curlywedge x_j\, f = \bar{f}_0 \wedge f_1; \qquad \mathsf{N} x_j\, f = f_0 \wedge \bar{f}_1. \tag{66}$$

Булева разность, D, — наиболее важный из указанных кванторов: $\mathsf{D} x_j\, f$ истинно для всех значений $\{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\}$, таких, что f зависит от x_j. Если мультилинейное представление f имеет вид $f = (x_j g + h) \bmod 2$, где g и h являются мультилинейными полиномами от $\{x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n\}$, то $\mathsf{D} x_j\, f = g \bmod 2$. (См. 7.1.1–(19).) Таким образом, D действует над конечным полем как производная в дифференциальном исчислении.

Булева функция $f(x_1, \ldots, x_n)$ является монотонной (неубывающей) тогда и только тогда, когда $\bigvee_{j=1}^n \mathsf{N} x_j\, f = 0$, что то же самое, что сказать, что $\mathsf{N} x_j\, f = 0$ для всех j. Однако в упр. 105 представлен более быстрый способ проверки БДР на монотонность.

Давайте теперь рассмотрим особенно поучительный детальный пример квантификации существования. Если G представляет собой произвольный граф, можно образовать булевы функции $\mathrm{IND}(x)$ и $\mathrm{KER}(x)$ для его независимых множеств и ядер следующим образом, где x — битовый вектор с одной записью x_v для каждой вер-

шины v графа G:

$$\mathrm{IND}(x) = \neg \bigvee_{u-v} (x_u \wedge x_v); \tag{67}$$

$$\mathrm{KER}(x) = \mathrm{IND}(x) \wedge \bigwedge_{v} \Big(x_v \vee \bigvee_{u-v} x_u\Big). \tag{68}$$

Можно образовать новый граф $\mathcal{G}$, вершины которого являются ядрами G, а именно векторами x, такими, что $\mathrm{KER}(x) = 1$. Будем говорить, что два ядра, x и y, являются *смежными* в $\mathcal{G}$, если они отличаются только в двух записях для u и v, где $(x_u, x_v) = (1, 0)$ и $(y_u, y_v) = (0, 1)$, и в этом случае мы также имеем u — v. Ядра можно рассматривать как определенные способы размещения маркеров на вершинах G; перемещение маркера из одной вершины в соседнюю дает смежное ядро. Формально мы определяем

$$\mathrm{ADJ}(x, y) = [\nu(x \oplus y) = 2] \wedge \mathrm{KER}(x) \wedge \mathrm{KER}(y). \tag{69}$$

Тогда x — y в $\mathcal{G}$ тогда и только тогда, когда $\mathrm{ADJ}(x, y) = 1$.

Заметим, что если $x = x_1 \ldots x_n$, то функция $[\nu(x) = 2]$ является симметричной функцией $S_2(x_1, \ldots, x_n)$. Кроме того, $f(x \oplus y)$ имеет узлов не более чем в три раза больше, чем $f(x)$, если чередовать переменные в стиле застежки-молнии, так что порядок ветвления имеет вид $(x_1, y_1, \ldots, x_n, y_n)$. Так что $B(\mathrm{ADJ})$ не может быть чрезвычайно большим, если только $B(\mathrm{KER})$ не велико.

Квантификация облегчает задачу выражения условия, что x является *изолированной вершиной* $\mathcal{G}$ (вершиной нулевой степени, ядром без соседей):

$$\mathrm{ISO}(x) \;=\; \mathrm{KER}(x) \wedge \neg\exists y\, \mathrm{ADJ}(x, y). \tag{70}$$

Предположим, например, что G представляет собой граф граничащих штатов США, как в (18). Тогда каждый вектор ядра x имеет 49 записей x_v для $v \in \{\mathtt{ME}, \mathtt{NH}, \ldots, \mathtt{CA}\}$. Граф $\mathcal{G}$ имеет 266 137 вершин, а ранее мы уже находили, что размеры БДР для $\mathrm{IND}(x)$ и $\mathrm{KER}(x)$ равны соответственно 428 и 780 (см. (17)). В этом случае $\mathrm{ADJ}(x, y)$ в (69) имеет бинарную диаграмму решений только с 7260 узлами, несмотря на то что это функция от 98 булевых переменных. БДР для выражения $\exists y\, \mathrm{ADJ}(x, y)$, описывающего все ядра x графа G, которые имеют хотя бы одного соседа, как выясняется, имеет 842 узла; а бинарная диаграмма решений для $\mathrm{ISO}(x)$ имеет их только 77. Мы находим, что G имеет ровно три изолированных ядра, а именно

(71)

а оставшееся является смесью двух приведенных. Полное вычисление, основанное на приведенных выше алгоритмах и начинающееся со списка вершин и ребер G (не $\mathcal{G}$), может быть выполнено с общей стоимостью около 4 миллионов обращений

к памяти и с использованием около 1.6 Мбайт памяти; это составляет всего лишь около 15 обращений к памяти на ядро графа G.

Подобным образом можно использовать бинарные диаграммы решений для работы с другими "неявными графами", которые имеют больше вершин, чем может быть представлено в памяти, если эти вершины могут быть охарактеризованы как векторы решений булевых функций. При не слишком сложных функциях можно отвечать на запросы об этих графах, на которые мы бы никогда не смогли ответить при явном представлении вершин и дуг.

***Функциональная композиция.** *Pièce de résistance** рекурсивных алгоритмов для работы с бинарными диаграммами решений является обобщенная процедура для вычисления $f(g_1, g_2, \ldots, g_n)$, где f — заданная функция от $\{x_1, x_2, \ldots, x_n\}$, и таковой же является каждый аргумент g_j. Предположим, что мы знаем число $m \geq 0$, такое, что $g_j = x_j$ для $m < j \leq n$; тогда процедуру можно выразить следующим образом.

$$\mathrm{COMPOSE}(f, g_1, \ldots, g_n) = \begin{cases} \text{Если } f = 0 \text{ или } f = 1, \text{ вернуть } f. \\ \text{В противном случае положим} \\ \quad f = (\bar{x}_v ?\ f_l\colon f_h), \text{ как в (50);} \\ \text{если } v > m, \text{ вернуть } f; \text{ в противном случае,} \\ \quad \text{если } f(g_1, \ldots, g_n) = r \text{ находится} \\ \quad \text{в кэше записей, вернуть } r. \\ \text{Вычислить } r_l \leftarrow \mathrm{COMPOSE}(f_l, g_1, \ldots, g_n) \\ \quad \text{и } r_h \leftarrow \mathrm{COMPOSE}(f_h, g_1, \ldots, g_n); \\ \text{установить } r \leftarrow \mathrm{MUX}(g_v, r_l, r_h) \\ \quad \text{с использованием (62);} \\ \text{поместить '} f(g_1, \ldots, g_n) = r \text{' в кэш и вернуть } r. \end{cases} \tag{72}$$

Представление записей кэша наподобие '$f(g_1, \ldots, g_n) = r$' в этом алгоритме вызывает определенные сложности; мы вскоре рассмотрим этот вопрос.

Хотя вычисления здесь выглядят в основном такими же, как и те, с которыми мы встречались в предыдущих рекурсиях, в действительности они существенно отличаются: функции r_l и r_h в (72) могут включать *все* переменные $\{x_1, \ldots, x_n\}$, а не только x вблизи нижнего края бинарных диаграмм решений. Так что время работы (72) может оказаться огромным. Но имеется множество случаев, когда все работает гармонично и эффективно. Например, вычисление $[\nu(x \oplus y) = 2]$ в (69) не представляет проблем.

Ключ записи наподобие '$f(g_1, \ldots, g_n) = r$' не должен быть полностью детальной спецификацией $(f, g_1, \ldots, g_n)$, поскольку мы хотим эффективно его хешировать. Поэтому мы храним только '$f[G] = r$', где G представляет собой идентификационный номер последовательности функций $(g_1, \ldots, g_n)$. При изменении этой последовательности можно использовать новый номер G; и мы можем помнить G для отдельных последовательностей функций, многократно встречающихся в определенном вычислении, пока отдельные функции g_j остаются "живыми". (См. также альтернативную схему в упр. 102.)

Вернемся к графу граничащих штатов, как к еще одному примеру. Это планарный граф; предположим, что мы хотим раскрасить его в четыре цвета. Поскольку

* Основное блюдо, лучший кусок (фр.) — *Примеч. пер.*

цвета могут быть заданы двухбитовыми кодами $\{00, 01, 10, 11\}$, легко выразить корректную раскраску как булеву функцию от 98 переменных, которая истинна тогда и только тогда, когда коды цветов ab различны у каждой пары граничащих штатов:

$$\begin{aligned}\mathrm{COLOR}(a_{\mathtt{ME}}, b_{\mathtt{ME}}, \ldots, a_{\mathtt{CA}}, b_{\mathtt{CA}}) \;=\; & \\ \mathrm{IND}(a_{\mathtt{ME}} \wedge b_{\mathtt{ME}}, \ldots, a_{\mathtt{CA}} \wedge b_{\mathtt{CA}}) \wedge \mathrm{IND}(a_{\mathtt{ME}} \wedge \bar{b}_{\mathtt{ME}}, \ldots, a_{\mathtt{CA}} \wedge \bar{b}_{\mathtt{CA}}) & \\ \wedge\, \mathrm{IND}(\bar{a}_{\mathtt{ME}} \wedge b_{\mathtt{ME}}, \ldots, \bar{a}_{\mathtt{CA}} \wedge b_{\mathtt{CA}}) \wedge \mathrm{IND}(\bar{a}_{\mathtt{ME}} \wedge \bar{b}_{\mathtt{ME}}, \ldots, \bar{a}_{\mathtt{CA}} \wedge \bar{b}_{\mathtt{CA}}).\end{aligned} \tag{73}$$

Каждая из четырех IND имеет бинарную диаграмму решений из 854 узлов, которые могут быть вычислены с помощью (72) со стоимостью около 70 тысяч обращений к памяти. Функция COLOR оказывается имеющей только 25 579 узлов бинарной диаграммы решений. Алгоритм C быстро выясняет, что общее количество вариантов раскраски данного графа четырьмя красками равно ровно 25 623 183 458 304 (или, если выполнить деление на 4! для удаления симметричных решений, около 1.1 триллиона).* Общее время, требующееся для данного вычисления, начиная с описания графа, составляет менее 3.5 миллионов обращений к памяти, а требуемое количество памяти — 2.2 Мбайт. (Мы можем также найти *случайную* раскраску четырьмя красками, и т. д.)

Трудные функции. Конечно, имеются также функции от 98 переменных, не столь приятные, как COLOR. Действительно, общее количество функций от 98 переменных равно $2^{2^{98}}$; в упр. 108 доказывается, что не более $2^{2^{46}}$ из них имеют бинарную диаграмму решений размером менее триллиона и что почти все булевы функции от 98 переменных в действительности имеют $B(f) \approx 2^{98}/98 \approx 3.2 \times 10^{27}$. Просто не существует способа сжатия 2^{98} бит данных до небольшого объема, если только эти данные не оказываются высоко избыточными.

Каков же наихудший случай? Если f представляет собой булеву функцию от n переменных, то как велико может быть значение $B(f)$? Ответ не так уж сложно найти, если рассмотреть *профиль* данной бинарной диаграммы решений, который представляет собой последовательность $(b_0, \ldots, b_{n-1}, b_n)$, когда имеется b_k узлов ветвления для переменной x_{k+1} и b_n стоков. Ясно, что

$$B(f) = b_0 + \cdots + b_{n-1} + b_n. \tag{74}$$

Мы также имеем $b_0 \le 1$, $b_1 \le 2$, $b_2 \le 4$, $b_3 \le 8$ и в общем случае

$$b_k \;\le\; 2^k, \tag{75}$$

поскольку каждый узел имеет только две ветви. Кроме того, $b_n = 2$, когда f не является константой; а $b_{n-1} \le 2$, поскольку имеется только два корректных выбора для ветвей LO и HI для $\textcircled{n}$. Действительно, мы знаем, что b_k представляет собой количество *бусин* порядка $n-k$ в таблице истинности f, а именно количество различных подфункций от $(x_{k+1}, \ldots, x_n)$, зависящих от x_{k+1} после указания значений $(x_1, \ldots, x_k)$. Возможны только $2^{2^m} - 2^{2^{m-1}}$ бусин порядка m, так что мы должны иметь

$$b_k \;\le\; 2^{2^{n-k}} - 2^{2^{n-k-1}} \qquad \text{для } 0 \le k < n. \tag{76}$$

---

* Для графа областей Украины соответствующая БДР содержит 2949 узлов, а общее количество вариантов раскраски равно 13 777 920 (574 080 после деления на 4!). — *Примеч. пер.*

При $n = 11$, например, (75) и (76) говорят нам, что $(b_0, \ldots, b_{11})$ не превышают

$$(1, 2, 4, 8, 16, 32, 64, 128, 240, 12, 2, 2). \tag{77}$$

Таким образом, $B(f) \le 1 + 2 + \cdots + 128 + 240 + \cdots + 2 = 255 + 256 = 511$, когда $n = 11$.

Эта верхняя граница в действительности достигается при таблице истинности, имеющей вид

$$00000000\ 00000001\ 00000010\ \ldots\ 11111110\ 11111111, \tag{78}$$

или при любой строке длиной 2^{11}, которая представляет собой перестановку 256 возможных 8-битовых байтов, поскольку очевидно наличие всех 8-битовых бусин и поскольку все подтаблицы длиной 16, 32, ..., 2^{11} очевидным образом являются бусинами. Аналогичные примеры можно построить для всех n (см. упр. 110). Следовательно, наихудший случай известен.

Теорема U. *Каждая булева функция $f(x_1, \ldots, x_n)$ имеет $B(f) \le U_n$, где*

$$U_n = 2 + \sum_{k=0}^{n-1} \min(2^k, 2^{2^{n-k}} - 2^{2^{n-k-1}}) = 2^{n-\lambda(n-\lambda n)} + 2^{2^{\lambda(n-\lambda n)}} - 1. \tag{79}$$

Кроме того, для всех n существуют явные функции f_n с $B(f_n) = U_n$. ▮

Если мы заменим λ на lg, то правая часть (79) примет вид $2^n/(n - \lg n) + 2^n/n - 1$. В общем случае U_n представляет собой u_n, умноженное на $2^n/n$, где множитель u_n лежит между 1 и $2 + O(\frac{\log n}{n})$. БДР с около $2^{n+1}/n$ узлами требует около $n + 1 - \lg n$ бит для каждого из двух указателей в каждом узле плюс $\lg n$ бит для указания переменной для ветвления. Так что общее количество памяти, занимаемое БДР для любой функции $f(x_1, \ldots, x_n)$, никогда не превышает около 2^{n+2} бит, что в четыре раза превышает размер ее таблицы истинности, даже если f оказывается одной из наихудших возможных функций с точки зрения представления БДР.

Средний случай оказывается почти таким же, как и наихудший, если выбирать таблицу истинности f случайным образом среди всех 2^{2^n} возможных вариантов. Вычисления вновь оказываются несложными: среднее число узлов $\boxed{k+1}$ составляет ровно

$$\hat{b}_k \;=\; \bigl(2^{2^{n-k}} - 2^{2^{n-k-1}}\bigr)\bigl(2^{2^n} - (2^{2^{n-k}} - 1)^{2^k}\bigr)/2^{2^n}, \tag{80}$$

поскольку имеется $2^{2^{n-k}} - 2^{2^{n-k-1}}$ бусин порядка $n-k$ и $(2^{2^{n-k}} - 1)^{2^k}$ таблиц истинности, в которых любая конкретная бусина не встречается. В упр. 112 показано, что эта сложно выглядящая величина $\hat{b}_k$ всегда лежит очень близко к оценке для наихудшего случая $\min(2^k,\ 2^{2^{n-k}} - 2^{2^{n-k-1}})$, за исключением двух значений k. Исключительные уровни встречаются, когда $k \approx 2^{n-k}$ и "min" имеет малое влияние. Например, средний профиль $(\hat{b}_0, \ldots, \hat{b}_{n-1}, \hat{b}_n)$ при $n = 11$ приближенно представляет собой

$$(1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 127.4, 151.9, 12.0, 2.0, 2.0) \tag{81}$$

при округлении до одного десятичного знака, и эти значения почти неотличимы от наихудшего случая (77) за исключением случаев $k = 7$ или 8.

Важной оказывается и родственная концепция *квази-БДР*, или "КДР" ("QDD"). Каждая функция имеет уникальную КДР, которая подобна ее БДР, за исключением

того, что корневой узел всегда (1), а каждый узел (k) для $k < n$ имеет ветвления в два узла (k+1); таким образом, каждый путь от корня до стока имеет длину n. Чтобы сделать это возможным, мы позволяем указателям LO и HI узла КДР быть идентичными. Но КДР должна оставаться приведенной в том смысле, что различные узлы не могут иметь одну и ту же пару указателей (LO, HI). Например, КДР для $\langle x_1 x_2 x_3 \rangle$ представляет собой

; (82)

она имеет в два раза больше узлов, чем соответствующая БДР на рис. 21. Обратите внимание, что в КДР поля V оказываются излишними, так что их незачем хранить в памяти.

Квазипрофиль функции представляет собой вектор $(q_0, \ldots, q_{n-1}, q_n)$, где q_{k-1} является количеством узлов (k) в КДР. Легко видеть, что q_k является также количеством различных *подтаблиц* порядка $n-k$ в таблице истинности, так же, как b_k является количеством различных бусин. Каждая бусина является подтаблицей, так что имеем

$$q_k \;\geq\; b_k \qquad \text{для } 0 \leq k \leq n. \tag{83}$$

Кроме того, в упр. 115 доказывается, что

$$q_k \leq 1 + b_0 + \cdots + b_{k-1} \;\text{ и }\; q_k \leq b_k + \cdots + b_n \quad \text{для } 0 \leq k \leq n. \tag{84}$$

Следовательно, каждый элемент квазипрофиля является нижней границей размера БДР:

$$B(f) \;\geq\; 2q_k - 1 \qquad \text{для } 0 \leq k \leq n. \tag{85}$$

Пусть $Q(f) = q_0 + \cdots + q_{n-1} + q_n$ — общий размер КДР для f. Очевидно, что $Q(f) \geq B(f)$ согласно (83). С другой стороны, $Q(f)$ не может быть намного больше $B(f)$, поскольку из (85) вытекает, что

$$Q(f) \;\leq\; \frac{n+1}{2}\bigl(B(f)+1\bigr). \tag{86}$$

В упр. 116 и 117 исследуются другие базовые свойства квазипрофилей.

Таблица истинности для наихудшего случая (78) в действительности соответствует знакомой функции, которую мы уже видели ранее, — восьмиканальному мультиплексору

$$M_3(x_9, x_{10}, x_{11}; x_1, \ldots, x_8) \;=\; x_{1+(x_9 x_{10} x_{11})_2}. \tag{87}$$

Однако мы перенумеровали переменные, так что мультиплексирование теперь выполняется по отношению к трем *последним* переменным (x_9, x_{10}, x_{11}), а не к трем первым, как в (30). Это простое изменение порядка переменных увеличивает размер БДР для M_3 от 17 до 511; а аналогичное изменение при $n = 2^m + m$ приводит к колоссальному прыжку $B(M_m)$ от $2n - 2m + 1$ до $2^{n-m+1} - 1$.

Р. Э. Брайант (R. E. Bryant) ввел интересный "самозацикленный" мультиплексор под названием *скрыто взвешенная битовая функция* (*hidden weighted bit func-*

tion), определяемый следующим образом:

$$h_n(x_1,\ldots,x_n) \;=\; x_{x_1+\cdots+x_n} \;=\; x_{\nu x}, \tag{88}$$

с соглашением, что $x_0 = 0$. Например, $h_4(x_1,x_2,x_3,x_4)$ имеет таблицу истинности 0000 0111 1001 1011. Он доказал [*IEEE Trans.* **C-40** (1991), 208–210], что h_n имеет большу́ю бинарную диаграмму решений, независимо от способов перенумерации ее переменных.

В случае стандартного порядка переменных профиль $(b_0,\ldots,b_{11})$ функции h_{11} представляет собой

$$(1,2,4,8,15,27,46,40,18,7,2,2); \tag{89}$$

следовательно, $B(h_{11}) = 172$. Первая половина этого профиля фактически представляет собой немного замаскированную последовательность Фибоначчи с $b_k = F_{k+4} - k - 2$. В общем случае h_n всегда имеет это значение b_k для $k < n/2$; таким образом, значения изначального профиля растут как ϕ^k, в то время как в наихудшем случае это рост 2^k. Эта скорость роста замедляется после того, как k превышает $n/2$, так что, например, $B(h_{32})$ имеет скромное значение 86 636. Но в конечном итоге экспоненциальный рост берет свое, и $B(h_{100})$ оказывается вне пределов видимости: 17 530 618 296 680. (Когда $n = 100$, максимальным элементом профиля является $b_{59} = 2\,947\,635\,944\,748$, по сравнению с которым $b_0+\cdots+b_{49} = 139\,583\,861\,115$ кажется карликом.) В упр. 125 доказывается, что $B(h_n)$ асимптотически равно $c\chi^n + O(n^2)$, где

$$\begin{aligned}\chi &= \frac{\sqrt[3]{27-\sqrt{621}}+\sqrt[3]{27+\sqrt{621}}}{\sqrt[3]{54}}\\ &= 1.32471\,79572\,44746\,02596\,09088\,54478\,09734\,07344\,04057\,90173+\end{aligned} \tag{90}$$

представляет собой так называемую “константу пластичности” (“plastic constant”), положительный корень $\chi^3 = \chi + 1$, а коэффициент c равен $7\chi - 1 + 14/(3+2\chi) \approx 10.75115$.

С другой стороны, можно поступить существенно лучше, если изменить порядок, в котором в бинарной диаграмме решений проверяются переменные. Если $f(x_1,\ldots,x_n)$ представляет собой произвольную булеву функцию и если π представляет собой некоторую перестановку $\{1,\ldots,n\}$, будем записывать

$$f^\pi(x_1,\ldots,x_n) \;=\; f(x_{1\pi},\ldots,x_{n\pi}). \tag{91}$$

Например, если $f(x_1,x_2,x_3,x_4) = (x_3 \vee (x_1 \wedge x_4)) \wedge (\bar x_2 \vee \bar x_4)$ и если $(1\pi,2\pi,3\pi,4\pi) = (3,2,4,1)$, то $f^\pi(x_1,x_2,x_3,x_4) = (x_4 \vee (x_3 \wedge x_1)) \wedge (\bar x_2 \vee \bar x_1)$; и мы имеем $B(f) = 10$, $B(f^\pi) = 6$, поскольку бинарные диаграммы решений имеют вид

(92)

БДР для f^π соответствует БДР для f, которая имеет нестандартный порядок, в котором ветви из (i) в (j) разрешены, только если $i\pi < j\pi$:

f : . (93)

Корнем является (i), где $i = 1\pi^-$ — индекс, для которого $i\pi = 1$. Когда переменные ветвления перечисляются сверху вниз, мы имеем $(4\pi, 2\pi, 1\pi, 3\pi) = (1, 2, 3, 4)$.

Применяя эти идеи к скрыто взвешенной битовой функции, мы имеем

$$h_n^\pi(x_1, \ldots, x_n) \;=\; x_{(x_1+\cdots+x_n)\pi}, \tag{94}$$

с соглашением, что $0\pi = 0$ и $x_0 = 0$. Например, $h_3^\pi(0, 0, 1) = 1$, если $(1\pi, 2\pi, 3\pi) = (3, 1, 2)$, поскольку $x_{(x_1+x_2+x_3)\pi} = x_3 = 1$. (См. упр. 120.)

Элемент q_k квазипрофиля подсчитывает количество различных подфункций, которые возникают, когда известны значения от x_1 до x_k. Используя (94), мы можем представить все такие подфункции посредством *реестра вариантов* $[r_0, \ldots, r_{n-k}]$, где r_j представляет собой результат подфункции, когда $x_{k+1} + \cdots + x_n = j$. Предположим, что $x_1 = c_1, \ldots, x_k = c_k$, и пусть $s = c_1 + \cdots + c_k$. Тогда $r_j = c_{(s+j)\pi}$, если $(s + j)\pi \le k$; в противном случае $r_j = x_{(s+j)\pi}$. Однако, если $s\pi > k$, мы устанавливаем $r_0 \leftarrow 0$, и $r_{n-k} \leftarrow 1$, если $(s+n-k)\pi > k$, так что первый и последний варианты каждого реестра являются константами.

Например, вычисления показывают, что следующая перестановка $1\pi \ldots 100\pi$ снижает размер бинарной диаграммы решений h_{100} от 17.5 триллиона до $B(h_{100}^\pi) = 1\,124\,432\,105$.

2	4	6	8	10	12	14	16	18	20	97	57	77	37	87	47	67	27	92	52
72	32	82	42	62	22	100	60	80	40	90	50	70	30	95	55	75	35	85	45
65	25	98	58	78	38	88	48	68	28	93	53	73	33	83	43	63	23	99	59
79	39	89	49	69	29	94	54	74	34	84	44	64	24	96	56	76	36	86	46
66	26	91	51	71	31	81	41	61	21	19	17	15	13	11	9	7	5	3	1

(95)

Такие вычисления могут основываться на подсчете всех реестров, которые могут возникнуть, для $0 \le s \le k \le n$. Предположим, что мы тестировали $x_1, \ldots, x_{83}$ и обнаружили, что $x_j = [j \le 42]$, скажем, для $1 \le j \le 83$. Тогда $s = 42$; а подфункция от оставшихся 17 переменных $(x_{84}, \ldots, x_{100})$ задается реестром $[r_0, \ldots, r_{17}] = [c_{25}, x_{98}, c_{58}, c_{78}, c_{38}, x_{88}, c_{48}, c_{68}, c_{28}, x_{93}, c_{53}, c_{73}, c_{33}, c_{83}, c_{43}, c_{63}, c_{23}, x_{99}]$, который сводится к

$$[1, x_{98}, 0, 0, 1, x_{88}, 0, 0, 1, x_{93}, 0, 0, 1, 0, 0, 0, 1, 1]. \tag{96}$$

Это одна из 2^{14} подфункций, подсчитанных q_{83} при $s = 42$. В упр. 124 поясняется, как поступить аналогичным образом с другими значениями k и s.

Теперь мы готовы к доказательству теоремы Брайанта (Bryant).

Теорема B. *Размер БДР для h_n^π превышает $2^{\lfloor n/5 \rfloor}$ для всех перестановок π.*

Доказательство. Сначала заметим, что две подфункции h_n^π равны тогда и только тогда, когда они имеют одинаковые реестры. Если $[r_0,\ldots,r_{n-k}] \neq [r'_0,\ldots,r'_{n-k}]$, предположим, что $r_j \neq r'_j$. Если и r_j, и r'_j являются константами, подфункции отличаются, когда $x_{k+1}+\cdots+x_n = j$. Если r_j является константой, но $r'_j = x_i$, мы имеем $0 < j < n-k$; подфункции отличаются, поскольку $x_{k+1}+\cdots+x_n$ может быть равно j с $x_i \neq r_j$. А если $r_j = x_i$, но $r'_j = x_{i'}$ с $i \neq i'$, мы можем иметь $x_{k+1}+\cdots+x_n = j$ с $x_i \neq x_{i'}$. (Последний случай может возникнуть, только когда реестры соответствуют разным смещениям s и s'.)

Поэтому q_k представляет собой количество различных реестров $[r_0,\ldots,r_{n-k}]$. В упр. 123 доказывается, что это число для любых заданных k, n и s, как описано выше, в точности равно

$$\binom{w}{w-s}+\binom{w}{w-s+1}+\cdots+\binom{w}{k-s} = \binom{w}{s+w-k}+\cdots+\binom{w}{s-1}+\binom{w}{s}, \tag{97}$$

где w — количество индексов j, таких, что $s \le j \le s+n-k$ и $j\pi \le k$.

Теперь рассмотрим случай $k = \lfloor 3n/5 \rfloor + 1$, и пусть $s = k - \lceil n/2 \rceil$, $s' = \lfloor n/2 \rfloor + 1$. (Представьте $n = 100$, $k = 61$, $s = 11$, $s' = 51$. Мы можем считать, что $n \ge 10$.) Тогда $w + w' = k - w''$, где w'' подсчитывает индексы, такие, что $j\pi \le k$ и либо $j < s$, либо $j > s'+n-k$. Поскольку $w'' \le (s-1)+(k-s') = 2k-2-n$, мы должны иметь $w+w' \ge n+2-k = \lceil 2n/5 \rceil + 1$. Следовательно, либо $w > \lfloor n/5 \rfloor$, либо $w' > \lfloor n/5 \rfloor$; и в обоих случаях (97) превосходит $2^{\lfloor n/5 \rfloor - 1}$. Утверждение теоремы следует из (85). ▮

И обратно, всегда имеется перестановка π, такая, что $B(h_n^\pi) = O(2^{0.2029n})$, хотя константа, скрытая в O-обозначении, достаточно велика. Этот результат доказан в B. Bollig, M. Löbbing, M. Sauerhoff and I. Wegener, *Theoretical Informatics and Applications* **33** (1999), 103–115, с применением перестановки наподобие (95): первые индексы, обладающие свойством $j\pi \le n/5$, поочередно поступают от $j > 9n/10$ и $j \le n/10$; прочие упорядочены с помощью чтения бинарного представления $9n/10 - j$ справа налево (*солексный порядок*).

Давайте также рассмотрим вкратце гораздо более простой пример: *функцию перестановки* $P_m(x_1,\ldots,x_{m^2})$, которая равна 1 тогда и только тогда, когда бинарная матрица с элементами $x_{(i-1)m+j}$ на пересечении строки i и столбца j является матрицей перестановки:

$$P_m(x_1,\ldots,x_{m^2}) = \bigwedge_{i=1}^{m} S_1(x_{(i-1)m+1}, x_{(i-1)m+2},\ldots,x_{(i-1)m+m}) \wedge \bigwedge_{j=1}^{m} S_1(x_j, x_{m+j},\ldots,x_{m^2-m+j}). \tag{98}$$

Несмотря на свою простоту, эта функция не может быть представлена небольшой бинарной диаграммой решений ни при каком переупорядочении ее переменных.

Теорема K. *Размер БДР для P_m^π превышает $m2^{m-1}$ для всех перестановок π.*

Доказательство. [См. I. Wegener, *Branching Programs and Binary Decision Diagrams* (SIAM, 2000), Theorem 4.12.3.] Заметим, что для заданной БДР для P_m^π каждый из $m!$ векторов x, таких, что $P_m^\pi(x) = 1$ отслеживает путь длиной $n = m^2$ от корня до $\boxed{\top}$; должна быть протестирована каждая переменная. Пусть $v_k(x)$ — узел, из

которого путь для x получает свою k-ю ветвь HI. Ветвление этого узла выполняется по значению в строке i и столбце j заданной матрицы для некоторой пары $(i,j) = (i_k(x), j_k(x))$.

Предположим, что $v_k(x) = v_{k'}(x')$, где $x \neq x'$. Построим x'', совпадающее с x до $v_k(x)$ и с x' после. Тогда $P_m^\pi(x'') = 1$; следовательно, мы должны иметь $k = k'$. Фактически эти рассуждения показывают, что мы должны также иметь

$$\begin{aligned} &\{i_1(x), i_2(x), \ldots, i_{k-1}(x)\} = \{i_1(x'), i_2(x'), \ldots, i_{k-1}(x')\} \\ \text{и } &\{j_1(x), j_2(x), \ldots, j_{k-1}(x)\} = \{j_1(x'), j_2(x'), \ldots, j_{k-1}(x')\}. \end{aligned} \tag{99}$$

Представим этикетки m цветов, при этом имеется $m!$ этикеток каждого цвета. Поместим этикетку цвета k на узел $v_k(x)$ для всех k и всех x. Тогда никакой узел не получит этикетки разных цветов и никакой узел с цветом k не получит всего более $(k-1)!\,(m-k)!$ этикеток согласно (99). Следовательно, как минимум $m!/((k-1)!\,(m-k)!) = k\binom{m}{k}$ различных узлов должны получить этикетки цвета k. Суммирование по k дает $m2^{m-1}$ нестоковых узлов. ▮

В упр. 184 показано, что $B(P_m)$ меньше, чем $m2^{m+1}$, так что нижняя граница в теореме K близка к оптимальной, за исключением множителя 4. Хотя размер растет экспоненциально, поведение не безнадежно плохо, поскольку $m = \sqrt{n}$. Например, $B(P_{20})$ равно всего лишь 38 797 317, хотя P_{20} представляет собой булеву функцию от 400 переменных.

***Оптимизация порядка.** Пусть $B_{\min}(f)$ и $B_{\max}(f)$ обозначают наименьшее и наибольшее значения $B(f^\pi)$, взятые по всем перестановкам π, которые могут предписывать порядок переменных. Мы видели несколько случаев, когда $B_{\min}$ и $B_{\max}$ существенно отличались друг от друга; например, 2^m-канальный мультиплексор имеет $B_{\min}(M_m) \approx 2n$ и $B_{\max}(M_m) \approx 2^n/n$ при $n = 2^m + m$. И действительно, простые функции, для которых критичен хороший порядок переменных, не так уж необычны. Рассмотрим, например,

$$f(x_1, x_2, \ldots, x_n) \;=\; (\bar{x}_1 \vee x_2) \wedge (\bar{x}_3 \vee x_4) \wedge \cdots \wedge (\bar{x}_{n-1} \vee x_n), \quad n \text{ четно}; \tag{100}$$

это важная *функция подмножества* $[x_1x_3\ldots x_{n-1} \subseteq x_2x_4\ldots x_n]$, и мы имеем $B(f) = B_{\min}(f) = n+2$. Но размер БДР взлетает до $B(f^\pi) = B_{\max}(f) = 2^{n/2+1}$, когда π представляет собой "порядок органных труб", а именно упорядочение, для которого

$$f^\pi(x_1, x_2, \ldots, x_n) \;=\; (\bar{x}_1 \vee x_n) \wedge (\bar{x}_2 \vee x_{n-1}) \wedge \cdots \wedge (\bar{x}_{n/2} \vee x_{n/2+1}). \tag{101}$$

Для упорядочения $[x_1\ldots x_{n/2} \subseteq x_{n/2+1}\ldots x_n]$ получается такое же плохое поведение. В этих случаях БДР должна "помнить" состояния $n/2$ переменных, в то время как исходная формулировка (100) требует очень малого количества памяти.

Каждая булева функция f имеет *главную профилограмму* (*master profile chart*), которая инкапсулирует множество всех ее возможных размеров $B(f^\pi)$. Если f имеет n переменных, эта профилограмма имеет 2^n вершин, по одной для каждого подмножества переменных; и она имеет $n2^{n-1}$ ребер, по одному для каждой пары подмножеств, отличающихся только одним элементом. Например, главная профи-

лограмма для функции из (92) и (93) имеет вид

. (102)

Каждое ребро имеет вес, показанный здесь количеством линий; например, вес между $\{1,2\}$ и $\{1,2,3\}$ равен 3. Профилограмма имеет следующую интерпретацию: *если X представляет собой подмножество k переменных, и если $x \notin X$, то вес между X и $X \cup x$ равен количеству подфункций f, которые зависят от x, когда переменные X заменяются константами всеми 2^k возможными способами.* Например, если $X = \{1,2\}$, мы имеем $f(0,0,x_3,x_4) = x_3$, $f(0,1,x_3,x_4) = f(1,1,x_3,x_4) = x_3 \wedge \bar{x}_4$ и $f(1,0,x_3,x_4) = x_3 \vee x_4$; все три эти подфункции зависят от x_3, но только две из них зависят от x_4, что и показано в виде весов под $\{1,2\}$.

Имеется $n!$ путей длиной n от $\emptyset$ до $\{1,\ldots,n\}$, и мы можем считать путь $\emptyset \to \{a_1\} \to \{a_1,a_2\} \to \cdots \to \{a_1,\ldots,a_n\}$ соответствующим перестановке π, если $a_1\pi = 1$, $a_2\pi = 2$, $\ldots$, $a_n\pi = n$. Тогда сумма весов на пути π равна $B(f^\pi)$, если добавить 2 для узлов стока. Например, путь $\emptyset \to \{4\} \to \{2,4\} \to \{1,2,4\} \to \{1,2,3,4\}$ дает единственный способ достичь $B(f^\pi) = 6$, как в (93).

Обратите внимание, что главная профилограмма представляет собой знакомый граф — n-мерный куб, ребра которого оформлены так, что они подсчитывают количество бусин в разных множествах подфункций. Граф имеет экспоненциальный размер, $n2^{n-1}$; все же это намного меньше, чем общее количество перестановок, $n!$. В упр. 138 показано, что, когда n равно, скажем, 25 или меньше, вся профилограмма может быть вычислена без особого труда, и мы можем найти оптимальную перестановку для любой заданной функции. Например, скрыто взвешенная битовая функция, как оказывается, имеет $B_{\min}(h_{25}) = 2090$ и $B_{\max}(h_{25}) = 35441$; минимум достигается при $(1\pi,\ldots,25\pi) = (3, 5, 7, 9, 11, 13, 15, 17, 25, 24, 23, 22, 21, 20, 19, 18, 16, 14, 12, 10, 8, 6, 4, 2, 1)$, в то время как максимум представляет собой результат странной перестановки (22, 19, 17, 25, 15, 13, 11, 10, 9, 8, 7, 24, 6, 5, 4, 3, 2, 12, 1, 14, 23, 16, 18, 20, 21), которая сначала проверяет много “средних” переменных.

Вместо вычисления всей главной профилограммы иногда можно сэкономить время, исследуя только то, что необходимо для определения пути с наименьшим весом (см. упр. 140.) Но когда n растет и функции становятся более причудливыми, вряд ли нам удастся точно определить $B_{\min}(f)$, поскольку задача поиска наилучшего упорядочения является NP-полной (см. упр. 137).

Мы определили профиль и квазипрофиль для одной булевой функции f, но те же идеи применимы и к произвольной базе БДР, которая содержит m функций $\{f_1,\ldots,f_m\}$. А именно, профиль представляет собой вектор $(b_0,\ldots,b_n)$, когда имеется b_k узлов на уровне k, а квазипрофиль имеет вид $(q_0,\ldots,q_n)$, когда имеется q_k

Рис. 26. Обмен двух верхних уровней базы БДР. Здесь (s_1, s_2, s_3, s_4) — функции-источники; (t_1, t_2, t_3, t_4) — целевые узлы, представляющие подфункции на нижних уровнях.

узлов на уровне k соответствующей базы КДР; таблицы истинности этих функций имеют b_k различных бусин порядка $n-k$ и q_k различных подтаблиц. Например, профиль функций $(4+4)$-битового сложения $\{f_1, f_2, f_3, f_4, f_5\}$ в (36) представляет собой $(2, 4, 3, 6, 3, 6, 3, 2, 2)$, а квазипрофиль выводится в упр. 144. Аналогично концепция главной профилограммы приложима к m функциям, переменные которых переупорядочиваются одновременно; и мы можем использовать ее для поиска $B_{\min}(f_1, \ldots, f_m)$ и $B_{\max}(f_1, \ldots, f_m)$, минимума и максимума $b_0 + \cdots + b_n$, взятых по всем профилям.

***Локальное переупорядочение.** Что произойдет с базой БДР, если мы решим сначала выполнить ветвление по x_2, а затем — по $x_1, x_3, \ldots, x_n$? На рис. 26 показано, что структура двух верхних уровней кардинально изменилась, но все прочие уровни остались прежними.

Более детальный анализ показывает, что этот процесс обмена уровнями нетрудно понять или реализовать. Узлы (1) до обмена можно разделить на два вида, "связанные" и "одиночные", в зависимости от того, имеют ли они узлы (2) в качестве потомков; например, слева на рис. 26 имеются три связанных узла, на которые указывают s_1, s_2 и s_3, в то время как s_4 указывает на одиночный узел. Аналогично узлы (2) до обмена являются либо "видимыми", либо "скрытыми", в зависимости от того, представляют ли они независимые функции-источники или доступны только через узлы (1); все четыре узла (2) в левой части рис. 26 являются скрытыми.

После обмена одиночные узлы (1) просто перемещаются на один уровень вниз; связанные же узлы превращаются в (2), в соответствии с процессом, который мы вскоре поясним. Скрытые узлы (2) (если таковые имеются) исчезают, а видимые просто перемещаются на один уровень вверх. В процессе превращения могут также появиться дополнительные узлы; такие узлы, помеченные (1), называются новичками. Например, в правой части рис. 26 над t_2 появляются два новичка. Этот процесс уменьшает общее количество узлов тогда и только тогда, когда скрытые узлы численно превосходят новичков.

Обращение обмена представляет собой, конечно же, такой же обмен, но с измененными ролями (1) и (2). Если начать с правой диаграммы на рис. 26, можно увидеть, что она имеет три связанных узла (с метками (2)) и один видимый (с меткой (1)); два узла скрыты, а одиночных узлов нет. Процесс обмена в общем случае превращает (связанные, одиночные, видимые, скрытые) узлы в (связанные, видимые, одиночные, новые) узлы соответственно — после чего новички становятся скрытыми при обратном обмене, а изначально скрытые узлы вновь возрождаются как новички.

Понять превращения проще всего, если рассматривать все узлы ниже верхних двух уровней так, как если бы они были стоками, имеющими константные значения. Тогда каждая функция-источник $f(x_1, x_2)$ зависит только от x_1 и x_2; следовательно, она принимает четыре значения $a = f(0,0)$, $b = f(0,1)$, $c = f(1,0)$ и $d = f(1,1)$, где a, b, c и d представляют стоки. Можно предположить, что имеется q стоков, $\boxed{1}$, $\boxed{2}$, ..., $\boxed{q}$, и что $1 \le a, b, c, d \le q$. Тогда $f(x_1, x_2)$ полностью описывается ее *расширенной таблицей истинности*, $f(0,0)f(0,1)f(1,0)f(1,1) = abcd$. А после обмена мы остаемся с $f(x_2, x_1)$, которая имеет расширенную таблицу истинности $acbd$. Например, рис. 26 можно перерисовать с использованием расширенных таблиц истинности в качестве меток узлов так, как показано на рис. 27.

Рис. 27. Другой способ представления преобразований на рис. 26.

В этих терминах функция-источник $abcd$ указывает на одиночный узел, когда $a = b \ne c = d$, и на видимый узел, когда $a = c \ne b = d$; в противном случае она указывает на связанный узел (кроме случая $a = b = c = d$, когда она указывает непосредственно на сток). Связанный узел $abcd$ обычно имеет LO $= ab$ и HI $= cd$, кроме случаев $a = b$ или $c = d$; в этих исключительных случаях LO или HI представляет собой сток. После превращения аналогичным образом LO $= ac$ и HI $= bd$, где эти узлы будут либо новичками, либо видимыми узлами, либо стоками (но не оба стоками одновременно). Интересным случаем является 1224, дочерние узлы которого 12 и 24 слева являются скрытыми узлами, в то время как 12 и 24 справа являются новичками.

В упр. 147 рассматривается эффективная реализация этого преобразования, разработанная Ричардом Руделлом (Richard Rudell) в *IEEE/ACM International Conf. Computer-Aided Design* **CAD-93** (1993), 42–47. Она обладает тем важным свойством, что не требует изменения указателей, за исключением указателей внутри узлов на двух верхних уровнях: все исходные узлы s_j продолжают указывать на одни и те же места в памяти компьютера, а все стоки сохраняют их предыдущую идентичность. Мы описывали такое преобразование как обмен между узлами $\textcircled{1}$ и $\textcircled{2}$, но в действительности такое же преобразование будет выполнять обмен $\textcircled{j}$ и $\textcircled{k}$, когда переменные x_j и x_k соответствуют ветвлениям на соседних уровнях. Дело в том, что верхние уровни любой базы БДР, по сути, определяют функции-источники для низших уровней, составляющих собственную базу БДР.

Из нашего изучения сортировки мы знаем, что *любое* переупорядочение переменных базы БДР может быть выполнено путем последовательности обменов между соседними уровнями. В частности, мы можем использовать смежные обмены для преобразования подъема, выносящего переменную x_k на верхний уровень без влияния на относительный порядок других переменных. Легко, например, перенести на верхний уровень x_4: мы просто выполняем обмены $\textcircled{4} \leftrightarrow \textcircled{3}$, затем $\textcircled{4} \leftrightarrow \textcircled{2}$,

затем $\textcircled{4} \leftrightarrow \textcircled{1}$, поскольку x_4 после второго обмена окажется на месте x_2 и будет смежной с x_1.

Поскольку многократные обмены могут приводить к любому упорядочению, иногда они способны увеличивать базу БДР до тех пор, пока она не оказывается слишком большой для работы с ней. Насколько плохим может оказаться один отдельный обмен? Если ровно (s, t, v, h, ν) узлов являются одиночными, связанными, видимыми, скрытыми и новичками, то верхние два уровня оказываются с $s+t+v+\nu$ узлами; и это количество при m функциях-источниках не превышает $m+\nu \le m+2t$, поскольку $m \ge s+t+v$. Таким образом, новый размер этих уровней не может превышать исходный более чем в два раза плюс количество источников.

Если один обмен может удвоить размер, то перенос x_k угрожает экспоненциальным увеличением размера, поскольку при этом выполняется $k-1$ обменов. Но, к счастью, в этом отношении переносы вверх не хуже отдельных обменов.

Теорема J$^+$. *После операции подъема* $B(f_1^\pi, \ldots, f_m^\pi) < m + 2B(f_1, \ldots, f_m)$.

Доказательство. Пусть $a_1 a_2 \ldots a_{2^k-1} a_{2^k}$ — расширенная таблица истинности для функции-источника $f(x_1, \ldots, x_k)$; узлы на более низких уровнях рассматриваются как стоки. После подъема расширенная таблица истинности для $f^\pi(x_1, \ldots, x_k) = f(x_{1\pi}, \ldots, x_{k\pi}) = f(x_2, \ldots, x_k, x_1)$ имеет вид $a_1 a_3 \ldots a_{2^k-1} a_2 a_4 \ldots a_{2^k}$. Таким образом, мы можем видеть, что каждая бусина на уровне j функции f^π выводится из некоторой бусины на уровне $j-1$ функции f для $1 \le j < k$; но каждая бусина на уровне $j-1$ функции f порождает в f^π не более двух бусин половинного размера. Следовательно, если соответствующие профили для $\{f_1, \ldots, f_m\}$ и $\{f_1^\pi, \ldots, f_m^\pi\}$ имеют вид $(b_0, \ldots, b_n)$ и $(b'_0, \ldots, b'_n)$, мы должны иметь $b'_0 \le m$, $b'_1 \le 2b_0$, $\ldots$, $b'_{k-1} \le 2b_{k-2}$, $b'_k = b_k$, $\ldots$, $b'_n = b_n$. Таким образом, итоговое значение $\le m + B(f_1, \ldots, f_m) + b_0 + \cdots + b_{k-2} - b_{k-1}$. ▮

Обратным к подъему является "спуск", который приводит к перемещению верхней переменной вниз на $k-1$ уровней. Как и ранее, эта операция может быть реализована с помощью $k-1$ обменов. Но в этот раз мы устанавливаем более слабую верхнюю границу для получающегося в результате размера.

Теорема J$^-$. *После операции спуска* $B(f_1^\pi, \ldots, f_m^\pi) < B(f_1, \ldots, f_m)^2$.

Доказательство. Теперь расширенная таблица истинности из предыдущего доказательства изменяется с $a_1 \ldots a_{2^k}$ на $a_1 \ldots a_{2^{k-1}} \ddagger a_{2^{k-1}+1} \ldots a_{2^k} = a_1 a_{2^{k-1}+1} \ldots a_{2^{k-1}} a_{2^k}$, "функцию застежки" 7.1.3–(76). В этом случае мы можем идентифицировать каждую бусину после спуска с помощью упорядоченной пары исходных подфункций, как в операции слияния (37) и (38). Например, при $k = 3$ таблица истинности 12345678 превращается в 15263748, бусина 1526 которой может рассматриваться как смесь $12 \diamond 56$. ▮

Это доказательство указывает, почему может получиться квадратичный рост. Если, например,

$$f(x_1, \ldots, x_n) \;=\; x_1 ?\; M_m(x_2, \ldots, x_{m+1}; x_{2m+2}, \ldots, x_n) : \\ M_m(x_{m+2}, \ldots, x_{2m+1}; \bar{x}_{2m+2}, \ldots, \bar{x}_n), \qquad (103)$$

где $n = 1 + 2m + 2^m$, спуск на $2m$ уровней заменяет $B(f) = 4n - 8m - 3$ на $B(f^\pi) = 2n^2 - 8m(n-m) - 2(n-2m) + 1 \approx$.

Поскольку подъем и спуск являются обратными друг к другу операциями, мы можем также использовать теоремы J^+ и J^- в обратном направлении: *операция подъема уменьшит размер БДР до порядка его квадратного корня, но операция спуска не может уменьшить размер сильнее, чем в два раза.* Это плохие новости для любителей спуска, хотя они могут утешиться знанием того, что спуск иногда оказывается единственным приемлемым способом получить из заданного упорядочения оптимальное.

Теоремы J^+ и J^- опубликованы в B. Bollig, M. Löbbing and I. Wegener, *Inf. Processing Letters* **59** (1996), 233–239. (См. также упр. 149.)

***Динамическое переупорядочение.** На практике естественный способ упорядочения переменных зачастую напрашивается сам собой, в частности на основе теоремы M и сетей булевых модулей (рис. 23). Но иногда подходящее упорядочение неочевидно, и мы можем надеяться только на везение; возможно также, что нам на помощь придет компьютер и найдет требуемое упорядочение. Кроме того, даже если мы знаем хороший способ начать вычисления, упорядочение переменных, ведущее себя наилучшим образом на начальных этапах, может оказаться совершенно неудовлетворительным на этапах более поздних. Следовательно, мы можем получить лучшие результаты, если не будем настаивать на фиксированном упорядочении. Вместо этого можно попытаться подстроить текущий порядок ветвления, когда база БДР станет громоздкой.

Например, можно попытаться выполнить обмен $x_{j-1} \leftrightarrow x_j$ для $1 < j \le n$, отменяя его, если в результате количество узлов при этом увеличивается; эти действия можно продолжать до тех пор, пока никакие обмены не будут давать улучшений. Этот метод легко реализуется, но, к сожалению, он слишком слаб и не дает большого снижения размера. Гораздо лучший метод переупорядочения был предложен Ричардом Руделлом (Richard Rudell) одновременно с алгоритмом обмена "на месте" (без привлечения дополнительной памяти) из упр. 147. Как доказано, его метод, названный "просеиванием" ("sifting") достаточно успешен. Идея заключается в том, чтобы взять переменную x_k и попытаться поднять или опустить ее на все прочие уровни, т. е., по сути, полностью удалить x_k из упорядочения, а затем вставить ее назад, выбрав место для вставки, которое позволит получить наименьший возможный размер БДР. Всю необходимую работу можно выполнить с помощью последовательности элементарных обменов.

Алгоритм J (*Просеивание переменной*). Этот алгоритм переносит переменную x_k в оптимальную позицию по отношению к текущему упорядочению других переменных $\{x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n\}$ в заданной базе БДР. Он работет путем многократного вызова процедуры из упр. 147 для обмена смежных переменных $x_{j-1} \leftrightarrow x_j$. Во всем этом алгоритме S обозначает текущий размер базы БДР (общее количество узлов); операция обмена обычно изменяет S.

J1. [Инициализация.] Установить $p \leftarrow 0$, $j \leftarrow k$ и $s \leftarrow S$. Если $k > n/2$, перейти к шагу J5.

J2. [Просев вверх.] Пока $j > 1$, обменивать $x_{j-1} \leftrightarrow x_j$ и устанавливать $j \leftarrow j - 1$, $s \leftarrow \min(S, s)$.

J3. [Конец прохода.] Если $p = 1$, перейти к шагу J4. В противном случае, пока $j \neq k$, устанавливать $j \leftarrow j + 1$ и обменивать $x_{j-1} \leftrightarrow x_j$; затем установить $p \leftarrow 1$ и перейти к шагу J5.

J4. [Завершение спуска.] Пока $s \neq S$, устанавливать $j \leftarrow j + 1$ и обменивать $x_{j-1} \leftrightarrow x_j$. Остановиться.

J5. [Просев вниз.] Пока $j < n$, устанавливать $j \leftarrow j + 1$, обменивать $x_{j-1} \leftrightarrow x_j$ и устанавливать $s \leftarrow \min(S, s)$.

J6. [Конец прохода.] Если $p = 1$, перейти к шагу J7. В противном случае, пока $j \neq k$, обменивать $x_{j-1} \leftrightarrow x_j$ и устанавливать $j \leftarrow j - 1$; затем устанавливать $p \leftarrow 1$ и перейти к шагу J2.

J7. [Завершение подъема.] Пока $s \neq S$, обменивать $x_{j-1} \leftrightarrow x_j$ и устанавливать $j \leftarrow j - 1$. Остановиться. ▮

Всякий раз, когда алгоритм J обменивает $x_{j-1} \leftrightarrow x_j$, исходная переменная x_k в этот момент называется либо x_{j-1}, либо x_j. Общее количество обменов варьируется от около n до примерно $2.5n$, в зависимости от k и оптимальной конечной позиции x_k. Но можно существенно улучшить время работы без серьезного влияния на получаемые результаты, если модифицировать шаги J2 и J5 так, чтобы выполнялся немедленный переход к шагам J3 и J6 соответственно, когда S становится больше, чем, скажем, $1.2s$ или даже $1.1s$ или $1.05s$. В таких случаях дальнейший просев в том же направлении вряд ли уменьшит s.

Процедура просева Руделла заключается в применении алгоритма J ровно n раз, по одному разу для каждой имеющейся в наличии переменной; см. упр. 151. Мы можем продолжать просев вновь и вновь, пока не прекратятся улучшения; но получаемый при этом выигрыш обычно не стоит затрачиваемых дополнительных усилий.

Давайте рассмотрим подробный пример, чтобы сделать абстрактные идеи вполне конкретными результатами. Мы заметили, что когда граничащие штаты расположены в порядке

```
ME NH VT MA RI CT NY NJ PA DE MD DC VA NC SC GA FL AL TN KY WV OH MI IN
IL WI MN IA MO AR MS LA TX OK KS NE SD ND MT WY CO NM AZ UT ID WA OR NV CA
```
(104)

как в (17), то это приводит к БДР размером 428 узлов для функции независимого множества

$$\neg\bigl((x_{\texttt{AL}} \wedge x_{\texttt{FL}}) \vee (x_{\texttt{AL}} \wedge x_{\texttt{GA}}) \vee (x_{\texttt{AL}} \wedge x_{\texttt{MS}}) \vee \cdots \vee (x_{\texttt{UT}} \wedge x_{\texttt{WY}}) \vee (x_{\texttt{VA}} \wedge x_{\texttt{WV}})\bigr). \tag{105}$$

Автор выбрал упорядочение (104) вручную, начав с исторического/географического перечисления штатов, которому его обучили еще ребенком, а затем пытался минимизировать размер границы между штатами-уже-перечисленными и штатами-еще-не-рассмотренными, так что БДР для (105) не нуждалась в “запоминании” слишком большого количества частичных результатов на любом уровне. Получающийся в результате размер, 428, достаточно хорош для функции от 49 переменных; но просеивание способно сделать его еще лучше. Например, рассмотрим WV. Вот некоторые возможности изменения его положения с изменяющимися размерами S.

RI	CT	NY	NJ	PA	DE	MD	DC	VA	NC	SC	GA	FL	AL	TN	KY	OH	MI	IN	IL
424	422	417	415	414	412	411	410	412	412	415	420	421	426	425	427	428	428	436	442	453

Таким образом, можно сэкономить $428 - 410 = 18$ узлов, если переместить WV в позицию между MD и DC. Применяя алгоритм J для просеивания всех переменных — сначала ME, затем NH, затем ..., затем CA, — мы закончим работу с упорядочением

```
VT MA ME NH CT RI NY NJ DE PA MD WV VA DC KY OH NC GA SC AL FL MS TN IN
IL MI AR TX LA OK MO IA WI MN CO NE KS MT ND WY SD UT AZ NM ID CA OR WA NV,
```
(106)

а размер БДР при этом уменьшится до 345(!). Всего процесс просеивания включает 4663 обмена, требуя менее 4 миллионов обращений к памяти.

Вместо аккуратного выбора упорядочения рассмотрим более ленивую альтернативу: начнем со списка штатов в алфавитном порядке

```
AL AR AZ CA CO CT DC DE FL GA IA ID IL IN KS KY LA MA MD ME MI MN MO MS
MT NC ND NE NH NJ NM NV NY OH OK OR PA RI SC SD TN TX UT VA VT WA WI WV WY
```
(107)

и будем работать с ним. Тогда БДР для (105) оказывается содержащей 306 214 узлов; ее можно вычислить с помощью алгоритма S (примерно за 380 миллионов обращений к памяти) или с помощью (55) и алгоритма U (565 миллионов обращений к памяти). В этом случае просеивание дает существенно иные результаты: упомянутые 306 214 узлов превращаются во всего лишь 2871 со стоимостью 430 миллионов дополнительных обращений к памяти. Кроме того, стоимость просеивания уменьшается от 430 Mμ до 210 Mμ, если циклы в алгоритме J прерывать при $S > 1.1s$. (Более радикальный выбор прерывания при $S > 1.05s$ снижает стоимость до 155 Mμ; но тогда размер БДР снижается лишь до 2946.)

В действительности мы можем поступить гораздо, гораздо лучше, если будем просеивать переменные *во время* вычисления (105), не дожидаясь, пока будет полностью вычислена вся длинная последовательность дизъюнкций. Например, предположим, что мы автоматически вызываем просеивание, когда размер БДР в два раза превышает количество узлов, имевшихся в наличии после предыдущего просеивания. Тогда вычисление (105), начиная с алфавитного порядка (107), автоматически доводит БДР до размера всего лишь 419 узлов, после всего лишь 60 миллионов обращений к памяти! Ни гениальность человека, ни "геометрические соображения" не требуются, чтобы получить упорядочение

```
NV OR ID WA AZ CA UT NM WY CO MT OK TX NE MO KS LA AR MS TN IA ND MN SD
GA FL AL NC SC KY WI MI IL OH IN WV MD VA DC PA NJ DE NY CT RI NH ME VT MA
```
(108)

которое выигрывает у авторского (104). Для его получения компьютер прибегает к автопросеиванию 39 раз на меньших БДР.

Какое же упорядочение штатов является *наилучшим* для функции (105)? Ответ на этот вопрос, вероятно, никогда не будет известен достоверно, но можно сделать весьма неплохие предположения. Прежде всего, немного дополнительных просеиваний (108) дают лучшее упорядочение

```
OR ID NV WA AZ CA UT NM WY CO MT SD MN ND IA NE OK KS TX MO LA AR MS TN
GA FL AL NC SC KY WI MI IL OH IN WV MD DC VA PA NJ DE NY CT RI NH ME VT MA
```
(109)

с БДР размером 354. Дальнейшие просеивания не улучшают (109); но просеивание обладает ограниченной силой, поскольку исследует только $(n-1)^2$ альтернативных упорядочений среди $n!$ возможных. (Действительно, в упр. 134 продемонстрирована функция от всего лишь четырех переменных, БДР которой не может быть улучшена путем просеивания, несмотря на то что упорядочение ее переменных не оптимально.)

В нашем колчане имеется, однако, и другая стрела: можно воспользоваться *главными профилограммами*, чтобы оптимизировать каждое окно из, скажем, шестнадцати последовательных уровней БДР. Имеется 34 таких окна; и алгоритм из упр. 139 достаточно быстро их оптимизирует. После примерно 9.6 Gμ вычислений этот алгоритм находит нового чемпиона

```
OR ID NV WA AZ CA UT NM WY CO MT SD MN ND IA NE OK KS TX MO LA AR MS WI
KY MI IN IL AL TN FL NC SC GA WV OH MD DC VA PA NJ DE NY CT RI NH ME VT MA
```
(110)

путем искусной перестановки 16 штатов в (109). Это упорядочение с БДР размером всего лишь 339, может оказаться оптимальным, поскольку не может быть улучшено ни просеиванием, ни оптимизацией ни одного окна размером 25. Однако эта гипотеза остается весьма шаткой: упорядочение

```
AL GA FL TN NC SC VA MS AR TX LA OK KY MO NM WV MD DC PA NJ DE OH IL MI
IN IA NE KS WI SD WY ND MN MT UT CO ID CA AZ OR WA NV NY CT RI NH ME VT MA
```
(111)

также оказывается неулучшаемым просеиваниями или оптимизацией окон размером 25, но его БДР при этом имеет 606 узлов и очень далека от оптимальности.

При улучшенном упорядочении (110) функция COLOR от 98 переменных из (73) требует только 22 037 узлов БДР вместо 25 579. Просеивание снижает это количество до 16098.

***Бесповторные функции.** Булевы функции, такие как $(x_1 \supset x_2) \oplus ((x_3 \equiv x_4) \wedge x_5)$, которые можно выразить в виде формул, в которых каждая переменная встречается только один раз, образуют важный класс, для которого легко вычисляется наилучшее упорядочение переменных. Формально будем говорить, что $f(x_1, \ldots, x_n)$ является *бесповторной функцией* (*функцией с однократным чтением*), если либо (i) $n = 1$ и $f(x_1) = x_1$; либо (ii) $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_k) \circ h(x_{k+1}, \ldots, x_n)$, где $\circ$ представляет собой один из бинарных операторов $\{\wedge, \vee, \overline{\wedge}, \overline{\vee}, \supset, \subset, \overline{\supset}, \overline{\subset}, \oplus, \equiv\}$, и где и g, и h являются бесповторными функциями. В случае (i) мы очевидным образом имеем $B(f) = 3$. А в случае (ii) в упр. 163 доказывается, что

$$B(f) = \begin{cases} B(g) + B(h) - 2, & \text{если } \circ \in \{\wedge, \vee, \overline{\wedge}, \overline{\vee}, \supset, \subset, \overline{\supset}, \overline{\subset}\}; \\ B(g) + B(h, \bar{h}) - 2, & \text{если } \circ \in \{\oplus, \equiv\}. \end{cases} \tag{112}$$

Чтобы получить рекуррентное соотношение, нам нужны также аналогичные формулы

$$B(f, \bar{f}) = \begin{cases} 4, & \text{если } n = 1; \\ 2B(g) + B(h, \bar{h}) - 4, & \text{если } \circ \in \{\wedge, \vee, \overline{\wedge}, \overline{\vee}, \supset, \subset, \overline{\supset}, \overline{\subset}\}; \\ B(g, \bar{g}) + B(h, \bar{h}) - 2, & \text{если } \circ \in \{\oplus, \equiv\}. \end{cases} \tag{113}$$

Особенно интересное семейство бесповторных функций получается, когда мы определяем

$$\begin{aligned} u_{m+1}(x_1, \ldots, x_{2^{m+1}}) &= v_m(x_1, \ldots, x_{2^m}) \wedge v_m(x_{2^m+1}, \ldots, x_{2^{m+1}}), \\ v_{m+1}(x_1, \ldots, x_{2^{m+1}}) &= u_m(x_1, \ldots, x_{2^m}) \oplus u_m(x_{2^m+1}, \ldots, x_{2^{m+1}}), \end{aligned} \tag{114}$$

и $u_0(x_1) = v_0(x_1) = x_1$; например, $u_3(x_1, \ldots, x_8) = \big((x_1 \wedge x_2) \oplus (x_3 \wedge x_4)\big) \wedge \big((x_5 \wedge x_6) \oplus (x_7 \wedge x_8)\big)$. В упр. 165 показано, что размеры БДР для этих функций, вычисляемые

с помощью (112) и (113), включают числа Фибоначчи:

$$\begin{aligned} B(u_{2m}) &= 2^m F_{2m+2} + 2, \qquad & B(u_{2m+1}) &= 2^{m+1} F_{2m+2} + 2; \\ B(v_{2m}) &= 2^m F_{2m+2} + 2, \qquad & B(v_{2m+1}) &= 2^m F_{2m+4} + 2. \end{aligned} \tag{115}$$

Таким образом, u_m и v_m представляют собой функции от $n = 2^m$ переменных, размеры БДР которых растут как

$$\Theta(2^{m/2}\phi^m) = \Theta(n^\beta), \qquad \text{где } \beta = 1/2 + \lg\phi \approx 1.19424. \tag{116}$$

В действительности размеры БДР в (115) являются наилучшими для функций u и v среди всех перестановок переменных в силу фундаментального результата, полученного М. Зауэрхофом (M. Sauerhoff), И. Вегенером (I. Wegener) и Р. Верхнером (R. Werchner).

Теорема W. *Если $f(x_1, \ldots, x_n) = g(x_1, \ldots, x_k) \circ h(x_{k+1}, \ldots, x_n)$ является бесповторной функцией, существует перестановка π, которая минимизирует $B(f^\pi)$ и $B(f^\pi, \bar{f}^\pi)$ одновременно и в которой переменные $\{x_1, \ldots, x_k\}$ встречаются либо первыми, либо последними.*

Доказательство. Любая перестановка $(1\pi, \ldots, n\pi)$ естественным образом приводит к "неперетасованной" перестановке $(1\sigma, \ldots, n\sigma)$, в которой первыми k элементами являются $\{1, \ldots, k\}$, а последними $n - k$ элементами являются $\{k+1, \ldots, n\}$ при сохранении порядка π в пределах каждой группы. Например, если $k = 7$, $n = 9$ и $(1\pi, \ldots, 9\pi) = (3, 1, 4, 5, 9, 2, 6, 8, 7)$, мы имеем $(1\sigma, \ldots, 9\sigma) = (3, 1, 4, 5, 2, 6, 7, 9, 8)$. В упр. 166 доказывается, что при соответствующих обстоятельствах $B(f^\sigma) \le B(f^\pi)$ и $B(f^\sigma, \bar{f}^\sigma) \le B(f^\pi, \bar{f}^\pi)$. ▮

Используя эту теорему вместе с (112) и (113), можно легко оптимизировать порядок переменных для БДР любой бесповторной функции. Рассмотрим, например, $(x_1 \vee x_2) \oplus (x_3 \wedge x_4 \wedge x_5) = g(x_1, x_2) \oplus h(x_3, x_4, x_5)$. Мы имеем $B(g) = 4$ и $B(g, \bar{g}) = 6$; $B(h) = 5$ и $B(h, \bar{h}) = 8$. Для общей формулы $f = g \oplus h$ теорема W гласит, что имеются два кандидата на наилучшее упорядочение $(1\pi, \ldots, 5\pi)$, а именно $(1, 2, 3, 4, 5)$ и $(4, 5, 1, 2, 3)$. Первое из них дает $B(f^\pi) = B(g) + B(h, \bar{h}) - 2 = 10$; второе с $B(f^\pi) = B(h) + B(g, \bar{g}) - 2 = 9$ превосходит его.

Алгоритм из упр. 167 находит наилучшую перестановку π для любой бесповторной функции $f(x_1, \ldots, x_n)$ за $O(n)$ шагов. Более того, тщательный анализ доказывает, что в наилучшем упорядочении $B(f^\pi) = O(n^\beta)$, где β — константа из (116). (См. упр. 168.)

***Умножение.** Одними из наиболее интересных в математическом отношении булевых функций являются $m + n$ бит, получающихся при умножении m-битового числа на n-битовое:

$$(x_m \ldots x_2 x_1)_2 \times (y_n \ldots y_2 y_1)_2 \;=\; (z_{m+n} \ldots z_2 z_1)_2. \tag{117}$$

В частности, "старший бит" z_{m+n} и "средний бит" z_n при $m = n$ особенно примечательны. Чтобы устранить зависимость этой записи от m и n, можно представить, что $m = n = \infty$, полагая $x_i = y_j = 0$ для всех $i > m$ и $j > n$; тогда каждое z_k является функцией от $2k$ переменных, $z_k = Z_k(x_1, \ldots, x_k; y_1, \ldots, y_k)$, а именно средним битом произведения $(x_k \ldots x_1)_2 \times (y_k \ldots y_1)_2$.

Таблица 1

Наилучшее и наихудшее упорядочения для среднего бита z_n умножения

$$\begin{array}{r} x_{11}x_{10}x_9x_7x_8x_6x_{13}x_{15} \\ \times\ x_{16}x_{14}x_{12}x_5x_4x_3x_2x_1 \\ B_{\min}(Z_8) = 756 \end{array} \qquad \begin{array}{r} x_{10}x_{11}x_9x_8x_7x_{16}x_6x_{15} \\ \times\ x_5x_4x_3x_{12}x_{13}x_2x_1x_{14} \\ B_{\max}(Z_8) = 6791 \end{array}$$

$$\begin{array}{r} x_{24}x_{20}x_{18}x_{16}x_9x_8x_{10}x_{11}x_7x_{12}x_{14}x_{21} \\ \times\ x_{22}x_{19}x_{17}x_{15}x_6x_5x_4x_3x_2x_1x_{13}x_{23} \\ B_{\min}(Z_{12}) = 21931 \end{array} \qquad \begin{array}{r} x_{16}x_{17}x_{15}x_{14}x_{24}x_{13}x_{12}x_{11}x_{20}x_{10}x_9x_{23} \\ \times\ x_8x_7x_6x_5x_{18}x_4x_{22}x_3x_2x_{19}x_1x_{21} \\ B_{\max}(Z_{12}) = 866283 \end{array}$$

Таблица 2

Наилучшее и наихудшее упорядочения для всех битов $\{z_1, \ldots, z_{m+n}\}$ умножения

$$\begin{array}{r} x_{11}x_{16}x_{15}x_{14}x_{13}x_{12}x_{10}x_9 \\ \times\ x_8x_7x_6x_5x_4x_3x_2x_1 \\ B_{\min}(Z_{8,8}^{(1)}, \ldots, Z_{8,8}^{(16)}) = 9700 \end{array} \qquad \begin{array}{r} x_{10}x_8x_9x_{13}x_2x_1x_{11}x_7 \\ \times\ x_{16}x_5x_{15}x_6x_4x_{14}x_3x_{12} \\ B_{\max}(Z_{8,8}^{(1)}, \ldots, Z_{8,8}^{(16)}) = 28678 \end{array}$$

$$\begin{array}{r} x_{15}x_{17}x_{24}x_{23}x_{22}x_{21}x_{20}x_{19}x_{18}x_{16}x_{14}x_{13} \\ \times\ x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}x_{11}x_{12} \\ B_{\min}(Z_{12,12}^{(1)}, \ldots, Z_{12,12}^{(24)}) = 648957 \end{array} \qquad \begin{array}{r} x_{17}x_{22}x_{14}x_{13}x_{16}x_{10}x_{20}x_3x_2x_1x_{19}x_{12} \\ \times\ x_{24}x_{15}x_9x_8x_{21}x_7x_6x_{11}x_{23}x_5x_4x_{18} \\ B_{\max}(Z_{12,12}^{(1)}, \ldots, Z_{12,12}^{(24)}) = 4224195 \end{array}$$

$$\begin{array}{r} x_{17}x_{16}x_{10}x_9x_{11}x_{12}\ldots x_{15}x_{18}x_{19}x_{24}x_{23}\ldots x_{20} \\ \times\ x_1x_2x_3x_4x_5x_6x_7x_8 \\ B_{\min}(Z_{16,8}^{(1)}, \ldots, Z_{16,8}^{(24)}) = 157061 \end{array} \qquad \begin{array}{r} x_{13}x_{14}x_{12}x_{15}x_{16}x_{17}x_{22}x_{10}x_8x_7x_{18}x_9x_2x_1x_{19}x_6 \\ \times\ x_{24}x_{11}x_{21}x_5x_4x_{23}x_3x_{20} \\ B_{\max}(Z_{16,8}^{(1)}, \ldots, Z_{16,8}^{(24)}) = 1236251 \end{array}$$

Средний бит оказывается трудным с точки зрения БДР, даже если y является константой. Пусть $Z_{n,a}(x_1, \ldots, x_n) = Z_n(x_1, \ldots, x_n; a_1, \ldots, a_n)$, где $a = (a_n \ldots a_1)_2$.

Теорема X. *Существует константа a, такая, что $B_{\min}(Z_{n,a}) > \frac{5}{288} \cdot 2^{\lfloor n/2 \rfloor} - 2$.*

Доказательство. [П. Вёльфель (P. Woelfel), *J. Computer and System Sci.* **71** (2005), 520–534.] Можно считать, что $n = 2t$ четно, поскольку $Z_{2t+1,2a} = Z_{2t,a}$. Пусть $x = (x_n \ldots x_1)_2$ и $m = ([n\pi \le t] \ldots [1\pi \le t])_2$. Тогда $x = p + q$, где $q = x \mathbin{\&} m$ представляет "известные" биты x после t ветвлений в БДР для $Z_{n,a}$ с упорядочением π, а $p = x \& \overline{m}$ представляет пока еще неизвестные биты. Пусть

$$P = \{x \mathbin{\&} \overline{m} \mid 0 \le x < 2^n\} \quad \text{и} \quad Q = \{x \mathbin{\&} m \mid 0 \le x < 2^n\}. \tag{118}$$

Для любого фиксированного a функция $Z_{n,a}$ имеет 2^t подфункций

$$f_q(p) = \big((pa + qa) \gg (n-1)\big) \mathbin{\&} 1, \qquad q \in Q. \tag{119}$$

Мы хотим показать, что некоторое n-битовое число a будет делать многие из этих подфункций различными; другими словами, мы хотим найти большое подмножество $Q^* \subseteq Q$, такое, что

$$q \in Q^* \text{ и } q' \in Q^*, \text{ и } q \ne q' \text{ влечет за собой } f_q(p) \ne f_{q'}(p) \quad \text{для некоторого } p \in P. \tag{120}$$

В упр. 176 детально рассматривается, как это может быть сделано. ▮

Хорошая верхняя граница для размера БДР для среднего бита, когда ни один операнд не является константой, была найдена в работе K. Amano and A. Maruoka, *Discrete Applied Math.* **155** (2007), 1224–1232.

Теорема A. *Пусть* $f(x_1,\ldots,x_{2n}) = Z_n(x_1,x_3,\ldots,x_{2n-1};x_2,x_4,\ldots,x_{2n})$. *Тогда*

$$B(f) \;\le\; Q(f) \;<\; \tfrac{19}{7}\,2^{\lceil 6n/5\rceil}. \tag{121}$$

Доказательство. Рассмотрим два n-битовых числа, $x = 2^k x_h + x_l$ и $y = 2^k y_h + y_l$, с $n-k$ неизвестными битами в каждой из их старших частей (x_h, y_h), в то время как обе их k-битовые младшие части (x_l, y_l) известны. Тогда средний бит xy определяется путем сложения трех $(n-k)$-битовых величин при $k \ge n/2$, а именно $x_h y_l \bmod 2^{n-k}$, $x_l y_h \bmod 2^{n-k}$ и $(x_l y_l \gg k) \bmod 2^{n-k}$. Следовательно, уровень $2k$ КДР требует "запоминания" только $n-k$ младших битов каждой из предшествующих величин x_l, y_l и $x_l y_l \gg k$, всего $3n-3k$ бит, и мы имеем $q_{2k} \le 2^{3n-3k}$ в квазипрофиле f. Упр. 177 завершает доказательство. ▮

Амано (Amano) и Маруока (Maruoka) открыли также другую важную верхнюю границу. Пусть $Z^{(p)}_{m,n}(x_1,\ldots,x_m;y_1,\ldots,y_n)$ означает p-й бит z_p произведения (117).

Теорема Y. *Для всех констант* $(a_m\ldots a_1)_2$ *и для всех* p *БДР и КДР для функции* $Z^{(p)}_{m,n}(a_1,\ldots,a_m;x_1,\ldots,x_n)$ *имеют менее* $3\cdot 2^{(n+1)/2}$ *узлов.*

Доказательство. В упр. 180 доказывается, что для этой функции $q_k \le 2^{n+1-k}$. Теорема будет доказана, когда мы объединим этот результат с очевидной верхней границей $q_k \le 2^k$. ▮

Теорема Y показывает, что нижняя граница теоремы X является наилучшей возможной, если не считать константного множителя. Она также показывает, что база БДР для всех $m+n$ функций произведения $Z^{(p)}_{m,n}(x_1,\ldots,x_m;x_{m+1},\ldots,x_{m+n})$ не столь велика, как величина $\Theta(2^{m+n})$, которую мы получаем почти для всех экземпляров $m+n$ функций от $m+n$ переменных.

Следствие Y. *Если* $m \le n$, $B(Z^{(1)}_{m,n},\ldots,Z^{(m+n)}_{m,n}) < 3(m+n)2^{m+(n+1)/2}$. ▮

Наилучшее упорядочение переменных для функции среднего бита Z_n и для полной базы БДР остается загадкой, но эмпирические результаты для малых m и n дают основания для гипотезы, что верхние границы из теоремы A и следствия Y недалеки от истины (табл. 1 и 2). Вот, например, наилучшие результаты Z_n при $n \le 12$.

$n =$	1	2	3	4	5	6	7	8	9	10	11	12
$B_{\min}(Z_n) =$	4	8	14	31	63	136	315	756	1717	4026	9654	21931
$2^{6n/5} \approx$	2	5	12	28	64	147	338	776	1783	4096	9410	21619

Значения $B_{\max}/B_{\min}$ по отношению к полной базе БДР $\{Z^{(1)}_{m,n},\ldots,Z^{(m+n)}_{m,n}\}$ в табл. 2 на удивление малы. Следовательно, все упорядочения для этой задачи могут оказаться грубо эквивалентными.

БДР с подавлением нулей: комбинаторная альтернатива. При применении БДР к комбинаторным задачам зачастую беглого взгляда на данные достаточно, чтобы увидеть, что большинство полей HI просто указывает на $\boxed{\perp}$. В таких случаях лучше использовать иную структуру данных, носящую название *бинарной диаграммы с подавлением нулей* (*zero-suppressed binary decision diagram*), или "НДР" ("ZDD") для краткости, введенную Шин-ичи Минато (Shin-ichi Minato) [*ACM/IEEE Design Automation Conf.* **30** (1993), 272–277]. НДР, подобно БДР, имеет узлы, но

интерпретируются они иначе: когда узел (i) выполняет ветвление в узел (j) для $j > i + 1$, это означает, что булева функция ложна, если только не выполняется условие $x_{i+1} = \cdots = x_{j-1} = 0$.

Например, БДР для независимых множеств и ядер в (12) имеют много узлов с HI $= \boxed{\bot}$. Эти узлы можно убрать из соответствующих НДР, хотя при этом и потребуется добавить несколько новых узлов.

(122)

Обратите внимание, что в НДР в силу новых соглашений мы можем иметь LO = HI. Кроме того, пример слева показывает, что НДР не обязана содержать $\boxed{\bot}$ вообще! Из каждой диаграммы (12) в результате оказывается удалено около 40% узлов.

Один из хороших способов понимания НДР заключается в рассмотрении ее как сжатого представления *семейства множеств*. Действительно, бинарные диаграммы решений с подавлением нулей в (122) представляют соответственно семейства всех независимых множеств и всех ядер C_6. Корневой узел НДР указывает наименьший элемент, имеющийся как минимум в одном из множеств; его ветви HI и LO представляют остаточные подсемейства, которые соответственно содержат и не содержат этот элемент; и т. д. Внизу $\boxed{\bot}$ представляет пустое семейство '$\emptyset$', а $\boxed{\top}$ представляет '$\{\emptyset\}$'. Например, правая НДР в (122) представляет семейство $\bigl\{\{1,3,5\}, \{1,4\}, \{2,4,6\}, \{2,5\}, \{3,6\}\bigr\}$, поскольку ветвь HI корня представляет $\{\{3,5\}, \{4\}\}$, а ветвь LO — $\{\{2,4,6\}, \{2,5\}, \{3,6\}\}$.

Каждая булева функция $f(x_1, \ldots, x_n)$, конечно, эквивалентна семейству подмножеств множества $\{1, \ldots, n\}$, и наоборот. Но концепция семейства дает нам точку зрения, отличную от функциональной. Например, семейство $\{\{1,3\}, \{2\}, \{2,5\}\}$ имеет одну и ту же НДР для всех $n \geq 5$; но если, скажем, $n = 7$, то БДР для функции $f(x_1, \ldots, x_7)$, которая определяет это семейство, нуждается в дополнительных узлах, обеспечивающих $x_4 = x_6 = x_7 = 0$, когда $f(x) = 1$.

Почти каждое понятие, которое мы рассматривали при обсуждении бинарных диаграмм решений, имеет своего двойника в теории бинарных диаграмм решений с подавлением нулей, хотя фактические структуры данных зачастую поразительно отличаются. Мы можем, например, взять таблицу истинности для любой заданной функции $f(x_1, \ldots, x_n)$ и построить ее уникальную НДР простым способом, аналогичным построению ее БДР, как показано в (5). Мы знаем, что узлы БДР для f соответствуют "бусинам" таблицы истинности f; аналогично узлы НДР соответствуют "*z-бусинам*", которые представляют собой бинарные строки вида $\alpha\beta$, где $|\alpha| = |\beta|$ и $\beta \neq 0\ldots0$, или $|\alpha| = |\beta|-1$. Любая бинарная строка, соответствующая уникальной

z-бусине, получается путем при необходимости многократного отсечения правой половины, пока строка не будет иметь нечетную длину или ее правая часть не станет ненулевой.

Уважаемый читатель, пожалуйста, поработайте сейчас над упр. 187. (Это не шутка.)

z-профиль $f(x_1,\ldots,x_n)$ представляет собой $(z_0,\ldots,z_n)$, где z_k — количество z-бусин порядка $n-k$ в таблице истинности f для $0 \le k < n$, а именно количество узлов $\overline{k+1}$ в НДР; z_n равно количеству стоков. Для общего количества узлов мы записываем $Z(f) = z_0 + \cdots + z_n$. Например, функции в (122) имеют z-профили $(1,1,2,2,2,1,1)$ и $(1,1,2,2,1,1,2)$ соответственно, так что $Z(f) = 10$ в каждом случае.

Основные соотношения (83)–(85) между профилями и квазипрофилями остаются истинными и для z-профилей, но q'_k подсчитывает только *ненулевые* подтаблицы порядка $n-k$:

$$q'_k \;\ge\; z_k \qquad \text{для } 0 \le k < n; \tag{123}$$

$$q'_k \le 1 + z_0 + \cdots + z_{k-1} \;\text{ и }\; q'_k \le z_k + \cdots + z_n \quad \text{для } 0 \le k \le n; \tag{124}$$

$$Z(f) \;\ge\; 2q'_k - 1 \qquad \text{для } 0 \le k \le n. \tag{125}$$

Следовательно, размеры БДР и НДР не могут отличаться слишком сильно:

$$Z(f) \;\le\; \frac{n}{2}\bigl(B(f)+1\bigr)+1 \qquad \text{и} \qquad B(f) \;\le\; \frac{n}{2}\bigl(Z(f)+1\bigr)+2. \tag{126}$$

С другой стороны, множитель 50, получающийся при $n = 100$, не настолько мал, чтобы его можно было не замечать.

Когда бинарные диаграммы решений с подавлением нулей используются для поиска независимых множеств и ядер граничащих штатов в исходном порядке (17), размеры БДР, равные 428 и 780, опускаются до 177 и 385 соответственно. Просеивание уменьшает эти размеры НДР до 160 и 335. Как вам такой результат? Это удивительно хорошо для сложных функций от 49 переменных.

Зная НДР для f и g, мы можем выполнить синтез для получения НДР для $f \wedge g$, $f \vee g$, $f \oplus g$ и так далее, используя алгоритмы, очень похожие на методы, применявшиеся для БДР. Кроме того, мы можем подсчитать и/или оптимизировать решения f с помощью аналогов алгоритмов C и B; фактически методы подсчета и оптимизации на основе НДР оказываются более простыми по сравнению с соответствующими алгоритмами на основе БДР. С помощью небольшой модификации БДР-методов можно также выполнять динамическое переупорядочение переменных путем просеивания. В упр. 197–209 обсуждаются детали всех основных НДР-процедур.

В общем случае НДР обычно оказывается лучше БДР при работе с функциями с *разреженными* решениями, в том смысле, что при $f(x) = 1$ значение νx достаточно мало́. И если $f(x)$ сама по себе разреженная, в том смысле, что имеет относительно немного решений, то ситуация оказывается еще лучше.

Например, НДР хорошо подходит для *задачи точного покрытия*, определяемой матрицей из нулей и единиц размером $m \times n$: мы хотим найти все способы выбора строк, которые дают в сумме $(1,1,\ldots,1)$. Наша цель может быть, скажем, покрытие

шахматной доски 32 костями домино, наподобие

(127)

Это задача точного покрытия, матрица которой имеет $8\times 8 = 64$ столбцов, по одному для каждой ячейки. Всего имеется $2\times 7\times 8 = 112$ строк, по одной для каждой пары смежных ячеек:

$$\begin{pmatrix}
1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\ldots 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\ldots 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\\
\qquad\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\vdots\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots 0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots 0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\\
0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\ldots 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,1
\end{pmatrix}. \tag{128}$$

Пусть переменная x_j представляет выбор (или напротив, не выбор) строки j. Таким образом, три решения в (127) имеют $(x_1, x_2, x_3, x_4, \ldots, x_{110}, x_{111}, x_{112}) = (1, 0, 0, 0, \ldots, 1, 0, 1)$, $(1, 0, 0, 0, \ldots, 1, 0, 1)$ и $(0, 1, 0, 1, \ldots, 1, 0, 0)$ соответственно. В общем случае решение задачи точного покрытия представляется функцией

$$f(x_1, \ldots, x_m) = \bigwedge_{j=1}^{n} S_1(X_j) = \bigwedge_{j=1}^{n} [\nu X_j = 1], \tag{129}$$

где $X_j = \{x_i \mid a_{ij} = 1\}$ и (a_{ij}) — заданная матрица.

Как оказывается, НДР для покрытия шахматной доски домино имеет только $Z(f) = 2300$ узлов, несмотря на то что в данном случае функция f имеет $m = 112$ переменных. Мы можем использовать ее для доказательства того факта, что имеется ровно 12 988 816 покрытий, таких как (127).

Аналогично можно исследовать и более экзотические виды покрытий. Например, в

(130)

шахматная доска покрывается мономино, домино и/или тримино, т. е. многоугольниками, состоящими из одного, двух или трех квадратов, соединенных сторонами. Всего имеется ровно 92 109 458 286 284 989 468 604 способа такого покрытия(!); это число мы можем вычислить почти мгновенно, выполнив всего 75 миллионов обращений к памяти и образовав НДР размером 512 227 от 468 переменных.

Можно разработать специальный алгоритм для поиска НДР для любой заданной задачи точного покрытия; можно также синтезировать результат, используя (129). См. упр. 212.

Кстати, задача покрытия костями домино, как в (127), эквивалентна поиску совершенного паросочетания графа решетки $P_8 \mathbin{\square} P_8$, являющегося двудольным. В разделе 7.5.1 мы увидим, что имеются эффективные алгоритмы, позволяющие изучать совершенные паросочетания графов, гораздо бóльших, чем те, с которыми можно работать методами, основанными на БДР/НДР. Фактически имеется даже явная формула для количества покрытий костями домино решетки размером $m \times n$. Напротив, обобщенные покрытия, такие как (130), оказываются в более широкой категории задач на гиперграфах, для которых вряд ли существуют методы решения с полиномиальным временем при $m, n \to \infty$.

Интересный вариант покрытия костями домино под названием "обрезанная шахматная доска" был рассмотрен Максом Блеком (Max Black) в его книге *Critical Thinking* (1946), с. 142 и 394: предположим, что мы удалили противоположные углы шахматной доски и пытаемся покрыть остатки 31 костью домино. Очень легко разместить 30 из них, например, как показано здесь; но затем мы застреваем. Действительно, если рассмотреть соответствующую задачу точного покрытия 108×62, но отбросить два последних ограничения из (129), то можно получить НДР с 1224 узлами, из которой можно вывести, что имеется 324 480 способов выбрать строки, которые суммируются в $(1, 1, \ldots, 1, 1, *, *)$. Но каждое из этих решений имеет как минимум две единицы в столбце 61; следовательно, НДР приводится к $\boxed{\bot}$ после выполнения И в ограничении $[\nu X_{61} = 1]$. ("Критичное мышление" поясняет, почему; см. упр. 213.) Этот пример напоминает нам, что (i) размер окончательной НДР или БДР в вычислении может быть гораздо меньше, чем время, необходимое для ее вычисления; и что (ii) использование мозга может сэкономить огромное количество машинного времени...

НДР как словари. Давайте теперь немного сменим тему и заметим, что НДР имеют массу преимуществ и в приложениях совершенно иного типа. Их можно использовать, например, для представления *пятибуквенных английских слов*, множества `WORDS`(5757) из Stanford GraphBase, с которыми мы уже встречались в начале этой главы. Один из способов сделать это заключается в рассмотрении функции $f(x_1, \ldots, x_{25})$, которая определена таким образом, что она возвращает 1 тогда и только тогда, когда пять чисел $(x_1 \ldots x_5)_2$, $(x_6 \ldots x_{10})_2$, $\ldots$, $(x_{21} \ldots x_{25})_2$ кодируют буквы английского слова (здесь $\mathtt{a} = (00001)_2$, $\ldots$, $\mathtt{z} = (11010)_2$). Например, $f(0,0,1,1,1,0,1,1,1,1,0,1,1,1,1,0,0,1,1,0,1,1,0,0,x_{25}) = x_{25}$. Эта функция от 25 переменных имеет $Z(f) = 6233$ узла, что не так уж плохо для представления 5757 слов.

Конечно, в главе 6 мы изучали множество других способов представления 5757 слов. Подход с использованием НДР не может тягаться с бинарными деревьями, лучами или хеш-таблицами, если все, что нам нужно, — это простой поиск. Но в случае НДР мы можем также получать частично специфицированные данные, или данные, которые соответствуют ключу только приближенно; можно легко обрабатывать многие сложные запросы.

Кроме того, при использовании НДР нам не нужно слишком беспокоиться о большом количестве переменных. Вместо работы с 25 переменными x_j, рассмотренными выше, можно представить эти пятибуквенные слова как разреженную функцию $F(a_1, \ldots, z_1, a_2, \ldots, z_2, \ldots, a_5, \ldots, z_5)$, которая имеет $26 \times 5 = 130$ переменных, где переменная a_2 (например) контролирует, является ли вторая буква слова буквой ‘a’. Чтобы указать, что crazy является словом, мы делаем F истинной, когда $c_1 = r_2 = a_3 = z_4 = y_5 = 1$, а все другие переменные равны 0. Или, что то же самое, мы рассматриваем F как семейство, состоящее из 5757 подмножеств: $\{\mathtt{w}_1, \mathtt{h}_2, \mathtt{i}_3, \mathtt{c}_4, \mathtt{h}_5\}$, $\{\mathtt{t}_1, \mathtt{h}_2, \mathtt{e}_3, \mathtt{r}_4, \mathtt{e}_5\}$ и т. д. В этом случае 130 переменных размер НДР $Z(F)$ оказывается равным только 5020 вместо 6233.

Кстати, $B(F)$ равно 46 189 — что более чем в девять раз превосходит $Z(F)$. Но в случае 25 переменных отношение $B(f)/Z(f)$ равно всего лишь $8870/6233 \approx 1.4$. Мир НДР, несмотря на схожесть алгоритмов и теории, во многом отличается от мира БДР.

Одним из следствий этого различия является необходимость в новых примитивных операциях, с помощью которых можно легко построить сложные семейства подмножеств из элементарных семейств. Обратите внимание, что простое подмножество $\{\mathtt{f}_1, \mathtt{u}_2, \mathtt{n}_3, \mathtt{n}_4, \mathtt{y}_5\}$ в действительности представляет собой очень скучную булеву функцию

$$\bar{a}_1 \wedge \cdots \wedge \bar{e}_1 \wedge f_1 \wedge \bar{g}_1 \wedge \cdots \wedge \bar{t}_2 \wedge u_2 \wedge \bar{v}_2 \wedge \cdots \wedge \bar{x}_5 \wedge y_5 \wedge \bar{z}_5, \tag{131}$$

минитерм из 130 булевых переменных. В упр. 203 рассматривается важная *алгебра семейств*, в которой это подмножество выражается более естественным путем как ‘$\mathtt{f}_1 \sqcup \mathtt{u}_2 \sqcup \mathtt{n}_3 \sqcup \mathtt{n}_4 \sqcup \mathtt{y}_5$’. С помощью алгебры семейств можно легко описать и вычислить много интересных коллекций слов и фрагментов слов (см. упр. 222).

НДР для представления простых путей. Важная связь между произвольными ориентированными ациклическими графами и специальным классом НДР показана на рис. 28. Когда каждая вершина-источник ориентированного ациклического графа имеет исходящую степень 1, а каждая вершина-сток имеет входящую степень, равную 1, НДР для всех ориентированных путей от источника до стока имеет, по сути, ту же “форму”, что и исходный ориентированный ациклический граф. Переменные в этой НДР являются *дугами* ориентированного ациклического графа в подходящем топологическом порядке. (См. упр. 224.)

Рис. 28. Ориентированный ациклический граф и НДР для его путей от источника к стоку. Дуги графа соответствуют вершинам НДР. Все ветви к $\boxed{\perp}$ в НДР опущены, чтобы более ясно показать структурное подобие.

Можно также использовать НДР для представления простых путей в *неориентированном* графе. Например, имеется 12 способов пройти от верхнего левого угла решетки 3×3 в нижний правый угол, не посещая

никакую из вершин дважды.

(132)

Эти пути можно представить с помощью НДР, показанной справа, которая характеризует все множества подходящих вершин. Например, мы получаем первый путь, беря HI-ветви в (13), (36), (68) и (89) этой НДР. (Как и на рис. 28, эта диаграмма упрощена путем отбрасывания всех не интересующих нас ветвей LO, которые просто ведут в $\boxed{\bot}$.) Конечно, эта НДР не является наилучшим способом представления (132), поскольку это семейство путей имеет только 12 членов. Но на большей решетке $P_8 \square P_8$ количество простых путей из угла в угол оказывается равным 789 360 053 252; и все они могут быть представлены НДР не более чем с 33 580 узлами. В упр. 225 поясняется, как быстро построить такую НДР.

Аналогичный алгоритм, который рассматривается в упр. 226, строит НДР, которая представляет все *циклы* заданного графа. С помощью НДР с размером 22 275 можно вывести, что решетка $P_8 \square P_8$ имеет ровно 603 841 648 931 простой цикл. Эта НДР может также являться наилучшим способом представления всех этих циклов в компьютере и наилучшим способом их систематической генерации при необходимости.

Те же идеи работают и в случае графов из "реального мира", не имеющих изящной математической структуры. Например, их можно использовать для ответа на вопрос, заданный в 2008 году автору Рэндалом Брайантом (Randal Bryant): "Предположим, я хочу совершить тур по континентальной части США, посещая все столицы штатов и проезжая через каждый штат только один раз. Какой маршрут мне выбрать, чтобы минимизировать общее расстояние?" Приведенная далее диаграмма показывает кратчайшие расстояния между соседними столицами штатов при ограничении, что каждый путь пересекает только одну границу штата.

(133)

Задача заключается в выборе подмножества ребер, которые образуют гамильтонов путь с наименьшей общей длиной.*

* Традиционно приведем соответствующий граф для областей и областных центров Украины (для экономии места названия областных центров пришлось сократить до двух букв). Данные

Понятно, что каждый гамильтонов путь в этом графе должен начинаться или заканчиваться в Огасте, штат Мэн (ME). Предположим, что мы начинаем путешествие из Сакраменто, штат Калифорния (CA). Действуя, как и ранее, мы можем найти НДР, характеризующую все пути от CA до ME; оказывается, что эта НДР имеет только 7850 узлов и быстро находит, что всего возможно 437 525 772 584 простых пути от CA до ME. Фактически производящая функция по числу ребер имеет вид

$$4z^{11} + 124z^{12} + 1539z^{13} + \cdots + 33385461z^{46} + 2707075z^{47}; \tag{134}$$

так что наидлиннейшие такие пути оказываются гамильтоновыми, и их имеется ровно 2 707 075. Кроме того, в упр. 227 показано, как построить наименьшую НДР размером 4726, которая описывает только гамильтоновы пути от CA до ME.

Этот эксперимент можно повторить для каждого из штатов вместо Калифорнии. (Начальную точку лучше выбирать вне Новой Англии, если мы хотим быть приняты в Нью-Йорке, являющемся точкой сочленения этого графа.) Например, имеется 483 194 гамильтоновых пути от NJ до ME. Но в упр. 228 показано, как построить *единую* НДР размером 28808 для семейства всех гамильтоновых путей от ME до *любого* другого конечного штата, общее число каковых составляет 68 656 026. Ответ для задачи Брайанта получается немедленно с помощью алгоритма B. (Читатель может попытаться найти кратчайший маршрут вручную, перед тем как обратиться к упр. 230 и узнать, каково же наилучшее решение этой задачи.)

***НДР и простые импликанты.** В заключение давайте рассмотрим поучительное приложение, в котором одновременно используются и БДР, и НДР. Согласно теореме 7.1.1Q каждая монотонная булева функция f имеет единственное кратчайшее двухуровневое представление в виде ИЛИ, примененного к ряду И, именуемое дизъюнктивной простой формой — дизъюнкции всех ее простых импликант. Простые импликанты соответствуют минимальным точкам, где $f(x) = 1$, а именно бинарным векторам x, для которых мы имеем $f(x') = 1$ и $x' \subseteq x$ тогда и только тогда, когда $x' = x$. Если, например,

$$f(x_1, x_2, x_3) = x_1 \vee (x_2 \wedge x_3), \tag{135}$$

то простыми импликантами f являются x_1 и $x_2 \wedge x_3$, в то время как минимальными решениями являются $x_1x_2x_3 = 100$ и 011. Эти минимальные решения можно также удобно выразить как e_1 и $e_2 \sqcup e_3$, используя алгебру семейств (см. упр. 203).

о расстояниях между областными центрами Украины взяты на сайте http://e-meridian.eu/. — *Примеч. пер.*

(133, y)

В общем случае $x_{i_1} \wedge \cdots \wedge x_{i_s}$ является простой импликантой монотонной функции f тогда и только тогда, когда $e_{i_1} \sqcup \cdots \sqcup e_{i_s}$ является минимальным решением f. Таким образом, мы можем рассматривать простые импликанты f $\mathrm{PI}(f)$ как ее семейство минимальных решений. Заметим, однако, что $x_{i_1} \wedge \cdots \wedge x_{i_s} \subseteq x_{j_1} \wedge \cdots \wedge x_{j_t}$ тогда и только тогда, когда $e_{i_1} \sqcup \cdots \sqcup e_{i_s} \supseteq e_{j_1} \sqcup \cdots \sqcup e_{j_t}$; так что говорить о том, что одна простая импликанта "содержит" другую, не совсем корректно. Вместо этого мы говорим, что короткая импликанта "поглощает" длинную.

Любопытное явление показано в примере (135): диаграмма представляет собой не только БДР для f, но и НДР для $\mathrm{PI}(f)$! Аналогично на рис. 21 в начале этого раздела показана не только БДР для $\langle x_1 x_2 x_3 \rangle$, но и НДР для $\mathrm{PI}(\langle x_1 x_2 x_3 \rangle)$. С другой стороны, пусть $g = (x_1 \wedge x_3) \vee x_2$. Тогда БДР для g представляет собой , но НДР для $\mathrm{PI}(g)$ имеет вид . Что же тут происходит?

Ключ к решению этой тайны лежит в рекурсивной структуре, на которой основаны БДР и НДР. Каждая булева функция может быть представлена в виде

$$f(x_1, \ldots, x_n) \;=\; (\bar{x}_1 ?\; f_0 \colon f_1) \;=\; (\bar{x}_1 \wedge f_0) \vee (x_1 \wedge f_1), \tag{136}$$

где f_c — значение f, когда x_1 заменено на c. Когда f монотонна, мы также имеем $f = f_0 \vee (x_1 \wedge f_1)$, поскольку $f_0 \subseteq f_1$. Если $f_0 \neq f_1$, БДР для f получается путем создания узла $\textcircled{1}$, ветви LO и HI которого указывают на БДР для f_0 и f_1. Аналогично нетрудно видеть, что простыми импликантами f являются

$$\mathrm{PI}(f) \;=\; \mathrm{PI}(f_0) \cup \big(e_1 \sqcup (\mathrm{PI}(f_1) \setminus \mathrm{PI}(f_0))\big). \tag{137}$$

(См. упр. 253.) Это рекуррентное соотношение определяет НДР для $\mathrm{PI}(f)$, если мы добавим к нему условие завершения рекурсии для константных функций: НДР для $\mathrm{PI}(0)$ и $\mathrm{PI}(1)$ представляют собой $\boxed{\bot}$ и $\boxed{\top}$.

Будем называть булеву функцию f "*легкой*" (*sweet*), если она монотонна и если НДР для $\mathrm{PI}(f)$ в точности такая же, как и БДР для f. Ясно, что константные функции — легкие. А неконстантную легкость легко охарактеризовать.

Теорема S. *Булева функция, зависящая от x_1, является легкой тогда и только тогда, когда ее простыми импликантами являются $P \cup (x_1 \sqcup Q)$, где P и Q — легкие и не зависят от x_1 и каждый член P поглощается некоторым членом Q.*

Доказательство. См. упр. 246. (То, что "P и Q являются легкими" означает, что каждое из них является семейством простых импликант, которое определяет легкую булеву функцию.) ▮

Следствие S. *Функция связности любого графа является легкой.*

Доказательство. Простые импликанты функции связности f представляет собой остовные деревья графа. Каждое остовное дерево, не включающее дугу x_1, имеет как минимум одно поддерево, которое будет остовным при добавлении к нему дуги x_1. Кроме того, все подфункции f представляют собой функции связности меньших графов. ▮

Таким образом, например, БДР на рис. 22, которая определяет все 431 связанных подграфа $P_3 \,\square\, P_3$, представляет также собой НДР, которая определяет все его 192 остовных дерева.

Независимо от того, является ли f легкой, мы можем использовать (137) для вычисления НДР для $\mathrm{PI}(f)$, когда f монотонна. При этом мы можем в действительности полагать узлы БДР и узлы НДР *сосуществующими* в одной и той же большой базе данных: два узла с идентичными полями (V, LO, HI) могут встречаться в памяти только один раз, несмотря на то что они имеют совершенно разные значения в разных контекстах. Мы используем одну подпрограмму для синтеза $f \wedge \bar{g}$, когда f и g указывают на бинарные диаграммы решений, и другую подпрограмму для образования $f \setminus g$, когда f и g указывают на бинарные диаграммы решений с подавлением нулей; никаких неприятностей из-за совместного использования узлов этими подпрограммами не возникает, пока переменные не переупорядочиваются. (Конечно, записи кэша должны отличать факты, относящиеся к БДР, от фактов, относящихся к НДР.)

Например, в упр. 7.1.1–67 определен интересный класс самодуальных функций, называющихся Y-функциями, а БДР для Y_{12} (которая представляет собой функцию от 91 переменной) имеет 748 416 узлов. Эта функция имеет 2 178 889 774 простых импликанты; а $Z(\mathrm{PI}(Y_{12}))$ равно только 217 388. (Стоимость поиска этой НДР составляет около 13 миллиардов обращений к памяти и 660 Мбайт памяти.)

Краткая историческая справка. Зерна бинарных диаграмм решений были неявно посеяны Клодом Шенноном (Claude Shannon) [*Trans. Amer. Inst. Electrical Engineers* **57** (1938), 713–723] в его иллюстрациях релейно-контактных схем. В разделе 4 указанной статьи показано, что любая симметричная булева функция от n переменных имеет БДР не более чем с $\binom{n+1}{2}$ узлами ветвления. Шеннон предпочитал работать с булевой алгеброй; но Ч. Ю. Ли (C. Y. Lee), в *Bell System Tech. J.* **38** (1959), 985–999, указал на некоторые преимущества того, что он назвал бинарными решающими программами (binary-decision programs), поскольку любая функция от n переменных могла быть вычислена в такой программе не более чем за n инструкций ветвления.

Ш. Акерс (S. Akers) придумал название “бинарные диаграммы решений” и продолжил развитие идей в *IEEE Trans.* **C-27** (1978), 509–516. Он показал, как получить БДР из таблицы истинности восходящим методом или нисходящим методом из алгебраических подфункций. Он пояснил, как подсчитать пути от корня к $\boxed{\top}$ или $\boxed{\bot}$, и заметил, что эти пути делят n-мерный куб на непересекающиеся подкубы.

Тем временем очень похожая модель булевых вычислений возникла при теоретическом изучении автоматов. Например, А. Кобхэм (A. Cobham) [*FOCS* **7** (1966), 78–87] связал минимальные размеры ветвящихся программ для последовательности функций $f_n(x_1, \ldots, x_n)$ с пространственной сложностью неоднородных машин Тьюринга, вычисляющих эту последовательность. Что еще более важно, С. Форчун (S. Fortune), Д. Хопкрофт (J. Hopcroft) и Э. М. Шмидт (E. M. Schmidt) [*Lecture Notes in Comp. Sci.* **62** (1978), 227–240] рассмотрели “свободные B-схемы”, ныне известные как FBDD, в которых ни на каком пути никакие булевы переменные не тестируются дважды (см. упр. 35). Среди прочего они разработали алгоритм с полиномиальным временем работы для проверки, выполняется ли условие $f = g$ для данных FBDD для f и g при условии, что как минимум одна из этих FBDD последовательно упорядочена, как БДР. Была также разработана теория конечных автоматов, которая имеет глубокую связь со структурой БДР; таким образом, од-

новременно ряд исследователей работали над задачами, эквивалентными анализу размера $B(f)$ для различных функций f. (См. упр. 261.)

Все эти работы были концептуальными, не реализованными в виде компьютерных программ, хотя программисты сразу нашли применение для лучей и деревьев метода "Патриция", которые подобны бинарным диаграммам решений, но отличаются тем, что являются деревьями, а не ориентированными ациклическими графами (см. раздел 6.3). Но позже Рэндал Э. Брайант (Randal E. Bryant) открыл, что бинарные диаграммы решений достаточно важны на практике, если потребовать, чтобы они были *упорядочены* и *приведены*. Его введение в предмет [*IEEE Trans.* **C-35** (1986), 677–691] стало на многие годы наиболее цитируемой статьей в области компьютерных наук, поскольку революционизировало структуры данных, используемые для представления булевых функций.

В своей статье Брайант указал, что БДР для любой функции при таких условиях, по сути, уникальна и что большинство функций, встречающихся на практике, имеют БДР вполне разумного размера. Он представил эффективный алгоритм для синтеза БДР для $f \wedge g$ и $f \oplus g$ и других функций на основе БДР для f и g. Он также показал, как вычислить лексикографически наименьшее x, такое, что $f(x) = 1$, и многое другое.

Ли (Lee), Акерс (Akers) и Брайант (Bryant) заметили, что многие функции могут с выгодой сосуществовать в базе БДР, совместно используя общие подфункции. Высокопроизводительный "пакет" для операций с базами БДР, разработанный К. С. Брейсом (K. S. Brace), Р. Л. Руделлом (R. L. Rudell) и Р. Э. Брайантом (R. E. Bryant) [*ACM/IEEE Design Automation Conf.* **27** (1990), 40–45], сильно повлиял на все последующие реализации инструментария для работы с БДР. Брайант резюмировал ранние применения БДР в работе *Computing Surveys* **24** (1992), 293–318.

Как упоминалось ранее, в 1993 году Шин-ичи Минато (Shin-ichi Minato) представил бинарные диаграммы решений с подавлением нулей, повышавшие производительность комбинаторной работы. В своей работе *Software Tools for Technology Transfer* **3** (2001), 156–170, он привел ретроспективный обзор ранних приложений НДР.

Начало применению булевых методов в теории графов было положено Х. Мафутом (K. Maghout) [*Comptes Rendus Acad. Sci.* **248** (Paris, 1959), 3522–3523], который показал, как выразить максимальные независимые множества и минимальные доминирующие множества любого графа или ориентированного графа в виде простых импликант монотонной функции. Позже Р. Форте (R. Fortet) [*Cahiers du Centre d'Etudes Recherche Operationelle* **1**, 4 (1959), 5–36] рассмотрел булев подход к ряду других задач; например, он ввел идею 4-раскраски графа путем назначения каждой вершине двух булевых переменных, как мы делали это в (73). П. Камен (P. Camion) в том же журнале [**2** (1960), 234–289] преобразовал задачи целочисленного программирования в эквивалентные задачи булевой алгебры в надежде решить их методами символьной логики. Эта работа была продолжена и расширена другими, в особенности П. Л. Хаммером (P. L. Hammer) и С. Рудеану (S. Rudeanu), книга которых *Boolean Methods in Operations Research* (Springer, 1968) резюмировала эти идеи. К сожалению, их подход потерпел неудачу, поскольку в то время не имелось хороших методов булевых вычислений. Сторонники булевых методов были

вынуждены ждать пришествия бинарных диаграмм решений, чтобы решить общую задачу булева программирования (7) с помощью алгоритма B. Частный случай алгоритма B, когда все веса неотрицательны, был предложен Б. Линем (B. Lin) и Ф. Соменци (F. Somenzi) [*International Conf. Computer-Aided Design* **CAD-90** (IEEE, 1990), 88–91]. Ш. Минато (S. Minato) [*Formal Methods in System Design* **10** (1997), 221–242] разработал программное обеспечение, которое автоматически преобразует линейные неравенства между целочисленными переменными в БДР, с которыми можно удобно работать (на возможность этого надеялся ряд исследователей в 1960-х годах).

Классическая задача поиска ДНФ минимального размера для заданной функции также стала впечатляюще простой, когда стали понятны методы БДР. Последние методы решения этой задачи выходят за рамки данной книги, но вы можете обратиться к отличному обзору Оливье Кудера (Olivier Coudert) по этой теме в *Integration* **17** (1994), 97–140.

Отличная книга Инго Вегенера (Ingo Wegener) *Branching Programs and Binary Decision Diagrams* (SIAM, 2000) содержит обзор огромного количества литературы по данной теме и тщательную разработку математических основ, а также рассматривает различные пути обобщения и развития базовых идей.

Примечание переводчика: отличный программный пакет *CUDD: CU Decision Diagram Package* для работы с различными диаграммами решений, разработанный Ф. Соменци (F. Somenzi), на момент перевода этой книги находился по адресу `http://vlsi.colorado.edu/~fabio/CUDD/`.

Предостережение. Мы видели десятки примеров, когда применение БДР и/или НДР позволяло решить широкий спектр комбинаторных задач с удивительной эффективностью, а приведенные далее упражнения содержат десятки дополнительных примеров блестящего применения этих методов. Но структуры БДР и НДР не являются панацеей; это только два вида вооружения из нашего арсенала. Главным образом они применяются к задачам, у которых решений гораздо больше, чем можно рассмотреть поочередно, и к задачам, решения которых обладают локальной структурой, позволяющей нашим алгоритмам работать в каждый момент только с относительно небольшими подзадачами. В последующих разделах *Искусства программирования* мы изучим дополнительные методы, позволяющие справиться с комбинаторными задачами других видов.

Упражнения

▶ **1.** [*20*] Начертите БДР для всех 16 булевых функций $f(x_1, x_2)$. Чему равны их размеры?

▶ **2.** [*21*] Начертите планарный ориентированный ациклический граф с шестнадцатью вершинами, каждая из которых представляет собой корень одной из 16 БДР в упр. 1.

3. [*16*] Сколько булевых функций $f(x_1, \ldots, x_n)$ имеют БДР размером не более 3?

4. [*21*] Предположим, что поля | V | LO | HI | упакованы в 64-битовое слово x, где `V` занимает 8 бит, а два других поля занимают по 28 бит. Покажите, что пяти широкословных инструкций достаточно для выполнения преобразования $x \mapsto x'$, где x' равно x, за исключением того, что нулевое значение `LO` или `HI` заменяется на 1 и наоборот. (Повторение этой операции в каждом узле ветвления x бинарной диаграммы решений для f даст БДР для комплементарной функции $\bar{f}$.)

5. [*20*] Если взять БДР для $f(x_1,\ldots,x_n)$ и обменять указатели LO и HI в каждом узле и если также обменять два стока $\boxed{\bot} \leftrightarrow \boxed{\top}$, то что вы получите?

6. [*10*] Пусть $g(x_1,x_2,x_3,x_4) = f(x_4,x_3,x_2,x_1)$, где f имеет БДР из (6). Что собой представляет таблица истинности g и каковы ее бусины?

7. [*21*] Пусть для заданной булевой функции $f(x_1,\ldots,x_n)$

$$g_k(x_0,x_1,\ldots,x_n) = f(x_0,\ldots,x_{k-2},x_{k-1}\vee x_k,x_{k+1},\ldots,x_n) \qquad \text{для } 1\le k\le n.$$

Найдите простое соотношение между (а) таблицами истинности и (б) БДР f и g_k.

8. [*22*] Выполните упр. 7 с $x_{k-1}\oplus x_k$ вместо $x_{k-1}\vee x_k$.

9. [*16*] Поясните, как для заданной БДР для функции $f(x) = f(x_1,\ldots,x_n)$, представленной последовательно, как в (8), определить лексикографически наибольшее x, такое, что $f(x)=0$.

▶ **10.** [*21*] Разработайте для двух заданных БДР, которые определяют булевы функции f и f' и представлены последовательно, как в (8) и (10), алгоритм, проверяющий выполнение равенства $f=f'$.

11. [*20*] Дает ли алгоритм С корректный ответ при применении к бинарной диаграмме решений, которая (а) упорядочена, но не приведена? (б) приведена, но не упорядочена?

▶ **12.** [*M21*] *Ядро* ориентированного графа представляет собой множество вершин K, такое, что

$$\begin{array}{llll} \text{из} & v\in K & \text{вытекает} & v \not\longrightarrow u \text{ для всех } u\in K; \\ \text{из} & v\notin K & \text{вытекает} & v \longrightarrow u \text{ для некоторого } u\in K. \end{array}$$

а) Покажите, что, когда ориентированный граф представляет собой обычный граф (т. е. когда $u\longrightarrow v$ тогда и только тогда, когда $v\longrightarrow u$), ядро представляет собой то же самое, что и максимальное независимое множество.

б) Опишите ядра *ориентированного* цикла $\vec{C_n}$.

в) Докажите, что ориентированный ациклический граф имеет *единственное* ядро.

13. [*M15*] Каким образом концепция ядра графа связана с концепцией (а) максимальной клики? (б) минимального вершинного покрытия?

14. [*M24*] Насколько большим (требуется точное значение) является БДР для (а) всех независимых множеств цикла C_n и (б) всех ядер C_n при $n\ge 3$? (Нумеруйте вершины, как в (12).)

15. [*M23*] Сколько (а) независимых множеств и (б) ядер имеет C_n при $n\ge 3$?

▶ **16.** [*22*] Разработайте алгоритм для последовательной генерации всех векторов $x_1\ldots x_n$, для которых $f(x_1,\ldots,x_n)=1$, для заданной БДР для f.

17. [*32*] Если это возможно, улучшите алгоритм из упр. 16, так, чтобы его время работы составляло $O(B(f))+O(N)$ при наличии N решений.

18. [*13*] Пройдите алгоритм В с БДР (8) и $(w_1,\ldots,w_4)=(1,-2,-3,4)$.

19. [*20*] Каковы наибольшее и наименьшее возможные значения переменной m_k в алгоритме В, основанном на весах $(w_1,\ldots,w_n)$, но не на всех деталях функции f?

20. [*15*] Разработайте быстрый способ вычисления весов Тью–Морзе (15) для $1\le j\le n$.

21. [*05*] Может ли алгоритм В вместо максимизации *минимизировать* $w_1x_1+\cdots+w_nx_n$?

▶ **22.** [*M21*] Предположим, что шаг В3 был упрощен таким образом, что '$W_{v+1}-W_{v_l}$' и '$W_{v+1}-W_{v_h}$' были удалены из формул. Докажите, что алгоритм будет оставаться работоспособным при применении к БДР, представляющим ядра графов.

▶ **23.** [*M20*] Все пути от корня до $\boxed{\top}$ на рис. 22 состоят ровно из восьми сплошных дуг. Почему это не случайное совпадение?

24. [*M22*] Предположим, ребрам решетки на рис. 22 назначены 12 весов ($w_{12}, w_{13}, \ldots, w_{89}$). Поясните, как найти наименьшее остовное дерево в таком графе (а именно, остовное дерево, ребра которого имеют наименьший общий вес) путем применения алгоритма B к показанной здесь бинарной диаграмме решений.

25. [*M20*] Измените алгоритм C так, чтобы он вычислял производящую функцию для решений уравнения $f(x_1, \ldots, x_n) = 1$, а именно функцию $G(z) = \sum_{x_1=0}^{1} \cdots \sum_{x_n=0}^{1} z^{x_1+\cdots+x_n} \times f(x_1, \ldots, x_n)$.

26. [*M20*] Измените алгоритм C так, чтобы он вычислял полином надежности для заданных вероятностей, а именно

$$F(p_1, \ldots, p_n) = \sum_{x_1=0}^{1} \cdots \sum_{x_n=0}^{1} (1-p_1)^{1-x_1} p_1^{x_1} \ldots (1-p_n)^{1-x_n} p_n^{x_n} f(x_1, \ldots, x_n).$$

▶ **27.** [*M26*] Предположим, что $F(p_1, \ldots, p_n)$ и $G(p_1, \ldots, p_n)$ представляют собой полиномы надежности для булевых функций $f(x_1, \ldots, x_n)$ и $g(x_1, \ldots, x_n)$, где $f \neq g$. Пусть q — простое число, и выберем независимые случайные целые числа $q_1, \ldots, q_n$, равномерно распределенные в диапазоне $0 \le q_k < q$. Докажите, что $F(q_1, \ldots, q_n) \bmod q \neq G(q_1, \ldots, q_n) \bmod q$ с вероятностью $\ge (1 - 1/q)^n$. (В частности, если $n = 1000$ и $q = 2^{31} - 1$, при такой схеме различные функции приводят к различным "хеш-значениям" с вероятностью, не меньшей 0.9999995.)

28. [*M16*] Пусть $F(p)$ представляет собой значение полинома надежности $F(p_1, \ldots, p_n)$ при $p_1 = \cdots = p_n = p$. Покажите, что $F(p)$ легко вычислить из производящей функции $G(z)$.

29. [*HM20*] Измените алгоритм C таким образом, чтобы он вычислял полином надежности $F(p)$ из упр. 28, а также его производную $F'(p)$ для заданного p и БДР для f.

▶ **30.** [*M21*] Полином надежности представляет собой сумму вкладов всех "минитермов" $(1-p_1)^{1-x_1} p_1^{x_1} \ldots (1-p_n)^{1-x_n} p_n^{x_n}$ по всем решениям $f(x_1, \ldots, x_n) = 1$. Поясните, как для заданной БДР для f и последовательности вероятностей $(p_1, \ldots, p_n)$ найти решение $x_1 \ldots x_n$, вклад которого в общую надежность наибольший.

31. [*M21*] Измените алгоритм C таким образом, чтобы он вычислял полностью детализированную таблицу истинности f, формализуя процедуру, с помощью которой (24) было получено из рис. 21.

▶ **32.** [*M20*] Какие интерпретации '$\circ$', '$\bullet$', '$\bot$', '$\top$', '$\bar{x}_j$' и 'x_j' делают из общего алгоритма из упр. 31 специализированные версии из упр. 25, 26, 29 и 30?

▶ **33.** [*M22*] Специализируйте упр. 31 так, чтобы можно было эффективно вычислить

$$\sum_{f(x)=1} (w_1 x_1 + \cdots + w_n x_n) \qquad \text{и} \qquad \sum_{f(x)=1} (w_1 x_1 + \cdots + w_n x_n)^2$$

из БДР булевой функции $f(x) = f(x_1, \ldots, x_n)$.

34. [*M25*] Специализируйте упр. 31 так, чтобы можно было эффективно вычислить

$$\max\{\max_{1 \le k \le n} (w_1 x_1 + \cdots + w_{k-1} x_{k-1} + w'_k x_k + w_{k+1} x_{k+1} + \cdots + w_n x_n + w''_k) \mid f(x) = 1\}$$

из БДР для f, для заданных $3n$ произвольных весов $(w_1, \ldots, w_n, w'_1, \ldots, w'_n, w''_1, \ldots, w''_n)$.

▶ **35.** [*22*] *Свободная бинарная диаграмма решений* (*free binary decision diagram* — FBDD) представляет собой бинарную диаграмму решений, такую как

,

в которой переменные ветвления не обязаны находиться в некотором строгом порядке, но ни одна переменная не может встречаться на пути вниз от корня более одного раза. (FBDD "свободна" в том смысле, что возможен каждый путь в ориентированном ациклическом графе: никакое ветвление никак не ограничивает другое.)

а) Разработайте алгоритм для проверки того, что предполагаемая FBDD действительно свободная.

б) Покажите, что легко вычислить полином надежности $F(p_1, \ldots, p_n)$ булевой функции $f(x_1, \ldots, x_n)$ для заданных $(p_1, \ldots, p_n)$ и FBDD, которая определяет f, и вычислить количество решений $f(x_1, \ldots, x_n) = 1$.

36. [*25*] Поясните путем расширения упр. 31, как вычислить полностью детализированную таблицу истинности для любой заданной FBDD, если абстрактные операторы $\circ$ и $\bullet$ коммутативны, а также дистрибутивны и ассоциативны. (Таким образом, мы можем находить наилучшие решения, как в алгоритме B, или решать задачи, такие как в упр. 30 и 33, с использованием FBDD так же, как и с применением БДР.)

37. [*M20*] (Р. Л. Ривест (R. L. Rivest) и Ж. Вюллемин (J. Vuillemin), 1976.) Булева функция $f(x_1, \ldots, x_n)$ называется *уклончивой*, если каждая FBDD для f содержит нисходящий путь длиной n. Пусть $G(z)$ представляет собой производящую функцию для f, как в упр. 25. Докажите, что f уклончива, если $G(-1) \neq 0$.

▶ **38.** [*27*] Пусть $I_{s-1}, \ldots, I_0$ представляют собой инструкции ветвления, которые определяют неконстантную булеву функцию $f(x_1, \ldots, x_n)$, как в (8) и (10). Разработайте алгоритм, который вычисляет переменные состояния $t_1 \ldots t_n$, где

$$t_j = \begin{cases} +1, & \text{если } f(x_1, \ldots, x_n) = 1 \text{ при } x_j = 1; \\ -1, & \text{если } f(x_1, \ldots, x_n) = 1 \text{ при } x_j = 0; \\ 0 & \text{в противном случае.} \end{cases}$$

(Если $t_1 \ldots t_n \neq 0 \ldots 0$, функция f является канализирующей, как определено в разделе 7.1.1.) Время работы вашего алгоритма должно составлять $O(n + s)$.

39. [*M20*] Чему равен размер БДР для пороговой функции $[x_1 + \cdots + x_n \geq k]$?

▶ **40.** [*22*] Пусть g представляет собой "сжатие" f, получающееся путем установки $x_{k+1} \leftarrow x_k$, как в (27).

а) Докажите, что $B(g) \leq B(f)$. [*Указание*: рассмотрите подтаблицы и бусины.]

б) Предположим, что h получается из f путем установки $x_{k+2} \leftarrow x_k$. Выполняется ли соотношение $B(h) \leq B(f)$?

41. [*M25*] В предположении, что $n \geq 4$, найдите размер БДР для пороговых функций Фибоначчи (a) $\langle x_1^{F_1} x_2^{F_2} \ldots x_{n-2}^{F_{n-2}} x_{n-1}^{F_{n-1}} x_n^{F_{n-2}} \rangle$ и (b) $\langle x_n^{F_1} x_{n-1}^{F_2} \ldots x_3^{F_{n-2}} x_2^{F_{n-1}} x_1^{F_{n-2}} \rangle$.

42. [*22*] Изобразите базу БДР для всех симметричных булевых функций от трех переменных.

▶ **43.** [*22*] Чему равно $B(f)$, когда (a) $f(x_1, \ldots, x_{2n}) = [x_1 + \cdots + x_n = x_{n+1} + \cdots + x_{2n}]$? (б) $f(x_1, \ldots, x_{2n}) = [x_1 + x_3 + \cdots + x_{2n-1} = x_2 + x_4 + \cdots + x_{2n}]$?

▶ **44.** [*M32*] Определите Σ_n — наибольший возможный размер $B(f)$, когда f является симметричной булевой функцией от n переменных.

45. [*22*] Дайте точные описания булевых модулей, которые вычисляют функцию "три единицы подряд", как в (33) и (34), и покажите, что эта сеть вполне определена.

46. [*M23*] Каков истинный размер БДР функции "три единицы подряд"?

47. [*M21*] Разработайте и докажите *обращение* теоремы M: каждая булева функция f с малой БДР может быть реализована эффективной сетью модулей.

48. [*M22*] Реализуйте скрыто взвешенную битовую функцию с помощью сети модулей наподобие показанной на рис. 23, с использованием $a_k = 2 + \lambda k$ и $b_k = 1 + \lambda(n-k)$ соединительных проводов для $1 \le k < n$. Из теоремы B заключите, что верхняя граница в теореме M не может быть улучшена до $\sum_{k=0}^{n} 2^{p(a_k,b_k)}$ для любого полинома p.

49. [*20*] Изобразите базу БДР для следующих множеств симметричных булевых функций: (а) $\{S_{\ge k}(x_1,x_2,x_3,x_4) \mid 1 \le k \le 4\}$; (б) $\{S_k(x_1,x_2,x_3,x_4) \mid 0 \le k \le 4\}$.

50. [*22*] Изобразите базу БДР для функций 7-сегментного дисплея (7.1.2–(42)).

51. [*22*] Опишите базу БДР для бинарного сложения, когда входные биты нумеруются справа налево, а именно $(f_{n+1}f_nf_{n-1}\ldots f_1)_2 = (x_{2n-1}\ldots x_3x_1)_2 + (x_{2n}\ldots x_4x_2)_2$, вместо нумерации слева направо, как в (35) и (36).

52. [*20*] Имеется смысл, в котором база БДР для m функций $\{f_1,\ldots,f_m\}$ не очень отличается от БДР всего с одним корнем: рассмотрим *функцию соединения* $J(u_1,\ldots,u_n;v_1,\ldots,v_n) = (u_1?\ v_1{:}\ u_2?\ v_2{:}\ \cdots u_n?\ v_n{:}\ 0)$, и пусть

$$f(t_1,\ldots,t_{m+1},x_1,\ldots,x_n) = J(t_1,\ldots,t_{m+1};f_1(x_1,\ldots,x_n),\ldots,f_m(x_1,\ldots,x_n),1),$$

где $(t_1,\ldots,t_{m+1})$ являются новыми фиктивными переменными, которые размещаются перед $(x_1,\ldots,x_n)$. Покажите, что $B(f)$ имеет почти то же значение, что и размер базы БДР для $\{f_1,\ldots,f_m\}$.

▶ **53.** [*23*] Примените алгоритм R вручную к бинарной диаграмме решений с семью узлами ветвления в (2).

54. [*17*] Постройте БДР для $f(x_1,\ldots,x_n)$, исходя из таблицы истинности f, за $O(2^n)$ шагов.

55. [*M30*] Поясните, как построить "БДР связности" графа (наподобие показанной на рис. 22).

56. [*20*] Модифицируйте алгоритм R так, чтобы вместо помещения всех излишних узлов в стек `AVAIL` он создавал разновидность новой БДР, состоящей из последовательных инструкций I_{s-1}, ..., I_1, I_0, имеющих компактный вид $(\bar{v}_k?\ l_k{:}\ h_k)$, принятый в алгоритмах B и C. (Исходные входные узлы алгоритма могут затем быть удалены все вместе.)

57. [*25*] Укажите дополнительные действия, которые должны быть предприняты между шагами R1 и R2, когда алгоритм R расширяется для вычисления ограничения функции. Будем считать, что `FIX[`v`]` $= t \in \{0,1\}$, если переменная v получает фиксированное значение t; в противном случае `FIX[`v`]` < 0.

58. [*20*] Докажите, что "смешанная" диаграмма, определяемая рекурсивным использованием (37), является приведенной.

▶ **59.** [*M28*] Пусть $h(x_1,\ldots,x_n)$ является булевой функцией. Опишите смешанную БДР $f \diamond g$ в терминах БДР для h, когда (а) $f(x_1,\ldots,x_{2n}) = h(x_1,\ldots,x_n)$ и $g(x_1,\ldots,x_{2n}) = h(x_{n+1},\ldots,x_{2n})$; (б) $f(x_1,x_2,\ldots,x_{2n}) = h(x_1,x_3,\ldots,x_{2n-1})$ и $g(x_1,x_2,\ldots,x_{2n}) = h(x_2,x_4,\ldots,x_{2n})$. [В обоих случаях очевидно, что $B(f) = B(g) = B(h)$.]

60. [*M22*] Предположим, что $f(x_1,\ldots,x_n)$ и $g(x_1,\ldots,x_n)$ имеют профили $(b_0,\ldots,b_n)$ и $(b'_0,\ldots,b'_n)$ соответственно, а соответствующими квазипрофилями являются $(q_0,\ldots,q_n)$ и $(q'_0,\ldots,q'_n)$. Покажите, что их смесь $f \diamond g$ имеет $B(f \diamond g) \le \sum_{j=0}^{n}(q_jb'_j + b_jq'_j - b_jb'_j)$ узлов.

▶ **61.** [*M27*] Докажите, что если α и β являются узлами соответствующих БДР для f и g, то в смешанной БДР $f \diamond g$

$$\text{входящая степень}(\alpha \diamond \beta) \le \text{входящая степень}(\alpha) \cdot \text{входящая степень}(\beta).$$

(Представим, что корень БДР имеет входящую степень, равную 1.)

▶ **62.** [*M21*] Если $f(x) = \bigvee_{j=1}^{\lfloor n/2 \rfloor}(x_{2j-1} \wedge x_{2j})$ и $g(x) = (x_1 \wedge x_n) \vee \bigvee_{j=1}^{\lceil n/2 \rceil - 1}(x_{2j} \wedge x_{2j+1})$, то чему равны асимптотические значения $B(f)$, $B(g)$, $B(f \diamond g)$ и $B(f \vee g)$ при $n \to \infty$?

63. [*M27*] Пусть $f(x_1, \ldots, x_n) = M_m(x_1 \oplus x_2, x_3 \oplus x_4, \ldots, x_{2m-1} \oplus x_{2m}; x_{2m+1}, \ldots, x_n)$ и $g(x_1, \ldots, x_n) = M_m(x_2 \oplus x_3, \ldots, x_{2m-2} \oplus x_{2m-1}, x_{2m}; \bar{x}_{2m+1}, \ldots, \bar{x}_n)$, где $n = 2m + 2^m$. Чему равны $B(f)$, $B(g)$ и $B(f \wedge g)$?

64. [*M21*] Медиану $\langle f_1 f_2 f_3 \rangle$ трех булевых функций можно вычислить, образуя

$$f_4 = f_1 \vee f_2, \quad f_5 = f_1 \wedge f_2, \quad f_6 = f_3 \wedge f_4, \quad f_7 = f_5 \vee f_6.$$

Тогда

$$\begin{aligned} B(f_4) &= O\bigl(B(f_1)B(f_2)\bigr), \\ B(f_5) &= O\bigl(B(f_1)B(f_2)\bigr), \\ B(f_6) &= O\bigl(B(f_3)B(f_4)\bigr) = O\bigl(B(f_1)B(f_2)B(f_3)\bigr); \end{aligned}$$

следовательно, $B(f_7) = O\bigl(B(f_5)B(f_6)\bigr) = O\bigl(B(f_1)^2 B(f_2)^2 B(f_3)\bigr)$. Докажите, однако, что $B(f_7)$ в действительности представляет собой лишь $O\bigl(B(f_1)B(f_2)B(f_3)\bigr)$ и что время ее вычисления из f_5 и f_6 также составляет $O\bigl(B(f_1)B(f_2)B(f_3)\bigr)$.

▶ **65.** [*M25*] Докажите, что если $h(x_1, \ldots, x_n) = f(x_1, \ldots, x_{j-1}, g(x_1, \ldots, x_n), x_{j+1}, \ldots, x_n)$, то $B(h) = O\bigl(B(f)^2 B(g)\bigr)$. Можно ли улучшить эту верхнюю границу до $O\bigl(B(f)B(g)\bigr)$ в общем случае?

66. [*20*] Завершите алгоритм S, пояснив, что следует делать на шаге S1, если $f \circ g$ оказывается тривиальной константой.

67. [*24*] Набросайте действия алгоритма S, когда f и g определяются в (41), а $op = 1$.

68. [*20*] Ускорьте работу шага S10 путем выделения и оптимизации распространенного случая `LEFT(`t`)` < 0.

69. [*21*] В алгоритме S должна быть одна или несколько инструкций проверок, таких как "если `NTOP` $>$ `TBOT`, завершить работу алгоритма неудачей", для случаев нехватки памяти. Где лучше всего размещать такие инструкции?

70. [*21*] Рассмотрите присвоение переменной b значения $\lfloor \lg$ `LCOUNT[`l`]` $\rfloor$ вместо значения $\lceil \lg$ `LCOUNT[`l`]` $\rceil$ на шаге S4.

71. [*20*] Подумайте, как распространить алгоритм S на тернарные операторы.

72. [*25*] Поясните, как избавиться от хеширования в алгоритме S.

▶ **73.** [*25*] Рассмотрите применение "виртуальных адресов" вместо действительных в качестве связей БДР: каждый указатель p имеет вид $\pi(p)2^e + \sigma(p)$, где $\pi(p) = p \gg e$ является "страницей" p, а $\sigma(p) = p \bmod 2^e$ — "слотом" p; параметр e можно выбирать из соображений удобства. Покажите, что при таком подходе в узле БДР нужны только два поля (`LO`,`HI`), поскольку идентификатор переменной $V(p)$ можно вывести из самого́ виртуального адреса p.

▶ **74.** [*M23*] Поясните, как путем модификации (49) подсчитать количество *самодуальных* монотонных булевых функций от n переменных.

75. [*M20*] Пусть $\rho_n(x_1, \ldots, x_{2^n})$ представляет собой булеву функцию, которая истинна тогда и только тогда, когда $x_1 \ldots x_{2^n}$ является таблицей истинности *регулярной* функции (см. упр. 7.1.1–110). Покажите, что БДР для ρ_n может быть вычислена процедурой, подобной таковой для вычисления μ_n в (49).

▶ **76.** [*M22*] "Клаттер"* представляет собой семейство $\mathcal{S}$ взаимно несравнимых множеств; другими словами, $S \not\subseteq S'$, когда S и S' являются различными членами $\mathcal{S}$. Каждое множество $S \subseteq \{0, 1, \ldots, n-1\}$ может быть представлено в виде n-битового целого числа $s = \sum\{2^e \mid e \in S\}$; так что каждое семейство таких множеств соответствует бинарному вектору $x_0 x_1 \ldots x_{2^n-1}$ с $x_s = 1$ тогда и только тогда, когда s представляет множество семейства.

Покажите, что БДР для функции '[$x_0 x_1 \ldots x_{2^n-1}$ соответствует клаттеру]' имеет простую связь с БДР для функции монотонной функции $\mu_n(x_1, \ldots, x_{2^n})$.

▶ **77.** [*M30*] Покажите, что существует бесконечная последовательность $(b_0, b_1, b_2, \ldots) = (1, 2, 3, 5, 6, \ldots)$, такая, что профиль БДР для μ_n имеет вид $(b_0, b_1, \ldots, b_{2^{n-1}-1}, b_{2^{n-1}-1}, \ldots, b_1, b_0, 2)$. (См. рис. 25.) У скольких узлов ветвления такой БДР LO $= \boxed{\perp}$?

▶ **78.** [*25*] Воспользуйтесь бинарными диаграммами решений для определения количества графов с 12 помеченными вершинами, максимальная степень вершины которых равна d для $0 \le d \le 11$.

79. [*20*] Вычислите для $0 \le d \le 11$ вероятность того, что граф на вершинах $\{1, \ldots, 12\}$ имеет максимальную степень d, если каждое ребро присутствует в графе с вероятностью $1/3$.

80. [*23*] Рекурсивный алгоритм (55) вычисляет $f \wedge g$ методом поиска в глубину, в то время как алгоритм S выполняет поиск в ширину. Одинаково ли часто оба алгоритма сталкиваются с одной и той же подзадачей $f' \wedge g'$ во время работы (хотя и в разном порядке) или один из алгоритмов рассматривает меньше таких случаев, чем другой?

▶ **81.** [*20*] Модифицируя (55), поясните, как вычислить $f \oplus g$ в базе БДР.

▶ **82.** [*25*] Поясните, как в случае, когда узлы базы БДР снабжены полями REF, следует работать с этими узлами в (55) и в алгоритме U.

83. [*M20*] Докажите, что если и f, и g имеют количество ссылок, равное 1, то нам нет необходимости обращаться к кэшу записей при вычислении И(f, g) согласно (55).

84. [*24*] Предложите стратегии для выбора размера кэша записей и размеров таблиц уникальности при реализации алгоритмов для баз БДР. Предложите также хороший способ периодической сборки мусора.

85. [*16*] Сравните размер базы БДР для 32 функций 16×16-битового бинарного умножения с альтернативой, заключающейся в хранении таблицы всех возможных произведений.

▶ **86.** [*21*] В подпрограмме MUX в (62) упоминаются "очевидные" значения. Что собой представляют эти значения?

87. [*20*] Если оператор медианы $\langle fgh \rangle$ реализован с помощью рекурсивной подпрограммы, аналогичной (62), то что собой представляют "очевидные" значения в этом случае?

▶ **88.** [*M25*] Найдите функции f, g и h, для которых рекурсивное тернарное вычисление $f \wedge g \wedge h$ превосходит любые бинарные вычисления $(f \wedge g) \wedge h$, $(g \wedge h) \wedge f$, $(h \wedge f) \wedge g$.

89. [*15*] Истинны или ложны следующие квантифицированные формулы? (а) $\exists x_1 \exists x_2 f = \exists x_2 \exists x_1 f$. (б) $\forall x_1 \forall x_2 f = \forall x_2 \forall x_1 f$. (в) $\forall x_1 \exists x_2 f \le \exists x_2 \forall x_1 f$. (г) $\forall x_1 \exists x_2 f \ge \exists x_2 \forall x_1 f$.

---

* Дословно — "беспорядок". — *Примеч. пер.*

90. [*M20*] При $l = m = n = 3$ формула (64) соответствует операции `MOR` компьютера `MMIX`. Имеется ли аналогичная формула, соответствующая инструкции `MXOR` (матричное умножение по модулю 2)?

▶ **91.** [*26*] На практике зачастую желательно упростить булеву функцию f по отношению к некоторому интересующему нас множеству g путем поиска функции $\hat{f}$ с малым значением $B(\hat{f})$, такой, что

$$f(x) \wedge g(x) \;\le\; \hat{f}(x) \;\le\; f(x) \vee \bar{g}(x) \qquad \text{для всех } x.$$

Другими словами, $\hat{f}(x)$ должна совпадать с $f(x)$, когда x удовлетворяет условию $g(x) = 1$, значения же $\hat{f}(x)$ при $g(x) = 0$ нам безразличны. Привлекательным кандидатом в такие $\hat{f}$ является функция $f \downarrow g$, "f, ограниченное g", определяемая следующим образом: если $g(x)$ тождественно равна 0, $f \downarrow g = 0$. В противном случае $(f \downarrow g)(x) = f(y)$, где y — первый элемент последовательности x, $x \oplus 1$, $x \oplus 2$, ..., такой, что $g(y) = 1$. (Здесь мы рассматриваем x и y как n-битовые числа $(x_1 \ldots x_n)_2$ и $(y_1 \ldots y_n)_2$. Таким образом, $x \oplus 1 = x \oplus 0 \ldots 01 = x_1 \ldots x_{n-1}\bar{x}_n$; $x \oplus 2 = x \oplus 0 \ldots 010 = x_1 \ldots x_{n-2}\bar{x}_{n-1}x_n$; и т. д.)

а) Что собой представляют $f \downarrow 1$, $f \downarrow x_j$ и $f \downarrow \bar{x}_j$?
б) Докажите, что $(f \wedge f') \downarrow g = (f \downarrow g) \wedge (f' \downarrow g)$.
в) Истинно или ложно утверждение: $\bar{f} \downarrow g = \overline{f \downarrow g}$.
г) Упростите формулу $f(x_1, \ldots, x_n) \downarrow (x_2 \wedge \bar{x}_3 \wedge \bar{x}_5 \wedge x_6)$.
д) Упростите формулу $f(x_1, \ldots, x_n) \downarrow (x_1 \oplus x_2 \oplus \cdots \oplus x_n)$.
е) Упростите формулу $f(x_1, \ldots, x_n) \downarrow ((x_1 \wedge \cdots \wedge x_n) \vee (\bar{x}_1 \wedge \cdots \wedge \bar{x}_n))$.
ж) Упростите формулу $f(x_1, \ldots, x_n) \downarrow (x_1 \wedge g(x_2, \ldots, x_n))$.
з) Найдите функции $f(x_1, x_2)$ и $g(x_1, x_2)$, такие, что $B(f \downarrow g) > B(f)$.
и) Разработайте рекурсивный способ вычисления $f \downarrow g$, аналогичный (55).

92. [*M27*] Операция $f \downarrow g$ из упр. 91 иногда зависит от порядка переменных. Докажите, что для заданной $g = g(x_1, \ldots, x_n)$ для всех перестановок π множества $\{1, \ldots, n\}$ и для всех функций $f = f(x_1, \ldots, x_n)$ условие $(f^\pi \downarrow g^\pi) = (f \downarrow g)^\pi$ выполняется тогда и только тогда, когда $g = 0$ или g является подкубом (конъюнкцией литералов).

93. [*36*] Для заданного графа G на вершинах $\{1, \ldots, n\}$ постройте булевы функции f и g, обладающие тем свойством, что приближенная функция $\hat{f}$ с малым значением $B(\hat{f})$ (см. упр. 91) существует тогда и только тогда, когда G может быть раскрашен тремя красками. (Следовательно, задача минимизации $B(\hat{f})$ является NP-полной.)

94. [*21*] Поясните, почему (65) корректно выполняет квантификацию существования.

▶ **95.** [*20*] Улучшите (65) путем проверки условия $r_l = 1$ перед вычислением r_h.

96. [*20*] Покажите, как достичь (а) универсальной квантификации $\forall x_{j_1} \ldots \forall x_{j_m}\, f = f \,\mathrm{A}\, g$ и (б) разностной квантификации $\Box x_{j_1} \ldots \Box x_{j_m}\, f = f \,\mathrm{D}\, g$ путем модификации (65).

97. [*M20*] Докажите, что можно вычислить произвольную квантификацию низа БДР, такую как $\exists x_{n-5} \forall x_{n-4} \Box x_{n-3} \exists x_{n-2} \curlywedge x_{n-1} \forall x_n f(x_1, \ldots, x_n)$, за $O(B(f))$ шагов.

▶ **98.** [*22*] В дополнение к (70) поясните, как определить вершины $\mathrm{ENDPT}(x)$ графа $\mathcal{G}$, которые имеют степень ≤ 1. Охарактеризуйте также $\mathrm{PAIR}(x, y)$, компоненты с размером 2.

99. [*20*] (Р. Э. Брайант (R. E. Bryant), 1984.) Каждая раскраска карты США в четыре цвета, рассматривавшаяся в разделе, соответствует 24 решениям функции COLOR (73) при перестановке цветов. Как эффективно избежать этой избыточности?

▶ **100.** [*24*] Сколькими способами можно раскрасить граф США четырьмя красками так, чтобы каждым цветом было окрашено ровно по 12 штатов? (Уберите из графа вершину `DC`.)

101. [*20*] Продолжая выполнять упр. 100 и используя цвета $\{1, 2, 3, 4\}$, найдите раскраску, максимизирующую величину $\sum$ (вес штата) $\times$ (цвет штата), где веса штатов такие же, как и в (18).

102. [*23*] Разработайте метод кэширования результатов функциональной композиции с использованием следующих соглашений: система в любой момент поддерживает массив функций $[g_1, \ldots, g_n]$, по одной для каждой переменной x_j. Изначально g_j представляет собой просто проецирующую функцию x_j для $1 \le j \le n$. Этот массив может быть изменен только подпрограммой NEWG(j, g), которая заменяет g_j на g. Подпрограмма COMPOSE(f) всегда выполняет функциональную композицию по отношению к текущему массиву функций замещения.

▶ **103.** [*20*] Преподаватель хочет вычислить формулу

$$\exists y_1 \ldots \exists y_m((y_1 = f_1(x_1, \ldots, x_n)) \wedge \cdots \wedge (y_m = f_m(x_1, \ldots, x_n)) \wedge g(y_1, \ldots, y_m))$$

для ряда функций f_1, ..., f_m и g. Но его студент Шустрый нашел гораздо более простую формулу для той же самой задачи. В чем заключается идея Шустрого?

▶ **104.** [*21*] Разработайте эффективный способ выяснения по заданным БДР для f и g, выполняется ли $f \le g$ или $f \ge g$, или $f \parallel g$ (здесь $f \parallel g$ означает, что f и g несравнимы).

105. [*25*] Булева функция $f(x_1, \ldots, x_n)$ называется *унатной* (*unate*) с полярностями $(y_1, \ldots, y_n)$, если функция $h(x_1, \ldots, x_n) = f(x_1 \oplus y_1, \ldots, x_n \oplus y_n)$ монотонна.

а) Покажите, что f может быть проверена на унатность с использованием кванторов $\curlywedge$ и N.

б) Разработайте рекурсивный алгоритм для проверки унатности для заданной БДР для f не более чем за $O(B(f)^2)$ шагов. Если функция f унатная, ваш алгоритм должен также находить соответствующие полярности.

106. [*25*] Пусть $f\$g\$h$ означает отношение "для всех x и y из $f(x) = g(y) = 1$ вытекает $h(x \wedge y) = 1$". Покажите, что это отношение может быть вычислено не более чем за $O(B(f)B(g)B(h))$ шагов. [*Мотивация*: теорема 7.1.1H гласит, что f является функцией Хорна тогда и только тогда, когда $f\$f\$f$; таким образом, мы можем проверить, является ли функция функцией Хорна, за $O(B(f)^3)$ шагов.]

107. [*26*] Продолжая выполнять упр. 106, покажите, что за $O(B(f)^4)$ шагов можно определить, является ли функция f функцией Крома. [*Указание*: см. теорему 7.1.1S.]

108. [*HM24*] Пусть $b(n, s)$ представляет собой количество булевых функций от n переменных, таких, что $B(f) \le s$. Докажите, что $(s-3)!\, b(n, s) \le (n(s-1)^2)^{s-2}$ при $s \ge 3$, и исследуйте варианты этого неравенства при $s = \lfloor 2^n/(n + 1/\ln 2) \rfloor$. *Указание*: см. доказательство теоремы 7.1.2S.

▶ **109.** [*HM17*] Продолжая выполнять упр. 108, покажите, что почти все булевы функции от n переменных имеют $B(f^\pi) > 2^n/(n + 1/\ln 2)$ для всех перестановок π множества $\{1, \ldots, n\}$ при $n \to \infty$.

110. [*25*] Постройте явные функции наихудшего случая f_n с $B(f_n) = U_n$ из теоремы U.

111. [*M21*] Проверьте формулу суммирования (79) из теоремы U.

112. [*HM23*] Докажите, что $\min(2^k, 2^{2^{n-k}} - 2^{2^{n-k-1}}) - \hat{b}_k$ (где $\hat{b}_k$ — число, определенное в (80)) очень мало́, за исключением случая, когда $n - \lg n - 1 < k < n - \lg n + 1$.

113. [*20*] Вместо двух стоковых узлов, по одному для каждой булевой константы, мы могли бы иметь 2^{16} стоков, по одному для каждой булевой функции от четырех переменных. Тогда БДР могла бы завершаться четырьмя уровнями ранее, после ветвления по x_{n-4}. Насколько хороша эта идея?

114. [*20*] Существует ли функция с профилем $(1, 1, 1, 1, 1, 2)$ и квазипрофилем $(1, 2, 3, 4, 3, 2)$?

▶ **115.** [*M22*] Докажите неравенства для квазипрофилей (84) и (124).

116. [*M21*] Что можно сказать о случайном квазипрофиле (а) в наихудшем случае и (б) в среднем?

117. [*M20*] Сравните $Q(f)$ с $B(f)$, когда $f = M_m(x_1, \ldots, x_m; x_{m+1}, \ldots, x_{m+2^m})$.

118. [*M23*] Покажите, что с точки зрения раздела 7.1.2 скрыто взвешенная битовая функция имеет стоимость $C(h_n) = O(n)$. Чему равно точное значение $C(h_4)$?

119. [*20*] Истинно или ложно следующее утверждение? Каждая симметричная булева функция от n переменных является частным случаем h_{2n+1}. (Например, $x_1 \oplus x_2 = h_5(0, 1, 0, x_1, x_2)$.)

120. [*18*] Поясните формулу (94).

▶ **121.** [*M22*] Если $f(x_1, \ldots, x_n)$ — произвольная булева функция, то *дуальная* функция f^D представляет собой $\bar{f}(\bar{x}_1, \ldots, \bar{x}_n)$, а ее *отображением* f^R является $f(x_n \ldots, x_1)$. Заметим, что $f^{DD} = f^{RR} = f$ и $f^{DR} = f^{RD}$.

а) Покажите, что $h_n^{DR}(x_1, \ldots, x_n) = h_n(x_2, \ldots, x_n, x_1)$.

б) Покажите, что скрыто взвешенная битовая функция удовлетворяет рекуррентному соотношению

$$h_1(x_1) = x_1, \quad h_{n+1}(x_1, \ldots, x_{n+1}) = (x_{n+1}?\ h_n(x_2, \ldots, x_n, x_1) \colon h_n(x_1, \ldots, x_n)).$$

в) Определим $x\psi$, перестановку на множестве всех бинарных строк x, с помощью рекурсивных правил

$$\epsilon\psi = \epsilon, \quad (x_1 \ldots x_n 0)\psi = (x_1 \ldots x_n \psi)0, \quad (x_1 \ldots x_n 1)\psi = (x_2 \ldots x_n x_1)\psi 1.$$

Например, $1101\psi = (101\psi)1 = (01\psi)11 = (0\psi)111 = (\psi)0111 = 0111$; и мы также имеем $0111\psi = 1101$. Является ли ψ инволюцией?

г) Покажите, что $h_n(x) = \hat{h}_n(x\psi)$, где функция $\hat{h}_n$ имеет очень малую БДР.

122. [*27*] Постройте FBDD для h_n, которая имеет менее чем n^2 узлов при $n > 1$.

123. [*M20*] Докажите формулу (97), подсчитывающую все реестры для смещения s.

▶ **124.** [*27*] Разработайте эффективный алгоритм для вычисления профиля и квазипрофиля h_n^π для заданной перестановки π. *Указание*: когда реестр $[r_0, \ldots, r_{n-k}]$ соответствует бусине?

▶ **125.** [*HM34*] Докажите, что $B(h_n)$ можно точно выразить через последовательности

$$A_n = \sum_{k=0}^{n} \binom{n-k}{2k} \qquad \text{и} \qquad B_n = \sum_{k=0}^{n} \binom{n-k}{2k+1}.$$

126. [*HM42*] Проанализируйте $B(h_n^\pi)$ для перестановки в порядке органных труб $\pi = (2, 4, \ldots, n, \ldots, 3, 1)$.

127. [*46*] Найдите перестановку π, которая минимизирует $B(h_{100}^\pi)$.

▶ **128.** [*25*] Поясните, как для заданной перестановки π множества $\{1, \ldots, m + 2^m\}$ вычислить профиль и квазипрофиль переставленного 2^m-канального мультиплексора

$$M_m^\pi(x_1, \ldots, x_m; x_{m+1}, \ldots, x_{m+2^m}) = M_m(x_{1\pi}, \ldots, x_{m\pi}; x_{(m+1)\pi}, \ldots, x_{(m+2^m)\pi}).$$

129. [*M25*] Определим $Q_m(x_1, \ldots, x_{m^2})$ как равное 1 тогда и только тогда, когда 0–1-матрица $(x_{(i-1)m+j})$ не имеет полностью нулевых строк и полностью нулевых столбцов. Докажите, что $B(Q_m^\pi) = \Omega(2^m/m^2)$ для всех π.

130. [*HM31*] Матрица смежности неориентированного графа G на вершинах $\{1, \ldots, m\}$ состоит из $\binom{m}{2}$ записей $x_{uv} = [u — v \text{ в } G]$ для $1 \le u < v \le m$. Пусть $C_{m,k}$ представляет собой булеву функцию [G имеет k-клику] для некоторого упорядочения этих $\binom{m}{2}$ переменных.

а) Докажите, что если $1 < k \le \sqrt{m}$, то $B(C_{m,k}) \ge \binom{s+t}{s}$, где $s = \binom{k}{2} - 1$ и $t = m + 2 - k^2$.

б) Следовательно, $B(C_{m,\lceil m/2\rceil}) = \Omega(2^{m/3}/\sqrt{m})$, независимо от упорядочения переменных.

131. [*M28*] (*Покрывающая функция.*) Булева функция

$$C(x_1, x_2, \ldots, x_p; y_{11}, y_{12}, \ldots, y_{1q}, y_{21}, \ldots, y_{2q}, \ldots, y_{p1}, y_{p2}, \ldots, y_{pq})$$
$$= ((x_1 \wedge y_{11}) \vee (x_2 \wedge y_{21}) \vee \cdots \vee (x_p \wedge y_{p1})) \wedge \cdots \wedge ((x_1 \wedge y_{1q}) \vee (x_2 \wedge y_{2q}) \vee \cdots \vee (x_p \wedge y_{pq}))$$

является истинной тогда и только тогда, когда все столбцы произведения матриц

$$x \cdot Y \;=\; (x_1 x_2 \ldots x_p) \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1q} \\ y_{21} & y_{22} & \cdots & y_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ y_{p1} & y_{p2} & \cdots & y_{pq} \end{pmatrix}$$

положительны, т. е. когда строки Y, отобранные с помощью x, "покрывают" каждый столбец этой матрицы. Полином надежности для C играет важную роль в анализе отказоустойчивости систем.

а) Покажите, что, когда БДР для C тестирует переменные в порядке

$$x_1, y_{11}, y_{12}, \ldots, y_{1q}, x_2, y_{21}, y_{22}, \ldots, y_{2q}, \ldots, x_p, y_{p1}, y_{p2}, \ldots, y_{pq},$$

количество узлов асимптотически равно $pq2^{q-1}$ для фиксированного q при $p \to \infty$.

б) Найдите упорядочение, для которого размер асимптотически равен $pq2^{p-1}$ для фиксированного p при $q \to \infty$.

в) Докажите, что в общем случае $B_{\min}(C) = \Omega(2^{\min(p,q)/2})$.

132. [*32*] Какие булевы функции $f(x_1, x_2, x_3, x_4, x_5)$ имеют наибольшее $B_{\min}(f)$?

133. [*20*] Поясните, как вычислить $B_{\min}(f)$ и $B_{\max}(f)$ из главной профилограммы f.

134. [*24*] Постройте главную профилограмму, аналогичную (102), для булевой функции $x_1 \oplus ((x_2 \oplus (x_1 \vee (\bar{x}_2 \wedge x_3))) \wedge (x_3 \oplus x_4))$. Что собой представляют $B_{\min}(f)$ и $B_{\max}(f)$? *Указание*: тождество $f(x_1, x_2, x_3, x_4) = f(x_1, x_2, \bar{x}_4, \bar{x}_3)$ сэкономит вам половину работы.

135. [*M27*] Для всех $n \ge 4$ найдите булеву функцию $\theta_n(x_1, \ldots, x_n)$, *уникально тонкую* в том смысле, что $B(\theta_n^\pi) = n + 2$ ровно для одной перестановки π. (См. (93) и (102).)

▶ **136.** [*M34*] Что собой представляет главная профилограмма функции медианы медиан

$$\langle\langle x_{11} x_{12} \ldots x_{1n}\rangle \langle x_{21} x_{22} \ldots x_{2n}\rangle \ldots \langle x_{m1} x_{m2} \ldots x_{mn}\rangle\rangle,$$

при нечетных целых m и n? Какое упорядочение является наилучшим? (Всего имеется mn переменных.)

137. [*M38*] *Задача оптимального линейного размещения* заданного графа заключается в поиске такой перестановки вершин π, которая минимизирует $\sum_{u-v} |u\pi - v\pi|$. Постройте булеву функцию f, для которой это минимальное значение характеризуется оптимальным размером БДР $B_{\min}(f)$.

▶ **138.** [*M36*] Целью этого упражнения является разработка эффективного алгоритма, который вычисляет главную профилограмму для функции f, заданной ее КДР (а не БДР).

а) Поясните, как найти $\binom{n+1}{2}$ весов главной профилограммы из одной КДР.

б) Покажите, что операция подъема может быть легко осуществлена в КДР, без сборки мусора или хеширования. *Указание*: см. "карманную сортировку" в алгоритме R.

в) Рассмотрим 2^{n-1} упорядочений переменных, где $(i+1)$-е получается из i-го путем подъема с глубины $\rho i + \nu i$ до глубины $\nu i - 1$. Например, мы получим

12345 21345 32145 31245 43125 41325 42135 42315 54231 52431 53241 53421 51342 51432 51243 51234

при $n = 5$. Покажите, что каждое k-элементное подмножество множества $\{1, \ldots, n\}$ встречается на верхних k уровнях одного из этих упорядочений.

г) Объедините эти идеи для разработки искомого алгоритма построения профилограммы.

д) Проанализируйте используемую память и время работы вашего алгоритма.

139. [*22*] Обобщите алгоритм из упр. 138 так, чтобы (i) он вычислял общую профилограмму для всех функций базы БДР, а не для одной функции; и (ii) ограничивал профилограмму переменными $\{x_a, x_{a+1}, \ldots, x_b\}$, сохраняя $\{x_1, \ldots, x_{a-1}\}$ вверху и $\{x_{b+1}, \ldots, x_n\}$ внизу.

140. [*27*] Поясните, как найти $B_{\min}(f)$ без знания главной профилограммы всех f.

141. [*30*] Истинно или ложно следующее утверждение? Если X_1, X_2, ..., X_m представляют собой непересекающиеся множества переменных, то оптимальное упорядочение БДР для переменных $g(h_1(X_1), h_2(X_2), \ldots, h_m(X_m))$ может быть найдено путем ограниченного рассмотрения случаев, когда переменные каждого X_j являются последовательными.

▶ **142.** [*HM32*] Представление пороговых функций с помощью БДР удивительно загадочное. Рассмотрим самодуальную функцию $f(x) = \langle x_1^{w_1} \ldots x_n^{w_n} \rangle$, где каждое w_j представляет собой положительное целое число, а $w_1 + \cdots + w_n$ нечетно. Мы видели в (28), что $B(f) = O(w_1 + \cdots + w_n)^2$; и $B(f)$ зачастую представляет собой $O(n)$, даже когда веса растут экспоненциально, как в (29) или в упр. 41.

а) Докажите, что, когда $w_1 = 1$, $w_k = 2^{k-2}$ для $1 < k \le m$ и $w_k = 2^m - 2^{n-k}$ для $m < k \le 2m = n$, $B(f)$ при $n \to \infty$ растет экспоненциально, но $B_{\min}(f) = O(n^2)$.

б) Найдите веса $\{w_1, \ldots, w_n\}$, для которых $B_{\min}(f) = \Omega(2^{\sqrt{n}/2})$.

143. [*24*] Продолжая выполнять упр. 142, (а), найдите оптимальное упорядочение переменных для функции $\langle x_1 x_2 x_3^2 x_4^4 x_5^8 x_6^{16} x_7^{32} x_8^{64} x_9^{128} x_{10}^{256} x_{11}^{512} x_{12}^{768} x_{13}^{896} x_{14}^{960} x_{15}^{992} x_{16}^{1008} x_{17}^{1016} x_{18}^{1020} x_{19}^{1022} x_{20}^{1023} \rangle$.

144. [*16*] Что собой представляет квазипрофиль в случае функций сложения $\{f_1, f_2, f_3, f_4, f_5\}$ в (36)?

145. [*24*] Найдите $B_{\min}(f_1, f_2, f_3, f_4, f_5)$ и $B_{\max}(f_1, f_2, f_3, f_4, f_5)$ для этих функций.

▶ **146.** [*M22*] Пусть $(b_0, \ldots, b_n)$ и $(q_0, \ldots, q_n)$ представляют собой профиль и квазипрофиль базы БДР.

а) Докажите, что $b_0 \le \min\bigl(q_0, (b_1 + q_2)(b_1 + q_2 - 1)\bigr)$, $b_1 \le \min\bigl(b_0 + q_0, q_2(q_2 - 1)\bigr)$ и $b_0 + b_1 \ge q_0 - q_2$.

б) И обратно, если b_0, b_1, q_0 и q_2 являются неотрицательными целыми числами, удовлетворяющими указанным неравенствам, то существует база БДР с такими профилем и квазипрофилем.

▶ **147.** [*27*] Конкретизируйте детали алгоритма Руделла, работающего без привлечения дополнительной памяти, используя соглашения из алгоритма U и счетчики ссылок из упр. 82.

148. [*M21*] Истинно или ложно следующее утверждение? После обмена $① \leftrightarrow ②$ выполняется неравенство $B(f_1^\pi, \ldots, f_m^\pi) \le 2B(f_1, \ldots, f_m)$.

149. [*M20*] (Боллиг (Bollig), Лёббинг (Löbbing) и Вегенер (Wegener).) Покажите, что в дополнение к теореме J^- мы также имеем $B(f_1^\pi, \ldots, f_m^\pi) \le (2^k - 2)b_0 + B(f_1, \ldots, f_m)$ после операции подъема-спуска $k - 1$ уровней, где профилем $\{f_1, \ldots, f_m\}$ является $(b_0, \ldots, b_n)$.

150. [*30*] Когда для реализации подъемов или спусков используются многократные обмены, промежуточные результаты могут оказаться гораздо бóльшими, чем начальная или конечная БДР. Покажите, что перемещения переменных могут быть выполнены более непосредственным методом, время работы которого составляет $O\bigl(B(f_1, \ldots, f_m) + B(f_1^\pi, \ldots, f_m^\pi)\bigr)$ в наихудшем случае.

151. [*20*] Предложите способ вызова алгоритма J, такой, чтобы каждая переменная просеивалась только однократно.

152. [*25*] Скрыто взвешенная битовая функция h_{100} имеет более чем 17.5 триллиона узлов в своей БДР. Насколько сильно просеивание снижает это число? *Указание*: воспользуйтесь упр. 124 вместо реального построения бинарных диаграмм решений.

153. [*30*] Поместите функцию "крестиков-ноликов" $\{y_1, \ldots, y_9\}$ из упр. 7.1.2–65 в базу БДР. Каким будет ее размер при тестировании переменных в порядке x_1, x_2, ..., x_9, o_1, o_2, ..., o_9 сверху вниз? Чему равно значение $B_{\min}(y_1, \ldots, y_9)$?

154. [*20*] Сравнивая (104) и (106), можете ли вы сказать, насколько далеко сместился каждый штат при просеивании?

▶ **155.** [*25*] Пусть f_1 представляет собой функцию независимого множества (105) для графа смежных штатов США и пусть f_2 является соответствующей функцией ядра (см. (68)). Найдите упорядочение штатов π, такое, чтобы (а) $B(f_2^\pi)$ и (б) $B(f_1^\pi, f_2^\pi)$ были настолько малы, насколько вы сумеете этого достичь. (Заметим, что упорядочение (110) дает $B(f_1^\pi) = 339$, $B(f_2^\pi) = 795$ и $B(f_1^\pi, f_2^\pi) = 1129$.)

156. [*30*] Теоремы J^+ и J^- говорят, что мы можем сэкономить время переупорядочения, выполняя при просеивании только подъемы и не беспокоясь о спусках. Получается, что мы можем убрать шаги J3, J5, J6 и J7 из алгоритма J. Будет ли это разумно?

157. [*M24*] Покажите, что если $m + 2^m$ переменных 2^m-канального мультиплексора M_m размещаются в любом порядке, таком, что $B(M_m^\pi) > 2^{m+1} + 1$, то просеивание приведет к уменьшению размера БДР.

158. [*M24*] Если булева функция $f(x_1, \ldots, x_n)$ симметрична относительно переменных $\{x_1, \ldots, x_p\}$, естественно ожидать, что эти переменные будут находиться в последовательных позициях как минимум в одном из переупорядочений $f^\pi(x_1, \ldots, x_n)$, минимизирующих $B(f^\pi)$. Покажите, однако, что если

$$f(x_1, \ldots, x_n) = [x_1 + \cdots + x_p = \lfloor p/3 \rfloor] + [x_1 + \cdots + x_p = \lceil 2p/3 \rceil]\, g(x_{p+1}, \ldots, x_{p+m}),$$

где $p = n - m$ и $g(y_1, \ldots, y_m)$ представляет собой любую неконстантную булеву функцию, то $B(f^\pi) = \frac{1}{3}n^2 + O(n)$ при $n \to \infty$, когда $\{x_1, \ldots, x_p\}$ являются последовательными в π, но $B(f^\pi) = \frac{1}{4}n^2 + O(n)$, когда π размещает около половины этих переменных в начале и половину в конце.

159. [*20*] Базовые правила игры Джона Конвея (John Conway) "Жизнь" (см. упр. 7.1.3–167) представляет собой булеву функцию $L(x_{\mathrm{NW}}, x_{\mathrm{N}}, x_{\mathrm{NE}}, x_{\mathrm{W}}, x, x_{\mathrm{E}}, x_{\mathrm{SW}}, x_{\mathrm{S}}, x_{\mathrm{SE}})$. Какое упорядочение этих девяти переменных сделает БДР как можно меньше?

▶ **160.** [*24*] (*Шахматная "Жизнь"*.) Рассмотрим матрицу $X = (x_{ij})$ размером 8×8, состоящую из нулей и единиц, и окруженную со всех сторон бесконечным количеством нулей. Пусть $L_{ij}(X) = L(x_{(i-1)(j-1)}, \ldots, x_{ij}, \ldots, x_{(i+1)(j+1)})$ представляет собой базовое правило Конвея в позиции (i, j). Назовем X "ручной", если $L_{ij}(X) = 0$ при $i \notin [1\,..\,8]$ или $j \notin [1\,..\,8]$; в противном случае X является "дикой", поскольку активирует ячейки вне матрицы.

а) Сколько ручных конфигураций X исчезают за один шаг "Жизни", делая все $L_{ij}(X) = 0$?

б) Каков максимальный вес $\sum_{i=1}^{8} \sum_{j=1}^{8} x_{ij}$ среди всех таких решений?

в) Сколько диких конфигураций исчезает *внутри* матрицы после одного шага "Жизни"?

г) Чему равны минимальный и максимальный веса среди всех таких решений?

д) Сколько конфигураций X делают $L_{ij}(X) = 1$ для $1 \le i, j \le 8$?

е) Найдите ручных 8×8-предшественников следующих шаблонов.

(1) (2) (3) (4) (5)

(Здесь, как и в разделе 7.1.3, черные ячейки обозначают единицы в матрице.)

161. [*28*] Продолжая выполнять упр. 160, запишем $L(X) = Y = (y_{ij})$, если X представляет собой ручную матрицу, такую, что $L_{ij}(X) = y_{ij}$ для $1 \le i, j \le 8$.

а) Сколько X удовлетворяют условию $L(X) = X$ ("натюрморт")?

б) Найдите натюрморт размером 8×8 с весом 35.

в) "Перевертыш" представляет собой пару различных матриц, обладающих тем свойством, что $L(X)=Y$, $L(Y)=X$. Подсчитайте их.

г) Найдите перевертыш, у которого и X, и Y имеют вес 28.

▶ **162.** [*30*] (*Запертая "Жизнь"*) Если X и $L(X)$ ручные, но $L(L(X))$ дикая, мы говорим, что X "сбегает" из клетки за три шага. Сколько матриц 6×6 сбегают из своих клеток 6×6 ровно после k шагов для $k = 1, 2, \ldots$?

163. [*23*] Докажите формулы (112) и (113) для размеров БДР бесповторных функций.

▶ **164.** [*M27*] Чему равен максимум $B(f)$ среди всех бесповторных функций $f(x_1, \ldots, x_n)$?

165. [*M21*] Проверьте формулы с числами Фибоначчи (115) для $B(u_m)$ и $B(v_m)$.

166. [*M29*] Завершите доказательство теоремы W.

167. [*21*] Разработайте эффективный алгоритм для вычисления перестановки π, для которой для заданной бесповторной функции $f(x_1, \ldots, x_n)$ минимизированы как $B(f^\pi)$, так и $B(f^\pi, \bar{f}^\pi)$.

▶ **168.** [*HM40*] Рассмотрим следующие бинарные операции над упорядоченными парами $z = (x, y)$:

$$z \circ z' = (x, y) \circ (x', y') = \big(x + x', \min(x + y', x' + y)\big);$$
$$z \bullet z' = (x, y) \bullet (x', y') = \big(x + x' + \min(y, y'), \max(y, y')\big).$$

(Эти операции являются ассоциативными и коммутативными.) Пусть $S_1 = \{(1, 0)\}$ и

$$S_n = \bigcup_{k=1}^{n-1} \{z \circ z' \mid z \in S_k,\ z' \in S_{n-k}\} \cup \bigcup_{k=1}^{n-1} \{z \bullet z' \mid z \in S_k,\ z' \in S_{n-k}\} \text{ для } n > 1.$$

Таким образом, $S_2 = \{(2, 0), (2, 1)\}$; $S_3 = \{(3, 0), (3, 1), (3, 2)\}$; $S_4 = \{(4, 0), \ldots, (4, 3), (5, 1)\}$ и т. д.

а) Докажите, что существует бесповторная функция $f(x_1, \ldots, x_n)$, для которой мы имеем $\min_\pi B(f^\pi) = c$ и $\min_\pi B(f^\pi, \bar{f}^\pi) = c'$ тогда и только тогда, когда $(\frac{1}{2}c' - 1, c - \frac{1}{2}c' - 1) \in S_n$.

б) Истинно или ложно следующее утверждение? $0 \le y < x$ для всех $(x, y) \in S_n$.

в) Покажите, что если $z^T = (x + y, x - y)/\sqrt{2}$, то $z^T \circ z'^T = (z \bullet z')^T$ и $z^T \bullet z'^T = (z \circ z')^T$.

г) Докажите, что если β представляет собой константу в (116), то $x^2 + y^2 \le n^{2\beta}$ для всех $(x, y) \in S_n$. *Указания*: пусть $|z|^2 = x^2 + y^2$; достаточно доказать, что $|z \bullet z'| \le 2^\beta = \sqrt{2}\phi$ при $0 \le y \le x$, $0 \le y' \le x'$, $|z| = r = (1-\delta)^\beta$, $|z'| = r' = (1+\delta)^\beta$ и $0 \le \delta \le 1$. Если также $y = y'$, $z \bullet z'$ лежит внутри эллипса $(a \cos\theta + b \sin\theta, b \sin\theta)$, где $a = r + r'$ и $b = \sqrt{rr'}$.

169. [*M46*] Выполняется ли условие $\min_\pi B(f^\pi) \le B(v_{2m+1})$ для каждой бесповторной функции f от 2^{2m+1} переменных?

▶ **170.** [*M25*] Будем называть булеву функцию "худой" ("skinny"), если ее БДР включает все переменные простейшим возможным способом: худая БДР имеет ровно один узел ветвления $\textcircled{j}$ для каждой переменной x_j, и либо LO, либо HI в каждом ветвлении является стоковым узлом.

а) Сколько булевых функций $f(x_1, \ldots, x_n)$ является худыми в указанном смысле?

б) Сколько из них монотонных?

в) Покажите, что $f_t(x_1, \ldots, x_n) = [(x_1 \ldots x_n)_2 \geq t]$ является худой, когда $0 < t < 2^n$ и t нечетно.

г) Что собой представляет функция, *дуальная* функции f_t из п. (в)?

д) Поясните, как найти кратчайшие формулы CNF и DNF для f_t для заданного t.

171. [*M26*] Продолжая выполнять упр. 170, покажите, что функция является *бесповторной* и *регулярной* тогда и только тогда, когда она худая и монотонная.

172. [*M27*] Сколько худых функций являются также функциями Хорна? Сколько из них обладают тем свойством, что как f, *так и* $\bar{f}$ удовлетворяют условию Хорна?

▶ **173.** [*HM28*] Сколько в точности булевых функций $f(x_1, \ldots, x_n)$ являются худыми после некоторого переупорядочения переменных $f(x_{1\pi}, \ldots, x_{n\pi})$?

▶ **174.** [*M39*] Пусть S_n представляет собой количество булевых функций $f(x_1, \ldots, x_n)$, БДР которых является "тонкой" в том смысле, что она имеет ровно один узел, помеченный как $\textcircled{j}$, для $1 \leq j \leq n$. Покажите, что S_n также является количеством комбинаторных объектов следующих типов.

а) *Перестановки Деллака* порядка $2n$ (т. е. перестановки $p_1 p_2 \ldots p_{2n}$, такие, что $\lceil k/2 \rceil \leq p_k \leq n + \lceil k/2 \rceil$ для $1 \leq k \leq 2n$).

б) *Разупорядочения Дженоччи* порядка $2n + 2$ (а именно, перестановки $q_1 q_2 \ldots q_{2n+2}$, такие, что $q_k > k$ тогда и только тогда, когда k нечетно, для $1 \leq k \leq 2n+2$; кроме того, в разупорядочении $q_k \neq k$).

в) *Неприводимые пистолеты Дюмона* порядка $2n + 2$ (а именно, последовательности $r_1 r_2 \ldots r_{2n+2}$, такие, что $k \leq r_k \leq 2n + 2$ для $1 \leq k \leq 2n+2$ и $\{r_1, r_2, \ldots, r_{2n+2}\} = \{2, 4, 6, \ldots, 2n, 2n+2\}$, с тем особым свойством, что $2k \in \{r_1, \ldots, r_{2k-1}\}$ для $1 \leq k \leq n$).

г) Пути от $(1, 0)$ до $(2n + 2, 0)$ в ориентированном графе

$$\begin{array}{ccccccccccccccc}
 & & & & & & & & & & (7,3) & \longrightarrow & (8,3) & \longrightarrow & \cdots \\
 & & & & & & & & & & \uparrow & & \downarrow & & \\
 & & & & & & & & (5,2) & \longrightarrow & (6,2) & \longrightarrow & (7,2) & \longrightarrow & (8,2) \longrightarrow \cdots \\
 & & & & & & & & \uparrow & & \downarrow & & \uparrow & & \downarrow \\
 & & & & (3,1) & \longrightarrow & (4,1) & \longrightarrow & (5,1) & \longrightarrow & (6,1) & \longrightarrow & (7,1) & \longrightarrow & (8,1) \longrightarrow \cdots \\
 & & & & \uparrow & & \downarrow & & \uparrow & & \downarrow & & \uparrow & & \downarrow \\
(1,0) & \longrightarrow & (2,0) & \longrightarrow & (3,0) & \longrightarrow & (4,0) & \longrightarrow & (5,0) & \longrightarrow & (6,0) & \longrightarrow & (7,0) & \longrightarrow & (8,0) \longrightarrow \cdots
\end{array} .$$

(Обратите внимание, что подсчитать объекты типа (г) очень легко.)

175. [*M30*] Продолжая выполнять упр. 174, найдите способ подсчета булевых функций, БДР которых содержит ровно b_{j-1} узлов, помеченных $\textcircled{j}$, при заданном профиле $(b_0, \ldots, b_{n-1}, b_n)$.

176. [*M35*] Для завершения доказательства теоремы X мы воспользуемся упр. 6.4–78, которое гласит, что $\{h_{a,b} \mid a \in A$ и $b \in B\}$ является универсальным семейством хеш-функций, отображающих n бит на l бит, когда $h_{a,b}(x) = ((ax + b) \gg (n - l)) \bmod 2^l$, $A = \{a \mid 0 < a < 2^n,\ a$ нечетно$\}$, $B = \{b \mid 0 \leq b < 2^{n-l}\}$ и $0 \leq l \leq n$. Пусть $I = \{h_{a,b}(p) \mid p \in P\}$ и $J = \{h_{a,b}(q) \mid q \in Q\}$.

а) Покажите, что если $2^l - 1 \leq 2^{t-1}\epsilon/(1 - \epsilon)$, то существуют константы $a \in A$ и $b \in B$, для которых $|I| \geq (1 - \epsilon)2^l$ и $|J| \geq (1 - \epsilon)2^l$.

б) Для таких заданных a положим $J = \{j_1, \ldots, j_{|J|}\}$, где $0 = j_1 < \cdots < j_{|J|}$, и выберем $Q' = \{q_1, \ldots, q_{|J|}\} \subseteq Q$, так что $h_{a,b}(q_k) = j_k$ для $1 \leq k \leq |J|$. Пусть $g(q)$

обозначает средние $l-1$ бит aq, а именно $(aq \gg (n-l+1)) \bmod 2^{l-1}$. Докажите, что $g(q) \neq g(q')$, когда q и q' представляют собой различные элементы множества $Q'' = \{q_1, q_3, \ldots, q_{2\lceil |J|/2 \rceil - 1}\}$.

в) Докажите, что следующее множество Q^* удовлетворяет условию (120), когда $l \geq 3$ и $y = a$:

$$Q^* \;=\; \{q \mid q \in Q'',\ g(q) \text{ четно, и } g(p) + g(q) = 2^{l-1} \text{ для некоторого } p \in P\}.$$

г) Наконец покажите, что $|Q^*|$ достаточно велико для доказательства теоремы X.

177. [*M22*] Завершите доказательство теоремы A путем ограничения всего квази-профиля.

178. [*M24*] (Амано (Amano) и Маруока (Maruoka).) Улучшите константу в (121), используя лучшее упорядочение переменных: $Z_n(x_{2n-1}, x_1, x_3, \ldots, x_{2n-3}; x_{2n}, x_2, x_4, \ldots, x_{2n-2})$.

179. [*M47*] Удовлетворяет ли средний бит умножения условию $B_{\min}(Z_n) = \Theta(2^{6n/5})$?

180. [*M27*] Докажите теорему Y, используя указания из раздела.

181. [*M21*] Пусть $L_{m,n}$ является *функцией старшего бита* $Z_{m,n}^{(m+n)}(x_1, \ldots, x_m; y_1, \ldots, y_n)$. Докажите, что $B_{\min}(L_{m,n}) = O(2^m n)$ при $m \leq n$.

182. [*M38*] (И. Вегенер (I. Wegener).) Растет ли $B_{\min}(L_{n,n})$ экспоненциально при $n \to \infty$?

▶ **183.** [*M25*] Изобразите несколько первых уровней БДР для "ограничивающей функции старшего бита"

$$\big[(.x_1x_3x_5\ldots)_2 \cdot (.x_2x_4x_6\ldots)_2 \geq \tfrac{1}{2}\big],$$

которая имеет бесконечно много булевых переменных. Сколько узлов b_k имеется на уровне k? (Мы не можем позволить $(.x_1x_3x_5\ldots)_2$ или $(.x_2x_4x_6\ldots)_2$ заканчиваться бесконечным количеством единиц.)

184. [*M23*] Что собой представляют профили БДР и НДР функции перестановки P_m?

185. [*M25*] Насколько большим может быть значение $Z(f)$, когда f представляет собой симметричную булеву функцию от n переменных? (См. упр. 44.)

186. [*10*] Какая булева функция от $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ имеет НДР ' '?

▶ **187.** [*20*] Изобразите НДР для всех 16 булевых функций $f(x_1, x_2)$ от двух переменных.

188. [*16*] Выразите эти 16 булевых функций $f(x_1, x_2)$ в виде семейств подмножеств множества $\{1, 2\}$.

189. [*18*] Какие функции $f(x_1, \ldots, x_n)$ обладают НДР, совпадающей с БДР?

190. [*20*] Опишите все функции f, для которых (а) $Q(f) = B(f)$; (б) $Q(f) = Z(f)$.

▶ **191.** [*HM25*] Сколько функций $f(x_1, \ldots, x_n)$ не имеют $\boxed{\perp}$ в своих НДР?

192. [*M20*] Определим Z-преобразование бинарных строк следующим образом: $\epsilon^Z = \epsilon$, $0^Z = 0$, $1^Z = 1$ и

$$(\alpha\beta)^Z = \begin{cases} \alpha^Z\alpha^Z, & \text{если } |\alpha| = n \text{ и } \beta = 0^n; \\ \alpha^Z 0^n, & \text{если } |\alpha| = n \text{ и } \beta = \alpha; \\ \alpha^Z\beta^Z, & \text{если } |\alpha| = |\beta| - 1 \text{ или если } |\alpha| = |\beta| = n \text{ и } \alpha \neq \beta \neq 0^n. \end{cases}$$

а) Что собой представляет 11001001000011111^Z?

б) Истинно или ложно следующее утверждение? $(\tau^Z)^Z = \tau$ для всех бинарных строк τ.

в) Если $f(x_1, \ldots, x_n)$ представляет собой булеву функцию с таблицей истинности τ, обозначим как $f^Z(x_1, \ldots, x_n)$ булеву функцию, таблицей истинности которой является τ^Z. Покажите, что профиль f почти идентичен z-профилю f^Z, и наоборот. (Следовательно, теорема U выполняется как для НДР, так и для БДР, и статистика, такая как (80), корректна и для z-профилей.)

193. [*M21*] Продолжая выполнять упр. 192, ответьте, что собой представляет $S_k^Z(x_1, \ldots, x_n)$ при $0 \le k \le n$?

194. [*M25*] Сколько $f(x_1, \ldots, x_n)$ имеют z-профиль $(1, \ldots, 1)$? (См. упр. 174.)

195. [*24*] Найдите $Z(M_2)$, $Z_{\min}(M_2)$ и $Z_{\max}(M_2)$, где M_2 представляет собой 4-канальный мультиплексор.

196. [*M21*] Найдите функцию $f(x_1, \ldots, x_n)$, для которой $Z(f) = O(n)$ и $Z(\bar{f}) = \Omega(n^2)$.

197. [*25*] Модифицируйте алгоритм из упр. 138 так, чтобы он вычислял "главную z-профилограмму" f. (Тогда $Z_{\min}(f)$ и $Z_{\max}(f)$ можно найти, как в упр. 133.)

▶ **198.** [*23*] Поясните, как вычислить И(f, g) с использованием НДР вместо БДР (см. (55)).

199. [*21*] Аналогично реализуйте (а) ИЛИ(f, g), (б) ЛИБО(f, g), (в) НО-НЕ(f, g).

200. [*21*] И еще раз аналогично реализуйте MUX(f, g, h) для НДР (см. (62)).

201. [*22*] Каждая проецирующая функция x_j имеет простую трехузловую БДР, но их НДР-представление более сложное. Предложите способ реализации этих функций с использованием инструментария НДР общего назначения.

202. [*24*] Какие изменения требуется внести в алгоритм без привлечения дополнительной памяти из упр. 147, когда обмен уровней $(u) \leftrightarrow (v)$ выполняется не в базе БДР, а в базе НДР?

▶ **203.** [*M24*] (*Алгебра семейств.*) При работе с конечными семействами конечных подмножеств положительных целых чисел и их представлениями в виде НДР полезны следующие алгебраические соглашения. Простейшими такими семействами являются *пустое семейство*, обозначаемое как $\emptyset$ и представляемое $\boxed{\bot}$; *единичное семейство* $\{\emptyset\}$, обозначаемое как ϵ и представляемое $\boxed{\top}$; и *элементарные семейства* $\{\{j\}\}$ для $j \ge 1$, обозначаемые как e_j и представляемые узлом ветвления (j) с LO $= \boxed{\bot}$ и HI $= \boxed{\top}$. (В упр. 186 показана НДР для e_3.)

Два семейства, f и g, можно скомбинировать с помощью обычных операций с множествами.

- *Объединение* $f \cup g = \{\alpha \mid \alpha \in f$ или $\alpha \in g\}$ реализуется с помощью ИЛИ(f, g).
- *Пересечение* $f \cap g = \{\alpha \mid \alpha \in f$ и $\alpha \in g\}$ реализуется с помощью И(f, g).
- *Разность* $f \setminus g = \{\alpha \mid \alpha \in f$ и $\alpha \notin g\}$ реализуется с помощью НО-НЕ(f, g).
- *Симметричная разность* $f \oplus g = (f \setminus g) \cup (g \setminus f)$ реализуется с помощью ЛИБО(f, g).

Мы также определяем три новых способа построения семейств подмножеств.

- *Соединение* (*join*) $f \sqcup g = \{\alpha \cup \beta \mid \alpha \in f$ и $\beta \in g\}$, иногда записываемое просто как fg.
- *Встреча* (*meet*) $f \sqcap g = \{\alpha \cap \beta \mid \alpha \in f$ и $\beta \in g\}$.
- *Дельта* (*delta*) $f \boxplus g = \{\alpha \oplus \beta \mid \alpha \in f$ и $\beta \in g\}$.

Все три операции коммутативны и ассоциативны: $f \sqcup g = g \sqcup f$, $f \sqcup (g \sqcup h) = (f \sqcup g) \sqcup h$ и т. д.

а) Предположим, что $f = \{\emptyset, \{1, 2\}, \{1, 3\}\} = \epsilon \cup (e_1 \sqcup (e_2 \cup e_3))$ и $g = \{\{1, 2\}, \{3\}\} = (e_1 \sqcup e_2) \cup e_3$. Что собой представляют $f \sqcup g$ и $(f \sqcap g) \setminus (f \boxplus e_1)$?

б) Любое семейство f можно также рассматривать как булеву функцию $f(x_1, x_2, \ldots)$, где $\alpha \in f \iff f([1 \in \alpha], [2 \in \alpha], \ldots) = 1$. Опишите операции $\sqcup$, $\sqcap$ и $\boxplus$ через булевы логические формулы.

в) Какие из приведенных далее формул выполняются для всех семейств f, g и h? (i) $f \sqcup (g \cup h) = (f \sqcup g) \cup (f \sqcup h)$; (ii) $f \sqcap (g \cup h) = (f \sqcap g) \cup (f \sqcap h)$; (iii) $f \sqcup (g \sqcap h) = (f \sqcup g) \sqcap (f \sqcup h)$; (iv) $f \cup (g \sqcup h) = (f \cup g) \sqcup (f \cup h)$; (v) $f \boxplus \emptyset = \emptyset \sqcap g = h \sqcup \emptyset$; (vi) $f \sqcap \epsilon = \epsilon$.

г) Мы говорим, что f и g *ортогональны*, записывая это как $f \perp g$, если $\alpha \cap \beta = \emptyset$ для всех $\alpha \in f$ и всех $\beta \in g$. Какие из приведенных далее утверждений истинны

для всех семейств f и g? (i) $f \perp g \iff f \sqcap g = \epsilon$; (ii) $f \perp g \implies |f \sqcup g| = |f||g|$; (iii) $|f \sqcup g| = |f||g| \implies f \perp g$; (iv) $f \perp g \iff f \sqcup g = f \boxplus g$.

д) Опишите все семейства f, для которых справедливы следующие утверждения: (i) $f \cup g = g$ для всех g; (ii) $f \sqcup g = g$ для всех g; (iii) $f \sqcap g = g$ для всех g; (iv) $f \sqcup (e_1 \sqcup e_2) = f$; (v) $f \sqcup (e_1 \cup e_2) = f$; (vi) $f \boxplus ((e_1 \sqcup e_2) \cup e_3) = f$; (vii) $f \boxplus f = \epsilon$; (viii) $f \sqcap f = f$.

▶ **204.** [*M25*] Продолжая выполнять упр. 203, рассмотрим две другие важные операции над семействами f и g.

- *Частное* (*quotient*) $f/g = \{\alpha \mid \alpha \cup \beta \in f$ и $\alpha \cap \beta = \emptyset$ для всех $\beta \in g\}$.
- *Остаток* (*remainder*) $f \bmod g = f \setminus (g \sqcup (f/g))$.

Частное иногда также называют "кофактором" ("cofactor") f по отношению к g.

а) Докажите, что $f/(g \cup h) = (f/g) \cap (f/h)$.

б) Предположим, что $f = \{\{1,2\},\{1,3\},\{2\},\{3\},\{4\}\}$. Что собой представляют f/e_2 и $f/(f/e_2)$?

в) Упростите выражения $f/\emptyset$, f/ϵ, f/f и $(f \bmod g)/g$ для произвольных f и g.

г) Покажите, что $f/g = f/(f/(f/g))$. *Указание*: начните с отношения $g \subseteq f/(f/g)$.

д) Докажите, что f/g можно также определить как $\bigcup \{h \mid g \sqcup h \subseteq f$ и $g \perp h\}$.

е) Покажите, что для заданных f и j семейство f имеет единственное представление $(e_j \sqcup g) \cup h$ с $e_j \perp (g \cup h)$.

ж) Истинны или ложны следующие утверждения? $(f \sqcup g) \bmod e_j = (f \bmod e_j) \sqcup (g \bmod e_j)$; $(f \sqcap g)/e_j = (f/e_j) \sqcap (g/e_j)$.

205. [*M25*] Реализуйте пять базовых операций алгебры семейств, а именно (а) $f \sqcup g$, (б) $f \sqcap g$, (в) $f \boxplus g$, (г) f/g и (д) $f \bmod g$, используя соглашения из упр. 198.

206. [*M46*] Чему равно время работы алгоритмов из упр. 205 в наихудшем случае?

▶ **207.** [*M25*] Когда в приложениях требуется одна или несколько проецирующих функций x_j, как в упр. 201, очень удобным оказывается следующий "симметризующий" оператор:

$$(e_{i_1} \cup e_{i_2} \cup \cdots \cup e_{i_l}) \S k \;=\; S_k(x_{i_1}, x_{i_2}, \ldots, x_{i_l}), \qquad \text{целое } k \ge 0.$$

Например, $e_j \S 1 = x_j$; $e_j \S 0 = \bar{x}_j$; $(e_i \cup e_j) \S 1 = x_i \oplus x_j$; $(e_2 \cup e_3 \cup e_5) \S 2 = (x_2 \wedge x_3 \wedge \bar{x}_5) \vee (x_2 \wedge \bar{x}_3 \wedge x_5) \vee (\bar{x}_2 \wedge x_3 \wedge x_5)$. Покажите, что этот оператор легко реализуется. (Заметим, что $e_{i_1} \cup \cdots \cup e_{i_l}$ имеет при $l > 0$ очень простую НДР размером $l + 2$.)

▶ **208.** [*16*] Покажите путем модификации алгоритма C, что все решения булевой функции можно легко подсчитать и в ситуации, когда вместо БДР функции задана ее НДР.

209. [*M21*] Поясните, как вычислить полностью детализированную таблицу истинности булевой функции из ее НДР-представления. (См. упр. 31.)

▶ **210.** [*23*] Покажите, как при заданной НДР для f построить НДР для функции

$$g(x) = [f(x) = 1 \text{ и } \nu x = \max\{\nu y \mid f(y) = 1\}].$$

211. [*M20*] Справедливо ли соотношение $Z(f) \le B(f)$, когда f описывает решения задачи точного покрытия?

▶ **212.** [*25*] Укажите эффективный способ вычисления НДР для задачи точного покрытия.

213. [*16*] Почему нельзя покрыть костями домино обрезанную шахматную доску?

▶ **214.** [*21*] Когда некоторая фигура покрывается костями домино, мы будем говорить, что покрытие является *безошибочным*, если каждая прямая линия, пересекающая внутренность фигуры, пересекает также внутренность некоторого домино. Например, правое покрытие в (127) является безошибочным, в отличие от среднего, таковым не являющегося; о левом не приходится и говорить.

Сколько покрытий шахматной доски костями домино являются безошибочными?

215. [*21*] Японские циновки татами являются прямоугольниками 1×2, которые традиционно используются для покрытия прямоугольных полов таким образом, чтобы ни в одном углу не встретились 4 циновки одновременно. Например, на рис. 29, (а) показано покрытие прямоугольника 6×5 из издания 1641 года книги Мицуёши Ёшиды (Mitsuyoshi Yoshida) *Jinkōki*, которая впервые была опубликована в 1627 году.

Найдите все покрытия шахматной доски костями домино, отвечающие правилам покрытия татами.

Рис. 29. Два красивых примера:
(а) покрытие татами XVII века;
(б) трехцветное покрытие домино.

(а) (б)

▶ **216.** [*30*] На рис. 29, (б) показано покрытие шахматной доски красными, синими и белыми домино таким образом, что никакие два домино одного цвета не прилегают одно к другому.

а) Сколькими способами это можно сделать?

б) Сколько из 12 988 816 покрытий костями домино являются раскрашиваемыми в 3 цвета?

217. [*29*] Покрытие мономино/домино/тримино, показанное в (130), как оказывается, удовлетворяет дополнительному ограничению: *никакие две конгруэнтные фигуры не являются смежными.* Сколько из упомянутых в тексте 92 секстиллионов покрытий являются "раздельными" в указанном смысле?

▶ **218.** [*24*] Примените методы работы с БДР и НДР к задаче поиска пар Лэнгфорда, рассматривавшейся в начале этой главы.

219. [*20*] Чему равно $Z(F)$, когда F является семейством (а) `WORDS(1000)`; (б) `WORDS(2000)`; ...; (д) `WORDS(5000)`?

▶ **220.** [*21*] z-профиль для 5757 слов из Stanford GraphBase, представленных 130 переменными $a_1 \mathinner{..} z_5$, как рассматривалось в (131), имеет вид (1, 1, 1, ..., 1, 1, 1, 23, 3, ..., 6, 2, 0, 3, 2, 1, 1, 2).

а) Поясните элементы профиля 23 и 3, соответствующие переменным a_2 и b_2.

б) Поясните последние элементы 0, 3, 2, 1, 1, 2, соответствующие переменным v_5, w_5, x_5 и т. д.

▶ **221.** [*M27*] Чтобы представить 5757 наиболее распространенных пятибуквенных слов английского языка с использованием 130 переменных, достаточно 5020 узлов, что связано с лингвистическими свойствами обрабатываемых данных. Однако всего существует $26^5 = 11\,881\,376$ возможных пятибуквенных слов. Предположим, что мы выбираем 5757 из них случайным образом. Насколько большой, в среднем, окажется при этом НДР?

▶ **222.** [*27*] При применении алгебры семейств к пятибуквенным словам, как в (131), 130 переменных называются a_1, b_1, ..., z_5 вместо x_1, x_2, ..., x_{130}; а соответствующие элементарные семейства обозначаются символами $\mathtt{a}_1$, $\mathtt{b}_1$, ..., $\mathtt{z}_5$ вместо e_1, e_2, ..., e_{130}. Таким образом, семейство $F = \mathtt{WORDS(5757)}$ можно построить путем синтезирования формулы

$$F = (\mathtt{w}_1 \sqcup \mathtt{h}_2 \sqcup \mathtt{i}_3 \sqcup \mathtt{c}_4 \sqcup \mathtt{h}_5) \cup \cdots \cup (\mathtt{f}_1 \sqcup \mathtt{u}_2 \sqcup \mathtt{n}_3 \sqcup \mathtt{n}_4 \sqcup \mathtt{y}_5) \cup \cdots \cup (\mathtt{p}_1 \sqcup \mathtt{u}_2 \sqcup \mathtt{p}_3 \sqcup \mathtt{a}_4 \sqcup \mathtt{l}_5).$$

а) Пусть $\wp$ обозначает *универсальное семейство* всех подмножеств $\{a_1, \ldots, z_5\}$, именуемое также "показательным множеством". Что означает формула $F \sqcap \wp$?

б) Пусть $X = X_1 \sqcup \cdots \sqcup X_5$, где $X_j = \{\mathtt{a}_j, \mathtt{b}_j, \ldots, \mathtt{z}_j\}$. Поясните формулу $F \sqcap X$.

в) Найдите простую формулу для всех слов F, которые соответствуют шаблону `t*u*h`.
г) Найдите формулу для всех слов SGB, которые содержат ровно k гласных, для $0 \le k \le 5$ (рассматривайте как гласные только `a`, `e`, `i`, `o` и `u`). Положите $V_j = \mathtt{a}_j \cup \mathtt{e}_j \cup \mathtt{i}_j \cup \mathtt{o}_j \cup \mathtt{u}_j$.
д) Сколько имеется шаблонов, в которых определены ровно три буквы, соответствуют как минимум одному слову SGB? (Например, таким шаблоном является `m*tc*`.) Приведите формулу.
е) Сколько из этих шаблонов соответствуют не менее чем двум словам (как, например, шаблон `*atc*`)?
ж) Выразите все слова, которые остаются словами, когда '`b`' заменяется на '`o`'.
з) Какой смысл заключен в формуле F/V_2?
и) Сравните $(X_1 \sqcup V_2 \sqcup V_3 \sqcup V_4 \sqcup X_5) \cap F$ с $(X_1 \sqcup X_5) \setminus ((\wp \setminus F)/(V_2 \sqcup V_3 \sqcup V_4))$.

223. [*28*] "Медианное слово" представляет собой пятибуквенное слово $\mu = \mu_1 \ldots \mu_5$, которое может быть получено из трех слов $\alpha = \alpha_1 \ldots \alpha_5$, $\beta = \beta_1 \ldots \beta_5$, $\gamma = \gamma_1 \ldots \gamma_5$ согласно правилу $[\alpha_i = \mu_i] + [\beta_i = \mu_i] + [\gamma_i = \mu_i] = 2$ для $1 \le i \le 5$. Например, `mixed` является медианой слов {`fixed`, `mixer`, `mound`}, а также слов {`mated`, `mixup`, `nixed`}. Но `noted` не является медианой {`notes`, `voted`, `naked`}, поскольку каждое из этих слов имеет `e` в четвертой позиции.

а) Покажите, что $\{d(\alpha,\mu), d(\beta,\mu), d(\gamma,\mu)\}$ представляют собой либо $\{1,1,3\}$, либо $\{1,2,2\}$, когда μ является медианой $\{\alpha,\beta,\gamma\}$. (Здесь d означает расстояние Хэмминга.)
б) Сколько медиан можно получить из `WORDS(`n`)`, когда $n = 100$? 1000? 5757?
в) Сколько из этих медиан принадлежат `WORDS(`m`)`, когда $m = 100$? 1000? 5757?

▶ **224.** [*20*] Предположим, что мы образуем НДР для всех путей от источника к стоку в ориентированном ациклическом графе, как на рис. 28, когда этот ориентированный ациклический граф представляет собой лес; т. е. считаем, что каждая вершина ориентированного ациклического графа, не являющаяся источником, имеет входящую степень 1. Покажите, что соответствующая НДР, по сути, такая же, как и бинарное дерево, представляющее лес при условии "естественного соответствия между лесами и бинарными деревьями" (см. 2.3.2–(1)–2.3.2–(3)).

▶ **225.** [*30*] Разработайте алгоритм, который будет строить для заданного графа и двух различных вершин $\{s,t\}$ этого графа НДР для всех множеств ребер, образующих простой путь от s до t.

▶ **226.** [*20*] Модифицируйте алгоритм из упр. 225 так, чтобы он давал НДР для всех простых *циклов* заданного графа.

227. [*20*] Аналогично модифицируйте его так, чтобы он рассматривал только *гамильтоновы пути* от s до t.

228. [*21*] Измените его еще раз для гамильтоновых путей от s до *любой* другой вершины.

229. [*15*] Имеется 587 218 421 488 путей от `CA` до `ME` в графах (18), но в (133) только 437 525 772 584 таких пути. Поясните такое различие.

230. [*25*] Найдите гамильтоновы пути в (133), имеющие минимальную и максимальную общие длины. Чему равна *средняя* длина, если все гамильтоновы пути равновероятны?

231. [*23*] Сколькими способами шахматный король может пройти из одного угла шахматной доски в противоположный, не посещая одну и ту же клетку дважды? (Это простые пути из угла в угол графа $P_8 \boxtimes P_8$.)

▶ **232.** [*23*] Продолжая выполнять упр. 231, рассмотрим *обход королем* шахматной доски, который представляет собой гамильтонов цикл в $P_8 \boxtimes P_8$. Определите точное количество обходов шахматной доски королем. Какой обход является наидлиннейшим в смысле пройденного королем евклидова расстояния?

▶ **233.** [*25*] Разработайте алгоритм для построения НДР для семейства всех *ориентированных циклов* данного ориентированного графа. (См. упр. 226.)

234. [*22*] Примените алгоритм из упр. 233 к ориентированному графу из 49 почтовых кодов AL, AR, ..., WY из (18), с XY $\longrightarrow$ YZ, как в упр. 7–54, (б). Например, одним таким ориентированным циклом является NC $\longrightarrow$ CT $\longrightarrow$ TN $\longrightarrow$ NC. Сколько всего ориентированных циклов может быть? Какова минимальная и максимальная длина такого цикла?

235. [*22*] Постройте ориентированный граф для пятибуквенных английских слов, в котором $x \longrightarrow y$, когда последние три буквы x соответствуют первым трем буквам y (например, crown $\longrightarrow$ owner). Сколько ориентированных циклов имеется в таком графе? Какие из них самые длинные и самые короткие?

▶ **236.** [*M25*] Многие расширения алгебры семейств из упр. 203 напрашиваются сами собой при применении НДР к комбинаторным задачам, включая следующие пять операций над семействами множеств.

- *Максимальные элементы* $f^{\uparrow} = \{$Из $\alpha \in f \mid \beta \in f$ и $\alpha \subseteq \beta$ вытекает $\alpha = \beta\}$.
- *Минимальные элементы* $f^{\downarrow} = \{$Из $\alpha \in f \mid \beta \in f$ и $\alpha \supseteq \beta$ вытекает $\alpha = \beta\}$.
- *Неподмножества* $f \nearrow g = \{$Из $\alpha \in f \mid \beta \in g$ вытекает $\alpha \not\subseteq \beta\}$.
- *Ненадмножества* $f \searrow g = \{$Из $\alpha \in f \mid \beta \in g$ вытекает $\alpha \not\supseteq \beta\}$.
- *Минимальные совпадающие множества* $f^{\sharp} = \{$Из $\alpha \mid \beta \in f$ вытекает $\alpha \cap \beta \neq \emptyset\}^{\downarrow}$.

Например, когда f и g представляют собой семейства из упр. 203, (а), мы имеем $f^{\uparrow} = e_1 \sqcup (e_2 \cup e_3)$, $f^{\downarrow} = \epsilon$, $f^{\sharp} = \emptyset$, $g^{\uparrow} = g^{\downarrow} = g$, $g^{\sharp} = (e_1 \cup e_2) \sqcup e_3$, $f \nearrow g = e_1 \sqcup e_3$, $f \searrow g = \epsilon$, $g \nearrow f = g \searrow f = \emptyset$.

а) Докажите, что $f \nearrow g = f \setminus (f \sqcap g)$, и приведите подобную формулу для $f \searrow g$.

б) Пусть $f^C = \{\overline{\alpha} \mid \alpha \in f\} = f \boxplus U$, где $U = e_1 \sqcup e_2 \sqcup \cdots$ является "универсумом". Ясно, что $f^{CC} = f$, $(f \cup g)^C = f^C \cup g^C$, $(f \cap g)^C = f^C \cap g^C$, $(f \setminus g)^C = f^C \setminus g^C$. Покажите, что имеются также законы дуальности $f^{\uparrow C} = f^{C\downarrow}$, $f^{\downarrow C} = f^{C\uparrow}$; $(f \sqcup g)^C = f^C \sqcap g^C$, $(f \sqcap g)^C = f^C \sqcup g^C$; $(f \nearrow g)^C = f^C \searrow g^C$, $(f \searrow g)^C = f^C \nearrow g^C$; $f^{\sharp} = (\wp \nearrow f^C)^{\downarrow}$.

в) Истинны или ложны следующие утверждения? (i) $x_1^{\downarrow} = e_1$; (ii) $x_1^{\uparrow} = e_1$; (iii) $x_1^{\sharp} = e_1$; (iv) $(x_1 \vee x_2)^{\downarrow} = e_1 \cup e_2$; (v) $(x_1 \wedge x_2)^{\downarrow} = e_1 \sqcup e_2$.

г) Какие из следующих формул выполняются для всех семейств f, g и h? (i) $f^{\uparrow\uparrow} = f^{\uparrow}$; (ii) $f^{\uparrow\downarrow} = f^{\downarrow}$; (iii) $f^{\uparrow\downarrow} = f^{\uparrow}$; (iv) $f^{\downarrow\uparrow} = f^{\downarrow}$; (v) $f^{\sharp\downarrow} = f^{\sharp}$; (vi) $f^{\sharp\uparrow} = f^{\sharp}$; (vii) $f^{\downarrow\sharp} = f^{\sharp}$; (viii) $f^{\uparrow\sharp} = f^{\sharp}$; (ix) $f^{\sharp\sharp} = f^{\sharp}$; (x) $f \nearrow (g \cup h) = (f \nearrow g) \cap (f \nearrow h)$; (xi) $f \searrow (g \cup h) = (f \searrow g) \cap (f \searrow h)$; (xii) $f \searrow (g \cup h) = (f \searrow g) \searrow h$; (xiii) $f \nearrow g^{\uparrow} = f \nearrow g$; (xiv) $f \searrow g^{\uparrow} = f \searrow g$; (xv) $(f \sqcup g)^{\sharp} = (f^{\sharp} \cup g^{\sharp})^{\downarrow}$; (xvi) $(f \cup g)^{\sharp} = (f^{\sharp} \sqcup g^{\sharp})^{\downarrow}$.

д) Предположим, что $g = \bigcup_{u-v}(e_u \sqcup e_v)$ является семейством всех ребер графа, и пусть f является семейством всех независимых множеств. Используя операции расширенной алгебры семейств, найдите простые формулы для выражения (i) f через g; (ii) g через f.

237. [*25*] Реализуйте пять операций из упр. 236 в стиле упр. 205.

▶ **238.** [*22*] Используйте НДР для вычисления *максимальных порожденных двудольных подграфов* графа смежных штатов США G из (18), а именно максимальных подмножеств U, таких, что $G|U$ не имеет циклов нечетной длины. Сколько таких множеств U существует? Приведите примеры наименьшего из них и наибольшего. Рассмотрите также максимальные порожденные *трехдольные* (раскрашиваемые тремя цветами) подграфы.

▶ **239.** [*21*] Поясните, как вычислить *максимальную клику* графа G с использованием алгебры семейств, когда G определен своими ребрами g, как в упр. 236, (д). Найдите максимальное множество вершин, которое может быть покрыто k кликами, для $k = 1, 2, \ldots$, когда G представляет собой граф (18).

▶ **240.** [*22*] Множество вершин U называется *доминирующим множеством* графа, если каждая вершина удалена не более чем на один шаг от U.

а) Докажите, что каждое ядро графа является минимальным доминирующим множеством.

б) Сколько минимальных доминирующих множеств имеет граф США (18)?

в) Найдите семь вершин (18), доминирующих над 36 прочими.

▶ **241.** [*28*] *Граф ферзя* Q_8 состоит из 64 полей шахматной доски, с u — v, когда поля u и v лежат в одном и том же ряду, столбце или на диагонали. Насколько велика НДР для его (а) ядер? (б) максимальных клик? (в) минимальных доминирующих множеств? (г) минимальных доминирующих множеств, являющихся кликами? (д) максимальных порожденных двудольных подграфов?

Проиллюстрируйте каждую из этих пяти категорий наименьшим и наибольшим примерами.

242. [*24*] Найдите все максимальные способы выбора точек на решетке 8×8 так, чтобы никакие три точки не лежали на прямой линии с любым наклоном.

243. [*M23*] *Замыканием* $f^\cap$ семейства множеств f является семейство всех множеств, которые могут быть получены путем пересечения одного или нескольких членов f.

а) Докажите, что $f^\cap = \{\alpha \mid \alpha = \bigcap\{\beta \mid \beta \in f \text{ и } \beta \supseteq \alpha\}\}$.

б) Как эффективно вычислить НДР для $f^\cap$ при заданной НДР для f?

в) Найдите производящую функцию для $F^\cap$ при $F = \texttt{WORDS(5757)}$, как в упр. 222.

244. [*25*] Что собой представляет НДР для функции связности для $P_3 \square P_3$ (рис. 22)? Какой вид имеет БДР для функции остовного дерева того же графа? (См. следствие S.)

▶ **245.** [*M22*] Покажите, что *простые дизъюнкты* монотонной функции f представляют собой $\mathrm{PI}(f)^\sharp$.

246. [*M21*] Докажите теорему S в предположении, что (137) истинно.

▶ **247.** [*M27*] Определите количество легких булевых функций от n переменных для $n \le 7$.

248. [*M22*] Истинно или ложно следующее утверждение? Если f и g легкие, то таковой же является и $f(x_1, \ldots, x_n) \wedge g(x_1, \ldots, x_n)$.

249. [*HM31*] Функция связности графа "сверхлегкая" в том смысле, что она легкая при всех перестановках ее переменных. Имеется ли эффективный способ описания сверхлегких булевых функций?

250. [*28*] Имеется 7581 монотонная булева функция $f(x_1, x_2, x_3, x_4, x_5)$. Чему равны средние значения $B(f)$ и $Z(\mathrm{PI}(f))$, когда одна из них выбрана случайным образом? Чему равна вероятность того, что $Z(\mathrm{PI}(f)) > B(f)$? Чему равно максимальное значение $Z(\mathrm{PI}(f))/B(f)$?

251. [*M46*] Выполняется ли $Z(\mathrm{PI}(f)) = O(B(f))$ для всех монотонных булевых функций f?

252. [*M30*] Когда булева функция не является монотонной, ее простые импликанты включают отрицательные литералы; например, простыми импликантами функции $(x_1?\ x_2\colon x_3)$ являются $x_1 \wedge x_2$, $\bar{x}_1 \wedge x_3$ и $x_2 \wedge x_3$. В таких случаях можно легко представить их с помощью НДР, если рассматривать их как слова на основе $2n$-буквенного алфавита $\{e_1, e'_1, \ldots, e_n, e'_n\}$. "Подкуб", такой как $01{*}0{*}$, при этом представляет собой $e'_1 \sqcup e_2 \sqcup e'_4$ в алгебре семейств (см. 7.1.1–(29)); а $\mathrm{PI}(x_1?\ x_2\colon x_3) = (e_1 \sqcup e_2) \cup (e'_1 \sqcup e_3) \cup (e_2 \sqcup e_3)$.

В упр. 7.1.1–116 показано, что симметричные функции от n переменных могут иметь $\Omega(3^n/n)$ простых импликант. Насколько большим может быть значение $Z(\mathrm{PI}(f))$, когда функция f симметрична?

▶ **253.** [*M26*] Продолжая выполнять упражнение 252, докажите, что если $f = (\bar{x}_1 \wedge f_0) \vee (x_1 \wedge f_1)$, то мы имеем $\mathrm{PI}(f) = A \cup (e_1' \sqcup B) \cup (e_1 \sqcup C)$, где $A = \mathrm{PI}(f_0 \wedge f_1)$, $B = \mathrm{PI}(f_0) \setminus A$ и $C = \mathrm{PI}(f_1) \setminus A$. (Уравнение (137) представляет собой частный случай, когда f монотонна.)

▶ **254.** [*M23*] Пусть функции f и g из (52) монотонны, а также $f \subseteq g$. Докажите, что

$$\mathrm{PI}(g) \setminus \mathrm{PI}(f) \;=\; \bigl(\mathrm{PI}(g_l) \setminus \mathrm{PI}(f_l)\bigr) \cup \bigl(\mathrm{PI}(g_h) \setminus \mathrm{PI}(f_h \cup g_l)\bigr).$$

▶ **255.** [*25*] *Мультисемейство* множеств, в котором члены f могут встречаться более одного раза, может быть представлено как последовательность НДР $(f_0, f_1, f_2, \ldots)$, в которой f_k является семейством множеств, которые встречаются в f $(\ldots a_2 a_1 a_0)_2$ раз, где $a_k = 1$. Например, если α появляется в мультисемействе ровно $9 = (1001)_2$ раз, α должно быть в f_3 и f_0.

а) Поясните, как вставлять элементы в такое представление мультисемейства и удалять их из него.

б) Реализуйте объединение мультимножеств $h = f \uplus g$ для мультисемейств.

256. [*M32*] Любое неотрицательное целое число x можно представить в виде семейства подмножеств двоичных степеней $U = \{2^{2^k} \mid k \geq 0\} = \{2^1, 2^2, 2^4, 2^8, \ldots\}$ следующим образом: если $x = 2^{e_1} + \cdots + 2^{e_t}$, где $e_1 > \cdots > e_t \geq 0$ и $t \geq 0$, соответствующее семейство имеет t множеств $E_j \subseteq U$, где $2^{e_j} = \prod\{u \mid u \in E_j\}$. И обратно, каждое конечное семейство конечных подмножеств U соответствует неотрицательному целому числу x. Например, число $41 = 2^5 + 2^3 + 1$ соответствует семейству $\{\{2^1, 2^4\}, \{2^1, 2^2\}, \emptyset\}$.

а) Найдите простую связь между бинарным представлением x и таблицей истинности булевой функции, которая соответствует семейству для x.

б) Пусть $Z(x)$ является размером НДР для семейства, которое представляет x, когда элементы U проверяются в обратном порядке $\ldots, 2^4, 2^2, 2^1$ (наивысшие степени ближе к корню); например, $Z(41) = 5$. Покажите, что $Z(x) = O(\log x / \log\log x)$.

в) Целое число x называется "разреженным", если $Z(x)$ существенно меньше верхней границы из п. (б). Докажите, что сумма разреженных целых чисел является разреженной в том смысле, что $Z(x + y) = O(Z(x)Z(y))$.

г) Всегда ли разреженна разность с насыщением разреженных целых чисел $x \dot{-} y$?

д) Всегда ли разреженно произведение разреженных целых чисел?

257. [*40*] (Ш. Минато (S. Minato).) Исследуйте применение НДР для представления полиномов с неотрицательными целыми коэффициентами. *Указание*: любые такие полиномы от x, y и z могут рассматриваться как семейства подмножеств $\{2, 2^2, 2^4, \ldots, x, x^2, x^4, \ldots, y, y^2, y^4, \ldots, z, z^2, z^4, \ldots\}$; например, $x^3 + 3xy + 2z$ естественным образом соответствует семейству $\{\{x, x^2\}, \{x, y\}, \{2, x, y\}, \{2, z\}\}$.

▶ **258.** [*25*] Каков минимальный размер БДР, имеющей ровно n решений, для заданного положительного целого числа n? Ответьте на этот вопрос и для минимального размера НДР.

▶ **259.** [*25*] Последовательность скобок можно закодировать как бинарную строку, в которой 0 представляет '(', а 1 представляет ')'. Например, `()))(()` кодируется как 011001.

Каждый лес из n узлов соответствует последовательности из $2n$ корректно *вложенных* скобок в том смысле, что левые и правые скобки соответствуют одна другой обычным образом. (См., например, 2.3.3–(1) или 7.2.1.6–(1).) Пусть

$$N_n(x_1, \ldots, x_{2n}) = [x_1 \ldots x_{2n} \text{ представляет корректно вложенные скобки}].$$

Например, $N_3(0, 1, 1, 0, 0, 1) = 0$ и $N_3(0, 0, 1, 0, 1, 1) = 1$; в общем случае N_n имеет $C_n \approx 4^n/(\sqrt{\pi}\, n^{3/2})$ решений, где C_n — число Каталана. Чему равны $B(N_n)$ и $Z(N_n)$?

▶ **260.** [*M27*] В разделе 7.2.1.5 мы увидим, что каждое разбиение $\{1,\ldots,n\}$ на непересекающиеся подмножества соответствует "ограниченно растущей строке" $a_1\ldots a_n$, которая является последовательностью неотрицательных целых чисел с тем свойством, что

$$a_1 = 0 \quad \text{и} \quad a_{j+1} \le 1 + \max(a_1,\ldots,a_j) \text{ для } 1 \le j < n.$$

Элементы j и k принадлежат одному и тому же подмножеству разбиения тогда и только тогда, когда $a_j = a_k$.

а) Пусть $x_{j,k} = [a_j = k]$ для $0 \le k < j \le n$ и пусть R_n представляет собой функцию от этих $\binom{n+1}{2}$ переменных, которая истинна тогда и только тогда, когда $a_1\ldots a_n$ представляет собой ограниченно растущую строку. (Изучая эту булеву функцию, можно изучить семейство всех разбиений множества, а накладывая дополнительные ограничения на R_n, можно изучить разбиения множеств со специальными свойствами. Имеется $\varpi_{100} \approx 5 \times 10^{115}$ разбиений множеств при $n = 100$.) Вычислите $B(R_{100})$ и $Z(R_{100})$. Приблизительно насколько велики $B(R_n)$ и $Z(R_n)$ при $n \to \infty$?

б) Покажите, что в случае надлежащего упорядочения переменных $x_{j,k}$ база БДР для $\{R_1,\ldots,R_n\}$ имеет то же количество узлов, что и БДР для одной R_n.

в) Можно также использовать меньшее количество переменных, примерно $n \lg n$ вместо $\binom{n+1}{2}$, если представить каждое a_k как бинарное целое число с $\lceil \lg k \rceil$ бит. Насколько велики базы БДР и НДР в *этом* представлении разбиений множеств?

261. [*HM21*] *"Детерминированный конечный автомат с наименьшим количеством состояний, принимающий любой заданный регулярный язык, является единственным."* Какая имеется связь между этой знаменитой теоремой теории автоматов и теорией бинарных диаграмм решений?

262. [*M26*] Определение оптимальных булевых цепочек в разделе 7.1.2 существенно ускоряется, если ограничиться рассмотрением булевых функций, являющихся *нормальными* в том смысле, что $f(0,\ldots,0) = 0$. (См. 7.1.2–(10).) Аналогично можно ограничить БДР так, что каждый из их узлов будет обозначать нормальную функцию.

а) Поясните, как это сделать путем введения "дополняющих связей", которые указывают на дополнение подфункции вместо нее самой.

б) Покажите, что каждая булева функция имеет единственную нормализованную БДР.

в) Изобразите нормализованные БДР для 16 функций из упр. 1.

г) Пусть $B^0(f)$ — размер нормализованной БДР для f. Найдите средний и наихудший случаи $B^0(f)$ и сравните $B^0(f)$ с $B(f)$. (См. (80) и теорему U.)

д) База БДР для умножения 3×3 из (58) имеет $B(F_1,\ldots,F_6) = 52$ узла. Чему равно $B^0(F_1,\ldots,F_6)$?

е) Как следует изменить (54) и (55) при реализации И с дополняющими связями?

263. [*HM25*] *Линейный блочный код* представляет собой множество бинарных векторов-столбцов $x = (x_1,\ldots,x_n)^T$, таких, что $Hx = 0$, где H — заданная "матрица проверки четности" (parity check matrix) размером $m \times n$.

а) Линейный блочный код с $n = 2^m - 1$, столбцы которого представляют собой ненулевые бинарные m-кортежи от $(0,\ldots,0,1)^T$ до $(1,\ldots,1,1)^T$, называется *кодом Хэмминга*. Докажите, что код Хэмминга исправляет одну ошибку в смысле упр. 7–23.

б) Пусть $f(x) = [Hx = 0]$, где H является матрицей размером $m \times n$, в которой нет полностью нулевых столбцов. Покажите, что профиль БДР для f имеет простую связь с рангами подматриц $H \bmod 2$, и вычислите $B(f)$ для кода Хэмминга.

в) В общем случае пусть $f(x) = [x$ является кодовым словом$]$ определяет *произвольный* блочный код. Предположим, что некоторые кодовые слова передаются через, возможно, зашумленный канал, и мы получили биты $y = y_1\ldots y_n$, причем канал передает сигнал так, что $y_k = x_k$ с вероятностью p_k для каждого k независимо. Поясните, как

определить наиболее вероятное кодовое слово x для заданных y, p_1, ..., p_n и БДР для f.

264. [*M46*] "Широкое обобщение" алгоритмов B и C в тексте раздела, основанное на (22), включает много важных приложений; но, похоже, оно не включает такие величины, как

$$\max_{f(x)=1}\Bigl(\sum_{k=1}^{n} w_k x_k + \sum_{k=1}^{n-1} w'_k x_k x_{k+1}\Bigr) \qquad \text{или} \qquad \max_{f(x)=1}\sum_{j=0}^{n-1}\Bigl(w_j \sum_{k=1}^{n-j} x_k \ldots x_{k+j}\Bigr),$$

которые также могут быть эффективно вычислены из БДР или НДР для f.

Разработайте еще более широкое обобщение.

▶ **265.** [*21*] Разработайте алгоритм, который находит m-е наименьшее в лексикографическом порядке $x_1 \ldots x_n$ решение уравнения $f(x) = 1$ для заданного m и БДР для булевой функции f от n переменных. Ваш алгоритм должен выполняться за $O(nB(f) + n^2)$ шагов.

▶ **266.** [*20*] Каждый лес F, узлы которого пронумерованы как $\{1, \ldots, n\}$ в прямом порядке обхода, определяет два семейства множеств:

$$a(F) = \{\mathrm{anc}(1), \ldots, \mathrm{anc}(n)\} \quad \text{и} \quad d(F) = \{\mathrm{dec}(1), \ldots, \mathrm{dec(n)}\},$$

где $\mathrm{anc}(k)$ и $\mathrm{dec(k)}$ представляют собой включительно предков и потомков узла k. Например, если F имеет вид

,

то $a(F) = \bigl\{\{1\}, \{1,2\}, \{3\}, \{3,4\}, \{3,5\}\bigr\}$ и $d(F) = \bigl\{\{1,2\}, \{2\}, \{3,4,5\}, \{4\}, \{5\}\bigr\}$. И обратно, F можно восстановить либо из $a(F)$, либо из $d(F)$.

Докажите, что НДР для семейства $a(F)$ имеет ровно $n + 2$ узлов.

267. [*HM32*] Продолжая выполнять упр. 266, найдите минимальный, максимальный и средний размеры НДР для семейства $d(F)$ при F, пробегающем все леса с n узлами.

Мы не смеем более удлинять эту книгу, дабы соблюсти пропорции и не отпугнуть читателя ее размерами.

— АЛЬФРИК (?LFR|C), *Проповеди* (*Catholic Homilies II*) (ок. 1000 г.)

7.2. ГЕНЕРАЦИЯ ВСЕХ ВОЗМОЖНЫХ ОБЪЕКТОВ

— Все в наличии или учтены, сэр!
(— All present or accounted for, sir.)
— Традиционный рапорт в американской армии

— Все в наличии и на месте, сэр!
(— All present and correct, sir.)
— Традиционный рапорт в английской армии

7.2.1. Генерация основных комбинаторных объектов

В этом разделе рассматриваются методы обхода всех возможных объектов некоторой комбинаторной генеральной совокупности, поскольку часто приходится сталкиваться с задачами, в которых необходим или желателен исчерпывающий перебор. Например, нам могут потребоваться все перестановки некоторого множества.

Одни авторы называют эту задачу *перечислением* (enumerating) всех возможностей; однако это не совсем верное слово, так как "перечисление" чаще всего означает, что мы хотим просто *подсчитать* общее количество вариантов, а не просмотреть их все. Если некто просит вас перечислить перестановки множества $\{1, 2, 3\}$, вы вполне обоснованно утверждаете, что ответом является $3! = 6$; вам не приходится давать более полный ответ $\{123, 132, 213, 231, 312, 321\}$.

Другие авторы говорят о *составлении списка* (listing) всех возможностей, но это также не слишком удачный термин. Ни один человек в здравом уме не станет составлять список из $10! = 3\,628\,800$ перестановок множества $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ни путем печати на тысячах листов бумаги, ни даже записывая их в текстовый файл. Все, что нам в действительности нужно, — это чтобы каждая из них некоторое время пребывала в определенной структуре данных с тем, чтобы программа могла по очереди поработать со всеми возможными перестановками.

Поэтому мы говорим о *генерации* (generating) всех требующихся нам комбинаторных объектов и о поочередном *посещении* (visiting) каждого объекта. Так же, как при изучении алгоритмов для обхода деревьев в разделе 2.3.1, где наша цель состояла в посещении каждого узла дерева, нам требуются алгоритмы для систематического обхода комбинаторного пространства возможностей.

Он всех их зачислил —
Он всех их зачислил;
И никто не упущен —
Никто не пропущен.
— УИЛЬЯМ С. ГИЛЬБЕРТ (WILLIAM S. GILBERT),
Микадо (The Mikado) (1885)

7.2.1.1. Генерация всех n-кортежей. Начнем с малого, рассмотрев обход всех 2^n строк, состоящих из n двоичных цифр. Иначе говоря, попробуем обойти все n-кортежи $(a_1, \ldots, a_n)$, где каждый элемент a_j является либо нулем, либо единицей.

Эта задача, по сути, эквивалентна просмотру всех подмножеств данного множества $\{x_1, \ldots, x_n\}$, поскольку можно сказать, что x_j находится в подмножестве тогда и только тогда, когда $a_j = 1$.

Конечно, такая задача имеет до абсурда простое решение. Все, что нужно,— начать с двоичного числа $(0\ldots00)_2 = 0$ и многократно добавлять к нему 1, пока не будет достигнуто число $(1\ldots11)_2 = 2^n - 1$. Однако вскоре, при более глубоком анализе, мы увидим, что эта очень простая задача имеет удивительно интересные особенности. Изучение n-кортежей окупится позже, при изучении особенностей генерации более сложных комбинаторных шаблонов.

Прежде всего, нетрудно заметить, что двоичная запись может быть расширена и на другие типы n-кортежей. Например, если нужно генерировать все варианты $(a_1, \ldots, a_n)$, где каждое a_j представляет собой одну из десятичных цифр $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, можно просто выполнить перечисление чисел от $(0\ldots00)_{10} = 0$ до $(9\ldots99)_{10} = 10^n - 1$ в десятичной системе счисления. А если нужно решить более общую задачу обхода всех случаев, в которых

$$0 \le a_j < m_j \qquad \text{для } 1 \le j \le n, \tag{1}$$

где верхние границы m_j разных компонентов вектора $(a_1, \ldots, a_n)$ могут отличаться, то задача, по сути, сводится к многократному прибавлению единицы к числу

$$\begin{bmatrix} a_1, & a_2, & \ldots, & a_n \\ m_1, & m_2, & \ldots, & m_n \end{bmatrix} \tag{2}$$

в смешанной позиционной системе счисления; см. уравнение 4.1–(9) и упр. 4.3.1–9.

Притормозим и опишем этот процесс более формально.

Алгоритм M (*Генерация в смешанной позиционной системе счисления*). Этот алгоритм посещает все n-кортежи, удовлетворяющие (1), путем многократного добавления 1 к числу в смешанной позиционной системе счисления (2) до тех пор, пока не будет достигнуто переполнение. Вспомогательные переменные a_0 и m_0 введены для удобства.

M1. [Инициализация.] Установить $a_j \leftarrow 0$ для $0 \le j \le n$ и установить $m_0 \leftarrow 2$.

M2. [Посещение.] Посетить n-кортеж $(a_1, \ldots, a_n)$. (Программа, которая должна посетить все n-кортежи, приступает к работе.)

M3. [Подготовка к прибавлению единицы.] Установить $j \leftarrow n$.

M4. [Перенос при необходимости.] Если $a_j = m_j - 1$, установить $a_j \leftarrow 0$, $j \leftarrow j - 1$, и повторить этот шаг.

M5. [Увеличение, если еще не выполнено.] Если $j = 0$, завершить работу алгоритма. В противном случае установить $a_j \leftarrow a_j + 1$ и вернуться к шагу M2. ▮

Алгоритм M прост и прямолинеен, но не следует забывать, что вложенные циклы при достаточно малых константах n еще проще. Например, для $n = 4$ можно было бы записать следующие инструкции.

Для $a_1 = 0, 1, \ldots, m_1 - 1$ (в указанном порядке) выполнить следующее:
 Для $a_2 = 0, 1, \ldots, m_2 - 1$ (в указанном порядке) выполнить следующее:
  Для $a_3 = 0, 1, \ldots, m_3 - 1$ (в указанном порядке) выполнить следующее: (3)
   Для $a_4 = 0, 1, \ldots, m_4 - 1$ (в указанном порядке) выполнить следующее:
    Посетить (a_1, a_2, a_3, a_4).

0000
*000
1000
00**
0100
*100
0110
**00
0010
*010
1010
0***
0001
*001
1001
01**
0101
*101
1101
0***
0011
*011
1011
11**
0111
*111
1111
1***
0111
*111
1111
10**
0110
*110
1110
1***
0110
*110
1110
10**
0010
*010
1010
1***
0001
*001
1001
11**
0101
*101
1101
1***
0011
*011
1011
01**
0111
*111
1111
0***
0100
*100
1100
00**
0000
*000
1000
0***
0010
*010
1010
0*10
0110
*110
1110
0*01
0101
*101
1101
0*11
0111
*111
1111
1*10
0110
*110
1110
1*01
0101
*101
1101
1*11
0111
*111
1111
11*1
0101
*101
1101
11*1
0001
*001
1001
10*1
0011
*011
1011
10*1
0010
*010
1010
10*1
0000
*000
1000
10*1
0100
*100
1100
10*1
0110
*110
1110
10*1
0111
*111
1111
10*1
0101
*101
1101
10

Другой способ определения последовательности $\Gamma_n = g(0),\ g(1),\ \ldots,\ g(2^n-1)$ заключается в указании явной формулы для отдельных элементов $g(k)$. Действительно, поскольку Γ_{n+1} начинается с $0\Gamma_n$, бесконечная последовательность

$$\begin{aligned}\Gamma_\infty &= g(0), g(1), g(2), g(3), g(4), \ldots \\ &= (0)_2, (1)_2, (11)_2, (10)_2, (110)_2, \ldots\end{aligned} \tag{6}$$

является перестановкой всех неотрицательных целых чисел, если рассматривать каждую строку из нулей и единиц как двоичное целое число с необязательными ведущими нулями. Тогда Γ_n состоит из первых 2^n элементов (6), преобразованных в n-битовые строки за счет вставки при необходимости ведущих нулей.

Когда $k = 2^n + r$, где $0 \le r < 2^n$, соотношение (5) говорит нам, что $g(k)$ равно $2^n + g(2^n - 1 - r)$. Следовательно, по индукции по n можно доказать, что целое число k с двоичным представлением $(\ldots b_2b_1b_0)_2$ имеет эквивалент $g(k)$ в бинарном коде Грея с представлением $(\ldots a_2a_1a_0)_2$, где

$$a_j \;=\; b_j \oplus b_{j+1} \qquad \text{для } j \ge 0. \tag{7}$$

(См. упр. 6.) Например, $g\big((111001000011)_2\big) = (100101100010)_2$. И наоборот, если дано $g(k) = (\ldots a_2a_1a_0)_2$, можно найти $k = (\ldots b_2b_1b_0)_2$, обращая систему уравнений (7) и получая

$$b_j \;=\; a_j \oplus a_{j+1} \oplus a_{j+2} \oplus \cdots \qquad \text{для } j \ge 0; \tag{8}$$

эта бесконечная сумма на самом деле конечна, поскольку $a_{j+t} = 0$ для всех больших t.

Одно из многочисленных приятных следствий уравнения (7) заключается в том, что $g(k)$ можно очень легко вычислить с помощью битовой арифметики:

$$g(k) \;=\; k \oplus \lfloor k/2 \rfloor. \tag{9}$$

Аналогично обратная функция в (8) удовлетворяет равенству

$$g^{[-1]}(l) = l \oplus \lfloor l/2 \rfloor \oplus \lfloor l/4 \rfloor \oplus \cdots; \tag{10}$$

однако для этой функции требуется больше вычислений (см. упр. 7.1.3–117). Из (7) можно также вывести, что если k и k' — произвольные неотрицательные целые числа, то

$$g(k \oplus k') \;=\; g(k) \oplus g(k'). \tag{11}$$

Еще одним следствием является то, что $(n+1)$-битовый бинарный код Грея может быть записан как

$$\Gamma_{n+1} \;=\; 0\Gamma_n,\ (0\Gamma_n) \oplus 110\ldots0;$$

этот шаблон наблюдается, например, в (4). Сравнивая этот шаблон с (5), можно заметить, что обращение порядка бинарного кода Грея эквивалентно дополнению первого бита:

$$\Gamma_n^R \;=\; \Gamma_n \oplus 1\overbrace{0\ldots0}^{n-1}, \text{ также записывается как } \Gamma_n \oplus 10^{n-1}. \tag{12}$$

В приведенных ниже упражнениях показано, что функция $g(k)$, определенная в (7), и обратная ей функция $g^{[-1]}$, определенная в (8), имеют много других интересных свойств и приложений. Порой эти функции рассматриваются как преобразующие бинарные строки в бинарные строки; их можно также рассматривать

как функции, преобразующие целые числа в целые числа с их промежуточным представлением в виде бинарных строк с игнорируемыми ведущими нулями.

Бинарный код Грея назван по имени Фрэнка Грея (Frank Gray), физика, прославившегося разработкой метода, долгое время использовавшегося для обеспечения совместимости цветного телевещания [*Bell System Tech. J.* **13** (1934), 464–515]. Он изобрел код Γ_n для применения в кодово-импульсной модуляции, методе аналоговой передачи цифровых сигналов [см. *Bell System Tech. J.* **30** (1951), 38–40; *U.S. Patent 2632058* (17 March 1953); W. R. Bennett, *Introduction to Signal Transmission* (1971), 238–240]. Однако сама идея "бинарного кода Грея" была известна задолго до начала его работы над этой проблемой, например в *U.S. Patent 2307868* Джорджа Стибитца (George Stibitz) (12 January 1943). Более важно другое его появление — код Γ_5 использовался в телеграфной машине, продемонстрированной в 1878 году Эмилем Бодо (Émile Baudot), (в его честь была названа единица измерения "бод"). Примерно в то же время аналогичный, но менее систематический код для телеграфии был независимо разработан Отто Шаффлером (Otto Schäffler) [см. *Journal Télégraphique* **4** (1878), 252–253; *Annales Télégraphiques* **6** (1879), 361, 382–383].*

Бинарный код Грея неявно присутствует в классической головоломке, которая веками забавляла людей и которая известна под общим названием "Китайские кольца", хотя ее часто называют попросту утомительными железками (tiring irons). На рис. 31 показан пример этой головоломки с семью кольцами. Задача заключается в том, чтобы снять кольца со стержня, но кольца переплетены так, что возможны только два основных действия (хотя это может и не быть очевидно при рассматривании рисунка).

а) Крайнее справа кольцо можно снять или одеть в любой момент.
б) Любое другое кольцо можно снять или одеть тогда и только тогда, когда кольцо справа от него находится на стержне, а все кольца правее него сняты.

Рис. 31.
Головоломка "Китайские кольца".

Текущее состояние головоломки можно представить в двоичной системе счисления, где 1 означает, что кольцо находится на стержне, а 0 — что кольцо снято. Таким образом, на рис. 31 показано состояние колец 1011000. (Второе кольцо слева обозначено нулем, поскольку оно полностью находится над стержнем.)

Французский судья Луи Грос (Louis Gros) продемонстрировал явную связь головоломки с двоичными числами в анонимно опубликованной брошюре *Théorie du Baguenodier* (*Теория безделья* [отлично!]) (Lyon: Aimé Vingtrinier, 1872). Если кольца находятся в состоянии $a_{n-1}\ldots a_0$, и если определить двоичное число $k = (b_{n-1}\ldots b_0)_2$ согласно (8), то ровно k шагов будет необходимо и достаточно для

* Некоторые авторы утверждают, что код Грея был изобретен Элишей Греем (Elisha Gray), который создал печатающую телеграфную машину в то же время, что и Бодо и Шаффлер. Эти заявления не соответствуют действительности, хотя с Элишей обошлись несправедливо в отношении приоритета в изобретении телефона [см. L. W. Taylor, *Amer. Physics Teacher* **5** (1937), 243–251].

Несомненно, не должно быть дома, в котором бы не было этой очаровательной, исторической и поучительной головоломки.

— ГЕНРИ Э. ДЬЮДЕНИ (HENRY E. DUDENEY) (1901)

того, чтобы решить головоломку. Таким образом, истинным первооткрывателем бинарного кода Грея является именно Грос.

Если кольца находятся в произвольном состоянии, отличном от $00\ldots0$ или $10\ldots0$, то возможны ровно два действия: первое — типа (а), а второе — типа (б). Только одно из этих действий дает продвижение к заданной цели, а другое представляет собой шаг назад. Действие типа (а) изменяет k на $k \oplus 1$; таким образом, его следует выполнять при нечетном k, поскольку тем самым будет выполнено уменьшение k. Действие типа (б) в позиции, заканчивающейся на $(10^{j-1})_2$ для $1 \le j < n$, изменяет k на $k \oplus (1^{j+1})_2 = k \oplus (2^{j+1} - 1)$. [В этой формуле '$1^{j+1}$' означает $j+1$ повторений '1', но '2^{j+1}' означает степень двойки.] Если k четно, нужно приравнять $k \oplus (2^{j+1} - 1)$ к $k - 1$, что означает, что k должно быть кратно 2^j, но не 2^{j+1}; другими словами,

$$j = \rho(k), \tag{13}$$

где ρ — "линеечная функция" из соотношений 7.1.3–(44). Следовательно, для корректного решения головоломки кольца должны следовать красивому шаблону. Если пронумеровать кольца 0, 1, ..., $n-1$ (начиная от свободного конца), то последовательность снятий и одеваний колец представляет собой последовательность чисел, которая заканчивается ..., $\rho(4)$, $\rho(3)$, $\rho(2)$, $\rho(1)$.

Идя в обратном направлении, нужно, начав с $00\ldots0$, последовательно одевать или снимать кольца до тех пор, пока не будет достигнуто конечное состояние $10\ldots0$ (которое, как заметил Джон Уоллис (John Wallis) в 1693 году, получить гораздо сложнее, чем на первый взгляд более сложное состояние $11\ldots1$), что приводит нас к алгоритму счета в бинарном коде Грея.

Алгоритм G (*Генерация бинарного кода Грея*). Этот алгоритм посещает все бинарные n-кортежи $(a_{n-1}, \ldots, a_1, a_0)$, начиная с $(0, \ldots, 0, 0)$ и изменяя только по одному биту за раз, а также поддерживая бит a_∞, такой, что

$$a_\infty = a_{n-1} \oplus \cdots \oplus a_1 \oplus a_0. \tag{14}$$

Он последовательно дополняет биты $\rho(1)$, $\rho(2)$, $\rho(3)$, ..., $\rho(2^n - 1)$, после чего алгоритм останавливается.

G1. [Инициализация.] Установить $a_j \leftarrow 0$ для $0 \le j < n$; установить также $a_\infty \leftarrow 0$.

G2. [Посещение.] Посетить n-кортеж $(a_{n-1}, \ldots, a_1, a_0)$.

G3. [Изменение четности.] Установить $a_\infty \leftarrow 1 - a_\infty$.

G4. [Выбор j.] Если $a_\infty = 1$, установить $j \leftarrow 0$. В противном случае пусть $j \ge 1$ — минимальное, при котором $a_{j-1} = 1$. (После k-го выполнения этого шага $j = \rho(k)$.)

G5. [Дополнение координаты j.] Завершить работу алгоритма, если $j = n$; в противном случае установить $a_j \leftarrow 1 - a_j$ и вернуться к шагу G2. ▌

Бит четности a_∞ пригодится при вычислении сумм наподобие

$$X_{000} - X_{001} - X_{010} + X_{011} - X_{100} + X_{101} + X_{110} - X_{111}$$

или

$$X_\emptyset - X_a - X_b + X_{ab} - X_c + X_{ac} + X_{bc} - X_{abc},$$

где знак зависит от четности бинарной строки или количества элементов подмножества. Такие суммы часто возникают в формулах "включения-исключения", таких, как 1.3.3–(29). Бит четности необходим также для эффективности: без него было бы не так просто выбрать один из двух способов определения значения j, которое соответствует выполнению действий типа (а) или типа (б) в "Китайских кольцах". Однако наиболее важной особенностью алгоритма G является то, что шаг G5 выполняет изменение только одной координаты. Следовательно, обычно требуется только простое изменение суммируемых членов X или иных структур, с которыми приходится иметь дело при посещении каждого n-кортежа.

Конечно, невозможно избежать неоднозначности в младшем разряде, за исключением способа, к которому, по слухам, прибегла одна ирландская железнодорожная компания, убрав во всех поездах последний вагон как наиболее уязвимый при столкновениях поездов.

— Д. Р. СТИБИТЦ (G. R. STIBITZ) и Д. А. ЛАРРИВИ (J. A. LARRIVEE), *Математика и компьютеры* (*Mathematics and Computers*) (1957)

Еще одно ключевое свойство бинарного кода Грея было открыто Д. Л. Уолшем (J. L. Walsh) в связи с важной последовательностью функций, которые ныне известны как *функции Уолша* [см. *Amer. J. Math.* **45** (1923), 5–24]. Пусть $w_0(x) = 1$ для всех действительных чисел x и

$$w_k(x) = (-1)^{\lfloor 2x \rfloor \lceil k/2 \rceil} w_{\lfloor k/2 \rfloor}(2x) \qquad \text{для } k > 0. \tag{15}$$

Например, $w_1(x) = (-1)^{\lfloor 2x \rfloor}$ изменяет знак всякий раз, когда x является целым числом или целым плюс $\frac{1}{2}$. Отсюда следует, что $w_k(x) = w_k(x+1)$ для всех k и что $w_k(x) = \pm 1$ для всех x. Более важно то, что $w_k(0) = 1$ и $w_k(x)$ *ровно* k *раз меняет знак в диапазоне* $(0\,..\,1)$, так что она стремится к $(-1)^k$ при x, стремящемся к 1 слева. Следовательно, $w_k(x)$ ведет себя скорее как тригонометрическая функция наподобие $\cos k\pi x$ или $\sin k\pi x$, и другие функции можно представить в виде линейной комбинации функций Уолша почти так же, как они традиционно представляются с помощью ряда Фурье. Этот факт, вместе с простой дискретной природой $w_k(x)$, делает функции Уолша исключительно полезными в компьютерных вычислениях, связанных с передачей информации, обработкой изображений и многими другими приложениями.

На рис. 32 показаны первые восемь функций Уолша вместе с их тригонометрическими собратьями. Инженеры обычно называют $w_k(x)$ функцией Уолша *порядка* k, по аналогии с тригонометрическими функциями $\cos k\pi x$ и $\sin k\pi x$ с *частотой* $k/2$. [См., например, книгу H. F. Harmuth, *Sequency Theory: Foundations and Applications* (New York: Academic Press, 1977).]

Хотя, на первый взгляд, выражение в (15) может показаться пугающим, на самом деле оно представляет простой способ увидеть по индукции, почему $w_k(x)$ имеет ровно k изменений знака. Если k четно, скажем, $k = 2l$, то мы имеем $w_{2l}(x) = w_l(2x)$ для $0 \le x < \frac{1}{2}$; в результате функция $w_l(x)$ просто сжимается вдвое, так что $w_{2l}(x)$ содержит l изменений знака. Тогда $w_{2l}(x) = (-1)^l w_l(2x) = (-1)^l w_l(2x-1)$ в диапазоне $\frac{1}{2} \le x < 1$; это позволяет присоединить вторую копию $w_l(2x)$, обращая

Рис. 32. Функции Уолша $w_k(x)$ для $0 \le k < 8$ и аналогичные тригонометрические функции $\sqrt{2}\cos k\pi x$ (показаны серым цветом).

знак из-за необходимости избежать изменения знака при $x = \frac{1}{2}$. Функция $w_{2l+1}(x)$ аналогична, но она *заставляет* изменить знак при $x = \frac{1}{2}$.

Какое отношение ко всему этому имеет бинарный код Грея? Уолш обнаружил, что его функции могут быть компактно выражены через более простые *функции Радемахера* [см. Hans Rademacher, *Math. Annalen* **87** (1922), 112–138],

$$r_k(x) \;=\; (-1)^{\lfloor 2^k x \rfloor}, \tag{16}$$

которые принимают значение $(-1)^{c_{-k}}$, где $(\ldots c_2c_1c_0.c_{-1}c_{-2}\ldots)_2$ является бинарным представлением x. Действительно, $w_1(x) = r_1(x)$, $w_2(x) = r_1(x)r_2(x)$, $w_3(x) = r_2(x)$ и в общем случае

$$w_k(x) = \prod_{j\ge 0} r_{j+1}(x)^{b_j \oplus b_{j+1}} \qquad \text{при } k = (\ldots b_2b_1b_0)_2. \tag{17}$$

(См. упр. 33.) Таким образом, согласно (7) степень $r_{j+1}(x)$ в $w_k(x)$ представляет собой j-й бит бинарного числа Грея $g(k)$, и мы имеем

$$w_k(x) \;=\; r_{\rho(k)+1}(x)\, w_{k-1}(x) \qquad \text{для } k > 0. \tag{18}$$

Из уравнения (17) следует удобная формула

$$w_k(x)\, w_{k'}(x) \;=\; w_{k\oplus k'}(x), \tag{19}$$

которая гораздо проще, чем соответствующие формулы для произведения синусов и косинусов. Это тождество легко проверить, поскольку $r_j(x)^2 = 1$ для всех j

и x, следовательно, $r_j(x)^{a\oplus b} = r_j(x)^{a+b}$. Отсюда, в частности, вытекает, что функция $w_k(x)$ *ортогональна* $w_{k'}(x)$, если $k \neq k'$, в том смысле, что среднее значение $w_k(x)w_{k'}(x)$ равно нулю. Мы также можем воспользоваться (17) для определения $w_k(x)$ для дробных значений k наподобие 1/2 или 13/8.

Преобразование Уолша 2^n чисел $(X_0, \ldots, X_{2^n-1})$ представляет собой вектор $(x_0, \ldots, x_{2^n-1})$, определяемый уравнением $(x_0, \ldots, x_{2^n-1})^T = W_n(X_0, \ldots, X_{2^n-1})^T$, где W_n является матрицей размером $2^n \times 2^n$, с элементами $w_j(k/2^n)$ на пересечении строки j и столбца k для $0 \le j, k < 2^n$. Например, рис. 32 говорит нам, что преобразование Уолша при $n = 3$ имеет вид

$$\begin{pmatrix} x_{000}\\ x_{001}\\ x_{010}\\ x_{011}\\ x_{100}\\ x_{101}\\ x_{110}\\ x_{111}\end{pmatrix} = \begin{pmatrix} 1&1&1&1&1&1&1&1\\ 1&1&1&1&\bar1&\bar1&\bar1&\bar1\\ 1&1&\bar1&\bar1&\bar1&\bar1&1&1\\ 1&1&\bar1&\bar1&1&1&\bar1&\bar1\\ 1&\bar1&\bar1&1&1&\bar1&\bar1&1\\ 1&\bar1&\bar1&1&\bar1&1&1&\bar1\\ 1&\bar1&1&\bar1&\bar1&1&\bar1&1\\ 1&\bar1&1&\bar1&1&\bar1&1&\bar1\end{pmatrix}\begin{pmatrix} X_{000}\\ X_{001}\\ X_{010}\\ X_{011}\\ X_{100}\\ X_{101}\\ X_{110}\\ X_{111}\end{pmatrix}. \tag{20}$$

(Здесь $\bar1$ обозначает -1, а подстрочные индексы рассматриваются как обычные бинарные строки 000–111, а не целые числа 0–7.) *Преобразование Адамара* определяется аналогично, но вместо W_n используется матрица H_n, где H_n на пересечении строки j и столбца k содержит элементы $(-1)^{j\cdot k}$; здесь '$j \cdot k$' обозначает скалярное произведение $a_{n-1}b_{n-1} + \cdots + a_0b_0$ бинарных представлений $j = (a_{n-1}\ldots a_0)_2$ и $k = (b_{n-1}\ldots b_0)_2$. Например, преобразование Адамара для $n = 3$ имеет вид

$$\begin{pmatrix} x'_{000}\\ x'_{001}\\ x'_{010}\\ x'_{011}\\ x'_{100}\\ x'_{101}\\ x'_{110}\\ x'_{111}\end{pmatrix} = \begin{pmatrix} 1&1&1&1&1&1&1&1\\ 1&\bar1&1&\bar1&1&\bar1&1&\bar1\\ 1&1&\bar1&\bar1&1&1&\bar1&\bar1\\ 1&\bar1&\bar1&1&1&\bar1&\bar1&1\\ 1&1&1&1&\bar1&\bar1&\bar1&\bar1\\ 1&\bar1&1&\bar1&\bar1&1&\bar1&1\\ 1&1&\bar1&\bar1&\bar1&\bar1&1&1\\ 1&\bar1&\bar1&1&\bar1&1&1&\bar1\end{pmatrix}\begin{pmatrix} X_{000}\\ X_{001}\\ X_{010}\\ X_{011}\\ X_{100}\\ X_{101}\\ X_{110}\\ X_{111}\end{pmatrix}. \tag{21}$$

Это то же самое, что и дискретное преобразование Фурье на n-мерном кубе 4.6.4–(38), и его можно легко вычислить "на месте", если адаптировать метод Ятса, рассматривавшийся в разделе 4.6.4.

Дано	Первый шаг	Второй шаг	Третий шаг
X_{000}	$X_{000}+X_{001}$	$X_{000}+X_{001}+X_{010}+X_{011}$	$X_{000}+X_{001}+X_{010}+X_{011}+X_{100}+X_{101}+X_{110}+X_{111}$
X_{001}	$X_{000}-X_{001}$	$X_{000}-X_{001}+X_{010}-X_{011}$	$X_{000}-X_{001}+X_{010}-X_{011}+X_{100}-X_{101}+X_{110}-X_{111}$
X_{010}	$X_{010}+X_{011}$	$X_{000}+X_{001}-X_{010}-X_{011}$	$X_{000}+X_{001}-X_{010}-X_{011}+X_{100}+X_{101}-X_{110}-X_{111}$
X_{011}	$X_{010}-X_{011}$	$X_{000}-X_{001}-X_{010}+X_{011}$	$X_{000}-X_{001}-X_{010}+X_{011}+X_{100}-X_{101}-X_{110}+X_{111}$
X_{100}	$X_{100}+X_{101}$	$X_{100}+X_{101}+X_{110}+X_{111}$	$X_{000}+X_{001}+X_{010}+X_{011}-X_{100}-X_{101}-X_{110}-X_{111}$
X_{101}	$X_{100}-X_{101}$	$X_{100}-X_{101}+X_{110}-X_{111}$	$X_{000}-X_{001}+X_{010}-X_{011}-X_{100}+X_{101}-X_{110}+X_{111}$
X_{110}	$X_{110}+X_{111}$	$X_{100}+X_{101}-X_{110}-X_{111}$	$X_{000}+X_{001}-X_{010}-X_{011}-X_{100}-X_{101}+X_{110}+X_{111}$
X_{111}	$X_{110}-X_{111}$	$X_{100}-X_{101}-X_{110}+X_{111}$	$X_{000}-X_{001}-X_{010}+X_{011}-X_{100}+X_{101}+X_{110}-X_{111}$

Обратите внимание, что строки H_3 являются перестановками строк W_3. Это верно и в общем случае, а потому преобразование Уолша можно выполнить за счет

перестановки элементов преобразования Адамара. Этот прием подробно рассматривается в упр. 36.

Ускорение. Анализируя 2^n возможностей, обычно хочется максимально сократить время вычислений. Алгоритм G должен выполнять дополнение только одного бита a_j на одно посещение $(a_{n-1}, \ldots, a_0)$, но на шаге G4 имеется цикл для выбора соответствующего значения j. Другой подход был предложен Гидеоном Эрлихом (Gideon Ehrlich) [*JACM* **20** (1973), 500–513], который ввел понятие комбинаторной генерации *без циклов*. В алгоритме без циклов количество операций между последовательными посещениями заранее ограничено, поэтому долго ждать генерации очередного шаблона не приходится.

В разделе 7.1.3 мы познакомились с некоторыми хитрыми способами быстрого определения количества ведущих или замыкающих нулей в двоичном числе. Эти методы можно использовать на шаге G4, чтобы сделать алгоритм G алгоритмом без циклов, если n не слишком велико. Однако метод Эрлиха существенно иной и более универсальный, так что он расширяет наш арсенал технологий эффективных вычислений. Вот как этот подход можно применить для генерации бинарных n-кортежей [см. Bitner, Ehrlich, and Reingold, *CACM* **19** (1976), 517–521].

Алгоритм L (*Генерация бинарного кода Грея без циклов*). Этот алгоритм, как и алгоритм G, посещает все бинарные n-кортежи $(a_{n-1}, \ldots, a_0)$ в порядке бинарного кода Грея. Однако вместо бита четности в нем используется массив "фокусных указателей" $(f_n, \ldots, f_0)$, важность которых рассматривается ниже.

L1. [Инициализация.] Установить $a_j \leftarrow 0$ и $f_j \leftarrow j$ для $0 \le j < n$; также установить $f_n \leftarrow n$. (Алгоритм без циклов может иметь цикл на этапе инициализации, если это оправдано с точки зрения эффективности; в конце концов, каждая программа должна быть загружена и запущена.)

L2. [Посещение.] Посетить n-кортеж $(a_{n-1}, \ldots, a_1, a_0)$.

L3. [Выбор j.] Установить $j \leftarrow f_0$, $f_0 \leftarrow 0$. (Если это k-е выполнение данного шага, то j равно $\rho(k)$.) Завершить работу алгоритма, если $j = n$; в противном случае установить $f_j \leftarrow f_{j+1}$ и $f_{j+1} \leftarrow j + 1$.

L4. [Дополнение координаты j.] Установить $a_j \leftarrow 1 - a_j$ и вернуться к шагу L2. ▮

Например, для $n = 4$ вычисления происходят следующим образом (элемент a_j в приведенной таблице подчеркнут, если соответствующий бит b_j равен 1 в бинарной строке $b_3b_2b_1b_0$, такой, что $a_3a_2a_1a_0 = g(b_3b_2b_1b_0)$).

a_3	0	0	0	0	0	0	0	0	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$
a_2	0	0	0	0	$\underline{1}$	$\underline{1}$	$\underline{1}$	$\underline{1}$	1	1	1	1	$\underline{0}$	$\underline{0}$	$\underline{0}$	$\underline{0}$
a_1	0	0	$\underline{1}$	$\underline{1}$	1	1	$\underline{0}$	$\underline{0}$	0	0	$\underline{1}$	$\underline{1}$	1	1	$\underline{0}$	$\underline{0}$
a_0	0	$\underline{1}$	1	$\underline{0}$	0	$\underline{1}$	1	$\underline{0}$	0	$\underline{1}$	1	$\underline{0}$	0	$\underline{1}$	1	$\underline{0}$
f_3	3	3	3	3	3	3	3	3	4	4	4	4	3	3	3	3
f_2	2	2	2	2	3	3	2	2	2	2	2	2	4	4	2	2
f_1	1	1	2	1	1	1	3	1	1	1	2	1	1	1	4	1
f_0	0	1	0	2	0	1	0	3	0	1	0	2	0	1	0	4

Хотя двоичное число $k = (b_{n-1} \ldots b_0)_2$ никогда явно не появляется в алгоритме L, фокусные указатели f_j неявно представляют его таким образом, что можно много-

кратно получать $g(k) = (a_{n-1} \ldots a_0)_2$ путем дополнения бита $a_{\rho(k)}$. Будем говорить, что бит a_j *пассивен*, когда он подчеркнут, и *активен* в противном случае. Тогда фокусные указатели удовлетворяют следующим инвариантным соотношениям.

1) Если бит a_j пассивен, а бит a_{j-1} активен, то f_j — наименьший индекс $j' > j$, такой, что бит $a_{j'}$ активен. (Биты a_n и a_{-1} рассматриваются в данном правиле как активные, хотя в алгоритме они реально не присутствуют.)
2) В противном случае $f_j = j$.

Таким образом, крайний справа элемент a_j блока пассивных элементов $a_{i-1} \ldots a_{j+1}a_j$ с уменьшающимися индексами имеет фокусный указатель f_j, который указывает на элемент a_i, находящийся первым слева от этого блока. Все прочие элементы a_j имеют фокусные указатели f_j, указывающие на эти элементы.

В рассмотренных терминах первые операции '$j \leftarrow f_0$, $f_0 \leftarrow 0$' на шаге L3 эквивалентны высказыванию "установить j равным индексу крайнего справа активного элемента и активизировать все элементы справа от a_j". Заметим, что если $f_0 = 0$, то операция $f_0 \leftarrow 0$ оказывается излишней, но это не причиняет никакого вреда. Две другие операции шага L3, '$f_j \leftarrow f_{j+1}$, $f_{j+1} \leftarrow j + 1$', эквивалентны высказыванию "сделать a_j пассивным", поскольку мы знаем, что a_j и a_{j-1} в этот момент вычислений активны. (И вновь операция $f_{j+1} \leftarrow j + 1$ может быть избыточной, но безвредной.) Общий результат активизации и пассивизации, таким образом, эквивалентен подсчету в бинарной записи (как в алгоритме M, с пассивными единичными битами и активными нулевыми).

Алгоритм L поразительно быстр, так как он выполняет только пять операций присваивания и одну проверку завершения алгоритма между посещениями генерируемых n-кортежей. Но можно улучшить и его. Чтобы увидеть, как это сделать, давайте рассмотрим его применение в занимательной лингвистике. Рудольф Кастаун (Rudolph Castown) в своей работе *Word Ways* **1** (1968), 165–169, заметил, что все 16 способов смешения букв слова `sins` (грехи) с соответствующими буквами слова `fate` (судьба) дают слова, которые имеются в достаточно большом словаре английского языка: `sine`, `sits`, `site` и т. д.; и все слова, кроме трех (а именно `fane`, `fite` и `sats`), достаточно распространены и, бесспорно, являются неотъемлемой частью английского языка. Следовательно, естественно задать вопрос для слов из пяти букв: какие две строки из пяти букв порождают максимальное количество слов из Stanford GraphBase, где буквы в соответствующих позициях обмениваются всеми 32 возможными способами?

Для ответа на этот вопрос не нужно проверять все $\binom{26}{2}^5 = 3\,625\,908\,203\,125$ существенно разных пар строк; достаточно просмотреть все $\binom{5757}{2} = 16\,568\,646$ пар слов из GraphBase, при условии, что как минимум одна из пар порождает не меньше 17 слов, поскольку каждое множество из 17 или более пятибуквенных слов, которые можно получить из двух пятибуквенных строк, должны содержать два слова, являющиеся "антиподами" (не имеющими одинаковых соответствующих букв). Для каждой пары антиподов следует максимально быстро определить, дают ли 32 возможные перестановки слова́ английского языка.

Каждое пятибуквенное слово можно представить в виде 25-битового числа с использованием пяти битов на букву, от `"a"` = 00000 до `"z"` = 11001. Таблица из 2^{25} битов или байтов позволит быстро определить, является ли данная пятибуквенная строка словом английского языка. Так что задача сводится к генерации битовых

шаблонов 32 потенциальных слов, получающихся смешением букв двух заданных слов, и поиску этих слов в таблице. Для каждой пары 25-битовых слов w и w' можно действовать следующим образом.

W1. [Проверка различий.] Установить $z \leftarrow w \oplus w'$. Отвергнуть пару слов (w, w'), если $m' \mathbin{\&} (z - m) \mathbin{\&} \bar{m} \neq 0$, где $m = 2^{20} + 2^{15} + 2^{10} + 2^5 + 1$ и $m' = 2^5 m$; эта проверка исключает случаи, когда w и w' имеют общую букву в некоторой позиции (см. 7.1.3–(90). Оказывается, что 10 614 085 из 16 568 646 пар слов не имеют таких общих букв).

W2. [Формирование индивидуальных масок.] Установить $m_0 \leftarrow z \mathbin{\&} (2^5 - 1)$, $m_1 \leftarrow z \mathbin{\&} (2^{10} - 2^5)$, $m_2 \leftarrow z \mathbin{\&} (2^{15} - 2^{10})$, $m_3 \leftarrow z \mathbin{\&} (2^{20} - 2^{15})$ и $m_4 \leftarrow z \mathbin{\&} (2^{25} - 2^{20})$ в качестве подготовки к следующему шагу.

W3. [Подсчет слов.] Установить $l \leftarrow 1$ и $A_0 \leftarrow w$; переменная l будет содержать количество найденных к этому моменту слов, начиная с w. Затем выполнить описанные ниже операции $swap(4)$.

W4. [Вывод максимального решения.] Если l больше или равно текущему максимуму, вывести A_j для $0 \le j < l$. ▮

Сердцем этого высокоскоростного метода является последовательность операций $swap(4)$, которая должна быть встроена в код (например, с помощью макропроцессора) для устранения всех накладных расходов. Она определяется в терминах базовой операции

$sw(j)$: установить $w \leftarrow w \oplus m_j$.
Затем, если w — слово, установить $A_l \leftarrow w$ и $l \leftarrow l + 1$.

Для заданной операции $sw(j)$, которая обменивает буквы в положении j, определим

$$\begin{aligned}
swap(0) &= sw(0);\\
swap(1) &= swap(0), sw(1), swap(0);\\
swap(2) &= swap(1), sw(2), swap(1);\\
swap(3) &= swap(2), sw(3), swap(2);\\
swap(4) &= swap(3), sw(4), swap(3).
\end{aligned} \tag{22}$$

Таким образом, $swap(4)$ раскрывается в последовательность из 31 шага $sw(0)$, $sw(1)$, $sw(0)$, $sw(2)$, …, $sw(0) = sw(\rho(1))$, $sw(\rho(2))$, …, $sw(\rho(31))$; эти шаги будут использоваться 10 миллионов раз. Очевидно, что выигрыша в скорости можно добиться путем вставки значений линеечной функции $\rho(k)$ непосредственно в программу, а не вычислять их для каждой пары слов с использованием алгоритмов M, G и L.

Пара-победитель генерирует набор из 21 слова, а именно

```
ducks, ducky, duces, dunes, dunks, dinks, dinky,
dines, dices, dicey, dicky, dicks, picks, picky,
pines, piney, pinky, pinks, punks, punky, pucks.
```
(23)

Если, например, $w =$ `ducks`, а $w' =$ `piney`, то $m_0 =$ `s` $\oplus$ `y`, так что первая операция $sw(0)$ меняет строку `ducks` на строку `ducky`, которая является словом. Следующая операция $sw(1)$ применяет m_1, что дает `k` $\oplus$ `e` в предпоследней позиции и приводит

к строке `ducey`, словом не являющейся. Еще одно применение $sw(0)$ меняет `ducey` на `duces` (корректное слово, за которым обычно следует слово `tecum`*). И так далее. Таким методом все пары слов можно обработать всего за несколько секунд.

Этот метод можно оптимизировать. Например, после обнаружения пары, которая дает k слов, можно отбрасывать пары после генерации ими $33 - k$ строк, не являющихся словами. Но рассмотренный нами метод и так достаточно быстрый и демонстрирует тот факт, что даже алгоритм L без циклов может быть превзойден.

Поклонники алгоритма L могут, конечно, возразить, что процесс ускорен только в частном случае $n = 5$, в то время как алгоритм L решает задачу генерации для произвольного n. Однако аналогичная идея применима и в общем случае для $n > 5$: программу можно расширить так, чтобы она быстро генерировала все 32 варианта для крайних справа битов $a_4a_3a_2a_1a_0$, как показано выше; затем можно применить алгоритм L после каждых 32 шагов, используя его для генерации последовательных изменений других битов $a_{n-1}\ldots a_5$. Такой подход снижает количество лишней работы, выполняемой алгоритмом L, почти в 32 раза.

Другие бинарные коды Грея. Бинарный код Грея $g(0)$, $g(1)$, $\ldots$, $g(2^n - 1)$ представляет собой только один из множества способов обхода всех возможных n-битовых строк с изменением на каждом шаге только одного бита. Назовем в общем случае "циклом Грея" для бинарных n-кортежей *любую* последовательность $(v_0, v_1, \ldots, v_{2^n-1})$, которая включает все возможные n-кортежи и обладает тем свойством, что v_k отличается от $v_{(k+1) \bmod 2^n}$ только одним битом. Таким образом, на языке теории графов цикл Грея является ориентированным Гамильтоновым циклом в n-мерном кубе. Мы можем считать, что индексы выбраны таким образом, что $v_0 = 0\ldots0$.

Если рассматривать v как двоичные числа, то существуют такие целые числа $\delta_0 \ldots \delta_{2^n-1}$, что

$$v_{(k+1) \bmod 2^n} \;=\; v_k \oplus 2^{\delta_k} \qquad \text{для } 0 \le k < 2^n. \tag{24}$$

Эта так называемая "дельта-последовательность" представляет собой еще один способ описания цикла Грея. Например, дельта-последовательность для стандартного бинарного кода Грея при $n = 3$ имеет вид 01020102; по сути, это линеечная функция $\delta_k = \rho(k + 1)$ из (13), но последнее значение δ_{2^n-1} равно $n - 1$, а не n, так что цикл замыкается. Отдельные элементы δ_k всегда находятся в диапазоне $0 \le \delta_k < n$ и называются координатами.

Пусть $d(n)$ — количество разных дельта-последовательностей, которые определяют n-битовый цикл Грея, а $c(n)$ — количество "канонических" дельта-последовательностей, в которых каждая координата k появляется перед первым появлением $k + 1$. Тогда $d(n) = n!\, c(n)$, поскольку каждая перестановка координат в дельта-последовательности очевидным образом порождает другую дельта-последовательность. Легко видеть, что единственными возможными каноническими дельта-последовательностями для $n \le 3$ являются

$$00; \qquad 0101; \qquad 01020102 \quad \text{и} \quad 01210121. \tag{25}$$

---

* *Duces tecum* (лат., юр.) — повестка о явке в суд для представления суду имеющихся у лица письменных доказательств. — *Примеч. пер.*

Рис. 33. (а) Комплементарный код Грея. (б) Сбалансированный код Грея.

Следовательно, $c(1) = c(2) = 1$, $c(3) = 2$; $d(1) = 1$, $d(2) = 2$ и $d(3) = 12$. Прямые компьютерные вычисления с помощью методов, предназначенных для перечисления Гамильтоновых циклов (они будут рассмотрены позже), дают следующие значения:

$$\begin{aligned} c(4) &= 112; & d(4) &= 2688; \\ c(5) &= 15\,109\,096; & d(5) &= 1\,813\,091\,520. \end{aligned} \tag{26}$$

Никакая простая закономерность не просматривается, а числа растут очень быстро (см. упр. 47); так что можно спорить, что никто не знает точные значения $c(8)$ и $d(8)$.

Поскольку количество возможностей так велико, имеет смысл поискать дополнительные полезные свойства циклов Грея. Например, на рис. 33, (а) показан четырехбитовый цикл Грея, в котором каждая строка $a_3a_2a_1a_0$ диаметрально противоположна своему дополнению $\overline{a}_3\overline{a}_2\overline{a}_1\overline{a}_0$. Такая схема кодирования возможна при четном количестве битов (см. упр. 49).

Еще более интересен цикл Грея, найденный Д. К. Тутиллом (G. C. Tootill) [*Proc. IEE* **103**, Part B Supplement (1956), 435] и приведенный на рис. 33, (б). Этот цикл имеет одинаковое количество изменений на каждой из четырех координатных дорожек, а следовательно, все координаты проявляют равную активность. Сбалансированные подобным образом циклы Грея могут быть построены для всех бóльших значений n с помощью следующего универсального метода расширения цикла от n до $n+2$ бит.

Теорема D. *Пусть $\alpha_1 j_1 \alpha_2 j_2 \ldots \alpha_l j_l$ представляет собой дельта-последовательность для n-битового цикла Грея, где каждое j_k является одной координатой, каждое α_k может быть пустой последовательностью координат, а l нечетно. Тогда*

$$\begin{aligned} &\alpha_1(n{+}1)\alpha_1^R n\alpha_1 \\ &\quad j_1\alpha_2 n\alpha_2^R(n{+}1)\alpha_2\ j_2\alpha_3(n{+}1)\alpha_3^R n\alpha_3\ \ldots\ j_{l-1}\alpha_l(n{+}1)\alpha_l^R n\alpha_l \\ &\qquad\qquad (n{+}1)\alpha_l^R j_{l-1}\alpha_{l-1}^R \ldots \alpha_2^R j_1 \alpha_1^R n \end{aligned} \tag{27}$$

является дельта-последовательностью $(n+2)$-битового цикла Грея.

Например, если начать с последовательности $01\underline{0}201\underline{02}$ для $n = 3$ и рассматривать три подчеркнутых элемента как j_1, j_2, j_3, то новая последовательность (27) для 5-битового цикла будет иметь вид

$$014103010201310242010434010201 03. \tag{28}$$

Доказательство. Пусть α_k имеет длину m_k и пусть v_{kt} является вершиной, достигнутой при начале с $0\ldots0$ и применении изменения координат $\alpha_1 j_1 \ldots \alpha_{k-1} j_{k-1}$ и первых t из α_k. Нам нужно доказать, что при использовании (27), для $1 \le k \le l$ и $0 \le t \le m_k$, встречаются все вершины $00v_{kt}$, $01v_{kt}$, $10v_{kt}$ и $11v_{kt}$. (Крайняя слева координата — $n{+}1$.)

Начиная с $000\ldots0 = 00v_{10}$, мы получаем вершины

$$00v_{11},\ \ldots,\ 00v_{1m_1},\ 10v_{1m_1},\ \ldots,\ 10v_{10}, 11v_{10},\ \ldots,\ 11v_{1m_1};$$

затем j_1 дает $11v_{20}$, за которой следует

$$11v_{21},\ \ldots,\ 11v_{2m_2},\ 10v_{2m_2},\ \ldots,\ 10v_{20}, 00v_{20},\ \ldots,\ 00v_{2m_2};$$

после этого идет $00v_{30}$, и т. д., и в конечном итоге мы достигаем $11v_{lm_l}$. В заключение третья строка из (27) используется для генерации всех недостающих вершин $01v_{lm_l}, \ldots, 01v_{10}$ и возврата к $000\ldots0$. ▮

Количества переходов (*transition counts*) $(c_0, \ldots, c_{n-1})$ дельта-последовательности определяются в предположении, что c_j — количество раз, когда $\delta_k = j$. Например, для (28) количества переходов представляют собой $(12, 8, 4, 4, 4)$ и получаются из последовательности с количествами переходов $(4, 2, 2)$. Если аккуратно выбрать исходную дельта-последовательность и подчеркнуть соответствующий элемент j_k, можно получить настолько близкие количества переходов, насколько это возможно.

Следствие B. *Для всех $n \ge 1$ существует n-битовый цикл Грея с количествами переходов $(c_0, c_1, \ldots, c_{n-1})$, удовлетворяющими условию*

$$|c_j - c_k| \ \le\ 2 \qquad \text{для } 0 \le j < k < n. \tag{29}$$

(Это наилучшее возможное условие сбалансированности, поскольку каждое количество c_j должно быть четным числом и должно выполняться $c_0 + c_1 + \cdots + c_{n-1} = 2^n$. Действительно, условие (29) выполняется тогда и только тогда, когда $n-r$ количеств равны $2q$, а r количеств равны $2q+2$, где $q = \lfloor 2^{n-1}/n \rfloor$ и $r = 2^{n-1} \bmod n$.)

Доказательство. Для заданной дельта-последовательности для n-битового цикла Грея с количествами переходов $(c_0, \ldots, c_{n-1})$ количества для цикла (27) получаются, начиная со значений $(c'_0, \ldots, c'_{n-1}, c'_n, c'_{n+1}) = (4c_0, \ldots, 4c_{n-1}, l{+}1, l{+}1)$, затем вычитанием 2 из c'_{j_k} для $1 \le k < l$ и вычитанием 4 из c'_{j_l}. Например, когда $n = 3$, можно получить 5-битовый сбалансированный код Грея с количествами переходов $(8-2, 16-10, 8, 6, 6) = (6, 6, 8, 6, 6)$, если применить теорему D к дельта-последовательности $\underline{01}2\underline{1}0\underline{1}2\underline{1}$. В упр. 51 подробно рассматриваются случаи с другими значениями n. ▮

Другой важный класс n-битовых циклов Грея, в которых каждая дорожка координат обладает равной ответственностью, получается при рассмотрении *длин серий*, т. е. расстояний между последовательными появлениями одного и того же значения δ. Стандартный бинарный код Грея имеет длину серии 2 в младшей позиции, и это может привести к потере точности при необходимости прецизионных измерений [см., например, обсуждение этого вопроса в G. M. Lawrence and W. E. McClintock, *Proc. SPIE* **2831** (1996), 104–111]. Но в замечательном 5-битовом

цикле Грея с дельта-последовательностью

$$(0123042103210423)^2 \tag{30}$$

все серии имеют длину не менее 4.

Пусть $r(n)$ — максимальное значение r, такое, что можно найти n-битовый цикл Грея, в котором все серии имеют длину $\geq r$. Ясно, что $r(1) = 1$, а $r(2) = r(3) = r(4) = 2$; легко увидеть, что $r(n)$ должно быть меньше n при $n > 2$, следовательно, (30) доказывает, что $r(5) = 4$. Исчерпывающий компьютерный поиск дает значения $r(6) = 4$ и $r(7) = 5$. Прямое вычисление с возвратом для случая $n = 7$ требует дерево примерно с 60 миллионами узлов, чтобы выяснить, что $r(7) < 6$, а в упр. 61, (a) строится 7-битовый цикл с длинами серий не короче 5. Точные значения $r(n)$ для $n \geq 8$ неизвестны; однако $r(10)$ почти наверняка равно 8, и имеются интересные построения, с помощью которых можно доказать, что $r(n) = n - O(\log n)$ при $n \to \infty$ (см. упр. 60–64.)

***Бинарные пути Грея.** Мы определили n-битовый цикл Грея как способ упорядочения всех бинарных n-кортежей в виде последовательности $(v_0, v_1, \ldots, v_{2^n-1})$, обладающей тем свойством, что v_k $(0 \leq k < 2^n - 1)$ смежно с v_{k+1} в n-мерном кубе, а кроме того, v_{2^n-1} смежно с v_0. Свойство цикличности прекрасно, но не всегда критично, и порой можно добиться большего, если от него отказаться. Итак, мы говорим, что n-битовый *путь Грея*, часто именуемый *кодом Грея*, представляет собой любую последовательность, которая удовлетворяет условиям цикла Грея, за исключением того, что последний элемент не обязан быть смежным с первым. Другими словами, цикл Грея представляет собой гамильтонов *цикл* на вершинах n-мерного куба, а код Грея является просто гамильтоновым *путем* в этом графе.

Наиболее важными бинарными путями Грея, не являющимися циклами Грея, являются n-битовые последовательности $(v_0, v_1, \ldots, v_{2^n-1})$, обладающие свойством *монотонности*, т. е. такие, что

$$\nu(v_k) \;\leq\; \nu(v_{k+2}) \qquad \text{для } 0 \leq k < 2^n - 2. \tag{31}$$

(Здесь, как и везде в книге, символ ν обозначает "вес" или "контрольную сумму" (sideways sum) бинарной строки, т. е. количество содержащихся в ней единиц.) Методом проб и ошибок можно найти, что имеются, по сути, только два монотонных n-битовых кода Грея для каждого $n \leq 4$, один из которых начинается с 0^n, а другой — с $0^{n-1}1$. Для $n = 3$ это последовательности

$$000,\ 001,\ 011,\ 010,\ 110,\ 100,\ 101,\ 111; \tag{32}$$

$$001,\ 000,\ 010,\ 110,\ 100,\ 101,\ 111,\ 011. \tag{33}$$

Для $n = 4$ они немного менее очевидны, но находятся без особого труда.

Поскольку $\nu(v_{k+1}) = \nu(v_k) \pm 1$ всякий раз, когда v_k смежно с v_{k+1}, очевидно, что мы не можем усилить условие (31) требованием, чтобы все n-кортежи были отсортированы по весу. Однако соотношение (31) достаточно строгое для того, чтобы определить вес каждого v_k для данного k и вес v_0, поскольку мы знаем, что среди n-кортежей ровно $\binom{n}{j}$ имеют вес j.

На рис. 34 подведены итоги нашего рассмотрения бинарных кодов Грея до текущего момента в виде семи из несметного количества кодов Грея, проходящих

(а) (б) (в) (г) (д) (е) (ж)

Рис. 34. Примеры 8-битовых кодов Грея:

а) стандартный;
б) сбалансированный;
в) комплементарный;
г) длинносерийный;
д) нелокальный;
е) монотонный;
ж) бестрендовый.

по всем 256 возможным восьмибитовым байтам. Черные квадратики представляют единицы, а белые — нули. На рис. 34, (а) показан стандартный бинарный код Грея, на рис. 34, (б) — сбалансированный код с $256/8 = 32$ переходами в каждой координатной позиции. На рис. 34, (в) показан код Грея, аналогичный коду на рис. 33, (а), в котором нижние 128 кодов являются дополнениями верхних 128 кодов. На рис. 34, (г) переходы в каждой координатной позиции осуществляются не ближе чем через 5 шагов; другими словами, все длины серий не меньше 5. Цикл на рис. 34, (д) *нелокальный* в смысле упр. 59. Монотонный путь для $n = 8$ показан на рис. 34, (е) — обратите внимание на черноту в нижней части рисунка. И наконец на рис. 34, (ж) показан полностью немонотонный код Грея, в том смысле, что центр тяжести черных квадратов находится точно в средней точке каждого столбца. Стандартный бинарный код Грея обладает этим свойством в семи позициях координат, но код на рис. 34, (ж) достигает полного равновесия во всех восьми столбцах. Такие коды называются *бестрендовыми* (*trend-free*) и имеют важное значение в аграрной области и других видах деятельности (см. упр. 75 и 76).

Карла Сэведж (Carla Savage) и Питер Винклер (Peter Winkler) [*J. Combinatorial Theory* **A70** (1995), 230–248] нашли элегантный способ построения монотонного бинарного кода Грея для всех $n > 0$. Такие пути обязательно строятся на основе подпутей P_{nj}, в которых все переходы осуществляются между n-кортежами с весами j и $j+1$. Сэведж и Винклер определили подходящие подпути рекурсивно, полагая $P_{10} = 0, 1$, и для всех $n > 0$

$$P_{(n+1)j} = 1P_{n(j-1)}^{\pi_n},\ 0P_{nj}; \tag{34}$$

$$P_{nj} = \emptyset, \quad \text{если } j < 0 \text{ или } j \geq n. \tag{35}$$

Здесь π_n — перестановка координат, которая будет определена позже, а запись P^π означает, что каждый элемент $a_{n-1}\ldots a_1a_0$ последовательности P заменяется элементом $b_{n-1}\ldots b_1b_0$, где $b_{j\pi} = a_j$. (Мы не определяем P^π, полагая $b_j = a_{j\pi}$, поскольку хотим, чтобы $(2^j)^\pi$ было равно $2^{j\pi}$.) Отсюда, например, следует, что

$$P_{20} = 0P_{10} = 00,\ 01 \tag{36}$$

поскольку последовательность $P_{1(-1)}$ пуста; также

$$P_{21} = 1P_{10}^{\pi_1} = 10,\ 11, \tag{37}$$

поскольку последовательность P_{11} пуста и π_1 должна быть тождественной перестановкой. В общем случае P_{nj} является последовательностью n-битовых строк, содержащих ровно $\binom{n-1}{j}$ строк с весом j, которые перемежаются $\binom{n-1}{j}$ строками с весом $j+1$.

Пусть α_{nj} и ω_{nj} являются первым и последним элементами P_{nj}. Тогда мы легко находим, что

$$\omega_{nj} = 0^{n-j-1}1^{j+1} \qquad \text{для } 0 \leq j < n; \tag{38}$$

$$\alpha_{n0} = 0^n \qquad \text{для } n > 0; \tag{39}$$

$$\alpha_{nj} = 1\alpha_{(n-1)(j-1)}^{\pi_{n-1}} \qquad \text{для } 1 \leq j < n. \tag{40}$$

В частности, α_{nj} всегда имеет вес j, а ω_{nj} всегда имеет вес $j+1$. Определим перестановку π_n множества $\{0, 1, \ldots, n-1\}$ так, что последовательности

$$P_{n0},\ P_{n1}^R,\ P_{n2},\ P_{n3}^R,\ \ldots \tag{41}$$

$$\text{и } P_{n0}^R,\ P_{n1},\ P_{n2}^R,\ P_{n3},\ \ldots \tag{42}$$

являются монотонными бинарными путями Грея для $n = 1, 2, 3, \ldots$. Действительно, монотонность очевидна, так что остаются только сомнения по поводу принадлежности к коду Грея. Последовательности (41) и (42) хорошо связываются, поскольку смежность элементов

$$\alpha_{n0} — \alpha_{n1} — \cdots — \alpha_{n(n-1)}, \qquad \omega_{n0} — \omega_{n1} — \cdots — \omega_{n(n-1)} \tag{43}$$

непосредственно вытекает из (34), вне зависимости от перестановок π_n. Таким образом, ключевым пунктом является переход в месте запятой в формуле (34), который делает $P_{(n+1)j}$ подпутем Грея тогда и только тогда, когда

$$\omega_{n(j-1)}^{\pi_n} = \alpha_{nj} \qquad \text{для } 0 < j < n. \tag{44}$$

Например, при $n = 2$ и $j = 1$ нам, в соответствии с (38)–(40), требуется, чтобы $(01)^{\pi_2} = \alpha_{21} = 10$; следовательно, перестановка π_2 должна переставлять координаты 0 и 1. Общая формула (см. упр. 71) имеет вид

$$\pi_n = \sigma_n \pi_{n-1}^2, \tag{45}$$

где σ_n является n-циклом $(n{-}1\ \ldots\ 1\,0)$. Первые несколько случаев, таким образом, имеют вид

$$\begin{aligned} \pi_1 &= (0), & \pi_4 &= (0\,3), \\ \pi_2 &= (0\,1), & \pi_5 &= (0\,4\,3\,2\,1), \\ \pi_3 &= (0\,2\,1), & \pi_6 &= (0\,5\,2\,4\,1\,3). \end{aligned}$$

Похоже, аналитической записи для магических перестановок π_n не существует. В упр. 73 показано, как можно эффективно генерировать коды Сэведж–Винклера.

Небинарные коды Грея. Мы подробно рассмотрели бинарные n-кортежи, поскольку это простейшая, наиболее классическая, широко применяемая и хорошо изученная тема. Но, конечно же, имеются многочисленные приложения, в которых требуется генерация кортежей $(a_1, \ldots, a_n)$ с целыми компонентами в более широких диапазонах $0 \le a_j < m_j$, чем в алгоритме M. Коды Грея прекрасно подходят и для этого случая.

Рассмотрим, например, десятичные цифры, где $0 \le a_j < 10$ для каждого j. Существует ли в десятичной системе счисления способ перечисления, аналогичный бинарному коду Грея, с изменением за один раз только одной цифры? Да; действительно, есть *две* естественные схемы. В первой, *рефлексивном десятичном коде Грея*, последовательность для перечисления чисел до тысячи с помощью строк из трех цифр, имеет вид

$$000, 001, \ldots, 009, 019, 018, \ldots, 011, 010, 020, 021, \ldots, 091, 090, 190, 191, \ldots, 900,$$

где каждая координата изменяется попеременно от 0 до 9, а затем назад от 9 до 0. Во второй схеме, которая называется *модульным десятичным кодом Грея*, цифры

всегда увеличиваются на 1 по модулю 10, “сворачиваясь” от 9 до 0:

$$000, 001, \ldots, 009, 019, 010, \ldots, 017, 018, 028, 029, \ldots, 099, 090, 190, 191, \ldots, 900.$$

В обоих случаях цифра, которая изменяется на шаге k, определяется линеечной функцией по основанию 10 — $\rho_{10}(k)$, наибольшей степенью 10, на которую делится k. Следовательно, каждый n-кортеж цифр появляется ровно однажды: мы генерируем 10^j различных состояний j крайних справа цифр перед тем, как изменять любые другие, для $1 \le j \le n$.

В общем случае рефлексивный код Грея в произвольной смешанной системе счисления можно рассматривать как перестановку неотрицательных целых чисел, т. е. как функцию, которая отображает обычное число в смешанной системе счисления

$$k = \begin{bmatrix} b_{n-1}, & \ldots, & b_1, & b_0 \\ m_{n-1}, & \ldots, & m_1, & m_0 \end{bmatrix} = b_{n-1}m_{n-2}\ldots m_1 m_0 + \cdots + b_1 m_0 + b_0 \tag{46}$$

в его эквивалент в рефлексивном коде Грея

$$\hat{g}(k) = \begin{bmatrix} a_{n-1}, & \ldots, & a_1, & a_0 \\ m_{n-1}, & \ldots, & m_1, & m_0 \end{bmatrix} = a_{n-1}m_{n-2}\ldots m_1 m_0 + \cdots + a_1 m_0 + a_0, \tag{47}$$

как это делается в (7) в частном случае двоичных чисел. Пусть

$$A_j = \begin{bmatrix} a_{n-1}, & \ldots, & a_j \\ m_{n-1}, & \ldots, & m_j \end{bmatrix}, \qquad B_j = \begin{bmatrix} b_{n-1}, & \ldots, & b_j \\ m_{n-1}, & \ldots, & m_j \end{bmatrix}, \tag{48}$$

где $A_n = B_n = 0$, так что при $0 \le j < n$ мы имеем

$$A_j = m_j A_{j+1} + a_j \qquad \text{и} \qquad B_j = m_j B_{j+1} + b_j. \tag{49}$$

Правило, связывающее a и b, нетрудно вывести по индукции по $n - j$:

$$a_j = \begin{cases} b_j, & \text{если } B_{j+1} \text{ четно;} \\ m_j - 1 - b_j, & \text{если } B_{j+1} \text{ нечетно.} \end{cases} \tag{50}$$

(Здесь координаты n-кортежей $(a_{n-1}, \ldots, a_1, a_0)$ и $(b_{n-1}, \ldots, b_1, b_0)$ нумеруются справа налево для согласованности с (7) и соглашениями смешанной системы счисления в 4.1–(9). Читатели, которые предпочитают запись наподобие $(a_1, \ldots, a_n)$, могут при желании заменить во всех формулах j на $n - j$.) Идя обратным путем, получим

$$b_j = \begin{cases} a_j, & \text{если } a_{j+1} + a_{j+2} + \cdots \text{ четно;} \\ m_j - 1 - a_j, & \text{если } a_{j+1} + a_{j+2} + \cdots \text{ нечетно.} \end{cases} \tag{51}$$

Забавно, что правило (50) и обратное ему правило (51) оказываются идентичными, если все основания m_j нечетны. В тернарном коде Грея, например, где $m_0 = m_1 = \cdots = 3$, мы имеем $\hat{g}\big((10010211012)_3\big) = (12210211010)_3$ и $\hat{g}\big((12210211010)_3\big) = (10010211012)_3$. В упр. 78 доказываются соотношения (50) и (51) и рассматриваются подобные формулы для модульного случая.

Такие последовательности Грея можно генерировать без привлечения циклов, обобщая алгоритмы M и L.

Алгоритм H (*Генерация рефлексивных кодов Грея без циклов*). Этот алгоритм посещает все n-кортежи $(a_{n-1}, \ldots, a_0)$, такие, что $0 \le a_j < m_j$ для $0 \le j < n$, изменяя на каждом шаге на ± 1 только один компонент. В нем поддерживаются массив

фокусных указателей $(f_n, \ldots, f_0)$ для управления действиями, как в алгоритме L, а также массив направлений $(o_{n-1}, \ldots, o_0)$. Предполагается, что каждое основание $m_j \geq 2$.

H1. [Инициализация.] Установить $a_j \leftarrow 0$, $f_j \leftarrow j$ и $o_j \leftarrow 1$ для $0 \leq j < n$; установить также $f_n \leftarrow n$.

H2. [Посещение.] Посетить n-кортеж $(a_{n-1}, \ldots, a_1, a_0)$.

H3. [Выбор j.] Установить $j \leftarrow f_0$ и $f_0 \leftarrow 0$. (Как и в алгоритме L, j представляет крайнюю справа активную координату; все элементы справа от нее должны быть реактивизированы.)

H4. [Изменение координаты j.] Завершить работу алгоритма, если $j = n$; в противном случае установить $a_j \leftarrow a_j + o_j$.

H5. [Отражение?] Если $a_j = 0$ или $a_j = m_j - 1$, установить $o_j \leftarrow -o_j$, $f_j \leftarrow f_{j+1}$ и $f_{j+1} \leftarrow j + 1$. (Координата j, таким образом, становится пассивной.) Вернуться к шагу H2. ▮

Аналогичный алгоритм генерирует модульную версию кода Грея (см. упр. 77).

***Подлеса.** Интересное и поучительное обобщение алгоритма H, открытое Я. Кода (Y. Koda) и Ф. Раски (F. Ruskey) [*J. Algorithms* **15** (1993), 324–340], проливает свет на дополнительные особенности кода Грея и генерации без применения циклов. Предположим, что у нас имеется лес из n узлов и нам нужно посетить все его "главные подлеса" ("principal subforests"), т. е. все подмножества узлов S, такие, что если x находится в S и не является корнем, то и родительский по отношению к x узел также находится в S. Например, семиузловой лес имеет 33 таких подмножества, соответствующих черным узлам в приведенных далее 33 диаграммах.

(52)

Обратите внимание, что если считывать верхнюю строку слева направо, среднюю строку — справа налево, а нижнюю — вновь слева направо, то на каждом шаге меняется состояние ровно одного узла.

Если заданный лес состоит из вырожденных неветвящихся деревьев, то главные подлеса эквивалентны числам со смешанным основанием. Например, лес наподобие

имеет $3 \times 2 \times 4 \times 2$ главных подлесов, соответствующих 4-кортежам (x_1, x_2, x_3, x_4), таким, что $0 \leq x_1 < 3$, $0 \leq x_2 < 2$, $0 \leq x_3 < 4$ и $0 \leq x_4 < 2$; значение x_j представляет собой количество узлов, выбранных в j-м дереве. Когда алгоритм Кода и Раски применяется к такому лесу, он посещает подлеса в том же порядке, что и рефлексивный код Грея для оснований $(3, 2, 4, 2)$.

Алгоритм K (*Рефлексивная генерация подлесов, не использующая циклов*). Для заданного леса, узлы которого при обходе в обратном порядке имеют номера $(1, \ldots, n)$, этот алгоритм посещает все бинарные n-кортежи $(a_1, \ldots, a_n)$, такие, что $a_p \geq a_q$, когда p является родительским по отношению к q узлом. (Таким образом, $a_p = 1$ означает, что p — узел в текущем подлесу.) Между двумя последовательными посещениями изменяется ровно один бит a_j. Фокусные указатели $(f_0, f_1, \ldots, f_n)$ аналогичны таковым в алгоритме L и используются вместе с дополнительными массивами указателей $(l_0, l_1, \ldots, l_n)$ и $(r_0, r_1, \ldots, r_n)$, которые представляют дважды связанный список, именуемый текущей каймой (current fringe). Текущая кайма содержит все узлы текущего подлеса и дочерние по отношению к ним; r_0 указывает на ее крайний слева узел, а l_0 — на ее крайний справа узел.

Вспомогательный массив $(c_0, c_1, \ldots, c_n)$ определяет лес следующим образом: если p не имеет дочерних узлов, то $c_p = 0$; в противном случае c_p является крайним слева (наименьшим) дочерним узлом p; c_0 же является крайним слева корнем самого леса. В начале работы алгоритма предполагается, что $r_p = q$ и $l_q = p$, если p и q — последовательные дочерние узлы одного и того же семейства. Таким образом, например, лес в (52) в обратном порядке обхода имеет следующую нумерацию узлов:

;

следовательно, в этом случае в начале шага K1 должно быть $(c_0, \ldots, c_7) = (2, 0, 1, 0, 0, 0, 4, 3)$ и $r_2 = 7$, $l_7 = 2$, $r_3 = 6$, $l_6 = 3$, $r_4 = 5$ и $l_5 = 4$.

K1. [Инициализация.] Установить $a_j \leftarrow 0$ и $f_j \leftarrow j$ для $1 \leq j \leq n$, тем самым делая исходный подлес пустым, а все узлы активными. Установить $f_0 \leftarrow 0$, $l_0 \leftarrow n$, $r_n \leftarrow 0$, $r_0 \leftarrow c_0$ и $l_{c_0} \leftarrow 0$, тем самым помещая все корни в текущую кайму.

K2. [Посещение.] Посетить подлес, определяемый $(a_1, \ldots, a_n)$.

K3. [Выбор p.] Установить $q \leftarrow l_0$, $p \leftarrow f_q$. (Теперь p — крайний справа активный узел каймы.) Также установить $f_q \leftarrow q$ (тем самым активизировав все узлы справа от p).

K4. [Проверка a_p.] Завершить работу алгоритма, если $p = 0$. В противном случае перейти к шагу K6, если $a_p = 1$.

K5. [Вставка дочерних узлов p.] Установить $a_p \leftarrow 1$. Затем, если $c_p \neq 0$, установить $q \leftarrow r_p$, $l_q \leftarrow p - 1$, $r_{p-1} \leftarrow q$, $r_p \leftarrow c_p$, $l_{c_p} \leftarrow p$ (тем самым помещая дочерние узлы p справа от p в кайме). Перейти к шагу K7.

K6. [Удаление дочерних узлов p.] Установить $a_p \leftarrow 0$. Затем, если $c_p \neq 0$, установить $q \leftarrow r_{p-1}$, $r_p \leftarrow q$, $l_q \leftarrow p$ (тем самым удаляя из каймы дочерние узлы p).

K7. [Пассивизация p.] (В этот момент мы знаем, что p — активен.) Установить $f_p \leftarrow f_{l_p}$ и $f_{l_p} \leftarrow l_p$. Вернуться к шагу K2. ▮

Читателю предлагается поэкспериментировать с данным алгоритмом на примерах наподобие (52), чтобы изучить прекрасный механизм, с помощью которого кайма растет и сокращается как раз в нужные моменты.

***Последовательности со сдвигом регистра.** Возможен и совершенно иной способ генерации всех n-кортежей из m-арных цифр. Можно последовательно генерировать по одной цифре за раз и многократно работать с n последними сгенерированными цифрами, таким образом переходя от одного n-кортежа $(x_0, x_1, \ldots, x_{n-1})$ к другому $(x_1, \ldots, x_{n-1}, x_n)$ путем сдвига новой цифры вправо. Например, на рис. 35 показано, как можно получить все пятибитовые числа как блоки из пяти последовательных битов в некоторой циклической структуре длиной 32. Эта общая идея уже рассматривалась в некоторых упражнениях из разделов 2.3.4.2 и 3.2.2, и теперь можно приступить к ее дальнейшему исследованию.

Рис. 35.
Цикл де Брейна для 5-битовых чисел.

Алгоритм S (*Обобщенная генерация со сдвигом регистра*). Этот алгоритм посещает все n-кортежи $(a_1, \ldots, a_n)$, такие, что $0 \le a_j < m$ для $1 \le j \le n$, при условии использования на шаге S3 подходящей функции f.

S1. [Инициализация.] Установить $a_j \leftarrow 0$ для $-n < j \le 0$ и $k \leftarrow 1$.

S2. [Посещение.] Посетить n-кортеж $(a_{k-n}, \ldots, a_{k-1})$. Завершить работу алгоритма, если $k = m^n$.

S3. [Продвижение.] Установить $a_k \leftarrow f(a_{k-n}, \ldots, a_{k-1})$, $k \leftarrow k+1$ и вернуться к шагу S2. ▮

Каждая функция f, делающая алгоритм S корректным, соответствует циклу из m^n цифр в системе счисления с основанием m, таких, что в этом цикле последовательно встречается каждая комбинация n цифр. Например, случай $m = 2$ и $n = 5$, проиллюстрированный на рис. 35, соответствует бинарному циклу

$$00000100011001010011101011011111; \tag{53}$$

а первые m^2 цифр бесконечной последовательности

$$0011021220313233041424344\ldots \tag{54}$$

дают соответствующий цикл для $n = 2$ и произвольного m. Такие циклы обычно называются m-арными *циклами де Брейна*, по имени Н. Г. де Брейна (N. G. de Bruijn), который рассмотрел бинарный случай для произвольного n в *Indagationes Mathematicæ* **8** (1946), 461–467.

Таблица 1

Параметры для алгоритма A

$3:1$	$8:1,5$	$13:1,3$	$18:7$	$23:5$	$28:3$
$4:1$	$9:4$	$14:1,11$	$19:1,5$	$24:1,3$	$29:2$
$5:2$	$10:3$	$15:1$	$20:3$	$25:3$	$30:1,15$
$6:1$	$11:2$	$16:2,3$	$21:2$	$26:1,7$	$31:3$
$7:1$	$12:3,4$	$17:3$	$22:1$	$27:1,7$	$32:1,27$

Записи '$n:s$' и '$n:s,t$' означают, что полиномы x^n+x^s+1 и $x^n+(x^s+1)(x^t+1)$ примитивны по модулю 2. Дополнительные значения до $n=168$ табулированы в работе W. Stahnke, *Math. Comp.* **27** (1973), 977–980.

В упр. 2.3.4.2–23 доказывается, что ровно $m!^{m^{n-1}}/m^n$ функций f обладают требуемыми свойствами. Это очень большое число, но только некоторые из этих функций могут быть быстро вычислены. Мы рассмотрим три типа функций f, которые представляются наиболее полезными.

Первый важный случай — когда m является простым числом, а f — почти линейным рекуррентным соотношением

$$f(x_1,\ldots,x_n)=\begin{cases} c_1, & \text{если } (x_1,x_2,\ldots,x_n)=(0,0,\ldots,0);\\ 0, & \text{если } (x_1,x_2,\ldots,x_n)=(1,0,\ldots,0);\\ (c_1x_1+c_2x_2+\cdots+c_nx_n)\bmod m & \text{в противном случае.}\end{cases} \tag{55}$$

Здесь коэффициенты $(c_1,\ldots,c_n)$ должны быть такими, что

$$x^n-c_nx^{n-1}-\cdots-c_2x-c_1 \tag{56}$$

является примитивным полиномом по модулю m в рассматривавшемся после формулы 3.2.2–(9) смысле. Количество таких полиномов равно $\varphi(m^n-1)/n$, достаточно много, чтобы позволить нам найти такой, в котором только несколько из коэффициентов c не равны нулю. [Это построение берет начало в пионерской работе Willem Mantel, *Nieuw Archief voor Wiskunde* (2) **1** (1897), 172–184.]

Предположим, например, что $m=2$. Тогда бинарные n-кортежи можно генерировать с помощью очень простой процедуры без циклов.

Алгоритм A (*Почти линейная генерация со сдвигом битов*). Этот алгоритм посещает все n-битовые векторы с помощью либо одного специального смещения s [случай 1], либо двух специальных смещений s и t [случай 2], указанных в табл. 1.

A1. [Инициализация.] Установить $(x_0,x_1,\ldots,x_{n-1})\leftarrow(1,0,\ldots,0)$ и $k\leftarrow 0$, $j\leftarrow s$. В случае 2 также установить $i\leftarrow t$ и $h\leftarrow s+t$.

A2. [Посещение.] Посетить n-кортеж $(x_{k-1},\ldots,x_0,x_{n-1},\ldots,x_{k+1},x_k)$.

A3. [Проверка окончания.] Если $x_k\neq 0$, установить $r\leftarrow 0$; в противном случае установить $r\leftarrow r+1$ и перейти к шагу A6, если $r=n-1$. (Мы имеем r последовательных нулей.)

A4. [Сдвиг.] Установить $k\leftarrow(k-1)\bmod n$ и $j\leftarrow(j-1)\bmod n$. В случае 2 также установить $i\leftarrow(i-1)\bmod n$ и $h\leftarrow(h-1)\bmod n$.

A5. [Вычисление нового бита.] Установить $x_k\leftarrow x_k\oplus x_j$ [случай 1] или $x_k\leftarrow x_k\oplus x_j\oplus x_i\oplus x_h$ [случай 2]. Вернуться к шагу A2.

A6. [Завершение.] Посетить $(0,\ldots,0)$ и завершить работу алгоритма. ∎

Подходящие параметры сдвига s и, возможно, t почти наверняка существуют для всех n, поскольку примитивных полиномов достаточно много; например, при $n = 32$ имеется восемь различных вариантов (s, t), и в табл. 1 указан просто наименьший из них. Однако строгое доказательство существования для всех случаев выходит за рамки современных математических знаний.

Наше первое построение циклов де Брейна в (55) было алгебраическим и основывалось на теории конечных полей. Подобный метод, применимый для составных m, имеется в упр. 3.2.2–21. Наше следующее построение, напротив, будет чисто комбинаторным. Фактически оно прочно связано с идеей модульных m-арных кодов Грея.

Алгоритм R (*Генерация рекурсивного цикла де Брейна*). Предположим, что $f()$ — сопрограмма, которая при многократном вызове выводит последовательные цифры m-арного цикла де Брейна длиной m^n, начиная с n нулей. Этот алгоритм представляет собой подобную подпрограмму, которая выводит цикл длиной m^{n+1}, при условии, что $n \geq 2$. Алгоритм поддерживает три закрытые переменные: x, y и t; переменная x изначально равна нулю.

R1. [Вывод.] Вывести x. Перейти к шагу R3, если $x \neq 0$ и $t \geq n$.

R2. [Вызов f.] Установить $y \leftarrow f()$.

R3. [Подсчет единиц.] Если $y = 1$, установить $t \leftarrow t + 1$; в противном случае установить $t \leftarrow 0$.

R4. [Пропуск единицы?] Если $t = n$ и $x \neq 0$, вернуться к шагу R2.

R5. [Подгонка x.] Установить $x \leftarrow (x + y) \bmod m$ и вернуться к шагу R1. ▮

Например, пусть $m = 3$ и $n = 2$. Если $f()$ производит бесконечный цикл длиной 9

$$001102122\ 001102122\ 0\ldots, \tag{57}$$

то алгоритм R будет производить следующий бесконечный цикл длиной 27 на шаге R1.

$$\begin{aligned} y &= 001021220011110212200102122\,001\ldots \\ t &= 001001000012340010000100100\,001\ldots \\ x &= 000110102220120020211122121\,0001\ldots \end{aligned}$$

Доказательство корректной работы алгоритма R интересно и поучительно (см. упр. 93). А доказательство следующего алгоритма, который *удваивает* размер окна n, еще более интересно и поучительно (см. упр. 95).

Алгоритм D (*Генерация удвоенного рекурсивного цикла де Брейна*). Предположим, что $f()$ и $f'()$ — сопрограммы, которые при многокоратном вызове выводят последовательные цифры m-арного цикла де Брейна длиной m^n, начиная с n нулей. (Эти два цикла могут быть идентичны, но они должны генерироваться независимыми сопрограммами, поскольку их значения будут использоваться с разной скоростью.) Этот алгоритм подобен сопрограмме, которая выводит цикл длиной m^{2n}. В нем поддерживаются шесть закрытых переменных: x, y, t, x', y' и t'; переменные x и x' изначально должны быть равны m.

Специальный параметр r должен быть установлен равным константе, для которой выполняются следующие условия:

$$0 \le r \le m \qquad \text{и} \qquad \gcd(m^n - r,\ m^n + r) = 2. \tag{58}$$

Обычно наилучшим выбором является $r = 1$ для нечетного m и $r = 2$ для четного m.

D1. [Возможный вызов f.] Если $t \ne n$ или $x \ge r$, установить $y \leftarrow f()$.

D2. [Подсчет повторов.] Если $x \ne y$, установить $x \leftarrow y$ и $t \leftarrow 1$. В противном случае установить $t \leftarrow t + 1$.

D3. [Вывод из f.] Вывести текущее значение x.

D4. [Вызов f'.] Установить $y' \leftarrow f'()$.

D5. [Подсчет повторов.] Если $x' \ne y'$, установить $x' \leftarrow y'$ и $t' \leftarrow 1$. В противном случае установить $t' \leftarrow t' + 1$.

D6. [Возможный отказ от f'.] Если $t' = n$ и $x' < r$ и либо $t < n$, либо $x' < x$, перейти к шагу D4. Если $t' = n$, $x' < r$ и $x' = x$, перейти к шагу D3.

D7. [Вывод из f'.] Вывести текущее значение x'. Вернуться к шагу D3, если $t' = n$ и $x' < r$; в противном случае вернуться к шагу D1. ▮

Основная идея алгоритма D состоит в поочередном выводе значений из $f()$ и $f'()$ со специальной обработкой, когда любая из последовательностей генерирует n последовательных x для $x < r$. Например, если $f()$ и $f'()$дают цикл (57) длиной 9, мы берем $r = 1$ и получаем

t на шаге D2: 12 31211112 12312111 12123121 11121231 21111212 ...
x на шаге D3: 00001102122 00011021 22000110 21220001 102122000 ...
t' на шаге D5: 1212111121212111112121211112121211112121211112121 ...
x' на шаге D7: 0 11021220 11021220 11021220 11021220 11021220 1 ...;

так что на шагах D3 и **D7** получается цикл 0**0**001**011**01**2**...2**222** 00**00**1... длиной 81.

Случай $m = 2$ алгоритма R был открыт Абрамом Лемпелем (Abraham Lempel) [*IEEE Trans.* **C-19** (1970), 1204–1209]. Алгоритм D был открыт только 25 лет спустя [C. J. Mitchell, T. Etzion, and K. G. Paterson, *IEEE Trans.* **IT-42** (1996), 1472–1478]. Используя их совместно, начиная с простых сопрограмм для $n = 2$ на основе (54), можно построить интересное семейство сотрудничающих сопрограмм, которые будут генерировать цикл де Брейна длиной m^n для любых заданных $m \ge 2$ и $n \ge 2$ с помощью всего лишь $O(\log n)$ простых вычислений для каждой выводимой цифры (см. упр. 96). Кроме того, в простейшем случае $m = 2$ этот комбинированный "метод R&D"* обладает тем свойством, что его k-й вывод можно вычислить непосредственно, как функцию от k, с помощью $O(n \log n)$ простых операций с n-битовыми числами. И наоборот, для любого заданного n-битового шаблона β его позицию в цикле также можно вычислить за $O(n \log n)$ шагов (см. упр. 97–99). В настоящее время неизвестны никакие иные семейства бинарных циклов де Брейна, которые обладали бы последним свойством.

---

* Игра слов: в английском языке *R&D* означает *Research & Development* — научно-исследовательские и опытно-конструкторские работы (НИОКР). — *Примеч. пер.*

Наше третье построение циклов де Брейна основано на теории простых строк, которые будут иметь огромное значение при изучении операций сопоставления с шаблоном в главе 9. Пусть $\gamma = \alpha\beta$ представляет собой конкатенацию двух строк; мы говорим, что α является *префиксом* γ, а β — *суффиксом*. Префикс и суффикс γ называются *собственными* (*proper*), если их длина положительна, но меньше длины γ. Таким образом, β является собственным суффиксом $\alpha\beta$ тогда и только тогда, когда $\alpha \neq \epsilon$ и $\beta \neq \epsilon$.

Определение P. *Строка является простой, если она непуста и (лексикографически) меньше любого собственного суффикса.* ▮

Например, 01101 не является простой строкой, поскольку она больше 01; строка же 01102 — простая, поскольку она меньше 1102, 102, 02 и 2. (Мы считаем, что строки состоят из букв, цифр или других символов линейно упорядоченного алфавита. Лексикографический, или словарный, порядок представляет собой обычный способ сравнения строк, поэтому мы записываем, что $\alpha < \beta$, и говорим, что α меньше β, если α лексикографически меньше, чем β. В частности, всегда $\alpha \leq \alpha\beta$, а $\alpha < \alpha\beta$ тогда и только тогда, когда $\beta \neq \epsilon$.)

Простые строки часто называют словами Линдона, поскольку они были введены Р. К. Линдоном (R. C. Lyndon) [*Trans. Amer. Math. Soc.* **77** (1954), 202–215]; сам Линдон называл их стандартными последовательностями. Термин "простой" более обоснован, так как связан с фундаментальной теоремой о разложении из упр. 101. Но из уважения к Линдону мы будем часто использовать для обозначения простой строки букву λ.

Ряд наиболее важных свойств простых строк был получен Ченом (Chen), Фоксом (Fox) и Линдоном (Lyndon) в важной статье по теории групп [*Annals of Math.* (2) **68** (1958), 81–95], включая следующий простой, но фундаментальный результат.

Теорема P. *Непустая строка, которая меньше всех своих циклических сдвигов, является простой.*

(Циклическими сдвигами строки́ $a_1 \ldots a_n$ являются стро́ки $a_2 \ldots a_n a_1$, $a_3 \ldots a_n a_1 a_2$, $\ldots$, $a_n a_1 \ldots a_{n-1}$.)

Доказательство. Пусть строка $\gamma = \alpha\beta$ не является простой, поскольку $\alpha \neq \epsilon$ и $\gamma \geq \beta \neq \epsilon$; допустим также, что γ меньше, чем ее циклический сдвиг $\beta\alpha$. Тогда из условий $\beta \leq \gamma < \beta\alpha$ вытекает, что $\gamma = \beta\theta$ для некоторой строки $\theta < \alpha$. Следовательно, если γ также меньше, чем ее циклический сдвиг $\theta\beta$, то $\theta < \alpha < \alpha\beta < \theta\beta$. Но это невозможно, поскольку α и θ имеют одну и ту же длину. ▮

Пусть $L_m(n)$ — количество m-арных простых строк длиной n. Каждая строка $a_1 \ldots a_n$ вместе со своими циклическими сдвигами дает d разных строк для некоторого делителя d значения n, соответствующего ровно одной простой строке длиной d. Например, из 010010 с помощью циклического сдвига можно получить строки 100100 и 001001, а наименьшей из периодических частей $\{010, 100, 001\}$ является простая строка 001. Следовательно, мы должны получить

$$\sum_{d \backslash n} d L_m(d) \;=\; m^n \qquad \text{для всех } m, n \geq 1. \tag{59}$$

Это семейство уравнений можно решить относительно $L_m(n)$ с помощью упр. 4.5.3–28, (a), и мы получим

$$L_m(n) \;=\; \frac{1}{n}\sum_{d\backslash n}\mu(d)m^{n/d}. \tag{60}$$

В 1970-х годах Гарольд Фредриксен (Harold Fredricksen) и Джеймс Майорана (James Maiorana) открыли простой красивый способ генерации всех m-арных простых строк длиной n или менее в возрастающем порядке [*Discrete Math.* **23** (1978), 207–210]. Для понимания этого алгоритма нужно рассмотреть *n-расширение* непустой строки λ, а именно — первых n символов бесконечной строки $\lambda\lambda\lambda\ldots$. Например, 10-расширение строки 123 есть 1231231231. В общем случае, если $|\lambda| = k$, ее n-расширением является $\lambda^{\lfloor n/k\rfloor}\lambda'$, где λ' — префикс λ с длиной $n \bmod k$.

Определение Q. *Строка является предпростой, если она представляет собой непустой префикс простой строки на некотором алфавите.* ▮

Теорема Q. *Строка длиной $n > 0$ является предпростой тогда и только тогда, когда она является n-расширением простой строки λ длиной $k \le n$. Эта простая строка определяется единственным образом.*

Доказательство. См. упр. 105. ▮

Теорема Q, по сути, гласит, что существует взаимно однозначное соответствие между простыми строками длиной $\le n$ и предпростыми строками длиной n. Приведенный далее алгоритм генерирует все m-арные экземпляры в порядке возрастания.

Алгоритм F (*Генерация простых и предпростых строк*). Этот алгоритм посещает все m-арные n-кортежи $(a_1, \ldots, a_n)$, такие, что строка $a_1 \ldots a_n$ является предпростой. Он также определяет индекс j, такой, что $a_1 \ldots a_n$ представляет собой n-расширение простой строки $a_1 \ldots a_j$.

F1. [Инициализация.] Установить $a_1 \leftarrow \cdots \leftarrow a_n \leftarrow 0$ и $j \leftarrow 1$; также установить $a_0 \leftarrow -1$.

F2. [Посещение.] Посетить $(a_1, \ldots, a_n)$ с индексом j.

F3. [Подготовка к увеличению.] Установить $j \leftarrow n$. Затем, если $a_j = m - 1$, уменьшать j, пока не будет найдено $a_j < m - 1$.

F4. [Прибавление единицы.] Завершить работу алгоритма, если $j = 0$. В противном случае установить $a_j \leftarrow a_j + 1$. (Сейчас $a_1 \ldots a_j$ — простая строка согласно упр. 105, (a).)

F5. [n-расширение.] Для $k \leftarrow j + 1, \ldots, n$ (в указанном порядке) установить $a_k \leftarrow a_{k-j}$. Вернуться к шагу F2. ▮

Например, алгоритм F посещает 32 тернарные предпростые строки при $m = 3$ и $n = 4$.

$$\begin{array}{cccccccc}
\underset{\wedge}{0}000 & 0011_{\wedge} & 0022_{\wedge} & 0111_{\wedge} & 0122_{\wedge} & 0212_{\wedge} & \underset{\wedge}{1}111 & \underset{\wedge}{1}212 \\
0001_{\wedge} & 0012_{\wedge} & 0\underset{\wedge}{1}01 & 0112_{\wedge} & 0\underset{\wedge}{2}02 & 0220_{\wedge} & 1112_{\wedge} & 122\underset{\wedge}{1} \\
0002_{\wedge} & 00\underset{\wedge}{2}0 & 0\underset{\wedge}{1}02 & 01\underset{\wedge}{2}0 & 02\underset{\wedge}{1}0 & 022\underset{\wedge}{1} & 11\underset{\wedge}{2}1 & 122\underset{\wedge}{2} \\
00\underset{\wedge}{1}0 & 002\underset{\wedge}{1} & 011\underset{\wedge}{0} & 012\underset{\wedge}{1} & 021\underset{\wedge}{1} & 022\underset{\wedge}{2} & 112\underset{\wedge}{2} & \underset{\wedge}{2}222
\end{array} \tag{61}$$

(Цифры, предшествующие символу '$\wedge$', представляют собой простые строки 0, 0001, 0002, 001, 0011, ..., 2.)

Теорема Q поясняет, почему этот алгоритм корректен. Шаги F3 и F4, очевидно, находят наименьшую m-арную простую строку длиной $\le n$, которая превосходит предыдущую предпростую строку $a_1 \ldots a_n$. Заметим, что после того, как a_1 увеличивается от 0 до 1, алгоритм посещает все $(m-1)$-арные простые и предпростые строки, увеличенные на $1\ldots1$.

Алгоритм F очень красив, но как он связан с циклами де Брейна? Изюминка вот в чем: если мы выводим цифры $a_1, \ldots, a_j$ на шаге F2 всякий раз, когда j является делителем n, то последовательность всех таких цифр образует цикл де Брейна! Например, в случае $m = 3$ и $n = 4$ выводится следующая 81 цифра:

$$0\,0001\,0002\,0011\,0012\,0021\,0022\,01\,0102\,0111\,0112$$
$$0121\,0122\,02\,0211\,0212\,0221\,0222\,1\,1112\,1122\,12\,1222\,2. \qquad (62)$$

(Здесь опущены простые строки 001, 002, 011, ..., 122 из (61), поскольку их длина не делит 4.) Причины этого почти магического свойства поясняются в упр. 108. Обратите внимание, что согласно (59) цикл имеет корректную длину.

В определенном смысле вывод этой процедуры фактически эквивалентен "дедушке" всех построений циклов де Брейна, работающих при всех m и n, а именно построения, впервые опубликованного в работе M. H. Martin, *Bull. Amer. Math. Soc.* **40** (1934), 859–864. Оригинальный цикл Мартина для $m = 3$ и $n = 4$ имел вид 2222122202211...10000, т. е. дополнение (62) до 2. Фредриксен (Fredricksen) и Майорана (Maiorana) открыли алгоритм F почти случайно, при поисках простого способа генерации последовательности Мартина. Явная связь между их алгоритмом и предпростыми строками не была замечена еще много лет, пока Раски (Ruskey), Сэведж (Savage) и Ванг (Wang) не выполнили тщательный анализ времени работы алгоритма [*J. Algorithms* **13** (1992), 414–430]. Основные результаты их анализа можно найти в упр. 107, в частности:

i) среднее значение $n - j$ на шагах F3 и F5 приближенно равно $1/(m-1)$;
ii) общее время работы алгоритма по генерации цикла де Брейна наподобие (62) составляет $O(m^n)$.

Упражнения

1. [*10*] Поясните, как сгенерировать все n-кортежи $(a_1, \ldots, a_n)$, в которых $l_j \le a_j \le u_j$, для заданных нижних границ l_j и верхних границ u_j для каждого компонента (считаем, что $l_j \le u_j$).

2. [*15*] Каков 1000000-й n-кортеж, посещенный алгоритмом M, если $n = 10$ и $m_j = j$ для $1 \le j \le n$? *Указание*: $\left[{0,\,0,\,1,\,2,\,3,\,0,\,2,\,7,\,1,\,0 \atop 1,\,2,\,3,\,4,\,5,\,6,\,7,\,8,\,9,\,10}\right] = 1000000$.

▶ **3.** [*M20*] Сколько раз алгоритм M выполняет шаг M4?

▶ **4.** [*18*] В большинстве компьютеров быстрее вести счет по убыванию до 0, чем по возрастанию до m. Переделайте алгоритм M так, чтобы он посещал все n-кортежи в обратном порядке, начиная с $(m_1 - 1, \ldots, m_n - 1)$ и заканчивая $(0, \ldots, 0)$.

▶ **5.** [*20*] Алгоритмы наподобие "быстрого преобразования Фурье" (упр. 4.6.4–14) часто завершаются массивом ответов в порядке с обращенными битами, $A[(b_0 \ldots b_{n-1})_2]$ там, где требуется $A[(b_{n-1} \ldots b_0)_2]$. Предложите эффективный способ переупорядочения ответов в правильном порядке. [*Указание*: обратите алгоритм M.]

6. [*M17*] Докажите (7) — основную формулу бинарного кода Грея.

7. [*20*] На рис. 30, (б) показан бинарный код Грея для диска, разделенного на 16 секторов. Каким должен быть код типа кода Грея для 12 или 60 секторов (для часов или минут на циферблате часов) или для 360 секторов (для градусов окружности)?

8. [*15*] Предложите простой способ обхода всех n-битовых строк с нулевой четностью с изменением на каждом шаге двух битов.

9. [*16*] Какой ход следует сделать при решении головоломки "Китайские кольца" в положении, показанном на рис. 31?

▶ **10.** [*M21*] Найдите простую формулу для общего количества шагов A_n и B_n, когда кольцо (а) снимается или (б) одевается, в случае кратчайшей процедуры снятия n китайских колец. Например, $A_3 = 4$ и $B_3 = 1$.

11. [*M22*] (Г. Д. Пуркисс (H. J. Purkiss), 1865.) Два самых маленьких кольца головоломки можно снимать или одевать одновременно. Сколько шагов потребуется для решения головоломки при таком ускорении?

▶ **12.** [*25*] *Композициями* n называются последовательности положительных целых чисел, дающих в сумме число n. Например, композициями 4 являются 1111, 112, 121, 13, 211, 22, 31 и 4. Целое число n имеет ровно 2^{n-1} композиций, соответствующих всем подмножествам точек $\{1, \ldots, n-1\}$, которые могут использоваться для разбиения интервала $(0\,..\,n)$ на подынтервалы с целочисленными длинами.

а) Разработайте алгоритм без циклов для генерации всех композиций n, представляющий каждую композицию в виде последовательного массива целых чисел $s_1 s_2 \ldots s_j$.

б) Аналогично разработайте алгоритм без циклов, который представляет композиции неявно в виде массива указателей $q_0 q_1 \ldots q_t$, где элементами композиции являются $(q_0 - q_1)(q_1 - q_2)\ldots(q_{t-1} - q_t)$ и $q_0 = n$, $q_t = 0$. Например, композиция 211 при использовании такой схемы должна быть представлена указателями $q_0 = 4$, $q_1 = 2$, $q_2 = 1$, $q_3 = 0$ и $t = 3$.

13. [*21*] Продолжая предыдущее упражнение, вычислите также мультиномиальный коэффициент $C = \binom{n}{s_1,\ldots,s_j}$ при посещении композиции $s_1 \ldots s_j$.

14. [*20*] Разработайте алгоритм для генерации всех строк $a_1 \ldots a_j$, таких, что $0 \le j \le n$ и $0 \le a_i < m_i$ для $1 \le i \le j$, в лексикографическом порядке. Например, если $m_1 = m_2 = n = 2$, ваш алгоритм должен последовательно посетить ϵ, 0, 00, 01, 1, 10, 11.

▶ **15.** [*25*] Разработайте алгоритм *без циклов* для генерации строк из предыдущего упражнения. Все строки одной и той же длины должны, как и ранее, посещаться в лексикографическом порядке, но строки с разными длинами могут быть перемешаны в любом удобном порядке. Например, при $m_1 = m_2 = n = 2$ таким допустимым порядком может быть 0, 00, 01, ϵ, 10, 11, 1.

16. [*23*] Очевидно, что алгоритм без циклов не в состоянии генерировать все бинарные векторы $(a_1, \ldots, a_n)$ в лексикографическом порядке, поскольку количество компонентов a_j, которые должны измениться между последовательными посещениями, не ограничено. Покажите, что, тем не менее, бесцикловая лексикографическая генерация становится возможной, если вместо последовательного представления использовать *связанное*. Предположим, что имеется $2n+1$ узлов $\{0, 1, \ldots, 2n\}$, каждый из которых содержит поле LINK. Бинарный n-кортеж $(a_1, \ldots, a_n)$ представляется с помощью следующих значений полей:

$$\begin{aligned}
&\texttt{LINK}(0) = 1 + na_1;\\
&\texttt{LINK}(j - 1 + na_{j-1}) = j + na_j \qquad \text{для } 1 < j \le n;\\
&\texttt{LINK}(n + na_n) = 0;
\end{aligned}$$

а другие n полей LINK могут содержать любые удобные значения.

17. [*20*] Хорошо известная конструкция, именуемая *картой Карно* [M. Karnaugh, *Amer. Inst. Elect. Eng. Trans.* **72**, part I (1953), 593–599], использует бинарный код Грея в двух измерениях для вывода всех 4-битовых чисел в торе размером 4×4.

0000	0001	0011	0010
0100	0101	0111	0110
1100	1101	1111	1110
1000	1001	1011	1010

(Элементы тора "склеиваются" слева и справа, а также сверху и снизу, как если бы они были частью мозаики, покрывающей плоскость.) Покажите, что можно аналогично упорядочить все 6-битовые числа в виде тора размером 8×8 так, что при перемещении в горизонтальном или вертикальном направлении на один элемент изменяется только одна координатная позиция.

▶ **18.** [*20*] *Вес Ли* вектора $u = (u_1, \ldots, u_n)$, где каждый компонент удовлетворяет условию $0 \le u_j < m_j$, определяется следующим соотношением:

$$\nu_L(u) = \sum_{j=1}^{n} \min(u_j, m_j - u_j);$$

расстояние Ли между двумя такими векторами u и v равно

$$d_L(u, v) = \nu_L(u - v), \qquad \text{где } u - v = ((u_1 - v_1) \bmod m_1, \ldots, (u_n - v_n) \bmod m_n).$$

(Это минимальное количество шагов, необходимых для преобразования u в v, если на каждом шаге изменять некоторую из компонент u_j на ± 1 (по модулю m_j).)

У кватернарного вектора $m_j = 4$ для $1 \le j \le n$, а у бинарного все $m_j = 2$. Найдите простое взаимно однозначное соответствие между кватернарными векторами $u = (u_1, \ldots, u_n)$ и бинарными векторами $u' = (u'_1, \ldots, u'_{2n})$, обладающими теми свойствами, что $\nu_L(u) = \nu(u')$ и $d_L(u, v) = \nu(u' \oplus v')$.

19. [*23*] (*Октакод.*) Пусть $g(x) = x^3 + 2x^2 + x - 1$.

а) Используйте один из алгоритмов данного раздела для вычисления полинома от переменных z_0, z_1, z_2 и z_3 — $\sum z_{u_0} z_{u_1} z_{u_2} z_{u_3} z_{u_4} z_{u_5} z_{u_6} z_{u_\infty}$, где суммирование выполняется по всем 256 полиномам

$$(v_0 + v_1 x + v_2 x^2 + v_3 x^3)\, g(x) \bmod 4 = u_0 + u_1 x + u_2 x^2 + u_3 x^3 + u_4 x^4 + u_5 x^5 + u_6 x^6$$

для $0 \le v_0, v_1, v_2, v_3 < 4$, где u_∞ выбирается так, что $0 \le u_\infty < 4$ и $(u_0 + u_1 + u_2 + u_3 + u_4 + u_5 + u_6 + u_\infty) \bmod 4 = 0$.

б) Постройте множество из 256 16-битовых чисел, которые отличаются одно от другого по меньшей мере в шести различных битовых позициях. (Такое множество, впервые открытое Нордстромом (Nordstrom) и Робинсоном (Robinson) [*Information and Control* **11** (1967), 613–616], по сути, является единственным.)

20. [*M36*] 16-битовые кодовые слова в предыдущем упражнении можно использовать для передачи 8 бит информации с возможностью исправления ошибок передачи данных в одном или двух поврежденных битах. Кроме того, код обнаруживает (но не всегда в состоянии исправить) ошибки в трех битах. Разработайте алгоритм, который либо находит ближайшее кодовое слово для данного 16-битового числа u', либо определяет, что в u' ошибочны по меньшей мере три бита. Как ваш алгоритм декодирует число $(1100100100001111)_2$? [*Указание*: воспользуйтесь тем фактом, что $x^7 \equiv 1$ (по модулю $g(x)$ и 4) и что каждый кватернарный полином степени < 3 сравним с $x^j + 2x^k$ (по модулю $g(x)$ и 4) для некоторого $j, k \in \{0, 1, 2, 3, 4, 5, 6, \infty\}$, где $x^\infty = 0$.]

21. [*M30*] t-подкуб n-мерного куба можно представить строкой наподобие $**10**0*$, содержащей t звездочек и $n-t$ конкретных битов. Если все 2^n бинарных n-кортежей записаны в лексикографическом порядке, то элемент, принадлежащий такому подкубу, появится в $2^{t'}$ кластерах последовательных записей, где t' — количество звездочек, лежащих слева от крайнего справа определенного бита. (В приведенном примере $n = 8$, $t = 5$ и $t' = 4$.) Но если записать n-кортежи в порядке бинарного кода Грея, то количество кластеров может быть уменьшено. Например, $(n-1)$-подкубы $*\ldots*0$ и $*\ldots*1$ при использовании бинарного кода Грея появляются только в $2^{n-2}+1$ и 2^{n-2} кластерах соответственно, а не в 2^{n-1} из них.

а) Поясните, как вычислить $C(\alpha)$, количество бинарных кластеров Грея подкуба, определяемого заданной строкой α, состоящей из звездочек, единиц и нулей. Чему равно $C(**10**0*)$?

б) Докажите, что $C(\alpha)$ всегда лежит между $2^{t'-1}$ и $2^{t'}$ включительно.

в) Чему равно среднее значение $C(\alpha)$, взятое по всем $2^{n-t}\binom{n}{t}$ возможным t-подкубам?

▶ **22.** [*22*] Назовем "правым подкубом" подкуб типа $0110**$, в котором все звездочки появляются после всех определенных битов. Любой бинарный луч (trie) (раздел 6.3) может рассматриваться как способ разбиения куба на непересекающиеся правые подкубы (рис. 36, (а)). Если поменять местами левый и правый подлучи каждого правого подлуча, проходя от корня сверху вниз, то мы получим *бинарный луч Грея* наподобие приведенного на рис. 36, (б).

Докажите, что если "листья" бинарного луча Грея обходятся слева направо, то последовательные листья соответствуют смежным подкубам. (Подкубы являются смежными, если они содержат смежные вершины. Например, $00**$ смежный с $011*$, поскольку первый содержит 0010, а второй — 0110; однако $011*$ не является смежным с $10**$.)

Рис. 36. (а) Обычный бинарный луч. (б) Бинарный луч Грея.

23. [*20*] Предположим, что $g(k) \oplus 2^j = g(l)$. Как проще всего найти l для заданных j и k?

24. [*M21*] Рассмотрим расширение бинарной функции Грея g на все 2-адические целые числа (см. раздел 7.1.3). Какой будет соответствующая обратная функция $g^{[-1]}$?

▶ **25.** [*M25*] Докажите, что если $g(k)$ и $g(l)$ отличаются $t > 0$ битами, и если $0 \le k, l < 2^n$, то $\lceil 2^t/3 \rceil \le |k-l| \le 2^n - \lceil 2^t/3 \rceil$.

26. [*25*] (Фрэнк Раски (Frank Ruskey).) Для каких целых чисел N можно сгенерировать все неотрицательные целые числа, меньшие N, таким образом, чтобы на каждом шаге изменялся только один бит в двоичном представлении этих чисел?

▶ **27.** [*20*] Пусть $S_0 = \{1\}$ и $S_{n+1} = 1/(2+S_n) \cup 1/(2-S_n)$; таким образом, например,

$$S_2 = \left\{ \frac{1}{2+\frac{1}{2+1}}, \frac{1}{2+\frac{1}{2-1}}, \frac{1}{2-\frac{1}{2+1}}, \frac{1}{2-\frac{1}{2-1}} \right\} = \left\{ \frac{3}{7}, \frac{1}{3}, \frac{3}{5}, 1 \right\},$$

и S_n содержит 2^n элементов, лежащих между $\frac{1}{3}$ и 1. Вычислите 10^{10}-й наименьший элемент S_{100}.

28. [*M27*] *Медианой* n-битовых строк $\{\alpha_1, \ldots, \alpha_t\}$, где α_k имеет битовое представление $\alpha_k = a_{k(n-1)} \ldots a_{k0}$, является строка $\hat{\alpha} = a_{n-1} \ldots a_0$, биты которой a_j для $0 \le j < n$ соответствуют большинству битов a_{kj} для $1 \le k \le t$. (Если t четно и биты α_{kj} поровну равны 0 и 1, то бит медианы a_j может быть как 0, так и 1.) Например, строки $\{0010, 0100, 0101, 1110\}$ имеют две медианы, 0100 и 0110, которые можно записать как 01∗0.

а) Найдите простой способ описания медиан $G_t = \{g(0), \ldots, g(t-1)\}$ — первых t бинарных строк, когда $0 < t \le 2^n$.

б) Докажите, что если $\alpha = a_{n-1} \ldots a_0$ является такой медианой и если $2^{n-1} < t < 2^n$, то строка β, полученная из α дополнением произвольного бита a_j, также является элементом G_t.

29. [*M24*] Если целочисленные значения k передаются как n-битовые бинарные коды Грея $g(k)$ и получаются с ошибками, описываемыми битовым шаблоном $p = (p_{n-1} \ldots p_0)_2$, то средняя числовая ошибка равна

$$\frac{1}{2^n} \sum_{k=0}^{2^n-1} \left| g^{[-1]}(g(k) \oplus p) - k \right|$$

в предположении равновероятности всех значений k. Покажите, что эта сумма равна $\sum_{k=0}^{2^n-1} |(k \oplus p) - k|/2^n$, как если бы бинарный код Грея не использовался, и вычислите ее.

▶ **30.** [*M27*] (*Перестановка Грея.*) Разработайте однопроходный алгоритм для замены элементов массива $(X_0, X_1, X_2, \ldots, X_{2^n-1})$ элементами $(X_{g(0)}, X_{g(1)}, X_{g(2)}, \ldots, X_{g(2^n-1)})$ с использованием только фиксированного количества вспомогательной памяти. *Указание*: рассматривая функцию $g(n)$ как перестановку всех неотрицательных целых чисел, покажите, что множество

$$L = \{0, 1, (10)_2, (100)_2, (100*)_2, (100*0)_2, (100*0*)_2, \ldots\}$$

является множеством *ведущих элементов цикла*, т. е. его наименьших элементов.

31. [*HM35*] (*Поля Грея.*) Пусть $f_n(x) = g(r_n(x))$ обозначает операцию отражения битов n-битовой бинарной строки, как в упр. 5, с последующим преобразованием в бинарный код Грея. Например, операция $f_3(x)$ имеет вид $(001)_2 \mapsto (110)_2 \mapsto (010)_2 \mapsto (011)_2 \mapsto (101)_2 \mapsto (111)_2 \mapsto (100)_2 \mapsto (001)_2$, поэтому все ненулевые возможности находятся в единственном цикле. Следовательно, можно использовать f_3 для определения поля из 8 элементов с $\oplus$ в качестве операции сложения и с умножением, определяемым правилом

$$f_3^{[j]}(1) \times f_3^{[k]}(1) = f_3^{[j+k]}(1) = f_3^{[j]}(f_3^{[k]}(1)).$$

Тем же приятным свойством обладают функции f_2, f_5 и f_6, но не функция f_4, поскольку $f_4((1011)_2) = (1011)_2$.

Найдите все $n \le 100$, для которых f_n определяет поле из 2^n элементов.

32. [*M20*] Истинно или ложно следующее утверждение? Функции Уолша удовлетворяют соотношению $w_k(-x) = (-1)^k w_k(x)$.

▶ **33.** [*M20*] Докажите закон Радемахера–Уолша (17).

34. [*M21*] *Функция Пейли* $p_k(x)$ определяется следующими соотношениями:

$$p_0(x) = 1 \qquad \text{и} \qquad p_k(x) = (-1)^{\lfloor 2x \rfloor k} p_{\lfloor k/2 \rfloor}(2x).$$

Покажите, что $p_k(x)$ можно достаточно просто выразить через функции Радемахера, аналогичные (17), и свяжите функции Пейли с функциями Уолша.

35. [*HM23*] Матрица Пейли P_n размером $2^n \times 2^n$ получается из функций Пейли так же, как матрица Уолша W_n получается из функций Уолша. (См. (20).) Найдите интересные соотношения между P_n, W_n и матрицей Адамара H_n. Докажите, что все три матрицы симметричны.

36. [*21*] Подробно опишите эффективный алгоритм вычисления преобразования Уолша $(x_0, \ldots, x_{2^n-1})$ заданного вектора $(X_0, \ldots, X_{2^n-1})$.

37. [*HM23*] Пусть z_{kl} — положение l-го изменения знака в $w_k(x)$ для $1 \le l \le k$ и $0 < z_{kl} < 1$. Докажите, что $|z_{kl} - l/(k+1)| = O((\log k)/k)$.

▶ **38.** [*M25*] Разработайте тернарное обобщение функций Уолша.

▶ **39.** [*HM30*] (Д. Д. Сильвестер (J. J. Sylvester).) Строки матрицы $\left(\begin{smallmatrix} a & b \\ b & -a \end{smallmatrix}\right)$ ортогональны одна другой и имеют одинаковую величину; следовательно, из матричного тождества

$$\begin{aligned}(\,A\;B\,)\begin{pmatrix} a^2+b^2 & 0 \\ 0 & a^2+b^2 \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} &= (\,A\;B\,)\begin{pmatrix} a & b \\ b & -a \end{pmatrix}\begin{pmatrix} a & b \\ b & -a \end{pmatrix}\begin{pmatrix} A \\ B \end{pmatrix} \\ &= (\,Aa+Bb\;\;Ab-Ba\,)\begin{pmatrix} aA+bB \\ bA-aB \end{pmatrix}\end{aligned}$$

вытекает тождество суммы двух квадратов $(a^2+b^2)(A^2+B^2) = (aA+bB)^2 + (bA-aB)^2$. Аналогично матрица

$$\begin{pmatrix} a & b & c & d \\ b & -a & d & -c \\ d & c & -b & -a \\ c & -d & -a & b \end{pmatrix}$$

приводит к тождеству суммы четырех квадратов

$$\begin{aligned}(a^2+b^2+c^2+d^2)(A^2+B^2+C^2+D^2) &= (aA+bB+cC+dD)^2 + (bA-aB+dC-cD)^2 \\ &\quad + (dA+cB-bC-aD)^2 + (cA-dB-aC+bD)^2.\end{aligned}$$

а) Присвойте знаки матрицы H_3 в (21) символам $\{a, b, c, d, e, f, g, h\}$ и получите матрицу с ортогональными строками и тождество суммы восьми квадратов.

б) Выполните обобщение на H_4 и матрицы более высокого порядка.

▶ **40.** [*21*] Будет ли схема вычисления на основе пятибуквенных слов давать правильные ответы, если на шаге W2 вычислять маски как $m_j = z \,\&\, (2^{5j} - 1)$ для $0 \le j < 5$?

41. [*25*] Если мы ограничимся только тремя тысячами наиболее распространенных пятибуквенных слов, тем самым опуская `ducky`, `duces`, `dunks`, `dinks`, `dinky`, `dices`, `dicey`, `dicky`, `dicks`, `picky`, `pinky`, `punky` и `pucks` из (23), то сколько корректных слов может быть сгенерировано из одной пары при таких ограничениях?

42. [*35*] (М. Л. Фредман (M. L. Fredman).) Алгоритм L использует $\Theta(n \log n)$ бит вспомогательной памяти для фокусных указателей при выборе бита a_j бинарного кода Грея, который будет дополнен на следующем шаге. Шаг L3 проверяет $\Theta(\log n)$ вспомогательных битов и изменяет $\Omega(\log n)$ из них.

Покажите, что с теоретической точки зрения можно поступить лучше: n-битовый бинарный код Грея может быть сгенерирован путем изменения между посещениями не более двух вспомогательных битов. (Мы позволяем себе исследовать на каждом шаге $O(\log n)$ вспомогательных битов, так что мы знаем, какие из них должны быть изменены.)

43. [*41*] Определите $d(6)$ — количество 6-битовых циклов Грея. (См. (26).)

44. [*M20*] Покажите, что $d(n) \le \binom{M(n)}{2}$, если n-мерный куб имеет $M(n)$ совершенных паросочетаний.

45. [*M40*] (Т. Федер (T. Feder) и К. Суби (C. Subi), 2009.) В этом упражнении строится большое количество циклов Грея в $(4r+2)$-мерном кубе $G = G_4 \square G_3 \square G_2 \square G_1 \square G_0 \square G_{-1}$, где G_i представляет собой r-мерный куб для $i > 0$, и $G_0 = G_{-1} = P_2$. Вершины v представляют собой $(4r+2)$-битовые строки $v_4 \ldots v_0 v_{-1}$, где v_i содержит r бит при $i > 0$ и один бит при $i \le 0$. "Сигнатурой" v является четырехбитовая строка $\sigma(v) = s_4 s_3 s_2 (s_1 \oplus v_0)$, где s_i — четность v_i. Мы рассматриваем битовые строки как двоичные числа.

Обозначим через $\mathcal{M}_l(v)$ $(1 \le l \le 4)$ совершенное паросочетание в G, такое, что v — $v' = v'_4 \ldots v'_0 v'_{-1}$ и $v'_i = v_i$ для $i \ne l$. (Заметим, что $\mathcal{M}_l(v') = v$.) Определим также $\mathcal{M}_0(v) = v \oplus 2$. Рассмотрим циклы, образованные ребрами v — $\mathcal{M}_{l(v)}(v)$, где $l(v)$ зависит от сигнатуры v.

$\sigma(v) =$	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000
$l(v) =$	0	2	0	3	1	2	0	4	1	2	1	3	1	2	0	4

а) Предположим, что $r = 2$ и $\mathcal{M}_l(v) = v \oplus 2^{2l+s_{l-1}}$ для $l > 1$ и $\mathcal{M}_1(v) = v \oplus 2^{2+(v_0 \oplus v_{-1})}$. Какому циклу принадлежит вершина $0 \ldots 0$ в этом случае?

б) Вершина, сигнатура которой представляет собой степень двойки, называется заземленной (ground) вершиной. Четыре вершины с одними и теми же $v_4 \ldots v_1$ называются братскими. Определим $u \equiv v$, если u и v находятся в одном и том же цикле или если u и v являются братскими заземленными вершинами, или если цепочка таких эквивалентностей ведет от u к v. Поясните, как построить циклы в G для каждого класса эквивалентности.

в) Кроме того, если u и v являются братскими заземленными вершинами, существует такой цикл, который включает ребра $\{u \oplus 2 — u, v \oplus 2 — v\}$ исходных циклов.

г) Наконец покажите, как преобразовать циклы из (б) и (в) в единый цикл.

д) Сколько различных Гамильтоновых циклов можно получить при изменении $\mathcal{M}_1, \ldots, \mathcal{M}_4$?

46. [*M23*] Распространите упражнение 45 на $(kr+2)$-мерный куб для четного k.

47. [*HM24*] Какова асимптотическая оценка $d(n)^{1/2^n}$, вытекающая из упр. 44 и 46?

48. [*HM48*] Определите асимптотическое поведение $d(n)^{1/2^n}$ при $n \to \infty$.

49. [*20*] Докажите, что для всех $n \ge 1$ существует $2n$-битовый цикл Грея, в котором $v_{k+2^{2n-1}}$ является дополнением v_k для всех $k \ge 0$.

▶ **50.** [*21*] Найдите конструкцию, подобную использованной в теореме D, но с четным l.

51. [*M24*] (*Сбалансированные циклы Грея.*) Завершите доказательство следствия B теоремы D.

52. [*M20*] Докажите, что если количество переходов n-битового цикла Грея удовлетворяет условию $c_0 \le c_1 \le \cdots \le c_{n-1}$, то $c_0 + \cdots + c_{j-1} \ge 2^j$, причем равенство достигается при $j = n$.

53. [*M46*] Если числа $(c_0, \ldots, c_{n-1})$ четные и удовлетворяют условию из предыдущего упражнения, то всегда ли существует n-битовый цикл Грея с указанными количествами переходов?

54. [*M20*] (Г. С. Шапиро (H. S. Shapiro), 1953.) Покажите, что если последовательность целых чисел $(a_1, \ldots, a_{2^n})$ содержит только n различных значений, то существует подпоследовательность для некоторых $0 \le k < l \le 2^n$, произведение $a_{k+1} a_{k+2} \ldots a_l$ которой является идеальным квадратом. Однако это утверждение может оказаться неверным, если запретить случай $l = 2^n$.

▶ **55.** [*35*] (Ф. Раски (F. Ruskey) и К. Сэведж (C. Savage), 1993.) Если $(v_0, \ldots, v_{2^n-1})$ — n-битовый цикл Грея, то пары $\{\, \{v_{2k}, v_{2k+1}\} \mid 0 \le k < 2^{n-1} \,\}$ образуют совершенное паросочетание между вершинами n-мерного куба с четной и нечетной четностями. Каждое ли

такое совершенное паросочетание возникает как “половина” некоторого n-битового цикла Грея?

56. [*M30*] (Э. Н. Гильберт (E. N. Gilbert), 1958.) Назовем два цикла Грея эквивалентными, если их дельта-последовательности можно сделать равными путем перестановки имен координат, обращения цикла и/или начала цикла с другого места. Покажите, что 2688 различных 4-битовых циклов Грея делятся на всего лишь 9 классов эквивалентности.

57. [*32*] Рассмотрим граф, вершинами которого являются 2688 возможных 4-битовых циклов Грея и в котором два цикла смежны, если они связаны друг с другом одним из следующих простых преобразований.

До После типа 1 После типа 2 После типа 3 После типа 4

(Изменение типа 1 проявляется, когда цикл может быть разбит на две части с обращением одной из них. Типы 2–4 означают, что цикл можно разбить на три части и собрать вновь после обращения 0, 1 или 2 частей. Части не обязаны иметь одинаковые размеры. Такие преобразования очень часто оказываются возможными для Гамильтоновых циклов.)

Напишите программу, которая выясняет, какие 4-битовые циклы Грея могут быть преобразованы один в другой, путем поиска связных компонентов графа. Ограничьтесь только одним из четырех типов преобразований.

▶ **58.** [*21*] Пусть α — дельта-последовательность n-битового цикла Грея, а β получается из α путем изменений q появлений 0 на n, где q — нечетно. Докажите, что $\beta\beta$ является дельта-последовательностью $(n+1)$-битового цикла Грея.

59. [*22*] 5-битовый цикл Грея из (30) является *нелокальным* в том смысле, что никакие 2^t последовательные элементы не принадлежат одному t-подкубу для $1 < t < n$. Докажите, что существуют нелокальные n-битовые циклы Грея для всех $n \geq 5$. [*Указание*: см. предыдущее упражнение.]

60. [*20*] Покажите, что функция границ длин серий удовлетворяет условию $r(n+1) \geq r(n)$.

61. [*M30*] Покажите, что $r(m+n) \geq r(m) + r(n) - 1$, если (а) $m = 2$ и $2 < r(n) < 8$; или (б) $m \leq n$ и $r(n) \leq 2^{m-3}$.

62. [*46*] Верно ли, что $r(8) = 6$?

63. [*30*] (Луи Годдин (Luis Goddyn).) Докажите, что $r(10) \geq 8$.

▶ **64.** [*HM35*] (Л. Годдин (L. Goddyn) и П. Гвоздяк (P. Gvozdjak).) n-битовым *потоком Грея* называется последовательность перестановок $(\sigma_0, \sigma_1, \ldots, \sigma_{l-1})$, в которой каждая σ_k представляет собой перестановку вершин n-мерного куба, перемещающую каждую вершину на место одного из ее соседей.

а) Предположим, что $(u_0, \ldots, u_{2^m-1})$ является m-битовым циклом Грея, а $(\sigma_0, \sigma_1, \ldots, \sigma_{2^m-1})$ — n-битовым потоком Грея. Пусть $v_0 = 0\ldots0$ и $v_{k+1} = v_k\sigma_k$, где $\sigma_k = \sigma_{k \bmod 2^m}$, если $k \geq 2^m$. При каких условиях последовательность

$$W = (u_0v_0,\ u_0v_1,\ u_1v_1,\ u_1v_2,\ \ldots,\ u_{2^{m+n-1}-1}v_{2^{m+n-1}-1},\ u_{2^{m+n-1}-1}v_{2^{m+n-1}})$$

является $(m+n)$-битовым циклом Грея?

б) Покажите, что если m достаточно велико, существует n-битовый поток Грея, удовлетворяющий условиям (а), для которого все длины серий из последовательности $(v_0, v_1, \ldots)$ не меньше $n-2$.

в) Примените эти результаты для доказательства того факта, что $r(n) \geq n - O(\log n)$.

65. [*30*] (Бретт Стивенс (Brett Stevens).) Пьеса Сэмюэля Беккетта (Samuel Beckett) *Четверка* (*Quad*) начинается и заканчивается пустой сценой; на нее выходят n актеров, появляясь на сцене во всех 2^n возможных подмножествах, и при этом покидающий сцену актер всегда тот, который появился на сцене раньше других присутствующих на ней актеров. При $n = 4$, как в реальной пьесе, некоторые подмножества вынужденно повторяются. Покажите, что при $n = 5$ имеется совершенный шаблон с ровно 2^n входами и выходами.

66. [*40*] Существует ли идеальный шаблон Беккетта–Грея для 8 актеров?

67. [*20*] Иногда желательно пройти по всем n-битовым бинарным строкам путем изменения как можно *большего* количества битов на каждом шаге, например при тестировании надежности схемы в наихудших условиях. Поясните, как обойти все бинарные n-кортежи так, чтобы на каждом шаге поочередно изменялись n или $n-1$ бит.

68. [*21*] Математик И. З. Вращенец намерен построить тернарный код *анти-Грея*, в котором каждое n-значное троичное число отличается от своих соседей *во всех* цифрах. Сможет ли он создать такой код для любого n?

▶ **69.** [*M25*] Изменим определение бинарного кода Грея (7), полагая

$$h(k) = \big(\ldots(b_6 \oplus b_5)(b_5 \oplus b_4)(b_4 \oplus b_3 \oplus b_2 \oplus b_0)(b_3 \oplus b_0)(b_2 \oplus b_1 \oplus b_0)b_1\big)_2,$$

когда $k = (\ldots b_5b_4b_3b_2b_1b_0)_2$.

а) Покажите, что последовательность $h(0), h(1), \ldots, h(2^n-1)$ проходит по всем n-битовым ($n > 3$) числам таким образом, что всякий раз изменяются ровно 3 бит.

б) Обобщите указанное правило для получения последовательностей, в которых на каждом шаге изменяются ровно t бит, где t нечетно и $n > t$.

70. [*21*] Каково количество монотонных n-битовых кодов Грея при $n = 5$ и $n = 6$?

71. [*M22*] Выведите (45) — рекуррентное соотношение, определяющее перестановки Сэведж–Винклера.

72. [*20*] Как выглядит код Сэведж–Винклера для перехода от 00000 к 11111?

▶ **73.** [*32*] Разработайте эффективный алгоритм для построения дельта-последовательности n-битового монотонного кода Грея.

74. [*M25*] (Сэведж (Savage) и Винклер (Winkler).) Насколько в монотонном коде Грея далеко друг от друга могут находиться смежные вершины n-мерного куба?

75. [*32*] Найдите все 5-битовые пути Грея $v_0, \ldots, v_{31}$ *без тренда*, т. е. такие, что в каждой координатной позиции j сумма $\sum_{k=0}^{31} k(-1)^{v_{kj}} = 0$.

76. [*M25*] Докажите, что бестрендовые n-битовые коды Грея существуют для всех $n \geq 5$.

77. [*21*] Модифицируйте алгоритм H так, чтобы можно было обойти n-кортежи со смешанным основанием в *модульном* порядке Грея.

78. [*M26*] Докажите формулы преобразования (50) и (51) для рефлексивных кодов Грея со смешанными основаниями и выведите аналогичные формулы для модульного случая.

▶ **79.** [*M22*] Когда последний n-кортеж (а) рефлексивного (б) модульного кода Грея со смешанным основанием смежен с первым?

80. [*M20*] Поясните, как пройти по всем делителям числа, если задано его разложение на простые множители $p_1^{e_1} \ldots p_t^{e_t}$, выполняя на каждом шаге умножение или деление на одно простое число.

81. [*M21*] Пусть $(a_0, b_0), (a_1, b_1), \ldots, (a_{m^2-1}, b_{m^2-1})$ является 2-значным m-арным модульным кодом Грея. Покажите, что если $m > 2$, то каждое ребро (x, y) — $(x, (y+1) \bmod m)$

и (x, y) — $((x+1) \bmod m, y)$ присутствует в одном из двух циклов:

$$(a_0, b_0) — (a_1, b_1) — \cdots — (a_{m^2-1}, b_{m^2-1}) — (a_0, b_0),$$
$$(b_0, a_0) — (b_1, a_1) — \cdots — (b_{m^2-1}, a_{m^2-1}) — (b_0, a_0).$$

▶ **82.** [*M25*] (Г. Рингель (G. Ringel), 1956.) Воспользуйтесь предыдущим упражнением для вывода того факта, что существует четыре 8-битовых цикла Грея, которые вместе покрывают все ребра 8-мерного куба.

83. [*41*] Можно ли покрыть все ребра 8-мерного куба четырьмя *сбалансированными* 8-битовыми циклами Грея?

▶ **84.** [*25*] (Говард Л. Дикман (Howard L. Dyckman).) На рис. 37 показана замечательная головоломка "Сумасшедшая петля", или "Гордиев узел", в которой нужно снять гибкую веревку с жестких окружающих ее петель. Покажите, что решение этой головоломки тесно связано с рефлексивным тернарным кодом Грея.

Рис. 37. Головоломка "Сумасшедшая петля".

▶ **85.** [*M25*] (Дана Ричардс (Dana Richards).) Если $\Gamma = (\alpha_0, \ldots, \alpha_{t-1})$ представляет собой произвольную последовательность t строк, а $\Gamma' = (\alpha'_0, \ldots, \alpha'_{t'-1})$ — произвольная последовательность t' строк, то их *двумерным произведением* (*boustrophedon product*) $\Gamma \wr \Gamma'$ является последовательность tt' строк, начинающаяся с

$$(\alpha_0\alpha'_0, \ldots, \alpha_0\alpha'_{t'-1}, \alpha_1\alpha'_{t'-1}, \ldots, \alpha_1\alpha'_0, \alpha_2\alpha'_0, \ldots, \alpha_2\alpha'_{t'-1}, \alpha_3\alpha'_{t'-1}, \ldots)$$

и заканчивающаяся $\alpha_{t-1}\alpha'_0$, если t четно, и $\alpha_{t-1}\alpha'_{t'-1}$, если t нечетно. Например, фундаментальное определение бинарного кода Грея в (5) можно выразить с помощью данного обозначения как $\Gamma_n = (0, 1) \wr \Gamma_{n-1}$ при $n > 0$. Докажите, что операция $\wr$ ассоциативна, следовательно, $\Gamma_{m+n} = \Gamma_m \wr \Gamma_n$.

▶ **86.** [*26*] Определите бесконечный код Грея, который проходит по всем возможным неотрицательным целым n-кортежам $(a_1, \ldots, a_n)$ таким образом, что если $(a'_1, \ldots, a'_n)$ следует за $(a_1, \ldots, a_n)$, то $\max(a_1, \ldots, a_n) \le \max(a'_1, \ldots, a'_n)$.

87. [*27*] Продолжая предыдущее упражнение, определите бесконечный код Грея, который проходит по *всем* целочисленным n-кортежам $(a_1, \ldots, a_n)$ таким образом, что если $(a'_1, \ldots, a'_n)$ следует за $(a_1, \ldots, a_n)$, то $\max(|a_1|, \ldots, |a_n|) \le \max(|a'_1|, \ldots, |a'_n|)$.

▶ **88.** [*25*] Что произойдет, если после того, как алгоритм K завершится на шаге K4, он будет немедленно перезапущен на шаге K2?

▶ **89.** [*25*] (*Код Грея для кода Морзе.*) Словами кода Морзе длиной n (упр. 4.5.3–32) являются строки из точек и тире, где n — количество точек плюс удвоенное количество тире.

а) Покажите, что можно сгенерировать все слова кода Морзе длиной n путем последовательной замены тире двумя точками и наоборот. Например, путь для $n = 3$ имеет вид •—, •••, —• или обратный к нему.

б) Какая строка следует за строкой •——••——•—• в вашей последовательности для $n = 15$?

90. [*26*] Для каких значений n можно расположить слова кода Морзе в виде *цикла* при использовании правил из упр. 89? [*Указание*: количество слов кода равно F_{n+1}.]

▶ **91.** [*34*] Разработайте алгоритм без циклов для посещения всех бинарных n-кортежей $(a_1, \ldots, a_n)$, таких, что $a_1 \le a_2 \ge a_3 \le a_4 \ge \cdots$. [Количество таких n-кортежей равно F_{n+2}.]

92. [*M30*] Существует ли такая бесконечная последовательность Φ_n, в которой первые m^n элементов при всех m образуют m-арный цикл де Брейна? [Случай $n = 2$ решен в (54).]

▶ **93.** [*M28*] Докажите, что алгоритм R, как и обещано, выводит цикл де Брейна.

94. [*22*] Каким будет вывод алгоритма D при $m = 5$, $n = 1$ и $r = 3$, если сопрограммы $f()$ и $f'()$ генерируют тривиальные циклы 01234 01234 01 ...?

▶ **95.** [*M23*] Предположим, что бесконечная последовательность $a_0 a_1 a_2 \ldots$ с периодом p перемежается с бесконечной последовательностью $b_0 b_1 b_2 \ldots$ с периодом q и образует бесконечную циклическую последовательность

$$c_0 c_1 c_2 c_3 c_4 c_5 \ldots = a_0 b_0 a_1 b_1 a_2 b_2 \ldots.$$

а) При каких условиях $c_0 c_1 c_2 \ldots$ имеет период pq? (Под "периодом" последовательности $a_0 a_1 a_2 \ldots$ в данном упражнении подразумевается наименьшее целое число $p > 0$, такое, что $a_k = a_{k+p}$ для всех $k \ge 0$.)

б) Какие $2n$-кортежи будут встречаться в последовательных выводах алгоритма D, если шаг D6 заменить на "Если $t' = n$ и $x' < r$, перейти к шагу D4"?

в) Докажите, что алгоритм D, как и обещано, выводит цикл де Брейна.

▶ **96.** [*M28*] Предположим, что у нас имеется семейство сопрограмм для генерации циклов де Брейна длиной m^n с использованием алгоритмов R и D, рекурсивно основанных на простых сопрограммах типа алгоритма S для базового случая $n = 2$, и с использованием алгоритма D, когда $n > 2$ четно.

а) Сколько может иметься сопрограмм каждого типа (R_n, D_n, S_n)?

б) Какое максимальное количество активизаций сопрограмм требуется для получения в выводе одной цифры верхнего уровня?

97. [*M29*] Цель этого упражнения — анализ циклов де Брейна, построенных алгоритмами R и D в важном частном случае $m = 2$. Пусть $f_n(k)$ представляет собой $(k+1)$-й бит 2^n-цикла, так что $f_n(k) = 0$ для $0 \le k < n$. Пусть также j_n — индекс, такой, что $0 \le j_n < 2^n$ и $f_n(k) = 1$ для $j_n \le k < j_n + n$.

а) Запишите циклы $\bigl(f_n(0) \ldots f_n(2^n - 1)\bigr)$ для $n = 2$, 3, 4 и 5.

б) Докажите, что для всех четных значений n существует такое число $\delta_n = \pm 1$, что

$$f_{n+1}(k) \equiv \begin{cases} \Sigma f_n(k), & \text{если } 0 < k \le j_n \text{ или } 2^n + j_n < k \le 2^{n+1}, \\ 1 + \Sigma f_n(k + \delta_n), & \text{если } j_n < k \le 2^n + j_n, \end{cases}$$

где сравнение выполняется по модулю 2. (В этой формуле Σf означает функцию суммирования $\Sigma f(k) = \sum_{j=0}^{k-1} f(j)$.) Следовательно, $j_{n+1} = 2^n - \delta_n$, когда n четно.

в) Пусть $\bigl(c_n(0) c_n(1) \ldots c_n(2^{2n} - 5)\bigr)$ является циклом, созданным путем применения упрощенной версии алгоритма D из упр. 95, (б) к $f_n()$. Где в этом цикле появляются $(2n-1)$-кортежи 1^{2n-1} и $(01)^{n-1}0$?

г) Воспользуйтесь результатами п. (в), чтобы выразить $f_{2n}(k)$ через $f_n()$.

д) Найдите (в определенной степени) простую формулу для j_n как функции от n.

98. [*M34*] Продолжая выполнение предыдущего упражнения, разработайте эффективный алгоритм для вычисления $f_n(k)$ для заданных $n \ge 2$ и $k \ge 0$.

▶ **99.** [*M23*] Воспользуйтесь технологиями из предыдущих упражнений для разработки эффективного алгоритма обнаружения любой заданной n-битовой строки в цикле $\bigl(f_n(0)$ $f_n(1) \ldots f_n(2^n - 1)\bigr)$.

100. [*40*] Является ли цикл де Брейна из упр. 97 при больших n полезным источником псевдослучайных битов?

▶ **101.** [*M30*] (*Уникальное разложение строк на неувеличивающиеся простые строки.*)

а) Докажите, что если λ и λ' простые, то $\lambda\lambda'$ является простой при $\lambda < \lambda'$.

б) Следовательно, каждую строку α можно записать в виде

$$\alpha = \lambda_1\lambda_2\ldots\lambda_t, \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_t, \qquad \text{где каждая } \lambda_j \text{ — простая.}$$

в) В действительности возможно только одно такое разложение. *Указание*: покажите, что λ_t должна быть лексикографически наименьшим непустым суффиксом α.

г) Истинно или ложно следующее утверждение? λ_1 представляет собой наидлиннейший простой префикс α.

д) Каковы простые множители строки 31415926535897932384626433832795028841 97?

102. [*HM28*] Выведите количество m-арных простых строк длиной n из теоремы об единственности разложения из предыдущего упражнения.

103. [*M20*] Воспользуйтесь (59) для доказательства малой теоремы Ферма о том, что $m^p \equiv m$ (по модулю p).

104. [*17*] Согласно формуле (60) примерно $1/n$ всех n-буквенных слов — простые. Сколько простых слов среди 5757 пятибуквенных слов GraphBase? Какое из них является наименьшей непростой строкой? А наибольшей простой строкой?

105. [*M31*] Пусть α — предпростая строка длиной n на бесконечном алфавите.

а) Покажите, что если увеличить последнюю букву α, то полученная строка будет простой.

б) Покажите, что если α разложена на простые строки, как в упр. 101, то она является n-расширением λ_1.

в) Кроме того, α не может быть n-расширением двух разных простых строк.

▶ **106.** [*M30*] Путем обратной реконструкции алгоритма F разработайте алгоритм, который посещает все m-арные простые и предпростые строки в *убывающем* порядке.

107. [*HM30*] Проанализируйте время работы алгоритма F для фиксированного m при $n \to \infty$.

108. [*M35*] Пусть $\lambda_1 < \cdots < \lambda_t$ представляют собой m-арные простые строки, длины которых являются множителями n, и пусть $a_1\ldots a_n$ является произвольной m-арной строкой. Цель данного упражнения — доказать, что $a_1\ldots a_n$ появляется в $\lambda_1\ldots\lambda_t\lambda_1\lambda_2$; следовательно, $\lambda_1\ldots\lambda_t$ представляет собой цикл де Брейна (поскольку его длина — m^n). Для удобства можно считать, что $m = 10$ и что строки соответствуют десятичным числам; те же самые аргументы применимы и для произвольного $m \geq 2$.

а) Покажите, что если $a_1\ldots a_n = \alpha\beta$ отличается от всех ее циклических сдвигов и если $\beta\alpha = \lambda_k$ — простая строка, то $\alpha\beta$ представляет собой подстроку $\lambda_k\lambda_{k+1}$, за исключением случая, когда $\alpha = 9^j$ для некоторого $j \geq 1$.

б) Где в $\lambda_1\ldots\lambda_t$ появится $\alpha\beta$, если $\beta\alpha$ является простой строкой, а α состоит только из девяток? *Указание*: покажите, что если $a_{n+1-l}\ldots a_n = 9^l$ на шаге F2 для некоторого $l > 0$ и если j не является делителем n, то на предыдущем шаге F2 $a_{n-l}\ldots a_n = 9^{l+1}$.

в) Рассмотрим теперь n-кортежи вида $(\alpha\beta)^d$, где $d > 1$ — делитель n, а $\beta\alpha = \lambda_k$ — простая строка.

г) Где появятся 899135, 997879, 913131, 090909, 909090 и 911911 при $n = 6$?

д) Является ли $\lambda_1\ldots\lambda_t$ лексикографически наименьшим m-арным циклом де Брейна длиной m^n?

109. [*M22*] m-арный тор де Брейна размером $m^2 \times m^2$ для окон 2×2 представляет собой матрицу m-арных цифр d_{ij}, такую, что каждая из m^4 подматриц

$$\begin{pmatrix} d_{ij} & d_{i(j+1)} \\ d_{(i+1)j} & d_{(i+1)(j+1)} \end{pmatrix}, \qquad 0 \le i, j < m^2$$

отличается от других (здесь индексы "свернуты в кольцо" по модулю m^2). Таким образом, каждая возможная m-арная подматрица 2×2 получается только один раз; Ян Стюарт (Ian Stewart) [*Game, Set, and Math* (Oxford: Blackwell, 1989), Chapter 4] назвал этот объект m-арным *ауротором* (*ourotorus*). Например,

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

является бинарным ауротором; фактически это единственная такая матрица для $m = 2$, если исключить сдвиги и/или транспонирование.

Рассмотрим бесконечную матрицу D, элемент которой в строке $i = (\ldots a_2 a_1 a_0)_2$ и столбце $j = (\ldots b_2 b_1 b_0)_2$ представляет собой $d_{ij} = (\ldots c_2 c_1 c_0)_2$, где

$$\begin{aligned} c_0 &= (a_0 \oplus b_0)(a_1 \oplus b_1) \oplus b_1; \\ c_k &= (a_{2k} a_0 \oplus b_{2k}) b_0 \oplus (a_{2k+1} a_0 \oplus b_{2k+1})(b_0 \oplus 1) \quad \text{для } k > 0. \end{aligned}$$

Покажите, что верхняя левая $2^{2n} \times 2^{2n}$-подматрица D представляет собой 2^n-арный ауротор для всех $n \ge 0$.

110. [*M25*] Продолжая выполнять предыдущее упражнение, постройте m-арные ауроторы для всех m.

111. [*20*] Число 100 можно получить путем вставки знаков + и − в последовательность 123456789 (в том числе и перед первой ее цифрой) двенадцатью способами:
$100 = 123-45-67+89 = 123-4-5-6-7+8-9 = 123+45-67+8-9 = 123+4-5+67-89 = 12-3-4+5-6+7+89 = 12+3-4+5+67+8+9 = 12+3+4+5-6-7+89 = 1+23-4+56+7+8+9 = 1+23-4+5+6+78-9 = 1+2+34-5+67-8+9 = 1+2+3-4+5+6+78+9 = -1+2-3+4+5+6+78+9$

а) Какое наименьшее положительное число нельзя получить таким образом?

б) Рассмотрите также вставку знаков + и − в последовательность 9876543210.

▶ **112.** [*25*] Продолжая предыдущее упражнение, как далеко вы сможете зайти, вставляя знаки среди цифр 12345678987654321? Например, $100 = -1234-5-6+7898-7-6543-2-1$.

7.2.1.2. Генерация всех перестановок. После n-кортежей следующим по важности элементом практически в любом списке пожеланий в области комбинаторной генерации является посещение всех *перестановок* некоторого заданного множества или мультимножества. Для решения этой задачи разработано много различных способов. В этом разделе мы изучим только наиболее важные генераторы перестановок, начав с классического метода — одновременно простого и гибкого.

Алгоритм L (*Лексикографическая генерация перестановок*). Для заданной последовательности из n элементов $a_1 a_2 \ldots a_n$, изначально отсортированной таким образом, что

$$a_1 \le a_2 \le \cdots \le a_n, \tag{1}$$

этот алгоритм генерирует все перестановки $\{a_1, a_2, \ldots, a_n\}$, посещая их в лексикографическом порядке. (Например, лексикографически упорядоченными перестановками $\{1, 2, 2, 3\}$ являются

1223, 1232, 1322, 2123, 2132, 2213, 2231, 2312, 2321, 3122, 3212, 3221.)

Вспомогательный элемент a_0 используется для удобства; значение a_0 должно быть строго меньше, чем значение наибольшего элемента a_n.

L1. [Посещение.] Посетить перестановку $a_1 a_2 \ldots a_n$.

L2. [Поиск j.] Установить $j \leftarrow n-1$. Если $a_j \geq a_{j+1}$, уменьшать j на 1, пока не будет выполнено условие $a_j < a_{j+1}$. Завершить работу алгоритма, если $j = 0$. (В этот момент j представляет собой наименьший индекс, такой, что все перестановки, начиная с $a_1 \ldots a_j$, уже были посещены. Так что лексикографически следующая перестановка увеличивает значение a_j.)

L3. [Увеличение a_j.] Установить $l \leftarrow n$. Если $a_j \geq a_l$, уменьшать l на 1, пока не будет выполнено условие $a_j < a_l$. Затем выполнить обмен $a_j \leftrightarrow a_l$. (Поскольку $a_{j+1} \geq \cdots \geq a_n$, элемент a_l является наименьшим элементом, который больше a_j и который может законным образом следовать в перестановке за $a_1 \ldots a_{j-1}$. Перед обменом мы имели $a_{j+1} \geq \cdots \geq a_{l-1} \geq a_l > a_j \geq a_{l+1} \geq \cdots \geq a_n$; после мы получаем $a_{j+1} \geq \cdots \geq a_{l-1} \geq a_j > a_l \geq a_{l+1} \geq \cdots \geq a_n$.)

L4. [Обращение $a_{j+1} \ldots a_n$.] Установить $k \leftarrow j+1$ и $l \leftarrow n$. Затем, пока $k < l$, выполнять обмен $a_k \leftrightarrow a_l$ и установки $k \leftarrow k+1$, $l \leftarrow l-1$. Вернуться к шагу L1. ▮

Этот алгоритм восходит к Нараяне Пандита (Nārāyaṇa Paṇḍita), жившему в XIV веке в Индии (см. раздел 7.2.1.7); он также упоминается в предисловии К. Ф. Гинденбурга (C. F. Hindenburg) к книге *Specimen Analyticum de Lineis Curvis Secundi Ordinis* Х. Ф. Рюдигера (C. F. Rüdiger) (Leipzig: 1784), xlvi–xlvii, и неоднократно переоткрывался впоследствии. Пояснения, как он работает, приводятся в скобках на шагах L2 и L3.

В общем случае объект, лексикографически следующий за данным объектом $a_1 \ldots a_n$, получается с помощью следующей трехшаговой процедуры.

1) Найти наибольшее j, такое, что a_j может быть увеличено.
2) Увеличить a_j на наименьшее допустимое значение.
3) Найти лексикографически наименьший способ расширения новой части $a_1 \ldots a_j$ до полного объекта.

Алгоритм L следует этой общей процедуре в случае генерации перестановок, так же как алгоритм 7.2.1.1M следует ей при генерации n-кортежей; позже мы увидим еще множество иных примеров при рассмотрении других комбинаторных шаблонов. Обратите внимание, что в начале шага L4 $a_{j+1} \geq \cdots \geq a_n$. Следовательно, первая перестановка, начинающаяся с текущего префикса $a_1 \ldots a_j$, имеет вид $a_1 \ldots a_j a_n \ldots a_{j+1}$, и шаг L4 генерирует ее с помощью $\lfloor (n-j)/2 \rfloor$ обменов.

На практике шаг L2 находит $j = n-1$ в половине случаев, если все элементы различны, поскольку ровно у $n!/2$ из $n!$ перестановок $a_{n-1} < a_n$. Следовательно, алгоритм L можно ускорить, выделяя этот частный случай без существенного усложнения самого алгоритма (см. упр. 1.) Аналогично вероятность того, что

Тин тан дин дан бим бам бом бо —
тан тин дин дан бам бим бо бом —
тин тан дан дин бим бам бом бо —
тан тин дан дин бам бим бо бом —
тан дан тин бам дин бо бим бом —
. . . Тин тан дин да бим бам бом бо.

— ДОРОТИ Л. СОЙЕРС (DOROTHY L. SAYERS),
Девять портных (*The Nine Tailors*) (1934)

Перестановка десяти десятичных цифр — это просто десятичное число из 10 цифр, в котором все цифры различны. Следовательно, все, что требуется сделать, — это сгенерировать все числа из десяти цифр и выбрать из них те, в которых все цифры различны. Разве не чудесно, что высокоскоростные компьютеры избавляют нас от необходимости думать? Мы просто программируем $k+1 \to k$ и проверяем цифры k на наличие нежелательных совпадений. Причем так мы получаем все перестановки в словарном порядке! По зрелом размышлении. . . нам не нужно задумываться о чем-либо ином.

— Д. Г. ЛЕМЕР (D. H. LEHMER) (1957)

$j \le n - t$, составляет лишь $1/t!$, когда все a различны; следовательно, обычно циклы на шагах L2–L4 очень быстрые. В упр. 6 анализируется время работы в общем случае и демонстрируется, что алгоритм L достаточно эффективен даже при наличии равных элементов, если только в мультимножестве $\{a_1, a_2, \ldots, a_n\}$ некоторые элементы не появляются значительно чаще других.

Смежные обмены. В разделе 7.2.1.1 мы видели, что коды Грея позволяют эффективно генерировать n-кортежи. Те же рассуждения применимы и к генерации перестановок. Простейшим возможным изменением перестановки является обмен соседних элементов, а из главы 5 мы знаем, что любая перестановка может быть упорядочена при использовании соответствующей последовательности таких обменов (например, так работает алгоритм пузырьковой сортировки 5.2.2B). Следовательно, можно пойти обратным путем и получить любую требуемую перестановку, если начать с упорядоченного размещения всех элементов и использовать только обмен в подходящих парах смежных элементов.

При этом возникает естественный вопрос “Можно ли обойти *все* перестановки заданного мультимножества, обменивая на каждом шаге местами только два соседних элемента?” Если можно, то программа обхода всех перестановок в целом зачастую будет проще и быстрее, поскольку будет требовать каждый раз вычисления только одного обмена вместо обработки всего нового массива $a_1 \ldots a_n$.

Увы, если мультимножество содержит повторяющиеся элементы, такую “грееобразную” последовательность можно найти не всегда. Например, шесть перестановок мультимножества $\{1, 1, 2, 2\}$ связаны одна с другой смежными обменами следующим образом:

$$\begin{array}{ccccccccc} & & & & 2112 & & & & \\ & & & \diagup & & \diagdown & & & \\ 1122 & — & 1212 & & & & 2121 & — & 2211; \\ & & & \diagdown & & \diagup & & & \\ & & & & 1221 & & & & \end{array} \tag{2}$$

у этого графа нет гамильтонова пути.

Однако большинство приложений работают с перестановками *различных* элементов, а в этом случае нас ждет хорошая новость: простой алгоритм позволяет сгенерировать все $n!$ перестановок с помощью $n! - 1$ смежных обменов. Кроме того, еще один смежный обмен возвращает нас в начальное состояние, так что мы получаем гамильтонов цикл, аналогичный бинарному коду Грея.

Идея заключается в том, чтобы взять такую последовательность для $\{1, \ldots, n-1\}$ и вставить число n в каждую из перестановок всеми возможными способами. Например, если $n = 4$, то последовательность $(123, 132, 312, 321, 231, 213)$ при вставке 4 во все четыре возможные позиции приводит к следующим столбцам массива.

$$\begin{array}{cccccc} 1234 & 1324 & 3124 & 3214 & 2314 & 2134 \\ 1243 & 1342 & 3142 & 3241 & 2341 & 2143 \\ 1423 & 1432 & 3412 & 3421 & 2431 & 2413 \\ 4123 & 4132 & 4312 & 4321 & 4231 & 4213 \end{array} \tag{3}$$

Теперь мы получаем искомую последовательность, читая первый столбец сверху вниз, второй — снизу вверх, третий — сверху вниз, ..., последний — снизу вверх: $(1234, 1243, 1423, 4123, 4132, 1432, 1342, 1324, 3124, 3142, \ldots, 2143, 2134)$.

В разделе 5.1.1 мы изучали инверсии перестановки, а именно пары элементов (не обязательно смежные), находящиеся не в упорядоченном состоянии. Каждый обмен смежных элементов изменяет общее количество инверсий на ± 1. Действительно, если мы рассмотрим так называемую таблицу инверсий $c_1 \ldots c_n$ из упр. 5.1.1–7, где c_j — количество элементов, лежащих справа от j и меньших j, то мы обнаружим, что перестановки в (3) имеют следующую таблицу инверсий.

$$\begin{array}{cccccc} 0000 & 0010 & 0020 & 0120 & 0110 & 0100 \\ 0001 & 0011 & 0021 & 0121 & 0111 & 0101 \\ 0002 & 0012 & 0022 & 0122 & 0112 & 0102 \\ 0003 & 0013 & 0023 & 0123 & 0113 & 0103 \end{array} \tag{4}$$

А если читать эти столбцы попеременно сверху вниз и снизу вверх, как и ранее, то получится в точности рефлексивный код Грея для смешанных оснований $(1, 2, 3, 4)$, как в уравнениях (46)–(51) из раздела 7.2.1.1. Как заметил Э. В. Дейкстра (E. W. Dijkstra) [*Acta Informatica* **6** (1976), 357–359], это свойство выполняется для всех n, что приводит нас к следующей формулировке алгоритма.

Алгоритм P (*Простые изменения*). Для заданной последовательности $a_1 a_2 \ldots a_n$ из n различных элементов этот алгоритм генерирует все их перестановки путем многократных обменов смежных элементов. Алгоритм использует вспомогательный массив $c_1 c_2 \ldots c_n$, который представляет инверсии, как было описано выше, и проходит по всем последовательностям целых чисел, таких, что

$$0 \le c_j < j \qquad \text{для } 1 \le j \le n. \tag{5}$$

Другой массив, $o_1 o_2 \ldots o_n$, управляет направлениями изменений элементов c_j.

P1. [Инициализация.] Установить $c_j \leftarrow 0$ и $o_j \leftarrow 1$ для $1 \le j \le n$.

P2. [Посещение.] Посетить перестановку $a_1 a_2 \ldots a_n$.

P3. [Подготовка к изменению.] Установить $j \leftarrow n$ и $s \leftarrow 0$. (Следующие шаги определяют координату j, для которой будет изменяться c_j, сохраняя (5); переменная s представляет собой количество индексов $k > j$, таких, что $c_k = k - 1$.)

P4. [Готовность к изменению?] Установить $q \leftarrow c_j + o_j$. Если $q < 0$, перейти к шагу P7; если $q = j$, перейти к шагу P6.

P5. [Изменение.] Обменять $a_{j-c_j+s} \leftrightarrow a_{j-q+s}$. Затем установить $c_j \leftarrow q$ и вернуться к шагу P2.

P6. [Увеличение s.] Завершить работу алгоритма, если $j = 1$; в противном случае установить $s \leftarrow s + 1$.

P7. [Переключение направления.] Установить $o_j \leftarrow -o_j$, $j \leftarrow j - 1$ и вернуться к шагу P4. ▮

Эта процедура, которая, очевидно, работает для всех $n \geq 1$, появилась в Англии XVII века, когда звонари придумали обычай перезвона множества колоколов во всех возможных перестановках. Алгоритм P они называли методом *простых изменений*. На рис. 38, (а) показана неупорядоченная специальная последовательность "Кембридж 48" из 48 перестановок 5 колоколов, которая использовалась в начале 1600-х годов, до того как принцип простых изменений позволил получить все $5! = 120$ возможных перестановок. Почтенная история алгоритма P прослеживается до манускрипта Питера Манди (Peter Mundy), написанного около 1653 года и хранящегося ныне в библиотеке имени Бодлея при Оксфордском университете. Эта история перепечатана Эрнестом Моррисом (Ernest Morris) в *The History and Art of Change Ringing* (1931), 29–30. Вскоре после этого известная книга *Tintinnalogia*, анонимно опубликованная в 1668 году (как ныне известно, она была написана Ричардом Даквортом (Richard Duckworth) и Фабианом Стедманом (Fabian Stedman)), посвятила первые 60 страниц детальному описанию метода простых изменений, начиная со случая $n = 3$ и заканчивая произвольно большими n.

> *Последовательность "*Кембридж 48*" многие годы была величайшим* Перезвоном*, который когда-либо звонил или был изобретен; но теперь ни "48", ни "100", ни "720" или какое-то иное число не может нас сдержать — мы открыли метод* изменения звона колоколов до бесконечности.
>
> *...Для четырех колоколов имеется* 24 разных варианта перезвона*; один из них называется "*Колебание*", еще три — "*Особые *Колокола*"; "*Колебание*" движется, постоянно* колеблясь *вверх и вниз...; два из "*Особых *колоколов*" меняются всякий раз, когда перед ними или после них следует "*Колебание*".*
>
> — Р. ДАКВОРТ (R. DUCKWORTH) и Ф. СТЕДМАН (F. STEDMAN), *Tintinnalogia* (*Тинтинналогия*, или *Искусство колокольного звона*) (1668)

Британские энтузиасты колокольного звона вскоре придумали более сложные схемы, в которых одновременно меняются местами две или более пар колоколов. Например, они придумали приведенный на рис. 38, (в) шаблон, известный под названием "Дубли деда", "наилучший и наиболее искусный перезвон, когда-либо придуманный для пяти колоколов" [*Tintinnalogia*, с. 95]. Такие методы, однако, более интересны, чем алгоритм P, с музыкальной точки зрения, но не с точки зрения вычислительной, так что мы не будем рассматривать их здесь. Заинтересованный читатель может обратиться к книге W. G. Wilson, *Change Ringing* (1965); см. также A. T. White, *AMM* **103** (1996), 771–778.

Х. Ф. Троттер (H. F. Trotter) опубликовал первую компьютерную реализацию алгоритма простых изменений в *CACM* **5** (1962), 434–435. Это достаточно эффек-

(неполностью)

(а) Последовательность “Кембридж 48”.

(б) Последовательность “Простые изменения”.

(в) Последовательность “Дубли деда”.

(г) Последовательность “Дубли Стедмана”.

Рис. 38. Четыре последовательности, использовавшиеся в Англии XVII века для перестановки пяти разных церковных колоколов. Последовательность (б) соответствует алгоритму P.

тивный алгоритм, особенно если он отлажен, как в упр. 16, поскольку $n-1$ из каждых n перестановок генерируются без использования шагов P6 и P7. У алгоритма же L наилучшие варианты встречаются только в половине случаев.

Тот факт, что алгоритм P выполняет только по одному обмену на посещение, означает, что генерируемые им перестановки попеременно четны и нечетны (см. упр. 5.1.1–13). Следовательно, можно сгенерировать все четные перестановки, пропуская нечетные. Действительно, таблицы c и o облегчают отслеживание текущего общего количества инверсий, $c_1 + \cdots + c_n$.

Во многих программах необходимо повторно генерировать одни и те же перестановки, и в таких случаях не требуется каждый раз проходить все шаги алгоритма P. Можно просто подготовить список подходящих переходов, используя описанный далее метод.

Алгоритм T (*Переходы простых изменений*). Этот алгоритм вычисляет таблицу $t[1]$, $t[2]$, $\ldots$, $t[n!-1]$, такую, что действия алгоритма P эквивалентны последовательным изменениям $a_{t[k]} \leftrightarrow a_{t[k]+1}$ для $1 \le k < n!$. Считаем, что $n \ge 2$.

T1. [Инициализация.] Установить $N \leftarrow n!$, $d \leftarrow N/2$, $t[d] \leftarrow 1$ и $m \leftarrow 2$.

T2. [Цикл по m.] Завершить работу алгоритма, если $m = n$. В противном случае установить $m \leftarrow m+1$, $d \leftarrow d/m$ и $k \leftarrow 0$. (Мы поддерживаем условие $d = n!/m!$.)

T3. [Колебание вниз.] Установить $k \leftarrow k+d$ и $j \leftarrow m-1$. Затем, пока $j > 0$, устанавливать $t[k] \leftarrow j$, $k \leftarrow k+d$ и $j \leftarrow j-1$.

T4. [Смещение.] Установить $t[k] \leftarrow t[k]+1$.

T5. [Колебание вверх.] Установить $k \leftarrow k+d$, $j \leftarrow 1$. Пока $j < m$, устанавливать $t[k] \leftarrow j$, $k \leftarrow k+d$, $j \leftarrow j+1$. Затем вернуться к шагу T3, если $k < N$, в противном случае вернуться к шагу T2. ▮

Например, если $n = 4$, мы получаем таблицу $(t[1], t[2], \ldots, t[23]) = (3, 2, 1, 3, 1, 2, 3, 1, 3, 2, 1, 3, 1, 2, 3, 1, 3, 2, 1, 3, 1, 2, 3)$.

Буквометики. Давайте теперь рассмотрим простую головоломку, в решении которой полезны перестановки: какое корректное суммирование зашифровано в приведенном далее примере, где каждая буква означает какую-то свою десятичную цифру, не совпадающую с другими?

$$\begin{array}{r} \texttt{SEND} \\ +\,\texttt{MORE} \\ \hline \texttt{MONEY} \end{array} \tag{6}$$

[H. E. Dudeney, *Strand* **68** (1924), 97, 214.] Такие головоломки часто называют буквометиками (alphametics) с легкой руки Д. А. Х. Хантера (J. A. H. Hunter) [*Globe and Mail* (Toronto: 27 October 1955), 27]; имеется еще одно название, предложенное С. Ватриквантом (S. Vatriquant) [*Sphinx* **1** (May 1931), 50]: "криптарифм" ("cryptarithm").

Классический буквометик (6) легко решить вручную (см. упр. 21). Но предположим, что необходимо решить большое количество сложных буквометиков, часть из которых неразрешима, а другие могут иметь десятки решений. В таком случае можно сэкономить время, написав программу, перебирающую все подстановки цифр и ищущие те из них, которые дают корректную сумму. [Такая программа была опубликована Джоном Бейдлером John Beidler в *Creative Computing* **4**, 6 (November–December 1978), 110–113.]

Мы можем не ограничивать себя и рассмотреть аддитивные буквометики в общем случае, т. е. не только простые суммы наподобие (6), но и примеры посложнее, наподобие

$$\texttt{VIOLIN} + \texttt{VIOLIN} + \texttt{VIOLA} \;=\; \texttt{TRIO} + \texttt{SONATA}.$$

Эта головоломка эквивалентна следующей:

$$2(\texttt{VIOLIN}) + \texttt{VIOLA} - \texttt{TRIO} - \texttt{SONATA} \;=\; 0, \tag{7}$$

где фигурирует сумма членов с целочисленными коэффициентами, которая в итоге должна быть равна нулю при верной подстановке различных десятичных цифр вместо различных букв. Каждая буква в такой задаче имеет "сигнатуру", получающуюся путем подстановки 1 вместо данной буквы и 0 вместо других букв; например, сигнатура для I в (7) имеет вид

$$2(010010) + 01000 - 0010 - 000000,$$

а именно 21010. Если произвольным образом присвоить индексы $(1, 2, \ldots, 10)$ буквам (V, I, O, L, N, A, T, R, S, X), то соответствующими сигнатурами для (7) будут

$$\begin{array}{lllll} s_1 = 210000, & s_2 = 21010, & s_3 = -7901, & s_4 = 210, & s_5 = -998, \\ s_6 = -100, & s_7 = -1010, & s_8 = -100, & s_9 = -100000, & s_{10} = 0. \end{array} \tag{8}$$

(Добавлена дополнительная буква X, поскольку нам требуется набор из десяти букв.) Теперь задача заключается в поиске всех перестановок $a_1 \ldots a_{10}$ множества $\{0, 1, \ldots, 9\}$, таких, что

$$a \cdot s \;=\; \sum_{j=1}^{10} a_j s_j \;=\; 0. \tag{9}$$

Обычно имеется одно побочное условие, заключающееся в том, что в буквометике числа не могут начинаться с нуля. Например, суммы

$$\begin{array}{r} 7316 \\ +\,0823 \\ \hline 08139 \end{array} \quad \text{и} \quad \begin{array}{r} 5731 \\ +\,0647 \\ \hline 06378 \end{array} \quad \text{и} \quad \begin{array}{r} 6524 \\ +\,0735 \\ \hline 07259 \end{array} \quad \text{и} \quad \begin{array}{r} 2817 \\ +\,0368 \\ \hline 03185 \end{array}$$

и еще двадцать подобных, несмотря на арифметическую корректность, решениями буквометика (6) *не* являются. В общем случае существует множество F первых букв, такое, что должно выполняться условие

$$a_j \neq 0 \qquad \text{для всех } j \in F; \tag{10}$$

множество F, соответствующее (7) и (8), имеет вид $\{1, 7, 9\}$.

Одним из способов работы с аддитивными буквометиками является использование алгоритма T для подготовки таблицы из $10! - 1$ переходов $t[k]$. Затем для каждой задачи, определенной последовательностью сигнатур $(s_1, \ldots, s_{10})$ и множеством первых букв F, можно выполнить исчерпывающий поиск решения следующим образом.

A1. [Инициализация.] Установить $a_1 a_2 \ldots a_{10} \leftarrow 01\ldots9$, $v \leftarrow \sum_{j=1}^{10}(j-1)s_j$, $k \leftarrow 1$ и $\delta_j \leftarrow s_{j+1} - s_j$ для $1 \le j < 10$.

A2. [Проверка.] Если $v = 0$ и если выполняется (10), вывести решение $a_1 \ldots a_{10}$.

A3. [Обмен.] Завершить работу алгоритма, если $k = 10!$. В противном случае установить $j \leftarrow t[k]$, $v \leftarrow v - (a_{j+1} - a_j)\delta_j$, $a_{j+1} \leftrightarrow a_j$, $k \leftarrow k + 1$ и вернуться к шагу A2. ▮

Шаг A3 оправдывается тем фактом, что обмен a_j с a_{j+1} просто уменьшает $a \cdot s$ на $(a_{j+1} - a_j)(s_{j+1} - s_j)$. Несмотря на то что $10!$ равно достаточно большому числу 3 628 800, операции на шаге A3 столь просты, что на современных компьютерах работа выполняется за доли секунды.

Буквометик называется *чистым*, если имеет единственное решение. К сожалению, буквометик (7) не является чистым. Перестановки 1764802539 и 3546281970 удовлетворяют условиям (9) и (10), следовательно, имеется два решения:

$$176478 + 176478 + 17640 = 2576 + 368020$$

и

$$354652 + 354652 + 35468 = 1954 + 742818.$$

Кроме того, в (8) $s_6 = s_8$, так что еще два решения можно получить путем обмена цифр, назначенных буквам `A` и `R`.

С другой стороны, буквометик (6) *является* чистым, хотя данный метод находит две различные решающие его перестановки. Дело в том, что буквометик (6) включает только восемь различных букв, следовательно, нам приходится использовать две фиктивные сигнатуры $s_9 = s_{10} = 0$. В общем случае буквометик с m различными цифрами включает $10 - m$ фиктивных сигнатур $s_{m+1} = \cdots = s_{10} = 0$, и каждое решение будет найдено $(10 - m)!$ раз, если только не потребовать, скажем, выполнения условия $a_{m+1} < \cdots < a_{10}$.

Общая структура. Для генерации перестановок различных объектов было предложено очень много алгоритмов, и наилучший способ понять их — применение мультипликативных свойств перестановок, которые мы изучали в разделе 1.3.3. С этой целью мы слегка изменим систему обозначений, используя индексирование с нуля и запись $a_0a_1\ldots a_{n-1}$ для перестановок $\{0,1,\ldots,n-1\}$ вместо записи $a_1a_2\ldots a_n$ для перестановок $\{1,2,\ldots,n\}$. Что более важно, мы будем рассматривать схемы для генерации перестановок, в которых основная работа будет выполняться *слева*, так что все перестановки $\{0,1,\ldots,k-1\}$ будут генерироваться на первых $k!$ шагах, $1\le k\le n$. Например, одной такой схемой для $n=4$ является

$$\begin{aligned}&0123, 1023, 0213, 2013, 1203, 2103, 0132, 1032, 0312, 3012, 1302, 3102,\\&0231, 2031, 0321, 3021, 2301, 3201, 1230, 2130, 1320, 3120, 2310, 3210,\end{aligned}\tag{11}$$

называемая "обратный солексикографический (солексный) порядок", поскольку, если обратить строки справа налево, мы получим 3210, 3201, 3120, ..., 0123, т. е. обратный лексикографический порядок. Еще один способ представления (11) заключается в рассмотрении элементов в виде $(n-a_n)\ldots(n-a_2)(n-a_1)$, где $a_1a_2\ldots a_n$ лексикографически проходит по всем перестановкам $\{1,2,\ldots,n\}$.

Вспомним из раздела 1.3.3, что перестановка наподобие $\alpha=250143$ может быть записана либо в двустрочном виде

$$\alpha=\binom{012345}{250143},$$

либо в более компактном циклическом виде

$$\alpha=(0\ 2)(1\ 5\ 3),$$

который означает, что в α имеет место $0\mapsto 2$, $1\mapsto 5$, $2\mapsto 0$, $3\mapsto 1$, $4\mapsto 4$ и $5\mapsto 3$; цикл из одного элемента типа '(4)' можно не указывать. Поскольку 4 представляет собой фиксированную точку данной перестановки, мы говорим, что "α фиксирует 4". Мы также записываем $0\alpha=2$, $1\alpha=5$ и т. д., говоря, что $j\alpha$ является "образом j при перестановке α". Произведение перестановок наподобие α, умноженной на β, где $\beta=543210$, легко записать либо в двустрочном виде

$$\alpha\beta=\binom{012345}{250143}\binom{012345}{543210}=\binom{012345}{250143}\binom{250143}{305412}=\binom{012345}{305412},$$

либо в циклическом

$$\alpha\beta=(0\ 2)(1\ 5\ 3)\ \cdot\ (0\ 5)(1\ 4)(2\ 3)\ =\ (0\ 3\ 4\ 1)(2\ 5).$$

Обратите внимание, что образом 1 при перестановке $\alpha\beta$ является $1(\alpha\beta)=(1\alpha)\beta=5\beta=0$, и т. д. *Предупреждение*: почти в половине всех книг о перестановках их умножают иначе (справа налево), полагая, что $\alpha\beta$ означает, что β применяется до α. Причина этого в том, что традиционная функциональная запись, в которой записывается $\alpha(1)=5$, естественным образом приводит к мысли, что $\alpha\beta(1)$ должно означать $\alpha(\beta(1))=\alpha(4)=4$. Однако в данной книге используется иная философия, и перестановки в ней умножаются слева направо.

Порядок умножения необходимо особенно внимательно учитывать, когда перестановки представлены массивами чисел. Например, если "применить" отражение $\beta=543210$ к перестановке $\alpha=250143$, то результат 341052 представляет собой

не $\alpha\beta$, а $\beta\alpha$. В общем случае операция замены перестановки $\alpha = a_0 a_1 \ldots a_{n-1}$ некоторым переупорядочением $a_{0\beta} a_{1\beta} \ldots a_{(n-1)\beta}$ дает $k \mapsto a_{k\beta} = k\beta\alpha$. Перестановка *позиции* с помощью β соответствует *умножению слева* (*предумножению*, *premultiplication*) на β, заменяющему α на $\beta\alpha$; перестановка *значений* с помощью β соответствует *умножению справа* (*постумножению*, *postmultiplication*) на β, заменяющему α на $\alpha\beta$. Таким образом, например, генератор перестановок, который обменивает $a_1 \leftrightarrow a_2$, умножает слева текущую перестановку на (1 2) и умножает ее справа на $(a_1\ a_2)$.

Следуя работе Эвариста Галуа (Évariste Galois) 1830 года, будем говорить, что непустое множество перестановок G образует *группу*, если оно замкнуто относительно операции умножения, т. е. если произведение $\alpha\beta$ принадлежит G тогда, когда α и β являются элементами G [см. *Écrits et Mémoires Mathématiques d'Évariste Galois* (Paris: 1962), 47]. Рассмотрим, например, четырехмерный куб, представленный в виде тора размером 4×4

$$\boxed{\begin{matrix} \mathtt{0} & \mathtt{1} & \mathtt{3} & \mathtt{2} \\ \mathtt{4} & \mathtt{5} & \mathtt{7} & \mathtt{6} \\ \mathtt{c} & \mathtt{d} & \mathtt{f} & \mathtt{e} \\ \mathtt{8} & \mathtt{9} & \mathtt{b} & \mathtt{a} \end{matrix}}, \tag{12}$$

как в упр. 7.2.1.1–17, и пусть G представляет собой множество всех перестановок вершин $\{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{f}\}$, сохраняющих отношение смежности: перестановка α находится в G тогда и только тогда, когда из u — v в четырехмерном кубе вытекает $u\alpha$ — $v\alpha$. (Здесь для обозначения целых чисел $(0, 1, \ldots, 15)$ мы используем шестнадцатеричные цифры $(\mathtt{0}, \mathtt{1}, \ldots, \mathtt{f})$. Метки в (12) выбраны так, что u — v тогда и только тогда, когда u и v отличаются на одну битовую позицию.) Это множество G, очевидно, является группой, а его элементы называются симметриями или "аутоморфизмами" четырехмерного куба.

Группы перестановок G удобно представлять в компьютере с помощью *таблицы Симса*, предложенной Чарльзом К. Симсом (Charles C. Sims) [*Computational Methods in Abstract Algebra* (Oxford: Pergamon, 1970), 169–183], которая представляет собой семейство подмножеств S_1, S_2, ... множества G, обладающих следующим свойством: S_k содержит ровно одну перестановку σ_{kj}, которая отображает $k \mapsto j$ и фиксирует значения всех элементов, бóльших k, если такая перестановка содержится в G. Перестановка σ_{kk} является тождественной перестановкой, которая всегда имеется в G; когда $0 \le j < k$, для роли σ_{kj} может быть выбрана любая подходящая перестановка. Главным преимуществом таблицы Симса является то, что она позволяет удобно представлять всю группу в целом.

Лемма S. *Пусть S_1, S_2, ..., S_{n-1} — таблица Симса для группы G перестановок $\{0, 1, \ldots, n-1\}$. Тогда каждый элемент α из G имеет единственное представление*

$$\alpha = \sigma_1 \sigma_2 \ldots \sigma_{n-1}, \qquad \text{где } \sigma_k \in S_k \text{ для } 1 \le k < n. \tag{13}$$

Доказательство. Если α имеет такое представление и если σ_{n-1} представляет собой перестановку $\sigma_{(n-1)j} \in S_{n-1}$, то α отображает $n-1 \mapsto j$, поскольку все элементы $S_1 \cup \cdots \cup S_{n-2}$ фиксируют значение $n-1$. И обратно: если α отображает $n-1 \mapsto j$, то $\alpha = \alpha' \sigma_{(n-1)j}$, где

$$\alpha' = \alpha\, \sigma^{-}_{(n-1)j}$$

является перестановкой G, которая фиксирует $n-1$. (Как и в разделе 1.3.3, σ^- обозначает инверсию σ.) Множество G' всех таких перестановок является группой, а $S_1, \ldots, S_{n-2}$ — таблицей Симса для G'; таким образом, результат следует по индукции по n. ▮

Например, вычисления показывают, что одной возможной таблицей Симса для аутоморфизма группы четырехмерного куба является

$$\begin{aligned}
S_{\mathtt{f}} &= \{(),\ (\mathtt{01})(\mathtt{23})(\mathtt{45})(\mathtt{67})(\mathtt{89})(\mathtt{ab})(\mathtt{cd})(\mathtt{ef}),\ \ldots,\\
&\qquad\qquad (\mathtt{0f})(\mathtt{1e})(\mathtt{2d})(\mathtt{3c})(\mathtt{4b})(\mathtt{5a})(\mathtt{69})(\mathtt{78})\};\\
S_{\mathtt{e}} &= \{(),\ (\mathtt{12})(\mathtt{56})(\mathtt{9a})(\mathtt{de}),\ (\mathtt{14})(\mathtt{36})(\mathtt{9c})(\mathtt{be}),\ (\mathtt{18})(\mathtt{3a})(\mathtt{5c})(\mathtt{7e})\};\\
S_{\mathtt{d}} &= \{(),\ (\mathtt{24})(\mathtt{35})(\mathtt{ac})(\mathtt{bd}),\ (\mathtt{28})(\mathtt{39})(\mathtt{6c})(\mathtt{7d})\};\\
S_{\mathtt{c}} &= \{()\};\\
S_{\mathtt{b}} &= \{(),\ (\mathtt{48})(\mathtt{59})(\mathtt{6a})(\mathtt{7b})\};\\
S_{\mathtt{a}} &= S_{\mathtt{9}} = \cdots = S_{\mathtt{1}} = \{()\};
\end{aligned} \tag{14}$$

здесь $S_{\mathtt{f}}$ содержит 16 перестановок $\sigma_{\mathtt{f}j}$ для $0 \le j \le 15$, которые соответственно отображают $i \mapsto i \oplus (15-j)$ для $0 \le i \le 15$. Множество $S_{\mathtt{e}}$ содержит только четыре перестановки, поскольку аутоморфизм, который фиксирует $\mathtt{f}$, должен отображать $\mathtt{e}$ в соседа $\mathtt{f}$; таким образом, образом $\mathtt{e}$ должно быть либо $\mathtt{e}$, либо $\mathtt{d}$, либо $\mathtt{b}$, либо $\mathtt{7}$. Множество $S_{\mathtt{c}}$ содержит только тождественную перестановку, поскольку аутоморфизм, который фиксирует $\mathtt{f}$, $\mathtt{e}$ и $\mathtt{d}$, должен фиксировать и $\mathtt{c}$. У большинства групп $S_k = \{()\}$ для всех малых значений k, как в данном примере; следовательно, таблица Симса обычно нуждается в наличии только достаточно малого количества перестановок, хотя сама по себе группа может быть весьма большой.

Представление Симса (13) упрощает проверку наличия данной перестановки α в группе G: сначала мы определяем $\sigma_{n-1} = \sigma_{(n-1)j}$, где α отображает $n-1 \mapsto j$, и полагаем $\alpha' = \alpha\sigma_{n-1}^-$; затем мы определяем $\sigma_{n-2} = \sigma_{(n-2)j'}$, где α' отображает $n-2 \mapsto j'$, и полагаем $\alpha'' = \alpha'\sigma_{n-2}^-$; и т. д. Если на каком-то этапе требуемая перестановка σ_{kj} отсутствует в S_k, то исходная перестановка α не принадлежит G. В случае (14) этот процесс должен привести α к тождественной перестановке после обнаружения $\sigma_{\mathtt{f}}$, $\sigma_{\mathtt{e}}$, $\sigma_{\mathtt{d}}$, $\sigma_{\mathtt{c}}$ и $\sigma_{\mathtt{b}}$.

Пусть, например, α является перестановкой $(\mathtt{14})(\mathtt{28})(\mathtt{3c})(\mathtt{69})(\mathtt{7d})(\mathtt{be})$, которая соответствует транспозиции (12) относительно главной диагонали $\{\mathtt{0}, \mathtt{5}, \mathtt{f}, \mathtt{a}\}$. Поскольку α фиксирует $\mathtt{f}$, $\sigma_{\mathtt{f}}$ будет тождественной перестановкой $()$, а $\alpha' = \alpha$. Тогда $\sigma_{\mathtt{e}}$ является членом $S_{\mathtt{e}}$, который отображает $\mathtt{e} \mapsto \mathtt{b}$, а именно $(\mathtt{14})(\mathtt{36})(\mathtt{9c})(\mathtt{be})$, и находим $\alpha'' = (\mathtt{28})(\mathtt{39})(\mathtt{6c})(\mathtt{7d})$. Эта перестановка принадлежит $S_{\mathtt{d}}$, так что α действительно является аутоморфизмом четырехмерного куба.

И наоборот, (13) также упрощает генерацию всех элементов соответствующей группы. Мы просто проходим по всем перестановкам вида

$$\sigma(1, c_1)\sigma(2, c_2)\ldots\sigma(n-1, c_{n-1}),$$

где $\sigma(k, c_k)$ является (c_k+1)-м элементом S_k для $0 \le c_k < s_k = |S_k|$ и $1 \le k < n$, с помощью любого алгоритма из раздела 7.2.1.1, который проходит по всем $(n-1)$-кортежам $(c_1, \ldots, c_{n-1})$ для соответствующих оснований $(s_1, \ldots, s_{n-1})$.

Применение общей структуры. Основной интерес вызывает группа *всех* перестановок $\{0, 1, \ldots, n-1\}$, а в этом случае каждое множество S_k таблицы Симса будет содержать $k+1$ элементов $\{\sigma(k,0), \sigma(k,1), \ldots, \sigma(k,k)\}$, где $\sigma(k,0)$ является тождественной перестановкой, а другие отображают k на значения $\{0, \ldots, k-1\}$ в некотором порядке. (Перестановка $\sigma(k,j)$ необязательно должна быть такой же, как и σ_{kj}, и обычно отличается от нее.) Каждая такая таблица Симса приводит к генератору перестановок согласно следующей схеме.

Алгоритм G (*Обобщенный генератор перестановок*). Этот алгоритм генерирует для заданной таблицы Симса $(S_1, S_2, \ldots, S_{n-1})$, где каждое множество S_k содержит $k+1$ элементов $\sigma(k,j)$, как было описано выше, все перестановки $a_0 a_1 \ldots a_{n-1}$ для $\{0, 1, \ldots, n-1\}$, используя вспомогательную управляющую таблицу $c_n \ldots c_2 c_1$.

G1. [Инициализация.] Установить $a_j \leftarrow j$ и $c_{j+1} \leftarrow 0$ для $0 \le j < n$.

G2. [Посещение.] (В этот момент число в системе счисления со смешанным основанием $\left[\begin{smallmatrix} c_{n-1}, & \ldots, & c_2, & c_1 \\ n, & \ldots, & 3, & 2 \end{smallmatrix}\right]$ представляет собой количество перестановок, посещенных до сих пор.) Посетить перестановку $a_0 a_1 \ldots a_{n-1}$.

G3. [Добавление 1 к $c_n \ldots c_2 c_1$.] Установить $k \leftarrow 1$. Пока $c_k = k$, устанавливать $c_k \leftarrow 0$ и $k \leftarrow k+1$. Завершить работу алгоритма, если $k = n$; в противном случае установить $c_k \leftarrow c_k + 1$.

G4. [Перестановка.] Применить перестановку $\tau(k, c_k)\omega(k-1)^-$ к $a_0 a_1 \ldots a_{n-1}$, как поясняется ниже, и вернуться к шагу G2. ▮

Применение перестановки π к $a_0 a_1 \ldots a_{n-1}$ означает замену a_j на $a_{j\pi}$ для $0 \le j < n$; это соответствует поясненному ранее умножению слева на π. Определим

$$\tau(k,j) = \sigma(k,j)\sigma(k,j-1)^- \qquad \text{для } 1 \le j \le k; \tag{15}$$

$$\omega(k) = \sigma(1,1)\ldots\sigma(k,k). \tag{16}$$

Тогда шаги G3 и G4 поддерживают следующее свойство:

$$a_0 a_1 \ldots a_{n-1} \text{ является перестановкой } \sigma(1,c_1)\sigma(2,c_2)\ldots\sigma(n-1,c_{n-1}), \tag{17}$$

и лемма S доказывает, что каждая перестановка посещается ровно один раз.

Рис. 39. Алгоритм G неявно обходит это дерево при $n = 4$.

Дерево на рис. 39 иллюстрирует алгоритм G в случае $n = 4$. Согласно (17) каждая перестановка $a_0 a_1 a_2 a_3$ множества $\{0, 1, 2, 3\}$ соответствует контрольной строке из трех цифр, $c_3 c_2 c_1$, где $0 \le c_3 \le 3$, $0 \le c_2 \le 2$ и $0 \le c_1 \le 1$. Некоторые узлы дерева помечены только одной цифрой, c_3; они соответствуют перестановкам $\sigma(3, c_3)$ используемой таблицы Симса. Другие узлы, помеченные двумя цифрами, $c_3 c_2$, соответствуют перестановкам $\sigma(2, c_2)\sigma(3, c_3)$. Толстая линия соединяет узел c_3

с узлом $c_3 0$, а узел $c_3 c_2$ с узлом $c_3 c_2 0$, поскольку $\sigma(2,0)$ и $\sigma(1,0)$ представляют собой тождественные перестановки и данные узлы, по сути, эквивалентны. Прибавление 1 к числу со смешанным основанием $c_3 c_2 c_1$ на шаге G3 соответствует перемещению от одного узла на рис. 39 к следующему за ним в прямом порядке обхода, а преобразование на шаге G4 соответственно изменяет перестановки. Например, когда $c_3 c_2 c_1$ меняется со 121 на 200, шаг G4 умножает слева текущую перестановку на

$$\tau(3,2)\,\omega(2)^- = \tau(3,2)\,\sigma(2,2)^-\sigma(1,1)^-;$$

умножение слева на $\sigma(1,1)^-$ дает переход от узла 121 к узлу 12, умножение слева на $\sigma(2,2)^-$ дает переход от узла 12 к узлу 1, а умножение слева на $\tau(3,2) = \sigma(3,2)\sigma(3,1)^-$ дает переход от узла 1 к узлу $2 \equiv 200$, который является преемником узла 121 в прямом порядке обхода. Иначе говоря, умножение слева на $\tau(3,2)\omega(2)^-$ — это именно то, что требуется для изменения $\sigma(1,1)\sigma(2,2)\sigma(3,1)$ на $\sigma(1,0)\sigma(2,0)\sigma(3,2)$, с сохранением (17).

Алгоритм G определяет огромное количество генераторов перестановок (см. упр. 37), так что не удивительно, что в литературе встречается множество его частных случаев. Конечно, одни из его вариантов более эффективны, чем другие, и хотелось бы найти примеры, когда выполняемые операции особенно эффективны на используемом компьютере.

Можно, например, получить перестановки в обратном солексном порядке как частный случай алгоритма G (см. (11)), делая $\sigma(k,j)$ $(j+1)$-циклом

$$\sigma(k,j) \;=\; (k{-}j \;\; k{-}j{+}1 \;\; \ldots \;\; k). \tag{18}$$

Дело в том, что $\sigma(k,j)$ должна быть перестановкой, которая соответствует $c_n \ldots c_1$ в обратном солексном порядке, когда $c_k = j$ и $c_i = 0$ для $i \neq k$, и эта перестановка $a_0 a_1 \ldots a_{n-1}$ имеет вид $01\ldots(k{-}j{-}1)(k{-}j{+}1)\ldots(k)(k{-}j)(k{+}1)\ldots(n{-}1)$. Например, когда $n = 8$ и $c_n \ldots c_1 = 00030000$, соответствующая перестановка в обратном солексном порядке — 01345267, или (2 3 4 5) в циклическом виде. Когда $\sigma(k,j)$ задается формулой (18), уравнения (15) и (16) приводят к формулам

$$\tau(k,j) = (k{-}j \;\; k); \tag{19}$$

$$\omega(k) = (0\,1)(0\,1\,2)\ldots(0\,1\,\ldots\,k) = (0\,k)(1\,k{-}1)(2\,k{-}2)\ldots = \phi(k); \tag{20}$$

здесь $\phi(k)$ — "$(k+1)$-переворот", который изменяет $a_0 \ldots a_k$ на $a_k \ldots a_0$. В этом случае $\omega(k)$ оказывается равной $\omega(k)^-$, поскольку $\phi(k)^2 = ()$.

Уравнения (19) и (20) неявно присутствуют за кулисами алгоритма L и его эквивалента в обратном солексном порядке (упр. 2), где шаг L3, по сути, применяет транспозицию, и шаг L4 — переворот. В действительности сначала шаг G4 выполняет переворот; но тождество

$$(k{-}j\;k)\phi(k-1) \;=\; \phi(k-1)(j{-}1\;k) \tag{21}$$

показывает, что переворот, за которым следует транспозиция, эквивалентен (другой) транспозиции, за которой следует переворот.

Действительно, уравнение (21) является частным случаем важного тождества

$$\pi^-\,(j_1\;j_2\;\ldots\;j_t)\,\pi \;=\; (j_1\pi\;j_2\pi\;\ldots\;j_t\pi), \tag{22}$$

которое справедливо для *любой* перестановки π и любого t-цикла $(j_1\ j_2\ \ldots\ j_t)$. В левой части (22) мы имеем, например, $j_1\pi \mapsto j_1 \mapsto j_2 \mapsto j_2\pi$ в соответствии с циклом в правой части. Следовательно, если α и π представляют собой произвольные перестановки, то перестановка $\pi^-\alpha\pi$ (называющаяся *сопряжением* α по отношению к π) имеет точно такую же циклическую структуру, как и α; мы просто заменяем каждый элемент j в каждом цикле на $j\pi$.

Еще один важный частный случай алгоритма G рассмотрен Р. Д. Орд-Смитом (R. J. Ord-Smith) [*CACM* **10** (1967), 452; **12** (1969), 638; см. также *Comp. J.* **14** (1971), 136–139], алгоритм которого получается установкой

$$\sigma(k,j) = (k\ \ldots\ 1\ 0)^j. \tag{23}$$

Теперь из (15) ясно, что

$$\tau(k,j) = (k\ \ldots\ 1\ 0); \tag{24}$$

и еще раз мы получаем

$$\omega(k) = (0\ k)(1\ k{-}1)(2\ k{-}2)\ldots = \phi(k), \tag{25}$$

поскольку перестановка $\sigma(k,k) = (0\ 1\ \ldots\ k)$ та же, что и ранее. Интересное в этом методе то, что перестановка, необходимая на шаге G4, а именно $\tau(k,c_k)\,\omega(k-1)^-$, не зависит от c_k:

$$\tau(k,j)\,\omega(k-1)^- = (k\ \ldots\ 1\ 0)\,\phi(k-1)^- = \phi(k). \tag{26}$$

Таким образом, алгоритм Орд-Смита является частным случаем алгоритма G, в котором на шаге G4 просто выполняются обмены $a_0 \leftrightarrow a_k$, $a_1 \leftrightarrow a_{k-1}$, $\ldots$; эта операция обычно очень быстрая, поскольку k мало́, и это экономит часть работы алгоритма L. (См. упр. 38 и ссылку на Г. С. Клюгеля (G. S. Klügel) в разделе 7.2.1.7.)

Можно добиться большего, поступая так, чтобы на шаге G4 каждый раз требовалось выполнять только одну транспозицию, примерно как в алгоритме P, но необязательно со смежными элементами. Возможно множество таких схем. Вероятно, наилучшим является следующий способ — положить

$$\tau(k,j)\,\omega(k-1)^- = \begin{cases} (k\ 0), & \text{если } k \text{ четно},\\ (k\ j{-}1), & \text{если } k \text{ нечетно}, \end{cases} \tag{27}$$

как предложено Б. Р. Хипом (B. R. Heap) [*Comp. J.* **6** (1963), 293–294]. Обратите внимание, что в методе Хипа всегда выполняется транспозиция $a_k \leftrightarrow a_0$, за исключением $k = 3, 5, \ldots$; а значение k в 5 шагах из 6 равно либо 1, либо 2. В упр. 40 доказывается, что метод Хипа действительно генерирует все перестановки.

Пропуск нежелательных блоков. Одним из важных преимуществ алгоритма G является то, что, прежде чем начать работу с a_k, он проходит по всем перестановкам $a_0 \ldots a_{k-1}$; затем перед очередным изменением a_k выполняются другие $k!$ циклов, и т. д. Следовательно, если в некоторый произвольный момент мы дойдем до последних элементов $a_k \ldots a_{n-1}$, которые не представляют важности для решаемой задачи, можно быстро пропустить все перестановки, которые заканчиваются не интересующими нас суффиксами. Точнее говоря, можно заменить шаг G2 следующими подшагами.

G2.0. [Приемлемо?] Если $a_k \ldots a_{n-1}$ — неприемлемый суффикс, перейти к шагу G2.1. В противном случае установить $k \leftarrow k-1$. Затем, если $k > 0$, повторить данный шаг; если $k = 0$, перейти к шагу G2.2.

G2.1. [Пропуск суффикса.] Пока $c_k = k$, применять $\sigma(k,k)^-$ к $a_0 \ldots a_{n-1}$ и устанавливать $c_k \leftarrow 0$, $k \leftarrow k+1$. Завершить работу алгоритма, если $k = n$; в противном случае установить $c_k \leftarrow c_k + 1$, применить $\tau(k, c_k)$ к $a_0 \ldots a_{n-1}$ и вернуться к шагу G2.0.

G2.2. [Посещение.] Посетить перестановку $a_0 \ldots a_{n-1}$. ▌

Шаг G1 должен также устанавливать $k \leftarrow n-1$. Обратите внимание на то, что на новых шагах сохраняется выполнение условия (17). Алгоритм становится более сложным, поскольку теперь в дополнение к перестановкам $\tau(k,j)\omega(k-1)^-$, появляющимся на шаге G4, требуется знать перестановки $\tau(k,j)$ и $\sigma(k,k)$. Однако дополнительные усложнения часто сто́ят затрачиваемых усилий, поскольку получающаяся программа может работать значительно быстрее.

Рис. 40. Нежелательные ветви с дерева на рис. 39 можно обрезать, если соответствующим образом расширить алгоритм G.

Например, на рис. 40 показано, что произойдет с деревом с рис. 39, если рассматривать как неприемлемые суффиксы $a_0a_1a_2a_3$, соответствующие узлам 00, 11, 121 и 2. (Каждый суффикс $a_k \ldots a_{n-1}$ перестановки $a_0 \ldots a_{n-1}$ соответствует *префиксу* $c_n \ldots c_k$ управляющей строки $c_n \ldots c_1$, поскольку перестановки $\sigma(1,c_1) \ldots \sigma(k-1, c_{k-1})$ не влияют на $a_k \ldots a_{n-1}$.) На шаге G2.1 выполняются умножение слева на $\tau(k,j)$ для перехода от узла $c_{n-1} \ldots c_{k+1}(j-1)$ к его правому брату $c_{n-1} \ldots c_{k+1}j$ и умножение слева на $\sigma(k,k)^-$ для перехода от узла $c_{n-1} \ldots c_{k+1}k$ к его родителю $c_{n-1} \ldots c_{k+1}$. Таким образом, чтобы перейти от отвергнутого префикса 121 к его преемнику в прямом порядке обхода, алгоритм выполняет умножение слева на $\sigma(1,1)^-$, $\sigma(2,2)^-$ и $\tau(3,2)$, тем самым переходя от узла 121 к узлу 12 и далее к узлам 1 и 2. (Это в определенной степени исключительный случай, поскольку префикс с $k = 1$ отвергается, только если нам не нужно посещать единственную перестановку $a_0a_1 \ldots a_{n-1}$, имеющую суффикс $a_1 \ldots a_{n-1}$.) После отвержения узла 2 $\tau(3,3)$ перемещает нас к узлу 3, и т. д.

Заметим, кстати, что пропуск суффикса $a_k \ldots a_{n-1}$ в этом расширении алгоритма G, по сути, представляет собой то же самое, что и пропуск префикса $a_1 \ldots a_j$ в наших исходных обозначениях, если вернуться к идее генерации перестановок $a_1 \ldots a_n$ для $\{1, \ldots, n\}$ и выполнять основную часть работы с правого конца. Наши исходные обозначения соответствуют выбору сначала a_1, затем a_2, ..., затем a_n; обозначения в алгоритме G, по сути, сначала выбирает a_{n-1}, затем a_{n-2}, ..., затем a_0. Соглашения алгоритма G могут показаться обратными, но они существенно упрощают формулы для работы с таблицей Симса. Хороший программист быстро научится без труда переходить от одной точки зрения к другой.

Эти идеи можно применить и к буквометикам, поскольку, например, ясно, что бóльшая часть вариантов выбора значений для букв D, E и Y делают невозможным равенство суммы SEND и MORE значению MONEY: в этой задаче требуется выполнение условия $(\mathtt{D}+\mathtt{E}-\mathtt{Y}) \bmod 10 = 0$. Следовательно, множество перестановок может вообще не рассматриваться.

В общем случае, если r_k — максимальная степень 10, которая делит значение сигнатуры s_k, можно сортировать буквы и присваивать коды $\{0,1,\ldots,9\}$ так, чтобы выполнялось условие $r_0 \geq r_1 \geq \cdots \geq r_9$. Например, для решения задачи (7) можно использовать $(0,1,\ldots,9)$ соответственно для $(\mathtt{X},\mathtt{S},\mathtt{V},\mathtt{A},\mathtt{R},\mathtt{I},\mathtt{L},\mathtt{T},\mathtt{O},\mathtt{N})$, получая сигнатуры

$$\begin{aligned} &s_0 = 0, \quad s_1 = -100000, \quad s_2 = 210000, \quad s_3 = -100, \quad s_4 = -100,\\ &s_5 = 21010, \quad s_6 = 210, \quad s_7 = -1010, \quad s_8 = -7901, \quad s_9 = -998; \end{aligned}$$

следовательно, $(r_0,\ldots,r_9) = (\infty,5,4,2,2,1,1,1,0,0)$. Теперь, если обратиться к шагу G2.0 за значением k с условием $r_{k-1} \neq r_k$, то можно сказать, что суффикс $a_k \ldots a_9$ неприемлем, если только $a_k s_k + \cdots + a_9 s_9$ не кратно $10^{r_{k-1}}$. Кроме того, (10) говорит нам, что суффикс $a_k \ldots a_9$ неприемлем, если $a_k = 0$ и $k \in F$; множество первых букв F теперь имеет вид $\{1,2,7\}$.

В нашем предыдущем подходе к буквометикам с шагами A1–A3, изложенном выше, использовался метод грубой силы для прохода по всем 10! вариантам. Он работает весьма быстро, так как метод транспозиции смежных элементов позволяет обойтись только шестью обращениями к памяти на одну перестановку; но 10! все еще равно 3 628 800, так что общая стоимость процесса — почти 22 миллиона обращений к памяти независимо от решаемого буквометика. Расширение же алгоритма G на основе метода Хипа с только что описанными пропусками находит все четыре решения (7) ценой всего лишь 128 *тысяч* обращений к памяти! Таким образом, технология пропуска суффиксов работает в 170 раз быстрее предыдущего метода, слепо перебирающего все варианты.

Большинство из 128 тысяч обращений к памяти в новом подходе тратятся на применение $\tau(k,c_k)$ на шаге G2.1. Другие обращения к памяти связаны с применением на этом шаге $\sigma(k,k)^-$, но τ требуется 7812 раз, в то время как σ^- требуется только 2162 раза. Причину легко понять из рис. 40, поскольку "сокращенное перемещение" $\tau(k,c_k)\,\omega(k-1)^-$ на шаге G4 почти никогда не применяется; в данном случае оно используется только четыре раза, по одному для каждого решения. Таким образом, обход дерева в прямом порядке выполняется почти полностью за τ шагов вправо и σ^- шагов вверх. В задачах наподобие приведенной, где посещается очень мало перестановок, доминируют τ шагов, поскольку каждый шаг $\sigma(k,k)^-$ предваряется k шагами $\tau(k,1)$, $\tau(k,2)$, …, $\tau(k,k)$.

Благодаря этому анализу ясно, что метод Хипа — который идет на все, чтобы оптимизировать перестановки $\tau(k,j)\,\omega(k-1)^-$, так что каждый переход на шаге G4 представляет собой простую транспозицию — *не* слишком хорош для расширенного алгоритма G, если только на шаге G2.0 не отвергается относительно небольшое количество суффиксов. Более простой обратный солексный порядок, для которого сама $\tau(k,j)$ всегда является простой транспозицией, теперь оказывается более привлекательным (см. (19)). Действительно, алгоритм G с обратным солексным порядком решает буквометик (7) ценой лишь 97 тысяч обращений к памяти.

Подобные результаты можно получить и для других буквометиков. Например, если применить расширенный алгоритм G к буквометикам из упр. 24, пп. (а)–(з), то вычисления потребуют соответственно тысяч обращений к памяти при использовании

$$\begin{array}{ll}(551,\ 110,\ 14,\ 8,\ 350,\ 84,\ 153,\ 1598) & \text{метода Хипа;}\\ (429,\ \ 84,\ 10,\ 5,\ 256,\ 63,\ 117,\ 1189) & \text{обратного солексного порядка.}\end{array} \tag{28}$$

"Коэффициент ускорения" для обратного солексного порядка в данных примерах по сравнению с методом грубой силы из алгоритма T находится в диапазоне от 18 в случае (з) до 4200 в случае (г) и составляет в среднем 80; в случае метода Хипа среднее ускорение — около 60.

Однако из алгоритма L известно, что лексикографический порядок легко обрабатывается *без* усложнения управляющей таблицы $c_n \ldots c_1$, используемой алгоритмом G. А более внимательное изучение алгоритма L показывает, что можно улучшить его поведение при частых пропусках перестановок, используя связанный список вместо последовательного массива. Улучшенный алгоритм хорошо подходит для широкого спектра алгоритмов генерации ограниченных классов перестановок.

Алгоритм X (*Лексикографические перестановки с ограниченными префиксами*). Этот алгоритм генерирует все перестановки $a_1 a_2 \ldots a_n$ для $\{1, 2, \ldots, n\}$, которые удовлетворяют заданной последовательности проверок

$$t_1(a_1), \quad t_2(a_1, a_2), \quad \ldots, \quad t_n(a_1, a_2, \ldots, a_n),$$

посещая их в лексикографическом порядке. Алгоритм использует вспомогательную таблицу связей $l_0, l_1, \ldots, l_n$ для поддержания циклического списка неиспользуемых элементов, так что если текущими доступными элементами являются

$$\{1, \ldots, n\} \setminus \{a_1, \ldots, a_k\} = \{b_1, \ldots, b_{n-k}\}, \qquad \text{где } b_1 < \cdots < b_{n-k}, \tag{29}$$

то мы имеем

$$l_0 = b_1, \quad l_{b_j} = b_{j+1} \quad \text{для } 1 \le j < n-k \quad \text{и} \quad l_{b_{n-k}} = 0. \tag{30}$$

Он также использует вспомогательную таблицу $u_1 \ldots u_n$ для отката операций, выполненных над массивом l.

X1. [Инициализация.] Установить $l_k \leftarrow k+1$ для $0 \le k < n$ и $l_n \leftarrow 0$. Затем установить $k \leftarrow 1$.

X2. [Вход на уровень k.] Установить $p \leftarrow 0$, $q \leftarrow l_0$.

X3. [Проверка $a_1 \ldots a_k$.] Установить $a_k \leftarrow q$. Если $t_k(a_1, \ldots, a_k)$ ложно, перейти к шагу X5. В противном случае при $k = n$ посетить $a_1 \ldots a_n$ и перейти к шагу X6.

X4. [Увеличение k.] Установить $u_k \leftarrow p$, $l_p \leftarrow l_q$, $k \leftarrow k+1$ и вернуться к шагу X2.

X5. [Увеличение a_k.] Установить $p \leftarrow q$, $q \leftarrow l_p$. Если $q \ne 0$, вернуться к шагу X3.

X6. [Уменьшение k.] Установить $k \leftarrow k-1$ и завершить работу алгоритма, если $k = 0$. В противном случае установить $p \leftarrow u_k$, $q \leftarrow a_k$, $l_p \leftarrow q$ и перейти к шагу X5. ▮

Этот элегантный алгоритм основан на идее, принадлежащей М. Ч. Эру (M. C. Er) [*Comp. J.* **30** (1987), 282]. Мы можем применить его для решения буквометиков,

слегка изменив систему обозначений, получая перестановки $a_0 \ldots a_9$ для $\{0, \ldots, 9\}$ и позволяя l_{10} играть прежнюю роль l_0. Получающийся в результате алгоритм требует только 49 тысяч обращений к памяти для решения буквометика (7), а буквометики из упр. 24, (а)–(з) он решает с использованием

$$(248,\ 38,\ 4,\ 3,\ 122,\ 30,\ 55,\ 553), \tag{31}$$

тысяч обращений к памяти соответственно. Таким образом, он работает примерно в 165 раз быстрее алгоритма, основанного на грубой силе.

Еще один способ применения алгоритма X для решения буквометиков зачастую оказывается еще более быстрым (см. упр. 49).

Рис. 41. Дерево, которое неявно обходит алгоритм X для $n = 4$, если посещаются все перестановки, за исключением начинающихся с 132, 14, 2, 314 или 4312.

***Дуальные методы.** Если $S_1, \ldots, S_{n-1}$ представляют собой таблицу Симса для группы перестановок G, то, как известно из леммы S, каждый элемент G можно единственным способом выразить в виде произведения $\sigma_1 \ldots \sigma_{n-1}$, где $\sigma_k \in S_k$; см. (13). В упр. 50 показано, что каждый элемент α также может быть выражен единственным способом в дуальном виде

$$\alpha = \sigma_{n-1}^- \ldots \sigma_2^- \sigma_1^-, \qquad \text{где } \sigma_k \in S_k \text{ для } 1 \le k < n, \tag{32}$$

и этот факт приводит к другому большому семейству генераторов перестановок. В частности, когда G является группой всех $n!$ перестановок, каждая перестановка может быть записана в виде

$$\sigma(n-1, c_{n-1})^- \ldots \sigma(2, c_2)^- \sigma(1, c_1)^-, \tag{33}$$

где $0 \le c_k \le k$ для $1 \le k < n$, а перестановки $\sigma(k, j)$ — те же, что и в алгоритме G. Однако теперь мы хотим варьировать c_{n-1} более быстро, а c_1 — менее быстро, так что получается алгоритм иного типа.

Алгоритм H (*Дуальный генератор перестановок*). Данный алгоритм для заданной таблицы Симса, как и в алгоритме G, генерирует все перестановки $a_0 \ldots a_{n-1}$ множества $\{0, \ldots, n-1\}$, используя вспомогательную таблицу $c_0 \ldots c_{n-1}$.

H1. [Инициализация.] Установить $a_j \leftarrow j$ и $c_j \leftarrow 0$ для $0 \le j < n$.

H2. [Посещение.] (В этот момент число со смешанным основанием $\left[\begin{smallmatrix} c_1, & c_2, & \ldots, & c_{n-1} \\ 2, & 3, & \ldots, & n \end{smallmatrix}\right]$ представляет собой количество посещенных до сих пор перестановок.) Посетить перестановку $a_0 a_1 \ldots a_{n-1}$.

H3. [Прибавление 1 к $c_0 c_1 \ldots c_{n-1}$.] Установить $k \leftarrow n-1$. Если $c_k = k$, установить $c_k \leftarrow 0$, $k \leftarrow k-1$ и повторять эти действия до тех пор, пока не будет выполнено

условие $k = 0$ или $c_k < k$. Завершить работу алгоритма, если $k = 0$; в противном случае установить $c_k \leftarrow c_k + 1$.

H4. [Перестановка.] Применить перестановку $\tau(k, c_k)\omega(k+1)^-$ к $a_0 a_1 \ldots a_{n-1}$, как поясняется ниже, и вернуться к шагу H2. ▌

Хотя этот алгоритм выглядит практически идентично алгоритму G, перестановки τ и ω, требующиеся на шаге H4, существенно отличаются от перестановок, требующихся на шаге G4. Новыми правилами, заменяющими (15) и (16), являются

$$\tau(k,j) = \sigma(k,j)^-\sigma(k,j-1) \qquad \text{для } 1 \le j \le k; \tag{34}$$

$$\omega(k) = \sigma(n-1,n-1)^-\sigma(n-2,n-2)^- \ldots \sigma(k,k)^-. \tag{35}$$

Количество вариантов так же велико, как и в случае алгоритма G, так что здесь мы ограничимся только небольшим количеством случаев, обладающих особыми достоинствами. Одним из таких случаев, естественно, является таблица Симса, которая заставляла алгоритм G генерировать перестановки в обратном солексном порядке, а именно

$$\sigma(k,j) \;=\; (k{-}j \;\; k{-}j{+}1 \;\; \ldots \;\; k), \tag{36}$$

как в (18). Получающийся в результате генератор перестановок оказывается очень похожим на метод простых изменений; так что можно сказать, что алгоритмы L и P, по сути, дуальны друг другу. (См. упр. 52.)

Другая естественная идея заключается в построении таблицы Симса, для которой на шаге H4 всегда выполняется одна транспозиция двух элементов, по аналогии с построением (27), достигающим максимальной эффективности шага G4. Но сейчас это невозможно: мы не можем достичь этого даже для $n = 4$. Если начать с тождественной перестановки $a_0 a_1 a_2 a_3 = 0123$, то переходы управляющей таблицы от $c_0 c_1 c_2 c_3 = 0000$ к 0001, к 0002, к 0003 должны перемещать 3; так что, если они являются транспозициями, то они должны иметь вид $(3\,a)$, $(a\,b)$ и $(b\,c)$ для некоторой перестановки abc множества $\{0,1,2\}$. Перестановка, соответствующая $c_0 c_1 c_2 c_3 = 0003$, теперь имеет вид $\sigma(3,3)^- = (b\,c)(a\,b)(3\,a) = (3\,a\,b\,c)$; а следующая перестановка, которая соответствует $c_0 c_1 c_2 c_3 = 0010$, будет $\sigma(2,1)^-$, которая должна фиксировать элемент 3. Единственная подходящая транспозиция — $(3\,c)$, следовательно, $\sigma(2,1)^-$ должна быть $(3\,c)(3\,a\,b\,c) = (a\,b\,c)$. Аналогично мы находим, что $\sigma(2,2)^-$ должна быть $(a\,c\,b)$, а перестановкой, соответствующей $c_0 c_1 c_2 c_3 = 0023$, будет $(3\,a\,b\,c)(a\,c\,b) = (3\,c)$. Теперь шаг H4, как предполагается, преобразует ее в перестановку $\sigma(1,1)^-$, которая соответствует управляющей таблице 0100, следующей за 0023. Но единственной транспозицией, преобразующей $(3\,c)$ в перестановку, которая фиксирует 2 и 3, является $(3\,c)$; а полученная в результате перестановка фиксирует также 1, так что это не может быть $\sigma(1,1)^-$.

Доказательство в предыдущем абзаце показывает, что алгоритм H нельзя использовать для генерации всех перестановок с помощью минимального количества транспозиций. Но из него также следует простая схема генерации, которая очень близка к этому минимуму, а получающийся в результате алгоритм очень привлекателен, поскольку требует выполнения дополнительной работы только один раз на $n(n-1)$ шагов (см. упр. 53.)

Наконец рассмотрим метод, дуальный методу Орд-Смита, когда

$$\sigma(k,j) = (k \;\; \ldots \;\; 1 \;\; 0)^j \tag{37}$$

как в (23). Значение $\tau(k,j)$ вновь не зависит от j,

$$\tau(k,j) = (0\ 1\ \ldots\ k), \tag{38}$$

и этот факт особенно благоприятен для алгоритма H, поскольку позволяет обойтись без управляющей таблицы $c_0 c_1 \ldots c_{n-1}$. Дело в том, что вследствие (32) на шаге H3 $c_{n-1} = 0$ тогда и только тогда, когда $a_{n-1} = n-1$; и действительно, когда $c_j = 0$ для $k < j < n$ на шаге H3, мы получаем $c_k = 0$ тогда и только тогда, когда $a_k = k$. Следовательно, можно переформулировать данный вариант алгоритма H следующим образом.

Алгоритм C (*Генерация перестановок циклическими сдвигами*). Этот алгоритм посещает все перестановки $a_1 \ldots a_n$ различных элементов $\{x_1, \ldots, x_n\}$.

C1. [Инициализация.] Установить $a_j \leftarrow x_j$ для $1 \le j \le n$.

C2. [Посещение.] Посетить перестановку $a_1 \ldots a_n$ и установить $k \leftarrow n$.

C3. [Сдвиг.] Заменить $a_1 a_2 \ldots a_k$ циклическим сдвигом $a_2 \ldots a_k a_1$ и вернуться к шагу C2, если $a_k \ne x_k$.

C4. [Уменьшение k.] Установить $k \leftarrow k-1$ и вернуться к шагу C3, если $k > 1$. ▮

Например, последовательные перестановки $\{1,2,3,4\}$, генерируемые при $n = 4$, имеют вид

1234, 2341, 3412, 4123, (1234),
2314, 3142, 1423, 4231, (2314),
3124, 1243, 2431, 4312, (3124), (1234),
2134, 1342, 3421, 4213, (2134),
1324, 3241, 2413, 4132, (1324),
3214, 2143, 1432, 4321, (3214), (2134), (1234),

где в скобках показаны непосещенные промежуточные перестановки. Этот алгоритм может оказаться самым простым генератором перестановок с точки зрения минимального размера кода программы. Он предложен Г. Д. Лангдоном-мл. (G. G. Langdon, Jr.) [*CACM* **10** (1967), 298–299; **11** (1968), 392]; аналогичный метод был ранее опубликован Ч. Томпкинсом (C. Tompkins) [*Proc. Symp. Applied Math.* **6** (1956), 202–205] и, в более явном виде, Р. Зайтцем (R. Seitz) [*Unternehmensforschung* **6** (1962), 2–15]. Эта процедура в особенности хороша для приложений, в которых эффективна операция циклического сдвига, например, когда последовательные перестановки хранятся в регистре компьютера, а не в массиве.

Главным недостатком дуальных методов является то, что они обычно плохо адаптируются к ситуациям, когда нужно пропустить большие блоки перестановок, поскольку множество всех перестановок с заданным значением первых управляющих элементов $c_0 c_1 \ldots c_{k-1}$ обычно не имеет большого значения. Однако частный случай (36) иногда является исключением, поскольку $n!/k!$ перестановок с $c_0 c_1 \ldots c_{k-1} = 00\ldots0$ в этом случае представляют собой именно те перестановки $a_0 a_1 \ldots a_{n-1}$, в которых 0 предшествует 1, 1 предшествует 2, …, а $k-2$ предшествует $k-1$.

***Метод обмена Эрлиха.** Гидеон Эрлих (Gideon Ehrlich) открыл совершенно иной подход к генерации перестановок, основанный на еще одном способе использования управляющей таблицы $c_1 \ldots c_{n-1}$. Его метод получает каждую перестановку из предыдущей путем обмена крайнего слева элемента с другим.

Алгоритм E (*Обмены Эрлиха*). Этот алгоритм генерирует все перестановки различных элементов $a_0 \ldots a_{n-1}$ с использованием вспомогательных таблиц $b_0 \ldots b_{n-1}$ и $c_1 \ldots c_n$.

E1. [Инициализация.] Установить $b_j \leftarrow j$ и $c_{j+1} \leftarrow 0$ для $0 \le j < n$.

E2. [Посещение.] Посетить перестановку $a_0 \ldots a_{n-1}$.

E3. [Поиск k.] Установить $k \leftarrow 1$. Затем, пока $c_k = k$, устанавливать $c_k \leftarrow 0$ и $k \leftarrow k + 1$. Завершить работу алгоритма, если $k = n$, в противном случае установить $c_k \leftarrow c_k + 1$.

E4. [Обмен.] Выполнить обмен $a_0 \leftrightarrow a_{b_k}$.

E5. [Переворот.] Установить $j \leftarrow 1$, $k \leftarrow k - 1$. Пока $j < k$, обменивать $b_j \leftrightarrow b_k$ и устанавливать $j \leftarrow j + 1$, $k \leftarrow k - 1$. Вернуться к шагу E2. ▮

Обратите внимание на то, что шаги E2 и E3 идентичны шагам G2 и G3 алгоритма G. Наиболее удивительное в этом алгоритме, как писал Эрлих Мартину Гарднеру в 1987 году, то, что он работает; доказательство этого приведено в упр. 55. Аналогичный метод, который упрощает операции на шаге E5, может быть проверен таким же образом (см. упр. 56). Среднее количество обменов, выполняемое алгоритмом на шаге E5, меньше 0.18 (см. упр. 57).

Алгоритм E не быстрее других рассматривавшихся нами методов. Но он обладает тем приятным свойством, что минимально изменяет каждую перестановку, используя только $n - 1$ различных транспозиций. В то время как алгоритм P использует обмены смежных элементов $a_{t-1} \leftrightarrow a_t$, алгоритм E обменивает с другими первый элемент $a_0 \leftrightarrow a_t$ (иногда такой обмен называют *звездной транспозицией* (*star ranspositions*)), используя для этого некоторую отобранную последовательность индексов $t[1]$, $t[2]$, $\ldots$, $t[n! - 1]$. И если требуется повторно генерировать перестановки для некоторого достаточно небольшого значения n, можно предвычислить эту последовательность (как это делалось в алгоритме T для последовательности индексов алгоритма P). Заметим также, что звездные транспозиции обладают по сравнению со смежными обменами тем преимуществом, что обмениваемое значение a_0 известно из предыдущего обмена, а значит, его не требуется считывать из памяти.

Пусть E_n — последовательность $n! - 1$ индексов t, таких, что алгоритм E на шаге E4 обменивает a_0 с a_t. Поскольку E_{n+1} начинается с E_n, можно рассматривать E_n как первые $n! - 1$ элементов бесконечной последовательности

$$E_\infty = 1212132121231212132121243131321313 12\ldots. \tag{39}$$

Например, если $n = 4$ и $a_0 a_1 a_2 a_3 = 1234$, то перестановки, посещенные алгоритмом E, имеют вид

$$\begin{aligned} &1234,\ 2134,\ 3124,\ 1324,\ 2314,\ 3214,\\ &4213,\ 1243,\ 2143,\ 4123,\ 1423,\ 2413,\\ &3412,\ 4312,\ 1342,\ 3142,\ 4132,\ 1432,\\ &2431,\ 3421,\ 4321,\ 2341,\ 3241,\ 4231. \end{aligned} \tag{40}$$

***Меньшие генераторы.** После знакомства с алгоритмами P и E вполне естественно было бы спросить, а не могут ли все перестановки быть получены путем использования только *двух* базовых операций вместо $n - 1$? Например, Ньенхьюз

(Nijenhuis) и Вильф (Wilf) [*Combinatorial Algorithms* (1975), Exercise 6] заметили, что все перестановки для $n = 4$ можно сгенерировать путем замены на каждом шаге $a_1a_2a_3\ldots a_n$ либо на $a_2a_3\ldots a_na_1$, либо на $a_2a_1a_3\ldots a_n$, и заинтересовались вопросом, существует ли такой метод для всех n.

В общем случае, если G — произвольная группа перестановок и если $\alpha_1, \ldots, \alpha_k$ являются элементами G, то *граф Кейли* для G с генераторами $(\alpha_1, \ldots, \alpha_k)$ представляет собой ориентированный граф, вершины которого являются перестановками π из G, а дуги идут от π к $\alpha_1\pi, \ldots, \alpha_k\pi$. [Arthur Cayley, *American J. Math.* **1** (1878), 174–176.] Вопрос Ньенхьюза и Вильфа эквивалентен вопросу о существовании гамильтонова пути в графе Кейли для всех перестановок $\{1, 2, \ldots, n\}$ с генераторами σ и τ, где σ — циклическая перестановка $(1\ 2\ \ldots\ n)$, а τ — транспозиция $(1\ 2)$.

Основная теорема Р. А. Ранкина (R. A. Rankin) [*Proc. Cambridge Philos. Soc.* **44** (1948), 17–25] позволяет нам сделать вывод о том, что во многих случаях графы Кейли с двумя генераторами не имеют гамильтонова цикла.

Теорема R. *Пусть G — группа из g перестановок. Если граф Кейли для G с генераторами (α, β) имеет гамильтонов цикл и если перестановки $(\alpha, \beta, \alpha\beta^-)$ имеют соответственно порядок (a, b, c), то либо c четно, либо g/a и g/b нечетны.*

(*Порядком* перестановки α называется наименьшее положительное целое число a, такое, что α^a является тождественной перестановкой.)

Доказательство. См. упр. 73. ▮

В частности, когда $\alpha = \sigma$ и $\beta = \tau$, как в случае выше, $g = n!$, $a = n$, $b = 2$ и $c = n-1$, поскольку $\sigma\tau^- = (2\ \ldots\ n)$. Следовательно, можно заключить, что гамильтонов цикл невозможен для четных $n \geq 4$. Однако легко построить гамильтонов *путь* для $n = 4$, поскольку можно соединить 12-шаговые циклы

$$\begin{aligned}
&1234 \to 2341 \to 3412 \to 4312 \to 3124 \to 1243 \to 2431 \\
&\qquad\qquad \to 4231 \to 2314 \to 3142 \to 1423 \to 4123 \to 1234, \\
&2134 \to 1342 \to 3421 \to 4321 \to 3214 \to 2143 \to 1432 \\
&\qquad\qquad \to 4132 \to 1324 \to 3241 \to 2413 \to 4213 \to 2134,
\end{aligned} \tag{41}$$

начиная с 2341 и переходя от 1234 к 2134, и заканчивая 4213.

Раски (Ruskey), Джанг (Jiang) и Вестон (Weston) [*Discrete Applied Math.* **57** (1995), 75–83] предприняли исчерпывающий поиск в σ–τ графе для $n = 5$ и выяснили, что он имеет пять существенно различных гамильтоновых циклов, один из которых ("самый красивый") показан на рис. 42, (a). Они также обнаружили гамильтонов путь для $n = 6$; это оказалось очень сложной задачей, так как потребовало бинарного дерева принятия решений с 720 шагами. К сожалению, открытое ими решение не имеет видимой логической структуры. Немного менее сложный путь описан в упр. 70, но даже этот путь нельзя назвать простым. Таким образом, подход на основе σ–τ графа, вероятно, не будет иметь практического интереса для бóльших значений n, если только не будут открыты какие-либо новые походы. Р. К. Комптон (R. C. Compton) и С. Д. Вильямсон (S. G. Williamson) [*Linear and Multilinear Algebra* **35** (1993), 237–293] доказали, что если допустить применение трех генераторов σ, σ^- и τ, то гамильтоновы циклы существуют для всех n. Их циклы обладают тем

интересным свойством, что каждое n-е преобразование является τ, а промежуточные $n-1$ преобразований являются преобразованиями либо только σ, либо только σ^{-}. Однако их метод слишком сложен, чтобы можно было кратко рассказать о нем в этой книге.

В упр. 69 описан общий алгоритм перестановок, достаточно простой и требующий только трех генераторов, каждый — второго порядка. На рис. 42, (б) этот метод проиллюстрирован для случая $n = 5$, побудительным мотивом для которого послужили ранее упоминавшиеся перезвоны пяти колоколов.

(а) Используются только переходы (1 2 3 4 5) и (1 2).

(б) Используются только переходы (1 2)(3 4), (2 3)(4 5) и (3 4).

Рис. 42. Гамильтоновы циклы для 5! перестановок.

Быстрее, еще быстрее. Но какой же способ генерации перестановок самый быстрый? Этот вопрос часто возникает в публикациях, посвященных информатике, поскольку для перебора всех $n!$ перестановок желательно сократить время работы как можно сильнее. Но в общем ответы оказываются противоречивыми, поскольку имеется много способов сформулировать вопрос. Давайте попытаемся разобраться в данном вопросе, изучая, как наиболее быстро генерировать перестановки на компьютере MMIX.

Сначала положим, что наша цель заключается в получении перестановок в массиве n последовательно расположенных в памяти слов (октабайтов). Наиболее быстрый способ сделать это среди всех рассмотренных в данном разделе, как подсказывает Р. Седжвик (R. Sedgewick) [*Computing Surveys* **9** (1977), 157–160], — применить оптимизированный метод Хипа (27).

Ключевая идея заключается в оптимизации кода для наиболее распространенных случаев на шагах G2 и G3, а именно для случаев, в которых вся деятельность сосредоточивается в начале массива. Если регистры u, v и w содержат первые три слова и если очередные шесть генерируемых перестановок получаются путем перестановок данных слов всеми шестью возможными способами, то можно выполнить эту работу следующим образом.

```
PUSHJ 0,Visit
STO v,A0;  STO u,A1;  PUSHJ 0,Visit
STO w,A0;  STO v,A2;  PUSHJ 0,Visit
STO u,A0;  STO w,A1;  PUSHJ 0,Visit
STO v,A0;  STO u,A2;  PUSHJ 0,Visit
STO w,A0;  STO v,A1;  PUSHJ 0,Visit
```
(42)

(Здесь A0 — адрес октабайта a_0, и т. д.) Полная программа, которая отвечает за корректное заполнение регистров u, v и w, приводится в упр. 77, но прочие инструк-

ции менее важны, так как выполняются только $\frac{1}{6}$ всего времени работы. Общая стоимость одной перестановки, без учета $4v$, необходимых для выполнения операций PUSHJ и POP при каждом вызове Visit, при таком подходе доходит приблизительно до $2.77\mu+5.69v$. Если использовать четыре регистра, u, v, w, x, и расширить (42) до 24 вызовов Visit, то время работы на одну перестановку сокращается до примерно $2.19\mu + 3.07v$. А в случае r регистров и $r!$ вызовов Visit, как показано в упр. 78, стоимость оказывается равной $(2 + O(1/r!))(\mu + v)$, т. е. очень близкой к стоимости двух инструкций STO.

Последнее значение, конечно, представляет собой минимально возможное время для любого метода генерации всех перестановок в последовательный массив... Или нет? Мы предполагали, что процедура посещения просматривает перестановки в последовательных местоположениях массива, но, возможно, эта процедура способна считывать перестановки из различных стартовых точек. Тогда можно организовать работу таким образом, чтобы зафиксировать a_{n-1} и хранить две копии других элементов поблизости:

$$a_0 a_1 \ldots a_{n-2} a_{n-1} a_0 a_1 \ldots a_{n-2}. \tag{43}$$

Если теперь позволить $a_0 a_1 \ldots a_{n-2}$ пробегать $(n-1)!$ перестановок, всегда изменяя обе копии одновременно с помощью двух команд STO вместо одной, можно обеспечить посещение каждым вызовом Visit не одной, а n последовательно размещающихся перестановок

$$a_0 a_1 \ldots a_{n-1}, \quad a_1 \ldots a_{n-1} a_0, \quad \ldots, \quad a_{n-1} a_0 \ldots a_{n-2}. \tag{44}$$

Стоимость одной перестановки при этом снижается до стоимости трех простых инструкций наподобие ADD, CMP, PBNZ, плюс $O(1/n)$. [См. Varol and Rotem, *Comp. J.* **24** (1981), 173–176.]

Кроме того, можно вообще не тратить время на сохранение перестановок в памяти. Предположим, например, что наша цель заключается в генерации всех перестановок $\{0, 1, \ldots, n-1\}$. Значение n вряд ли превысит 16, поскольку $16! = 20\,922\,789\,888\,000$, а $17! = 355\,687\,428\,096\,000$. Следовательно, вся перестановка может быть размещена в 16 полубайтах октабайта, и ее можно хранить в одном регистре. Преимущества этого способа хранения проявляются только в том случае, если подпрограмме посещения не требуется распаковывать отдельные полубайты; но пусть это и в самом деле так. Насколько же быстро можно генерировать перестановки в полубайтах 64-битового регистра?

Одна идея, основанная на предложении А. Д. Голдштейна (A. J. Goldstein) [*U.S. Patent 3383661* (14 May 1968)], состоит в предвычислении таблицы $(t[1], \ldots, t[5039])$ переходов простых изменений для семи элементов с использованием алгоритма T. Эти числа $t[k]$ находятся между 1 и 6, так что можно упаковать 20 таких чисел в одно 64-битовое слово. Оказывается удобным поместить число $\sum_{k=1}^{20} 2^{3k-1} t[20j+k]$ в слово j вспомогательной таблицы $(0 \le j < 252)$ и считать $t[5040] = 1$; например, таблица начинается с кодового слова

00|001|010|011|100|101|110|100|110|101|100|011|010|001|110|001|010|011|100|101|110|00.

Приведенная далее программа эффективно считывает эти коды.

```
Perm  ⟨Установить регистр a равным первой перестановке.⟩
0H    LDA  p,T        p ← адрес первого кодового слова.
      JMP  3F
1H    ⟨Посетить перестановку в регистре a.⟩
      ⟨Обмен полубайтов a, лежащих в t битах справа.⟩
      SRU  c,c,3      c ← c ≫ 3.
2H    AND  t,c,#1c    t ← c & (11100)_2.                    (45)
      PBNZ t,1B       Ветвление, если t ≠ 0.
      ADD  p,p,8
3H    LDO  c,p,0      c ← следующее кодовое слово.
      PBNZ c,2B       (За последним кодовым словом идет 0.)
      ⟨Если не конец, продвинуть ведущие n − 7
       полубайтов и вернуться к шагу 0B.⟩
```

В упр. 79 показано, как операция ⟨Обмен полубайтов...⟩ выполняется с помощью семи инструкций, с помощью побитовых операций, предусмотренных на большинстве компьютеров. Следовательно, стоимость одной перестановки чуть больше $10v$. (Стоимость инструкций для выборки новых кодовых слов равна всего лишь $(\mu + 5v)/20$; а стоимость инструкций продвижения $n - 7$ ведущих полубайтов пренебрежимо мала, так как делится на 5040.) Заметим, что сейчас нет необходимости в инструкциях PUSHJ и POP, как в (42); ранее мы их игнорировали, но их стоимость равна $4v$.

Однако можно получить еще лучший результат, если адаптировать метод циклического сдвига Лангдона, алгоритм C. Предположим, мы начинаем с лексикографически наибольшей перестановки и действуем следующим образом.

```
      GREG @
0H    OCTA #fedcba9876543210&(1<<(4*N)-1)
Perm  LDOU a,0B                          Установить a ← #...3210.
      JMP  2F
1H    SRU  a,a,4*(16-N)                  a ← ⌊a/16^(16−n)⌋.
      OR   a,a,t                         a ← a | t.            (46)
2H    ⟨Посещение перестановки в регистре a.⟩
      SRU  t,a,4*(N-1)                   t ← ⌊a/16^(n−1)⌋.
      SLU  a,a,4*(17-N)                  a ← 16^(17−n) a mod 16^16.
      PBNZ t,1B                          В 1B, если t ≠ 0.
      ⟨Продолжение работы методом Лангдона.⟩
```

Время работы на одну перестановку теперь равно всего лишь $5v + O(1/n)$, вновь без необходимости использовать PUSHJ и POP. В упр. 81 описан интересный способ расширения (46) до полной программы, что дает в результате замечательно короткую и быструю процедуру.

Быстрые генераторы перестановок интересны и занимательны, но на практике можно сэкономить больше времени, оптимизировав процедуру посещения, а не ускорив генератор.

Топологическая сортировка. Вместо обработки всех $n!$ перестановок $\{1, \ldots, n\}$ зачастую требуется просмотреть только те перестановки, которые удовлетворяют

определенным ограничениям. Например, нас могут интересовать только те перестановки, в которых 1 предшествует 3, 2 предшествует 3 и 2 предшествует 4; всего имеется пять таких перестановок множества $\{1,2,3,4\}$, а именно

$$1234,\ 1243,\ 2134,\ 2143,\ 2413. \tag{47}$$

Задача *топологической сортировки*, которую мы рассматривали в разделе 2.2.3 как первый пример нетривиальной структуры данных, является общей задачей поиска перестановки, которая удовлетворяет m условиям $x_1 \prec y_1, \ldots, x_m \prec y_m$, где $x \prec y$ означает, что x в перестановке должно предшествовать y. Эта задача часто возникает на практике, а потому имеет несколько названий; например, ее часто называют задачей *линейного вложения*, поскольку объекты следует выстроить в линию с сохранением определенных отношений упорядочения. Ее также называют задачей расширения частичного упорядочения до полного упорядочения (см. упр. 2.2.3–14).

Нашей целью в разделе 2.2.3 был поиск *одной* перестановки, удовлетворяющей всем соотношениям. Но теперь мы хотим найти *все* такие перестановки, т. е. все топологические сортировки. В этом разделе мы будем считать, что элементы x и y, для которых определены отношения, являются целыми числами от 1 до n и что, когда $x \prec y$, мы имеем $x < y$. Следовательно, перестановка $12\ldots n$ всегда топологически корректна. (Если это упрощенное предположение не удовлетворяется, мы можем предварительно обработать объекты с помощью алгоритма 2.2.3T, чтобы затем соответствующим образом их переименовать.)

Многие важные классы перестановок являются частными случаями этой задачи топологического упорядочения. Например, перестановки $\{1,\ldots,8\}$, такие, что

$$1 \prec 2, \quad 2 \prec 3, \quad 3 \prec 4, \quad 6 \prec 7, \quad 7 \prec 8,$$

эквивалентны перестановкам мультимножества $\{1,1,1,1,2,3,3,3\}$, поскольку можно отобразить $\{1,2,3,4\} \mapsto 1$, $5 \mapsto 2$ и $\{6,7,8\} \mapsto 3$. Мы знаем, как генерировать перестановки мультимножества с помощью алгоритма L, но сейчас мы изучим другой способ.

Обратите внимание, что x в перестановке $a_1 \ldots a_n$ предшествует y тогда и только тогда, когда $a'_x < a'_y$ в обратной перестановке $a'_1 \ldots a'_n$. Следовательно, рассматриваемый нами алгоритм находит также все перестановки $a'_1 \ldots a'_n$, такие, что $a'_j < a'_k$ всякий раз, когда $j \prec k$. Например, из раздела 5.1.4 нам известно, что таблица Юнга представляет собой такое упорядочение $\{1,\ldots,n\}$ по строкам и столбцам, что каждая строка возрастает слева направо, а каждый столбец возрастает сверху вниз. Таким образом, задача генерации всех таблиц Юнга размером 3×3 эквивалентна генерации всех $a'_1 \ldots a'_9$, таких, что

$$\begin{aligned} &a'_1 < a'_2 < a'_3, \quad a'_4 < a'_5 < a'_6, \quad a'_7 < a'_8 < a'_9, \\ &a'_1 < a'_4 < a'_7, \quad a'_2 < a'_5 < a'_8, \quad a'_3 < a'_6 < a'_9, \end{aligned} \tag{48}$$

и представляет собой особый вид топологической сортировки.

Можно также искать все *совершенные паросочетания* $2n$ элементов, т. е. все способы разбиения $\{1,\ldots,2n\}$ на n пар. Имеется $(2n-1)(2n-3)\ldots(1) = (2n)!/(2^n n!)$ таких способов, которые соответствуют перестановкам, удовлетворяющим условию

$$a'_1 < a'_2, \quad a'_3 < a'_4, \quad \ldots, \quad a'_{2n-1} < a'_{2n}, \qquad a'_1 < a'_3 < \cdots < a'_{2n-1}. \tag{49}$$

Элегантный алгоритм для исчерпывающей топологической сортировки был открыт Я. Л. Варолом (Y. L. Varol) и Д. Ротемом (D. Rotem) [*Comp. J.* **24** (1981), 83–84], которые сообразили, что можно использовать метод, аналогичный простым изменениям (алгоритм P). Предположим, что мы нашли способ топологически упорядочить $\{1, \ldots, n-1\}$ так, что $a_1 \ldots a_{n-1}$ удовлетворяет всем условиям, которые не включают n. Тогда можно легко записать все допустимые способы вставки последнего элемента n без изменения относительного порядка $a_1 \ldots a_{n-1}$: мы просто начинаем с $a_1 \ldots a_{n-1} n$, затем смещаем n по одному шагу влево, пока такой сдвиг не станет невозможным. Рекурсивное применение этой идеи дает следующую простую процедуру.

Алгоритм V (*Все топологические сортировки*). Для заданного отношения $\prec$ на множестве $\{1, \ldots, n\}$ с тем свойством, что из $x \prec y$ следует $x < y$, этот алгоритм генерирует все перестановки $a_1 \ldots a_n$ и обратные к ним $a'_1 \ldots a'_n$, обладающие тем свойством, что $a'_j < a'_k$ при $j \prec k$. Для удобства предполагаем, что $a_0 = a'_0 = 0$ и что $0 \prec k$ для $1 \le k \le n$.

V1. [Инициализация.] Установить $a_j \leftarrow j$ и $a'_j \leftarrow j$ для $0 \le j \le n$.

V2. [Посещение.] Посетить перестановку $a_1 \ldots a_n$ и обратную к ней $a'_1 \ldots a'_n$. Затем установить $k \leftarrow n$.

V3. [Можно переместить k влево?] Установить $j \leftarrow a'_k$ и $l \leftarrow a_{j-1}$. Если $l \prec k$, перейти к шагу V5.

V4. [Да, перемещение.] Установить $a_{j-1} \leftarrow k$, $a_j \leftarrow l$, $a'_k \leftarrow j-1$ и $a'_l \leftarrow j$. Перейти к шагу V2.

V5. [Нет, вернуть k.] Пока $j < k$, устанавливать $l \leftarrow a_{j+1}$, $a_j \leftarrow l$, $a'_l \leftarrow j$ и $j \leftarrow j+1$. Затем установить $a_k \leftarrow a'_k \leftarrow k$. Уменьшить k на 1 и вернуться к шагу V3, если $k > 0$. ▮

Например, теорема 5.1.4H гласит, что имеется ровно 42 таблицы Юнга размером 3×3. Если применить алгоритм V к отношениям (48) и записать обратную перестановку в виде массива

$$\boxed{\begin{array}{l} a'_1 a'_2 a'_3 \\ a'_4 a'_5 a'_6 \\ a'_7 a'_8 a'_9 \end{array}}, \tag{50}$$

то получим следующие 42 результата.

123 456 789	123 457 689	123 458 679	123 467 589	123 468 579	124 356 789	124 357 689	124 358 679	124 367 589	124 368 579	125 367 489	125 368 479	125 346 789	125 347 689

125 348 679	126 347 589	126 348 579	127 348 569	126 357 489	126 358 479	127 358 469	134 256 789	134 257 689	134 258 679	134 267 589	134 268 579	135 267 489	135 268 479

145 267 389	145 268 379	135 246 789	135 247 689	135 248 679	136 247 589	136 248 579	137 248 569	136 257 489	136 258 479	137 258 469	146 257 389	146 258 379	147 258 369

Пусть t_r — количество топологических сортировок, последние $n-r$ элементов которых находятся в их исходном положении $a_j = j$ для $r < j \le n$. Эквивалентно t_r является количеством топологических сортировок $a_1 \ldots a_r$ множества $\{1, \ldots, r\}$, если игнорировать отношения, затрагивающие элементы, бóльшие r. Тогда рекурсивный механизм, лежащий в основе алгоритма V, показывает, что шаг V2 выполняется N раз, а шаг V3 — M раз, где

$$M = t_n + \cdots + t_1 \qquad \text{и} \qquad N = t_n. \tag{51}$$

Кроме того, шаг V4 и операции цикла на шаге V5 выполняются $N-1$ раз; остальные операции шага V5 выполняются $M-N+1$ раз. Следовательно, общее время работы алгоритма представляет собой линейную комбинацию M, N и n.

При плохом выборе меток элементов M может оказаться гораздо больше N. Например, если ограничениями на входе алгоритма V являются

$$2 \prec 3, \quad 3 \prec 4, \quad \ldots, \quad n-1 \prec n, \tag{52}$$

то $t_j = j$ для $1 \le j \le n$, и мы имеем $M = \frac{1}{2}(n^2+n)$, $N = n$. Но эти ограничения после переименования элементов эквивалентны ограничениям

$$1 \prec 2, \quad 2 \prec 3, \quad \ldots, \quad n-2 \prec n-1; \tag{53}$$

M при этом сводится к $2n-1 = 2N-1$.

В упр. 89 показано, что простой предварительный шаг может найти такие метки элементов, что небольшая модификация алгоритма V оказывается способной генерировать все топологические сортировки за $O(N+n)$ шагов. Таким образом, топологическая сортировка всегда может быть выполнена эффективно.

Семь раз отмерь. В этом разделе мы познакомились с несколькими привлекательными алгоритмами генерации перестановок, но имеется множество алгоритмов, способных генерировать оптимальные перестановки для конкретных задач *без* полного перебора. Например, как показывает теорема 6.1S, можно найти наилучший способ упорядочения записей в последовательном хранилище просто путем сортировки в соответствии с определенными критериями стоимости, и этот процесс требует только $O(n \log n)$ шагов. В разделе 7.5.2 будет рассмотрена *задача о назначениях*, в которой требуется найти перестановку столбцов квадратной матрицы, максимизирующую сумму диагональных элементов. Эта задача может быть решена не более чем за $O(n^3)$ операций, так что было бы глупо использовать метод порядка $n!$, если только n не очень малó. Даже в случае задачи коммивояжера, для решения которой не известен эффективный алгоритм, обычно можно найти подход получше, чем перебор всех возможных решений. Генерацию перестановок лучше всего использовать только тогда, когда имеются основательные причины для рассмотрения каждой перестановки отдельно.

Упражнения

▶ **1.** [*20*] Поясните, как ускорить алгоритм L, оптимизируя его операции при значениях j, близких к n.

2. [*20*] Перепишите алгоритм L так, чтобы он генерировал все перестановки $a_1 \ldots a_n$ в обратном солексном порядке. (Другими словами, значения отражений $a_n \ldots a_1$ должны

быть лексикографически убывающими, как в (11). Этот тип алгоритма часто проще и быстрее исходного, поскольку от значения n зависит меньшее количество вычислений.)

▶ **3.** [*M21*] *Рангом* комбинаторного упорядочения X по отношению к алгоритму генерации является количество других упорядочений, посещаемых алгоритмом до X. Поясните, как вычислить ранг данной перестановки $a_1 \ldots a_n$ по отношению к алгоритму L, если $\{a_1, \ldots, a_n\} = \{1, \ldots, n\}$. Чему равен ранг 314592687?

4. [*M23*] Обобщая упр. 3, поясните, как вычислить ранг $a_1 \ldots a_n$ по отношению к алгоритму L, когда $\{a_1, \ldots, a_n\}$ является мультимножеством $\{n_1 \cdot x_1, \ldots, n_t \cdot x_t\}$; здесь $n_1 + \cdots + n_t = n$ и $x_1 < \cdots < x_t$. (Общее количество перестановок, конечно, представляет собой мультиномиальный коэффициент

$$\binom{n}{n_1, \ldots, n_t} = \frac{n!}{n_1! \ldots n_t!};$$

см. уравнение 5.1.2–(3).) Чему равен ранг 314159265?

5. [*HM25*] Вычислите среднее и дисперсию количества сравнений, выполняемых алгоритмом L на (а) шаге L2, (б) шаге L3, когда элементы $\{a_1, \ldots, a_n\}$ различны.

6. [*HM34*] Выведите производящую функцию для среднего количества сравнений, выполняемых алгоритмом L на (а) шаге L2, (б) шаге L3, когда $\{a_1, \ldots, a_n\}$ представляет собой обобщенное мультимножество, как в упр. 4. Представьте в аналитическом виде результат для случая, когда $\{a_1, \ldots, a_n\}$ представляет собой бинарное мультимножество $\{s \cdot 0, (n-s) \cdot 1\}$.

7. [*HM35*] Чему равен предел при $t \to \infty$ среднего количества сравнений, выполняемых на одну перестановку на шаге L2, если алгоритм L применяется к мультимножеству (а) $\{2 \cdot 1, 2 \cdot 2, \ldots, 2 \cdot t\}$? (б) $\{1 \cdot 1, 2 \cdot 2, \ldots, t \cdot t\}$? (в) $\{2 \cdot 1, 4 \cdot 2, \ldots, 2^t \cdot t\}$?

▶ **8.** [*21*] *Вариациями* (*variation*) множеств называются перестановки всех их подмультимножеств. Например, вариациями $\{1, 2, 2, 3\}$ являются

ϵ, 1, 12, 122, 1223, 123, 1232, 13, 132, 1322,

2, 21, 212, 2123, 213, 2132, 22, 221, 2213, 223, 2231, 23, 231, 2312, 232, 2321,

3, 31, 312, 3122, 32, 321, 3212, 322, 3221.

Покажите, как простое изменение алгоритма L позволяет генерировать все вариации данного мультимножества $\{a_1, a_2, \ldots, a_n\}$.

9. [*22*] Продолжая предыдущее упражнение, разработайте алгоритм для генерации всех r-вариаций мультимножества $\{a_1, a_2, \ldots, a_n\}$, именуемых также r-перестановками, а именно всех перестановок его r-элементных подмультимножеств. (Например, решение буквометика с r различными буквами представляет собой r-вариацию множества $\{0, 1, \ldots, 9\}$.)

10. [*20*] Каковы будут значения $a_1 a_2 \ldots a_n$, $c_1 c_2 \ldots c_n$ и $o_1 o_2 \ldots o_n$ в конце алгоритма P, если в начале $a_1 a_2 \ldots a_n = 12 \ldots n$?

11. [*M22*] Сколько раз выполняется каждый шаг алгоритма P (в предположении, что $n \geq 2$)?

▶ **12.** [*M23*] Какова 1 000 000-я перестановка, посещаемая (а) алгоритмом L, (б) алгоритмом P, (в) алгоритмом C, если $\{a_1, \ldots, a_n\} = \{0, \ldots, 9\}$? *Указание*: в записи со смешанными основаниями $1\,000\,000 = \left[\begin{smallmatrix} 2, & 6, & 6, & 2, & 5, & 1, & 2, & 2, & 0, & 0 \\ 10, & 9, & 8, & 7, & 6, & 5, & 4, & 3, & 2, & 1 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 0, & 0, & 1, & 2, & 3, & 0, & 2, & 7, & 1, & 0 \\ 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & 10 \end{smallmatrix}\right]$.

13. [*M21*] (Мартин Гарднер (Martin Gardner), 1974.) Истинно или ложно следующее утверждение? Если $a_1 a_2 \ldots a_n$ изначально равно $12 \ldots n$, алгоритм P начинает с посещения всех $n!/2$ перестановок, в которых 1 предшествует 2; затем следующей перестановкой является $n \ldots 21$.

14. [*M22*] Истинно или ложно следующее утверждение? Если в алгоритме P $a_1a_2\ldots a_n$ изначально равно $x_1x_2\ldots x_n$, то в начале шага P5 всегда имеем $a_{j-c_j+s} = x_j$.

15. [*M23*] (Селмер Джонсон (Selmer Johnson), 1963.) Покажите, что переменная смещения s в алгоритме P никогда не превышает 2.

16. [*21*] Поясните, как заставить алгоритм P работать быстрее, оптимизировав его операции для значений j, близких к n. (Эта задача аналогична упр. 1.)

▶ **17.** [*20*] Расширьте алгоритм P так, чтобы при посещении $a_1\ldots a_n$ на шаге P2 для обработки была доступна и *обратная перестановка* $a'_1\ldots a'_n$. (Обратная перестановка удовлетворяет условию "$a'_k = j$ тогда и только тогда, когда $a_j = k$".)

18. [*21*] (*Четочные перестановки.*) Разработайте эффективный способ генерации $(n-1)!/2$ перестановок, которые представляют все возможные неориентированные циклы на вершинах $\{1,\ldots,n\}$ (т. е. циклический сдвиг $a_1\ldots a_n$ или $a_n\ldots a_1$ не генерируется, если сгенерирован $a_1\ldots a_n$). Например, при $n = 4$ могут использоваться перестановки (1234, 1324, 3124).

19. [*25*] Разработайте алгоритм, который генерирует все перестановки n различных элементов *без использования циклов* в духе алгоритма 7.2.1.1L.

▶ **20.** [*20*] n-мерный куб имеет $2^n n!$ симметрий, по одной для каждого способа перестановки и/или дополнения координат. Такую симметрию обычно удобнее представлять как знаковую перестановку, т. е. перестановку с необязательными назначенными элементам знаками. Например, $23\overline{1}$ представляет собой знаковую перестановку, преобразующую вершины трехмерного куба с помощью замены $x_1x_2x_3$ на $x_2x_3\overline{x}_1$, так что $000 \mapsto 001$, $001 \mapsto 011$, $\ldots$, $111 \mapsto 110$. Разработайте простой алгоритм, который генерирует все знаковые перестановки множества $\{1,2,\ldots,n\}$, в котором на каждом шаге выполняется либо обмен двух смежных элементов, либо изменение знака первого элемента.

21. [*M21*] (Э. П. Мак-Крейви (E. P. McCravy), 1971.) Сколько решений имеет буквометик (6) в системе счисления с основанием b?

22. [*M15*] Истинно или ложно следующее утверждение? Если буквометик имеет решение в системе счисления с основанием b, то он имеет решение и в системе счисления с основанием $b+1$.

23. [*M20*] Истинно или ложно следующее утверждение? Чистый буквометик не может иметь две идентичные сигнатуры $s_j = s_k \neq 0$, когда $j \neq k$.

24. [*25*] Вручную или с помощью компьютера решите следующие буквометики.

а) SEND + A + TAD + MORE = MONEY.
б) ZEROES + ONES = BINARY. (Питер Мак-Дональд (Peter MacDonald), 1977.)
в) DCLIX + DLXVI = MCCXXV. (Вилли Энггрен (Willy Enggren), 1972.)
г) COUPLE + COUPLE = QUARTET. (Майкл Р. У. Бакли (Michael R. W. Buckley), 1977.)
д) FISH + N + CHIPS = SUPPER. (Роберт Винникомб (Bob Vinnicombe), 1978.)
е) SATURN + URANUS + NEPTUNE + PLUTO = PLANETS. (Вилли Энггрен (Willy Enggren), 1968.)
ж) EARTH + AIR + FIRE + WATER = NATURE. (Герман Нийон (Herman Nijon), 1977.)
з) AN+ACCELERATING+INFERENTIAL+ENGINEERING+TALE+ELITE+GRANT+FEE+ET+CETERA = ARTIFICIAL + INTELLIGENCE.
и) HARDY + NESTS = NASTY + HERDS.

▶ **25.** [*M21*] Разработайте быстрый способ вычисления $\min(a\cdot s)$ и $\max(a\cdot s)$ среди всех допустимых перестановок $a_1\ldots a_{10}$ множества $\{0,\ldots,9\}$ для заданного вектора сигнатуры $s = (s_1,\ldots,s_{10})$ и множества первых букв F буквометика. (Такая процедура позволяет быстро исключить многие случаи из большого семейства буквометиков, как в ряде примеров ниже, поскольку решение возможно только тогда, когда $\min(a\cdot s) \le 0 \le \max(a\cdot s)$.)

26. [*25*] Каково единственное решение буквометика

$$\texttt{NIIHAU} \pm \texttt{KAUAI} \pm \texttt{OAHU} \pm \texttt{MOLOKAI} \pm \texttt{LANAI} \pm \texttt{MAUI} \pm \texttt{HAWAII} = 0?$$

27. [*30*] Постройте чистые аддитивные буквометики, все слова которых состоят из пяти букв.

28. [*M25*] *Разбиением* целого числа n называется выражение вида $n = n_1 + \cdots + n_t$, где $n_1 \geq \cdots \geq n_t > 0$. Такое разбиение называется *дважды истинным*, если $\alpha(n) = \alpha(n_1) + \cdots + \alpha(n_t)$ является чистым буквометиком, где $\alpha(n)$ — "имя" n на некотором языке. Дважды истинные разбиения введены Аланом Уэйном (Alan Wayne) в *AMM* **54** (1947), 38, 412–414, где он предложил решение TWENTY = SEVEN + SEVEN + SIX и несколько других.

а) Найдите все дважды истинные разбиения на английском языке для $1 \leq n \leq 20$.

б) Уэйн привел также пример EIGHTY = FIFTY + TWENTY + NINE + ONE. Найдите все дважды истинные разбиения для $1 \leq n \leq 100$, все части которых *различны*, используя имена ONE, TWO, ..., NINETYNINE, ONEHUNDRED.

▶ **29.** [*M25*] Продолжая выполнять предыдущее упражнение, найдите все уравнения вида $n_1 + \cdots + n_t = n'_1 + \cdots + n'_{t'}$, которые истинны как с математической, так и с "буквометической" (на английском языке) точек зрения, когда все $\{n_1, \ldots, n_t, n'_1, \ldots, n'_{t'}\}$ — различные положительные целые числа, меньшие 20. Например,

$$\texttt{TWELVE} + \texttt{NINE} + \texttt{TWO} = \texttt{ELEVEN} + \texttt{SEVEN} + \texttt{FIVE}.$$

Все буквометики должны быть чистыми.

30. [*25*] Решите вручную или с помощью компьютера следующие мультипликативные буквометики.

а) TWO × TWO = SQUARE. (Г. Э. Дьюдени (H. E. Dudeney), 1929.)
(*Подобный мультипликативный буквометик на русском языке —* ДВА × ДВА = ЧЕТЫРЕ (ДВА × ДВА = ЧОТИРИ *на украинском*). — *Примеч. пер.*)

б) HIP × HIP = HURRAY. (Вилли Энггрен (Willy Enggren), 1970.)

в) PI × R × R = AREA. (Брайан Барвелл (Brian Barwell), 1981.)

г) NORTH/SOUTH = EAST/WEST. (Нобуюки Йошигахара (Nob Yoshigahara), 1995.)

д) NAUGHT × NAUGHT = ZERO × ZERO × ZERO. (Алан Уэйн (Alan Wayne), 2003.)

31. [*M22*] (Нобуюки Йошигахара (Nob Yoshigahara).) (а) Найдите единственное решение A/BC + D/EF + G/HI = 1, если $\{\texttt{A}, \ldots, \texttt{I}\} = \{1, \ldots, 9\}$. (б) Аналогично добейтесь выполнения условий AB $\bmod 2 = 0$, ABC $\bmod 3 = 0$ и т. д.

32. [*M25*] (Г. Э. Дьюдени (H. E. Dudeney), 1901.) Найдите все способы представления числа 100 путем вставки плюса и косой черты в перестановку цифр $\{1, \ldots, 9\}$. Например, $100 = 91 + 5742/638$. Плюс должен предшествовать косой черте.

33. [*25*] Продолжая предыдущее упражнение, найдите все натуральные числа, меньшие 150, которые (а) не могут быть представлены таким способом; (б) имеют единственное представление.

34. [*M26*] Сделайте уравнение EVEN + ODD + PRIME = x дважды истинным, когда (а) x является точной 5-й степенью; (б) x является точной 7-й степенью.

▶ **35.** [*M20*] Аутоморфизмы четырехмерного куба могут иметь много различных таблиц Симса, и в (14) показана только одна из них. Сколько различных таблиц Симса может быть для этой группы, если вершины пронумерованы так, как показано в (12)?

36. [*M23*] Найдите таблицу Симса для группы всех аутоморфизмов доски для игры "tic-tac-toe" (аналог крестиков-ноликов) размером 4×4

0	1	2	3
4	5	6	7
8	9	a	b
c	d	e	f

,

т. е. найдите все перестановки, которые преобразуют линии в линии, где "линия" — множество из четырех элементов, принадлежащих одной строке, столбцу или диагонали.

▶ **37.** [*HM22*] Сколько таблиц Симса можно использовать с алгоритмами G и H? Оцените логарифм этого количества при $n \to \infty$.

38. [*HM21*] Докажите, что среднее количество транспозиций на одну перестановку при использовании алгоритма Орд-Смита (26) примерно равно $\sinh 1 \approx 1.175$.

39. [*16*] Запишите 24 перестановки, генерируемые для $n = 4$ (а) методом Орд-Смита (26); (б) методом Хипа (27).

40. [*M23*] Покажите, что метод Хипа (27) соответствует корректной таблице Симса.

▶ **41.** [*M33*] Разработайте алгоритм, генерирующий все r-вариации множества $\{0, 1, \ldots, n-1\}$ путем обмена только двух элементов при переходе от одной вариации к следующей. (См. упр. 9.) *Указание*: обобщите метод Хипа (27), получая результаты в позициях $a_{n-r} \ldots a_{n-1}$ массива $a_0 \ldots a_{n-1}$. Например, одно из решений при $n = 5$ и $r = 2$ использует последние два элемента соответствующих перестановок 01234, 31204, 30214, 30124, 40123, 20143, 24103, 24013, 34012, 14032, 13042, 13402, 23401, 03421, 02431, 02341, 12340, 42310, 41320, 41230.

42. [*M20*] Постройте таблицу Симса для всех перестановок, в которых каждая $\sigma(k, j)$ и каждая $\tau(k, j)$ для $1 \le j \le k$ представляют собой цикл длиной ≤ 3.

43. [*M24*] Постройте таблицу Симса для всех перестановок, в которых каждая $\sigma(k, k)$, $\omega(k)$ и $\tau(k, j)\,\omega(k-1)^-$ для $1 \le j \le k$ является циклами длиной ≤ 3.

44. [*20*] Когда в расширенном алгоритме G пропускаются блоки нежелательных перестановок, превосходит ли таблица Симса метода Орд-Смита (23) таблицу Симса обратного солексного метода (18)?

45. [*20*] (а) Чему равны индексы $u_1 \ldots u_9$ при посещении алгоритмом X перестановки 314592687? (б) Какая перестановка посещается при $u_1 \ldots u_9 = 161800000$?

46. [*20*] Истинно или ложно следующее утверждение? Когда алгоритм X посещает $a_1 \ldots a_n$, мы имеем $u_k > u_{k+1}$ тогда и только тогда, когда $a_k > a_{k+1}$, $1 \le k < n$.

▶ **47.** [*M21*] Выразите количество выполнений каждого шага алгоритма X через величины N_0, N_1, $\ldots$, N_n, где N_k — число префиксов $a_1 \ldots a_k$, удовлетворяющих условию $t_j(a_1, \ldots, a_j)$ для $1 \le j \le k$.

▶ **48.** [*M25*] Сравните время работы алгоритмов X и L в случае, когда проверки $t_1(a_1)$, $t_2(a_1, a_2)$, $\ldots$, $t_n(a_1, a_2, \ldots, a_n)$ всегда истинны.

▶ **49.** [*28*] Предложенный в разделе метод решения аддитивных буквометиков с помощью алгоритма X, по сути, выбирает цифры справа налево; другими словами, он присваивает пробные значения младшим цифрам, перед тем как рассмотреть цифры, соответствующие более высоким степеням 10.

Исследуйте альтернативный подход, выбирающий цифры слева направо. Например, такой метод может сразу вывести, что в бувометике `SEND + MORE = MONEY` значение `M` $= 1$. *Указание*: см. упр. 25.

50. [*M15*] Объясните, почему из (13) следует дуальная формула (32).

51. [*M16*] Истинно или ложно следующее утверждение? Если множества $S_k = \{\sigma(k,0), \ldots, \sigma(k,k)\}$ образуют таблицу Симса для группы всех перестановок, то то же самое справедливо и для множеств $S_k^- = \{\sigma(k,0)^-, \ldots, \sigma(k,k)^-\}$.

▶ **52.** [*M22*] Какие перестановки $\tau(k,j)$ и $\omega(k)$ возникают при использовании алгоритма H с таблицей Симса (36)? Сравните полученный генератор с алгоритмом P.

▶ **53.** [*M26*] (Ф. М. Айвес (F. M. Ives).) Постройте таблицу Симса, для которой алгоритм H будет генерировать все перестановки с помощью только $n! + O((n-2)!)$ транспозиций.

54. [*20*] Будет ли алгоритм C корректно работать, если на шаге C3 выполнять правый циклический сдвиг и установку $a_1 \ldots a_{k-1}a_k \leftarrow a_k a_1 \ldots a_{k-1}$ вместо левого циклического сдвига?

55. [*M27*] Рассмотрим *факториальную линеечную функцию*

$$\rho_!(m) = \max\{k \mid m \bmod k! = 0\}.$$

Пусть σ_k и τ_k являются перестановками неотрицательных целых чисел, таких, что $\sigma_j\tau_k = \tau_k\sigma_j$, когда $j \le k$. Пусть α_0 и β_0 — тождественные перестановки, и определим для $m > 0$

$$\alpha_m = \beta_{m-1}^- \tau_{\rho_!(m)} \beta_{m-1}\alpha_{m-1}, \qquad \beta_m = \sigma_{\rho_!(m)}\beta_{m-1}.$$

Например, если σ_k — операция переворота $(1\ k{-}1)(2\ k{-}2)\ldots = (0\ k)\phi(k)$ и если $\tau_k = (0\ k)$, а алгоритм E начинается с $a_j = j$ для $0 \le j < n$, то α_m и β_m являются содержимым $a_0 \ldots a_{n-1}$ и $b_0 \ldots b_{n-1}$ после того, как шаг E5 будет выполнен m раз.

а) Докажите, что $\beta_{(n+1)!}\alpha_{(n+1)!} = \sigma_{n+1}\sigma_n^-\tau_{n+1}\tau_n^-(\beta_{n!}\alpha_{n!})^{n+1}$.

б) Воспользуйтесь результатом (а), чтобы установить правильность алгоритма E.

56. [*M22*] Докажите, что алгоритм E остается корректным при замене шага E5 на

E5′. [Транспозиция пар.] Если $k > 2$, обменять $b_{j+1} \leftrightarrow b_j$ для $j = k-2,\ k-4,\ \ldots,$ (2 или 1). Вернуться к шагу E2. ▮

57. [*HM22*] Каково среднее количество обменов на шаге E5?

58. [*M21*] Истинно или ложно следующее утверждение? Если алгоритм E начинается с $a_0 \ldots a_{n-1} = x_1 \ldots x_n$, то последняя посещенная перестановка начинается с $a_0 = x_n$.

59. [*M20*] Некоторые авторы определяют дуги графа Кейли как идущие от π к $\pi\alpha_j$, а не от π к $\alpha_j\pi$. Является ли это отличие существенным?

▶ **60.** [*21*] *Циклом Грея для перестановок* называется цикл $(\pi_0, \pi_1, \ldots, \pi_{n!-1})$, включающий каждую перестановку $\{1, 2, \ldots, n\}$ и обладающий тем свойством, что π_k отличается от $\pi_{(k+1) \bmod n!}$ на транспозицию смежных элементов. Он также может быть описан как гамильтонов цикл графа Кейли для группы всех перестановок множества $\{1, 2, \ldots, n\}$ с $n-1$ генераторами $((1\ 2), (2\ 3), \ldots, (n{-}1\ n))$. *Дельта-последовательность* такого цикла Грея представляет собой последовательность целых чисел $\delta_0\delta_1 \ldots \delta_{n!-1}$, такую, что $\pi_{(k+1) \bmod n!} = (\delta_k\ \delta_k{+}1)\,\pi_k$ (см. 7.2.1.1–(24), где описывается аналогичная ситуация для бинарных n-кортежей). Например, на рис. 43 показан цикл Грея, определяемый простыми изменениями при $n = 4$; его дельта-последовательность имеет вид $(32131231)^3$.

а) Найдите все циклы Грея для перестановок множества $\{1, 2, 3, 4\}$.

б) Два цикла Грея рассматриваются как эквивалентные, если их дельта-последовательности можно получить одну из другой путем циклического сдвига $(\delta_k \ldots \delta_{n!-1}\delta_0 \ldots \delta_{k-1})$ и/или обращения $(\delta_{n!-1} \ldots \delta_1\delta_0)$ и/или дополнения $((n{-}\delta_0)(n{-}\delta_1)\ldots(n{-}\delta_{n!-1}))$. Какие циклы Грея в (а) являются эквивалентными?

Рис. 43. Алгоритм P проходит по данному гамильтонову циклу усеченного октаэдра с рис. 5.1.

61. [*21*] Продолжим выполнение предыдущего упражнения. *Код Грея для перестановок* подобен циклу Грея, за исключением того, что последняя перестановка $\pi_{n!-1}$ не обязана быть смежной с начальной перестановкой π_0. Изучите множество всех кодов Грея для $n = 4$, которые начинаются с 1234.

▶ **62.** [*M23*] Какие перестановки могут быть достигнуты в качестве последнего элемента кода Грея, начинающегося с $12\ldots n$?

63. [*M25*] Оцените общее количество циклов Грея для перестановок множества $\{1,2,3,4,5\}$.

64. [*23*] "Двойной код Грея" для перестановок представляет собой цикл Грея с тем дополнительным свойством, что $\delta_{k+1} = \delta_k \pm 1$ для всех k. Комптон (Compton) и Вильямсон (Williamson) доказали, что такие коды существуют для всех $n \geq 3$. Сколько двойных кодов Грея имеется для $n = 5$?

65. [*M25*] Для каких целых чисел N существует путь Грея, проходящий через N лексикографически наименьших перестановок множества $\{1, \ldots, n\}$? (В упр. 7.2.1.1–26 решается аналогичная задача для бинарных n-кортежей.)

66. [*22*] Метод обмена Эрлиха предлагает другой тип циклов Грея для перестановок, в котором $n - 1$ генераторов представляют собой звездные транспозиции (1 2), (1 3), ..., (1 n). Например, на рис. 44 показан соответствующий граф для $n = 4$. Проанализируйте гамильтоновы циклы этого графа.

Рис. 44. Граф Кейли для перестановок множества $\{1, 2, 3, 4\}$, сгенерированных звездными транспозициями (1 2), (1 3) и (1 4), изображенный в виде скрученного тора.

67. [*26*] Продолжая выполнять предыдущее упражнение, найдите цикл Грея с обменом первого элемента для $n = 5$, в котором каждая звездная транспозиция (1 j), $2 \leq j \leq 5$, встречается 30 раз.

68. [*M30*] (В. Л. Компельмахер и В. А. Лисковец, 1975.) Пусть G — граф Кейли для всех перестановок множества $\{1,\ldots,n\}$ с генераторами $(\alpha_1,\ldots,\alpha_k)$, где каждый генератор α_j представляет собой транспозицию $(u_j\ v_j)$; пусть также A является графом с вершинами $\{1,\ldots,n\}$ и ребрами u_j — v_j для $1\le j\le k$. Докажите, что G имеет гамильтонов цикл тогда и только тогда, когда A — связный граф. (На рис. 43 показан частный случай, когда A является путем, а на рис. 44 — частный случай, когда A представляет собой "звезду".)

▶ **69.** [*28*] Приведенный ниже алгоритм генерирует все перестановки $A_1A_2A_3\ldots A_n$ множества $\{1,2,3,\ldots,n\}$ для $n\ge 4$ с применением только трех преобразований:

$$\rho=(1\,2)(3\,4)(5\,6)\ldots,\qquad \sigma=(2\,3)(4\,5)(6\,7)\ldots,\qquad \tau=(3\,4)(5\,6)(7\,8)\ldots,$$

никогда не применяя ρ и τ друг после друга. Поясните принцип работы данного алгоритма.

Z1. [Инициализация.] Установить $A_j\leftarrow j$ для $1\le j\le n$. Установить также $a_j\leftarrow 2j$ для $1\le j\le n/2$ и $a_{n-j}\leftarrow 2j+1$ для $1\le j<n/2$. Затем выполнить алгоритм P, но с параметром $n-1$ вместо n. Мы будем рассматривать этот алгоритм как сопрограмму, которая должна вернуть управление при "посещении" $a_1\ldots a_{n-1}$ на шаге P2. В ней совместно используются все переменные, за исключением n.

Z2. [Установка x и y.] Вновь выполнить алгоритм P, получая новую перестановку $a_1\ldots a_{n-1}$ и новое значение j. Если $j=2$, обменять $a_{1+s}\leftrightarrow a_{2+s}$ (тем самым отменяя действие шага P5) и повторить этот шаг; в таком случае мы находимся в срединной точке алгоритма P. Если $j=1$ (так что алгоритм P завершен), установить $x\leftarrow y\leftarrow 0$ и перейти к шагу Z3. В противном случае установить $x\leftarrow a_{j-c_j+s+[o_j=+1]}$, $y\leftarrow a_{j-c_j+s-[o_j=-1]}$; это два последних обменянных на шаге P5 элемента.

Z3. [Посещение.] Посетить перестановку $A_1\ldots A_n$. Затем перейти к шагу Z5, если $A_1=x$ и $A_2=y$.

Z4. [Применение ρ, затем σ.] Обменять $A_1\leftrightarrow A_2$, $A_3\leftrightarrow A_4$, $A_5\leftrightarrow A_6$, ... Посетить $A_1\ldots A_n$. Затем обменять $A_2\leftrightarrow A_3$, $A_4\leftrightarrow A_5$, $A_6\leftrightarrow A_7$, ... Завершить работу алгоритма, если $A_1\ldots A_n=1\ldots n$, в противном случае вернуться к шагу Z3.

Z5. [Применение τ, затем σ.] Обменять $A_3\leftrightarrow A_4$, $A_5\leftrightarrow A_6$, $A_7\leftrightarrow A_8$, ... Посетить $A_1\ldots A_n$. Затем обменять $A_2\leftrightarrow A_3$, $A_4\leftrightarrow A_5$, $A_6\leftrightarrow A_7$, ...и вернуться к шагу Z2. ▮

Указание: сначала покажите, что алгоритм работает, если изменить его так, что на шаге Z1 выполняются установки $A_j\leftarrow n+1-j$ и $a_j\leftarrow j$, и если на шагах Z4 и Z5 вместо ρ, σ и τ используются "переворотные" перестановки

$$\rho'=(1\ \ n)(2\ \ n{-}1)\ldots,\qquad \sigma'=(2\ \ n)(3\ \ n{-}1)\ldots,\qquad \tau'=(2\ \ n{-}1)(3\ \ n{-}2)\ldots.$$

При такой модификации шаг Z3 должен переходить к шагу Z5, если $A_1=x$ и $A_n=y$; шаг Z4 должен завершать работу алгоритма, когда $A_1\ldots A_n=n\ldots 1$.

▶ **70.** [*M33*] Два цикла из 12 элементов (41) можно рассматривать как σ–τ циклы для двенадцати перестановок множества $\{1,1,3,4\}$:

$$\begin{aligned}1134\to 1341\to 3411\to 4311\to 3114\to 1143\to 1431\\ \to 4131\to 1314\to 3141\to 1413\to 4113\to 1134.\end{aligned}$$

Замена $\{1,1\}$ на $\{1,2\}$ дает непересекающиеся циклы, и мы получаем гамильтонов путь путем перехода от одного цикла к другому. Можно ли подобным образом образовать σ–τ путь для всех перестановок из шести элементов, основываясь на цикле из 360 элементов для перестановок мультимножества $\{1,1,3,4,5,6\}$?

71. [*48*] Будет ли граф Кейли с генераторами $\sigma = (1\,2\,\ldots\,n)$ и $\tau = (1\,2)$ иметь гамильтонов цикл при нечетном $n \geq 3$?

72. [*M21*] Предположим, что для данного графа Кейли с генераторами $(\alpha_1, \ldots, \alpha_k)$ каждый α_j отображает $x \mapsto y$. (Например, и σ, и τ в упр. 71 отображают $1 \mapsto 2$.) Докажите, что любой гамильтонов путь, начинающийся с $12\ldots n$ в G, должен заканчиваться перестановкой, отображающей $y \mapsto x$.

▶ **73.** [*M30*] Пусть α, β и σ являются перестановками множества X, где $X = A \cup B$. Предположим, что $x\sigma = x\alpha$, когда $x \in A$, и $x\sigma = x\beta$, когда $x \in B$, и что порядок $\alpha\beta^-$ нечетен.

а) Докажите, что все три перестановки, α, β, σ, имеют один и тот же знак; т. е. либо все они четны, либо все нечетны. *Указание*: перестановка имеет нечетный порядок тогда и только тогда, когда все ее циклы имеют нечетную длину.

б) Выведите из п. (а) теорему R.

74. [*M30*] (Р. А. Ранкин (R. A. Rankin).) В предположении, что в теореме R $\alpha\beta = \beta\alpha$, докажите, что в графе Кейли для G гамильтонов цикл существует тогда и только тогда, когда имеется такое число k, что $0 \leq k \leq g/c$ и $t + k \perp c$, где $\beta^{g/c} = \gamma^t$, $\gamma = \alpha\beta^-$. *Указание*: представьте элементы группы в виде $\beta^j\gamma^k$.

75. [*M26*] Ориентированный тор $\vec{C_m} \times \vec{C_n}$ имеет mn вершин (x, y) $(0 \leq x < m,\ 0 \leq y < n)$ и дуги $(x, y) \longrightarrow (x, y)\alpha = ((x+1) \bmod m, y)$, $(x, y) \longrightarrow (x, y)\beta = (x, (y+1) \bmod n)$. Докажите, что если $m > 1$ и $n > 1$, то количество гамильтоновых циклов этого ориентированного графа равно

$$\sum_{k=1}^{d-1} \binom{d}{k} [\gcd((d-k)m, kn) = d], \qquad d = \gcd(m, n).$$

76. [*M31*] Пронумерованные числами 0, 1, …, 63 на рис. 45 ячейки иллюстрируют *северо-восточный обход конем* тора размером 8×8: если k находится в ячейке (x_k, y_k), то $(x_{k+1}, y_{k+1}) \equiv (x_k + 2, y_k + 1)$ или $(x_k + 1, y_k + 2)$ по модулю 8, а $(x_{64}, y_{64}) = (x_0, y_0)$. Сколько таких обходов может быть на торе $m \times n$ при $m, n \geq 3$?

29	24	19	14	49	44	39	34
58	53	48	43	38	9	4	63
23	18	13	8	3	62	33	28
52	47	42	37	32	27	22	57
17	12	7	2	61	56	51	46
6	41	36	31	26	21	16	11
35	30	1	60	55	50	45	40
0	59	54	25	20	15	10	5

Рис. 45. Северо-восточный обход конем.

▶ **77.** [*22*] Завершите `MMIX`-программу, внутренний цикл которой показан в (42), используя метод Хипа (27).

78. [*M23*] Проанализируйте время работы программы из упр. 77, обобщив ее так, чтобы внутренний цикл выполнял $r!$ посещений (с $a_0 \ldots a_{r-1}$ в глобальных регистрах).

79. [*20*] Какие семь инструкций `MMIX` выполняют ⟨Обмен полубайтов…⟩ в (45)? Например, если регистр `t` содержит значение 4, а регистр `a` — полубайты `#12345678`, то содержимое регистра `a` должно измениться на `#12345687`.

80. [*21*] Выполните предыдущее упражнение с помощью только пяти инструкций MMIX. *Указание*: воспользуйтесь MXOR.

▶ **81.** [*22*] Завершите MMIX-программу (46), указав, каким образом следует выполнить ⟨ Продолжение работы методом Лангдона. ⟩.

82. [*M21*] Проанализируйте время работы программы из упр. 81.

83. [*22*] Воспользуйтесь σ–τ-путем из упр. 70 для разработки MMIX-подпрограммы, аналогичной (42), для генерации всех перестановок #123456 в регистре a.

84. [*20*] Предложите эффективный способ генерации всех $n!$ перестановок множества $\{1, \ldots, n\}$ при наличии p параллельно работающих процессоров.

▶ **85.** [*25*] Будем считать, что n достаточно мало́, так что $n!$ помещается в компьютерном слове. Как эффективно преобразовать заданную перестановку $\alpha = a_1 \ldots a_n$ множества $\{1, \ldots, n\}$ в целое число $k = r(\alpha)$ в диапазоне $0 \le k < n!$? Функции $k = r(\alpha)$ и $\alpha = r^{[-1]}(k)$ должны быть вычислимы за $O(n)$ шагов.

86. [*20*] Отношение частичного упорядочения предполагается транзитивным, т. е. из $x \prec y$ и $y \prec z$ должно следовать $x \prec z$. Но алгоритм V не требует, чтобы его входное отношение обладало указанным свойством.

Покажите, что если $x \prec y$ и $y \prec z$, то результат работы алгоритма V не зависит от того, будет ли выполняться условие $x \prec z$.

87. [*20*] (Ф. Раски (F. Ruskey).) Рассмотрим таблицы инверсий $c_1 \ldots c_n$ перестановок, посещенных алгоритмом V. Каким замечательным свойством они обладают? (Сравните с таблицами инверсий (4) из алгоритма P.)

88. [*21*] Покажите, что алгоритм V может использоваться для генерации всех способов разбиения цифр $\{0, 1, \ldots, 9\}$ на два 3-элементных и два 2-элементных множества.

▶ **89.** [*M30*] Рассмотрим числа t_0, t_1, …, t_n, определенные перед (51). Ясно, что $t_0 = t_1 = 1$.

а) Назовем индекс j "тривиальным", если $t_j = t_{j-1}$. Например, 9 тривиально в смысле отношений таблицы Юнга (48). Поясните, как модифицировать алгоритм V так, чтобы переменная k принимала только нетривиальные значения.

б) Проанализируйте время работы модифицированного алгоритма. Какие формулы заменят (51)?

в) Будем говорить, что отрезок $[j\,..\,k]$ не является цепочкой, если существует индекс l, такой, что $j \le l < k$ и не выполняется $l \prec l + 1$. Докажите, что в таком случае $t_k \ge 2t_{j-1}$.

г) Каждая обратная топологическая сортировка $a'_1 \ldots a'_n$ определяет маркировку, которая соответствует отношениям $a'_{j_1} \prec a'_{k_1}$, …, $a'_{j_m} \prec a'_{k_m}$, эквивалентным исходным отношениям $j_1 \prec k_1$, …, $j_m \prec k_m$. Поясните, как найти такую маркировку, что $[j\,..\,k]$ не является цепочкой, когда j и k являются последовательными нетривиальными индексами.

д) Докажите, что при такой маркировке в формулах из п. (б) $M < 4N$.

90. [*M21*] Алгоритм V можно использовать для генерации всех H-упорядоченных перестановок для всех h из заданного множества, т. е. всех $a'_1 \ldots a'_n$, таких, что $a'_j < a'_{j+h}$ для $1 \le j \le n - h$ (см. раздел 5.2.1). Проанализируйте время работы алгоритма V при генерации им всех перестановок, являющихся одновременно 2- и 3-упорядоченными.

91. [*HM21*] Проанализируйте время работы алгоритма V, когда он используется с соотношениями (49) для поиска совершенных паросочетаний.

92. [*M18*] Каково вероятное количество посещенных алгоритмом V перестановок в "случайном" случае? Пусть P_n — количество частичных упорядочений $\{1, \ldots, n\}$, т. е. количество отношений, являющихся рефлексивными, антисимметричными и транзитивными. Пусть Q_n — количество таких отношений, обладающих тем дополнительным свойством,

что $j < k$, если $j \prec k$. Выразите ожидаемое количество способов топологической сортировки n элементов, усредненное по всем частичным упорядочениям, через P_n и Q_n.

93. [*35*] Докажите, что все топологические сортировки могут быть сгенерированы таким образом, что на каждом шаге выполняются только одна или две смежные транспозиции. (Пример $1 \prec 2$, $3 \prec 4$ показывает, что добиться выполнения одной транспозиции на шаг не всегда возможно, даже если допустить несмежные обмены, поскольку из шести подходящих перестановок нечетными оказываются только две.)

▶ **94.** [*25*] Покажите, что в случае совершенных паросочетаний использование отношений (49) позволяет генерировать все топологические сортировки с помощью одной транспозиции на шаг.

95. [*21*] Подумайте, как сгенерировать все *"холмистые" перестановки* множества $\{1, \ldots, n\}$, т. е. перестановки $a_1 \ldots a_n$, такие, что $a_1 < a_2 > a_3 < a_4 > \cdots$.

96. [*21*] Подумайте, как сгенерировать все *циклические перестановки* множества $\{1, \ldots, n\}$, т. е. перестановки $a_1 \ldots a_n$, циклическое представление которых состоит из одного цикла с n элементами.

97. [*21*] Подумайте, как сгенерировать все *неупорядочения* множества $\{1, \ldots, n\}$, т. е. такие $a_1 \ldots a_n$, у которых $a_1 \neq 1$, $a_2 \neq 2$, $a_3 \neq 3$, ...

98. [*HM23*] Проанализируйте асимптотическое время работы метода из предыдущего упражнения.

99. [*M30*] Покажите, что для заданного $n \geq 3$ все неупорядочения множества $\{1, \ldots, n\}$ можно генерировать, выполняя между посещениями не более двух транспозиций.

100. [*21*] Подумайте, как сгенерировать все *неприводимые* (*indecomposable*) перестановки множества $\{1, \ldots, n\}$, т. е. такие $a_1 \ldots a_n$, что $\{a_1, \ldots, a_j\} \neq \{1, \ldots, j\}$ для $1 \leq j < n$.

101. [*21*] Подумайте, как сгенерировать все *инволюции* множества $\{1, \ldots, n\}$, т. е. перестановки $a_1 \ldots a_n$, для которых $a_{a_1} \ldots a_{a_n} = 1 \ldots n$.

102. [*M30*] Покажите, что все инволюции множества $\{1, \ldots, n\}$ можно сгенерировать с использованием не более двух транспозиций между посещениями.

103. [*M32*] Покажите, что все четные перестановки множества $\{1, \ldots, n\}$ могут быть сгенерированы последовательными *циклическими сдвигами трех последовательных элементов.*

▶ **104.** [*M22*] Перестановка $a_1 \ldots a_n$ множества $\{1, \ldots, n\}$ называется *хорошо сбалансированной*, если

$$\sum_{k=1}^{n} k a_k = \sum_{k=1}^{n} (n + 1 - k)\, a_k.$$

Например, 3142 хорошо сбалансирована при $n = 4$.

а) Докажите, что для $n \bmod 4 = 2$ не существует хорошо сбалансированных перестановок.

б) Докажите, что если $a_1 \ldots a_n$ — хорошо сбалансированная перестановка, то таковыми же являются обратная перестановка $a_n \ldots a_1$, дополнение $(n+1-a_1) \ldots (n+1-a_n)$ и инверсия $a'_1 \ldots a'_n$.

в) Определите количество хорошо сбалансированных перестановок для небольших значений n.

▶ **105.** [*26*] *Слабым порядком* называется отношение $\preceq$, являющееся транзитивным (из $x \preceq y$ и $y \preceq z$ следует $x \preceq z$) и полным (всегда выполняется либо $x \preceq y$, либо $y \preceq x$). Мы можем записать $x \equiv y$, если $x \preceq y$ и $y \preceq x$; $x \prec y$, если $x \preceq y$ и $y \not\preceq x$. Для трех элементов $\{1, 2, 3\}$

имеется тринадцать слабых порядков, а именно

$$1 \equiv 2 \equiv 3, \quad 1 \equiv 2 \prec 3, \quad 1 \prec 2 \equiv 3, \quad 1 \prec 2 \prec 3, \quad 1 \equiv 3 \prec 2, \quad 1 \prec 3 \prec 2,$$
$$2 \prec 1 \equiv 3, \quad 2 \prec 1 \prec 3, \quad 2 \equiv 3 \prec 1, \quad 2 \prec 3 \prec 1, \quad 3 \prec 1 \equiv 2, \quad 3 \prec 1 \prec 2, \quad 3 \prec 2 \prec 1.$$

а) Поясните, как систематически сгенерировать все слабые порядки $\{1, \ldots, n\}$ как последовательности цифр, разделенных символами $\equiv$ или $\prec$.

б) Слабый порядок может также быть представлен как последовательность $a_1 \ldots a_n$, где $a_j = k$, если j предшествуют k знаков $\prec$. Например, при использовании такой записи тринадцатью слабыми порядками на множестве $\{1, 2, 3\}$ являются соответственно 000, 001, 011, 012, 010, 021, 101, 102, 100, 201, 110, 120, 210. Найдите простой способ генерации всех таких последовательностей длиной n.

106. [*M40*] Можно ли выполнить упр. 105, (б) с помощью кода наподобие кода Грея?

▶ **107.** [*30*] (Джон Х. Конвей (John H. Conway), 1973.) Пасьянс "topswops" начинается с тасования колоды из n карт, помеченных $\{1, \ldots, n\}$ и размещения их в стопку лицевой стороной вверх. Затем, если верхняя карта — $k > 1$, снимаем верхние k карт и возвращаем их по одной в стопку, тем самым заменяя перестановку $a_1 \ldots a_n$ перестановкой $a_k \ldots a_1 a_{k+1} \ldots a_n$. Так продолжается до тех пор, пока верхней картой не окажется 1. Например, при $n = 5$ может получиться семишаговая последовательность

$$31452 \to 41352 \to 53142 \to 24135 \to 42135 \to 31245 \to 21345 \to 12345.$$

Какая самая длинная последовательность может быть получена при $n = 13$?

108. [*M27*] Обозначим наибольшее количество ходов в пасьянсе "topswops" с n картами как $f(n)$. Докажите, что $f(n) \le F_{n+1} - 1$.

109. [*M47*] Найдите верхнюю и нижнюю границы для функции $f(n)$ из предыдущего упражнения.

▶ **110.** [*25*] Найдите все перестановки $a_0 \ldots a_9$ множества $\{0, \ldots, 9\}$, такие, что

$$\{a_0, a_2, a_3, a_7\} = \{2, 5, 7, 8\}, \qquad \{a_1, a_4, a_5\} = \{0, 3, 6\},$$
$$\{a_1, a_3, a_7, a_8\} = \{3, 4, 5, 7\}, \qquad \{a_0, a_3, a_4\} = \{0, 7, 8\}.$$

Предложите также алгоритм для решения больши́х задач такого типа.

▶ **111.** [*M25*] Было предложено несколько ориентированных на перестановки аналогов цикла де Брейна. Самый простой и красивый из них — *универсальный цикл перестановок*, предложенный Б. У. Джексоном (B. W. Jackson) в *Discrete Math.* **117** (1993), 141–150. Он представляет собой цикл из $n!$ цифр, таких, что каждая перестановка множества $\{1, \ldots, n\}$ возникает как блок из $n - 1$ последовательных цифр ровно один раз (с опущенным последним элементом, являющимся избыточным). Например, таким универсальным циклом перестановок для $n = 3$ является (121323), и он, по сути, единственный.

Докажите, что универсальный цикл перестановок существует для всех $n \ge 2$. Какой из универсальных циклов перестановок для $n = 4$ лексикографически наименьший?

▶ **112.** [*M30*] (А. Вильямс (A. Williams), 2007.) Продолжая выполнять упр. 111, *явно* постройте циклы.

а) Покажите, что универсальный цикл перестановок эквивалентен гамильтонову циклу графа Кейли с двумя генераторами: $\rho = (1\ 2\ \ldots\ n{-}1)$ и $\sigma = (1\ 2\ \ldots\ n)$.

б) Докажите, что любой гамильтонов путь в таком графе в действительности является гамильтоновым циклом.

в) Найдите такой путь в виде $\sigma^2 \rho^{n-3} \alpha_1 \ldots \sigma^2 \rho^{n-3} \alpha_{(n-1)!}$, $\alpha_j \in \{\rho, \sigma\}$ для $n \ge 3$.

113. [*HM43*] Сколько в точности имеется универсальных циклов для перестановок ≤ 9 объектов?

7.2.1.3. Генерация всех сочетаний. Комбинаторика часто описывается как "изучение перестановок, сочетаний и т. п.", так что теперь мы обратим наше внимание на сочетания. *Сочетание из n элементов по t*, которое часто называют просто t-сочетанием n элементов, представляет собой способ выбора подмножества размером t из данного множества размером n. Из уравнения 1.2.6–(2) мы знаем, что имеется ровно $\binom{n}{t}$ способов сделать это, а из раздела 3.4.2 — как выбрать t-сочетание случайным образом.

Выбор t из n объектов эквивалентен выбору $n-t$ оставшихся элементов. Эту симметрию можно подчеркнуть, полагая везде далее, что

$$n \;=\; s + t\,, \tag{1}$$

и говоря о t-сочетании n элементов как о "(s,t)-сочетании". Таким образом, (s,t)-сочетание представляет собой способ разделить $s+t$ объектов на два набора размером s и t.

Когда я спрашиваю, сколько существует способов выбрать 21 элемент из 25, на самом деле я спрашиваю, сколькими способами можно выбрать 4 элемента из 25. Способов выбрать 21 элемент столько же, сколько и оставить 4.

— ОГАСТЕС ДЕ МОРГАН (AUGUSTUS DE MORGAN), *Эссе о вероятностях (An Essay on Probabilities)* (1838)

Имеется два основных способа представления (s,t)-сочетаний: можно перечислить выбранные элементы $c_t \ldots c_2 c_1$, а можно работать с бинарными строками $a_{n-1} \ldots a_1 a_0$, для которых

$$a_{n-1} + \cdots + a_1 + a_0 \;=\; t. \tag{2}$$

Строковое представление содержит s нулей и t единиц, соответствующих невыбранным и выбранным элементам. Представление в виде списка $c_t \ldots c_2 c_1$ лучше подходит для случаев, когда элементы являются членами множества $\{0, 1, \ldots, n-1\}$ и мы перечисляем их в *убывающем* порядке:

$$n > c_t > \cdots > c_2 > c_1 \geq 0. \tag{3}$$

Бинарная запись отлично объединяет эти два представления, поскольку список элементов $c_t \ldots c_2 c_1$ соответствует сумме

$$2^{c_t} + \cdots + 2^{c_2} + 2^{c_1} \;=\; \sum_{k=0}^{n-1} a_k 2^k \;=\; (a_{n-1} \ldots a_1 a_0)_2. \tag{4}$$

Конечно, можно также перечислить позиции $b_s \ldots b_2 b_1$ нулей в строке $a_{n-1} \ldots a_1 a_0$, где

$$n > b_s > \cdots > b_2 > b_1 \geq 0. \tag{5}$$

Сочетания важны не только в силу своей вездесущести в математике, но и из-за их эквивалентности многим другим конфигурациям. Например, каждое (s,t)-сочетание соответствует сочетанию $s+1$ объекта по t *с повторениями*, именуемому

также *мультисочетанием* (multicombination) $s+1$ элемента, а именно последовательности целых чисел $d_t \ldots d_2 d_1$, обладающей тем свойством, что

$$s \geq d_t \geq \cdots \geq d_2 \geq d_1 \geq 0. \tag{6}$$

Одна из причин заключается в том, что $d_t \ldots d_2 d_1$ удовлетворяет (6) тогда и только тогда, когда $c_t \ldots c_2 c_1$ удовлетворяет (3), где

$$c_t = d_t + t - 1, \quad \ldots, \quad c_2 = d_2 + 1, \quad c_1 = d_1 \tag{7}$$

(см. упр. 1.2.6–60). Есть и другой полезный способ связать сочетания с повторениями и обычные сочетания, предложенный Соломоном Голомбом (Solomon Golomb) [*AMM* **75** (1968), 530–531], а именно — определить

$$e_j \;=\; \begin{cases} c_j, & \text{если } c_j \leq s; \\ e_{c_j - s}, & \text{если } c_j > s. \end{cases} \tag{8}$$

В этом виде числа $e_t \ldots e_1$ не обязательно находятся в убывающем порядке, но мультимножество $\{e_1, e_2, \ldots, e_t\}$ эквивалентно $\{c_1, c_2, \ldots, c_t\}$ тогда и только тогда, когда $\{e_1, e_2, \ldots, e_t\}$ является множеством (см. табл. 1 и упр. 1.)

(s,t)-сочетание также эквивалентно *композиции** (composition) $n+1$ из $t+1$ частей, а именно — упорядоченной сумме

$$n+1 \;=\; p_t + \cdots + p_1 + p_0, \qquad \text{где } p_t, \ldots, p_1, p_0 \geq 1. \tag{9}$$

Связь с (3) в этом случае имеет вид

$$p_t = n - c_t, \quad p_{t-1} = c_t - c_{t-1}, \quad \ldots, \quad p_1 = c_2 - c_1, \quad p_0 = c_1 + 1. \tag{10}$$

Аналогично, если $q_j = p_j - 1$, то мы получаем композицию s из $t+1$ *неотрицательных* частей

$$s \;=\; q_t + \cdots + q_1 + q_0, \qquad \text{где } q_t, \ldots, q_1, q_0 \geq 0, \tag{11}$$

которая связана с (6) соотношениями

$$q_t = s - d_t, \quad q_{t-1} = d_t - d_{t-1}, \quad \ldots, \quad q_1 = d_2 - d_1, \quad q_0 = d_1. \tag{12}$$

Кроме того, легко видеть, что (s,t)-сочетание эквивалентно пути длиной $s+t$ из угла в угол в сетке $s \times t$, потому что такой путь содержит s вертикальных шагов и t горизонтальных.

Таким образом, сочетания можно изучать как минимум в восьми различных обликах. В табл. 1 проиллюстрированы все $\binom{6}{3} = 20$ возможных сочетаний для случая $s = t = 3$.

На первый взгляд, такое обилие представлений может только запутать, но большинство из них можно легко вывести непосредственно из бинарного представления $a_{n-1} \ldots a_1 a_0$. Рассмотрим, например, "случайную" битовую строку

$$a_{23} \ldots a_1 a_0 \;=\; 011001001000011111101101, \tag{13}$$

---

* Здесь можно было бы говорить о *разбиении*, но во избежание путаницы в дальнейшем будем переводить composition как "композиция", чтобы отличать ее от неупорядоченной суммы, которой является разбиение. — *Примеч. пер.*

Таблица 1

(3, 3)-сочетания и их эквиваленты

$a_5a_4a_3a_2a_1a_0$	$b_3b_2b_1$	$c_3c_2c_1$	$d_3d_2d_1$	$e_3e_2e_1$	$p_3p_2p_1p_0$	$q_3q_2q_1q_0$	path
000111	543	210	000	210	4111	3000	
001011	542	310	100	310	3211	2100	
001101	541	320	110	320	3121	2010	
001110	540	321	111	321	3112	2001	
010011	532	410	200	010	2311	1200	
010101	531	420	210	020	2221	1110	
010110	530	421	211	121	2212	1101	
011001	521	430	220	030	2131	1020	
011010	520	431	221	131	2122	1011	
011100	510	432	222	232	2113	1002	
100011	432	510	300	110	1411	0300	
100101	431	520	310	220	1321	0210	
100110	430	521	311	221	1312	0201	
101001	421	530	320	330	1231	0120	
101010	420	531	321	331	1222	0111	
101100	410	532	322	332	1213	0102	
110001	321	540	330	000	1141	0030	
110010	320	541	331	111	1132	0021	
110100	310	542	332	222	1123	0012	
111000	210	543	333	333	1114	0003	

в которой содержится $s = 11$ нулей и $t = 13$ единиц, так что $n = 24$. Дуальное сочетание $b_s \ldots b_1$ перечисляет позиции нулей, а именно

$$23\ 20\ 19\ 17\ 16\ 14\ 13\ 12\ 11\ 4\ 1,$$

поскольку крайняя слева позиция имеет номер $n-1$, а крайняя справа — 0. Основное сочетание $c_t \ldots c_1$ перечисляет позиции единиц, а именно

$$22\ 21\ 18\ 15\ 10\ 9\ 8\ 7\ 6\ 5\ 3\ 2\ 0.$$

Соответствующее мультисочетание $d_t \ldots d_1$ указывает количество нулей справа от каждой единицы:

$$10\ 10\ 8\ 6\ 2\ 2\ 2\ 2\ 2\ 2\ 1\ 1\ 0.$$

Сочетание $p_t \ldots p_0$ перечисляет расстояния между последовательными единицами, если представить наличие фиктивных дополнительных единиц слева и справа от бинарной строки:

$$2\ 1\ 3\ 3\ 5\ 1\ 1\ 1\ 1\ 1\ 2\ 1\ 2\ 1.$$

И наконец неотрицательное сочетание $q_t \ldots q_0$ подсчитывает, сколько нулей находится между “ограждениями”-единицами:

$$1\ 0\ 2\ 2\ 4\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0;$$

т. е. мы имеем

$$a_{n-1} \ldots a_1 a_0 = 0^{q_t} 1 0^{q_{t-1}} 1 \ldots 1 0^{q_1} 1 0^{q_0}. \tag{14}$$

Пути в табл. 1 также имеют простую интерпретацию (см. упр. 2).

Лексикографическая генерация. В табл. 1 сочетания $a_{n-1}\ldots a_1a_0$ и $c_t\ldots c_1$ приведены в лексикографическом порядке, который также является лексикографическим порядком $d_t\ldots d_1$. Обратите внимание, что дуальные сочетания $b_s\ldots b_1$ и соответствующие композиции $p_t\ldots p_0$ и $q_t\ldots q_0$ расположены в *обратном* лексикографическом порядке.

Лексикографический порядок обычно предлагает наиболее удобный способ генерации комбинаторных конфигураций. В самом деле, алгоритм 7.2.1.2L уже решает задачу генерации сочетаний в форме $a_{n-1}\ldots a_1a_0$, поскольку (s,t)-сочетания в форме битовой строки представляют собой не что иное, как перестановки мультимножества $\{s\cdot 0, t\cdot 1\}$. Этот алгоритм общего назначения может быть упрощен при использовании для данного частного случая (см. также упр. 7.1.3–20, в котором представлена замечательная последовательность из семи битовых операций, преобразующих любое заданное бинарное число $(a_{n-1}\ldots a_1a_0)_2$ в лексикографически следующее t-сочетание в предположении, что n не превосходит длину машинного слова).

Однако обратимся к генерации сочетаний в ином виде, а именно $c_t\ldots c_2c_1$; этот вид в большей степени подходит для тех применений сочетаний, которые обычно требуются, и является более компактным, чем битовая строка, при значениях t, малых по сравнению с n. В первую очередь, мы должны вспомнить о том, что при очень небольших значениях t с задачей отлично справляются вложенные циклы. Например, при $t=3$ достаточно приведенных далее инструкций:

$$\begin{array}{l}\text{Для } c_3 = 2,\ 3,\ \ldots,\ n-1 \text{ (в указанном порядке) выполнить:}\\ \quad\text{Для } c_2 = 1,\ 2,\ \ldots,\ c_3-1 \text{ (в указанном порядке) выполнить:}\\ \qquad\text{Для } c_1 = 0,\ 1,\ \ldots,\ c_2-1 \text{ (в указанном порядке) выполнить:}\\ \qquad\quad\text{Посетить сочетание } c_3c_2c_1.\end{array}\tag{15}$$

(См. аналогичную ситуацию в 7.2.1.1–(3).)

С другой стороны, когда t — переменная или не столь малое значение, можно лексикографически генерировать сочетания с помощью следующего общего способа, обсуждавшегося после алгоритма 7.2.1.2L, а именно — поиска крайнего справа элемента c_j, который может быть увеличен, после чего все последующие элементы $c_{j-1}\ldots c_1$ получают минимально возможные значения.

Алгоритм L (*Сочетания в лексикографическом порядке*). Этот алгоритм генерирует все t-сочетания $c_t\ldots c_2c_1$ из n чисел $\{0,1,\ldots,n-1\}$ для данных $n\ge t\ge 0$. В качестве ограничителей используются дополнительные переменные c_{t+1} и c_{t+2}.

L1. [Инициализация.] Установить $c_j\leftarrow j-1$ для $1\le j\le t$; установить также $c_{t+1}\leftarrow n$ и $c_{t+2}\leftarrow 0$.

L2. [Посещение.] Посетить сочетание $c_t\ldots c_2c_1$.

L3. [Поиск j.] Установить $j\leftarrow 1$. Затем, пока $c_j+1=c_{j+1}$, установить $c_j\leftarrow j-1$ и $j\leftarrow j+1$. Эти действия повторяются до тех пор, пока в конечном итоге не выполнится условие $c_j+1\ne c_{j+1}$.

L4. [Выполнено?] Завершить выполнение алгоритма, если $j>t$.

L5. [Увеличение c_j.] Установить $c_j\leftarrow c_j+1$ и вернуться к шагу L2. ▮

Анализ времени работы данного алгоритма несложен. На шаге L3 устанавливается $c_j\leftarrow j-1$ сразу после посещения сочетания, для которого $c_{j+1}=c_1+j$, а количество

таких сочетаний равно количеству решений неравенств

$$n > c_t > \cdots > c_{j+1} \geq j; \tag{16}$$

однако эта формула представляет собой эквивалент $(t-j)$-сочетания из $n-j$ объектов $\{n-1, \ldots, j\}$, так что присваивание $c_j \leftarrow j-1$ выполняется ровно $\binom{n-j}{t-j}$ раз. Суммирование для $1 \leq j \leq t$ говорит нам, что цикл на шаге L3 выполняется

$$\binom{n-1}{t-1} + \binom{n-2}{t-2} + \cdots + \binom{n-t}{0} = \binom{n-1}{s} + \binom{n-2}{s} + \cdots + \binom{s}{s} = \binom{n}{s+1} \tag{17}$$

раз, или в среднем

$$\binom{n}{s+1} \Big/ \binom{n}{t} = \frac{n!}{(s+1)!\,(t-1)!} \Big/ \frac{n!}{s!\,t!} = \frac{t}{s+1} \tag{18}$$

выполнений за одно посещение. Это отношение меньше 1 при $t \leq s$, так что алгоритм L достаточно эффективен для таких случаев.

Однако величина $t/(s+1)$ может оказаться весьма большой, когда t близко к n, а s мало́. В самом деле, иногда алгоритм L выполняет присваивание $c_j \leftarrow j-1$ без необходимости, когда c_j и так равно $j-1$. Дальнейшее исследование показывает, что не всегда нужно искать индекс j, необходимый на шагах L4 и L5, поскольку корректное значение j зачастую можно предсказать на основе только что выполненных действий. Например, после того, как мы увеличили c_4 и сбросили $c_3c_2c_1$ в начальное значение 210, очередное сочетание неизбежно приведет к увеличению c_3. Эти наблюдения приводят к более эффективной версии алгоритма.

Алгоритм T (*Сочетания в лексикографическом порядке*). Этот алгоритм подобен алгоритму L, но работает быстрее. В нем также предполагается, что $0 < t < n$.

T1. [Инициализация.] Установить $c_j \leftarrow j-1$ для $1 \leq j \leq t$; затем установить $c_{t+1} \leftarrow n$, $c_{t+2} \leftarrow 0$ и $j \leftarrow t$.

T2. [Посещение.] (В этот момент j — наименьший индекс, такой, что $c_{j+1} > j$.) Посетить сочетание $c_t \ldots c_2c_1$. Затем, если $j > 0$, установить $x \leftarrow j$ и перейти к шагу T6.

T3. [Простой случай?] Если $c_1+1 < c_2$, установить $c_1 \leftarrow c_1+1$ и вернуться к шагу T2. В противном случае установить $j \leftarrow 2$.

T4. [Поиск j.] Установить $c_{j-1} \leftarrow j-2$ и $x \leftarrow c_j+1$. Если $x = c_{j+1}$, установить $j \leftarrow j+1$ и повторить шаг T4.

T5. [Выполнено?] Завершить выполнение алгоритма, если $j > t$.

T6. [Увеличение c_j.] Установить $c_j \leftarrow x$, $j \leftarrow j-1$ и вернуться к шагу T2. ▮

Теперь на шаге T2 $j = 0$ тогда и только тогда, когда $c_1 > 0$, так что присваивания на шаге T4 никогда не будут избыточными. Полный анализ алгоритма T можно найти в упр. 6.

Обратите внимание, что параметр n используется только инициализирующими шагами L1 и T1, но не в основной части алгоритмов L и T. Таким образом, можно рассматривать процесс генерации первых $\binom{n}{t}$ сочетаний из *бесконечного* списка, которые зависят только от t. Это упрощение является результатом того, что при наших соглашениях список t-сочетаний $n+1$ элемента начинается со списка t-сочетаний n элементов; именно по этой причине мы использовали лексикографический

порядок уменьшающихся последовательностей $c_t \ldots c_1$, а не возрастающие последовательности $c_1 \ldots c_t$.

Деррик Лемер (Derrick Lehmer) заметил еще одно приятное свойство алгоритмов L и T [*Applied Combinatorial Mathematics*, ed. by E. F. Beckenbach (1964), 27–30].

Теорема L. *Сочетание $c_t \ldots c_2 c_1$ посещается после посещения ровно*

$$\binom{c_t}{t} + \cdots + \binom{c_2}{2} + \binom{c_1}{1} \tag{19}$$

других сочетаний.

Доказательство. Имеется $\binom{c_k}{k}$ сочетаний $c'_t \ldots c'_2 c'_1$, таких, что $c'_j = c_j$ для $t \geq j > k$ и $c'_k < c_k$, а именно $c_t \ldots c_{k+1}$, за которым следуют k-сочетания множества $\{0, \ldots, c_k - 1\}$. ▮

Когда, например, $t = 3$, числа

$$\binom{2}{3} + \binom{1}{2} + \binom{0}{1},\ \binom{3}{3} + \binom{1}{2} + \binom{0}{1},\ \binom{3}{3} + \binom{2}{2} + \binom{0}{1},\ \ldots,\ \binom{5}{3} + \binom{4}{2} + \binom{3}{1},$$

соответствующие сочетаниям $c_3 c_2 c_1$ в табл. 1, просто пробегают последовательность $0, 1, 2, \ldots, 19$. Теорема L дает нам простой способ понимания *комбинаторной системы счисления* степени t, которая позволяет представить каждое неотрицательное целое число N в виде суммы

$$N = \binom{n_t}{t} + \cdots + \binom{n_2}{2} + \binom{n_1}{1}, \qquad n_t > \cdots > n_2 > n_1 \geq 0, \tag{20}$$

единственным образом. [See Ernesto Pascal, *Giornale di Matematiche* **25** (1887), 45–49.]

Биномиальные деревья. Семейство деревьев T_n, определяемое как

$T_0 = \circ$, $\qquad T_n =$

для $n > 0$, (21)

возникает в различных важных контекстах и проливает новый свет на генерацию сочетаний. Например, T_4 представляет собой

; (22)

а дерево T_5 в более художественном исполнении можно увидеть на фронтисписе тома 1 этого многотомника.

Заметим, что дерево T_n очень похоже на дерево T_{n-1}, за исключением дополнительной копии T_{n-1}; таким образом, всего в дереве T_n имеется 2^n узлов. Кроме того, количество узлов на уровне t равно биномиальному коэффициенту $\binom{n}{t}$ — именно благодаря этому факту и возникло название “биномиальное дерево”. В самом деле,

последовательность меток, встречающихся на пути от корня к каждому узлу на уровне t, определяет сочетание $c_t \ldots c_1$, и все сочетания слева направо располагаются в лексикографическом порядке. Таким образом, алгоритмы L и T могут рассматриваться как процедуры обхода узлов биномиального дерева T_n, находящихся на уровне t.

Бесконечное биномиальное дерево T_∞ получается из (21) при $n \to \infty$. Корень этого дерева имеет бесконечно много ветвей, но каждый узел, за исключением корня на уровне 0, представляет собой корень конечного биномиального поддерева. Все возможные t-сочетания появляются в лексикографическом порядке на уровне t дерева T_∞.

Давайте познакомимся с биномиальными деревьями поближе, рассмотрев все возможные способы упаковки рюкзака. Говоря более строго, предположим, что у нас есть n предметов, которые занимают соответственно $w_{n-1}, \ldots, w_1, w_0$ единиц емкости рюкзака, где

$$w_{n-1} \geq \cdots \geq w_1 \geq w_0 \geq 0. \tag{23}$$

Мы хотим сгенерировать все бинарные векторы $a_{n-1} \ldots a_1 a_0$, такие, что

$$a \cdot w = a_{n-1}w_{n-1} + \cdots + a_1 w_1 + a_0 w_0 \leq N, \tag{24}$$

где N — общая емкость рюкзака. Задачу можно сформулировать эквивалентным образом — найти все подмножества C множества $\{0, 1, \ldots, n-1\}$, такие, что $w(C) = \sum_{c \in C} w_c \leq N$; такие подмножества будем называть *допустимыми*. Мы будем записывать допустимые подмножества как $c_1 \ldots c_t$, где $c_1 > \cdots > c_t \geq 0$. Нумерация индексов отличается от нумерации, принятой в соглашении (3) выше, поскольку в этой задаче t является переменной.

Каждое допустимое подмножество соответствует узлу T_n, и наша цель состоит в том, чтобы обойти все допустимые узлы. Ясно, что родительский узел допустимого узла также является допустимым, и то же самое относится и к левому "братскому" узлу, если таковой имеется. Следовательно, описанная ниже простая процедура вполне работоспособна.

Алгоритм F (*Заполнение рюкзака*). Этот алгоритм генерирует все возможные способы $c_1 \ldots c_t$ заполнения рюкзака для заданных $w_{n-1}, \ldots, w_1, w_0$, и N. Обозначим $\delta_j = w_j - w_{j-1}$ для $1 \leq j < n$.

F1. [Инициализация.] Установить $t \leftarrow 0$, $c_0 \leftarrow n$ и $r \leftarrow N$.

F2. [Посещение.] Посетить сочетание $c_1 \ldots c_t$, которое использует $N - r$ единиц емкости рюкзака.

F3. [Попытка добавить w_0.] Если $c_t > 0$ и $r \geq w_0$, установить $t \leftarrow t + 1$, $c_t \leftarrow 0$, $r \leftarrow r - w_0$ и вернуться к шагу F2.

F4. [Попытка увеличить c_t.] Если $t = 0$, завершить работу алгоритма. В противном случае, если $c_{t-1} > c_t + 1$ и $r \geq \delta_{c_t+1}$, установить $c_t \leftarrow c_t + 1$, $r \leftarrow r - \delta_{c_t}$ и вернуться к шагу F2.

F5. [Удаление c_t.] Установить $r \leftarrow r + w_{c_t}$, $t \leftarrow t - 1$ и вернуться к шагу F4. ▮

Обратите внимание, что этот алгоритм неявно посещает узлы T_n в прямом порядке, пропуская недопустимые поддеревья. Элемент $c > 0$ размещается в рюкзаке, если он может поместиться в нем, сразу после того, как процедура исследует все

возможности использования на этом месте элемента $c-1$. Время работы алгоритма пропорционально количеству посещенных допустимых сочетаний (см. упр. 20).

Кстати, классическая "задача о рюкзаке" из исследования операций отличается от описанной: в ней требуется найти допустимое подмножество C, такое, что $v(C) = \sum_{c\in C} v(c)$ максимально, где каждому предмету c назначается значение $v(c)$. Алгоритм F не особенно хорош для решения этой задачи, так как он часто рассматривает случаи, которые могут быть исключены. Например, если C и C' представляют собой подмножества $\{1,\ldots,n-1\}$, причем $w(C) \le w(C') \le N - w_0$ и $v(C) \ge v(C')$, алгоритм F исследует как $C\cup 0$, так и $C'\cup 0$, но последнее подмножество никогда не улучшит максимум. Мы рассмотрим методы для решения задачи о рюкзаке позже; алгоритм F предназначен только для тех случаев, когда *все* допустимые решения могут оказаться потенциально оптимальными.

Коды Грея для сочетаний. Вместо простой генерации всех сочетаний зачастую предпочтительно их посещение таким образом, что каждое из них получается путем малого изменения его предшественника.

Например, мы можем потребовать то, что Ньенхьюз (Nijenhuis) и Вильф (Wilf) назвали "алгоритм двери-вертушки" (revolving door). Представим две комнаты, в которых находятся соответственно s и t человек, и дверь-вертушку между ними. Когда кто-то входит в одну из комнат, другой одновременно ее покидает. Можно ли разработать такую последовательность шагов, чтобы каждое (s,t)-сочетание встречалось ровно по одному разу?

Ответ — да, причем на самом деле существует огромное количество таких схем. Например, одну из них можно получить, если исследовать все n-битовые строки $a_{n-1}\ldots a_1a_0$ в хорошо известном порядке, соответствующем бинарному коду Грея (раздел 7.2.1.1), но выбрать из них только те, которые имеют ровно s нулей и t единиц. В результате получатся строки, образующие код двери-вертушки.

Вот доказательство этого факта. Бинарный код Грея определяется рекуррентным соотношением $\Gamma_n = 0\Gamma_{n-1},\ 1\Gamma^R_{n-1}$ из 7.2.1.1–(5), так что его (s,t)-подпоследовательность удовлетворяет рекуррентному соотношению

$$\Gamma_{st} \;=\; 0\Gamma_{(s-1)t},\ 1\Gamma^R_{s(t-1)}, \tag{25}$$

где $st > 0$. Мы также имеем $\Gamma_{s0} = 0^s$ и $\Gamma_{0t} = 1^t$. Следовательно, по индукции понятно, что Γ_{st} начинается с 0^s1^t и заканчивается 10^s1^{t-1} при $st > 0$. Переход, соответствующий запятой в (25), выполняется от последнего элемента $0\Gamma_{(s-1)t}$ к последнему элементу $1\Gamma_{s(t-1)}$, а именно от $010^{s-1}1^{t-1} = 010^{s-1}11^{t-2}$ к $110^s1^{t-2} = 110^{s-1}01^{t-2}$ при $t \ge 2$, и удовлетворяет ограничению, накладываемому правилом двери-вертушки. Случай $t = 1$ также подтверждается. Например, Γ_{33} задается столбцами

000111	011010	110001	101010
001101	011100	110010	101100
001110	010101	110100	100101
001011	010110	111000	100110
011001	010011	101001	100011

(26)

и Γ_{23} можно обнаружить в первых двух столбцах этого массива. Еще один поворот двери выполняет переход от последнего элемента к первому. [Эти свойства Γ_{st}

были открыты Д. Э. Миллер (J. E. Miller) в ее диссертации (Columbia University, 1971), а затем независимо Д. Т. Тангом (D. T. Tang) и Ч. Н. Лю (C. N. Liu), *IEEE Trans.* **C-22** (1973), 176–180. Реализация без использования циклов была представлена Д. Р. Битнером (J. R. Bitner), Г. Эрлихом (G. Ehrlich) и Э. М. Рейнгольдом (E. M. Reingold), *CACM* **19** (1976), 517–521.]

При преобразовании битовых строк $a_5a_4a_3a_2a_1a_0$ из (26) в соответствующие списки индексов $c_3c_2c_1$ схема сочетаний становится очевидной.

$$\begin{array}{llll} 210 & 431 & 540 & 531 \\ 320 & 432 & 541 & 532 \\ 321 & 420 & 542 & 520 \\ 310 & 421 & 543 & 521 \\ 430 & 410 & 530 & 510 \end{array} \tag{27}$$

Первый компонент c_3 появляется в неубывающем порядке, но для каждого фиксированного значения c_3 значения c_2 находятся в не*возрастающем* порядке. Для фиксированных же пар значений c_3c_2 значения c_1 вновь находятся в неубывающем порядке. То же самое справедливо и в общем случае: *все сочетания* $c_t \ldots c_2c_1$ *в коде Грея двери-вертушки* Γ_{st} *находятся в лексикографическом порядке*

$$(c_t,\ -c_{t-1},\ c_{t-2},\ \ldots,\ (-1)^{t-1}c_1). \tag{28}$$

Это свойство доказывается по индукции, поскольку при использовании списка индексов вместо битовой строки при $st > 0$ (25) записывается как

$$\Gamma_{st} \;=\; \Gamma_{(s-1)t},\ (s+t-1)\Gamma^R_{s(t-1)}. \tag{29}$$

Таким образом, эта последовательность может эффективно генерироваться с помощью следующего алгоритма В. Г. Пейна (W. H. Payne) [см. *ACM Trans. Math. Software* **5** (1979), 163–172].

Алгоритм R (*Сочетания двери-вертушки*). Этот алгоритм генерирует все t-сочетания $c_t \ldots c_2c_1$ из $\{0, 1, \ldots, n-1\}$ в лексикографическом порядке чередующейся последовательности (28), в предположении, что $n \geq t > 1$. Используется вспомагательная переменная c_{t+1}. Шаг R3 имеет два варианта — в зависимости от того, четно значение t или нечетно.

R1. [Инициализация.] Установить $c_j \leftarrow j-1$ для $t \geq j \geq 1$ и $c_{t+1} \leftarrow n$.

R2. [Посещение.] Посетить сочетание $c_t \ldots c_2c_1$.

R3. [Простой случай?] Если t нечетно: если $c_1+1 < c_2$, увеличить c_1 на 1 и вернуться к шагу R2, в противном случае установить $j \leftarrow 2$ и перейти к шагу R4. Если t четно: если $c_1 > 0$, уменьшить c_1 на 1 и вернуться к шагу R2, в противном случае установить $j \leftarrow 2$ и перейти к шагу R5.

R4. [Попытка уменьшения c_j.] (В этот момент $c_j = c_{j-1}+1$.) Если $c_j \geq j$, установить $c_j \leftarrow c_{j-1}$, $c_{j-1} \leftarrow j-2$ и вернуться к шагу R2. В противном случае увеличить j на 1.

R5. [Попытка увеличения c_j.] (В этот момент $c_{j-1} = j-2$.) Если $c_j + 1 < c_{j+1}$, установить $c_{j-1} \leftarrow c_j$, $c_j \leftarrow c_j + 1$ и вернуться к шагу R2. В противном случае увеличить j на 1 и, если $j \leq t$, перейти к шагу R4. В противном случае алгоритм завершает работу. ▌

В упр. 21–25 открываются новые свойства этой интересной последовательности. Одно из них представляет собой красивую пару для теоремы L: *сочетание* $c_t c_{t-1} \ldots c_2 c_1$ *посещается алгоритмом* R *после посещения ровно*

$$N = \binom{c_t+1}{t} - \binom{c_{t-1}+1}{t-1} + \cdots + (-1)^t \binom{c_2+1}{2} - (-1)^t \binom{c_1+1}{1} - [t \text{ нечетно}] \quad (30)$$

других сочетаний. Это представление числа N можно назвать знакопеременной комбинаторной системой счисления степени t. Одно из следствий, например, заключается в том, что любое положительное целое число может быть единственным образом представлено в виде $N = \binom{a}{3} - \binom{b}{2} + \binom{c}{1}$, где $a > b > c > 0$. Алгоритм R говорит нам, как добавить 1 к N в этой системе счисления.

Хотя строки (26) и (27) не находятся в лексикографическом порядке, они представляют собой пример более общей концепции, называющейся *облексным порядком* (genlex order), название которой придумано Тимоти Уолшем (Timothy Walsh). Говорят, что последовательность строк $\alpha_1, \ldots, \alpha_N$ находится в облексном порядке, когда все строки с общим префиксом расположены последовательно. Например, все 3-сочетания, начинающиеся с 53, в (27) находятся рядом.

Облексный порядок означает, что строки могут быть размещены в структуре луча (trie), как на рис. 31 в разделе 6.3, но с произвольным упорядочением дочерних узлов каждого узла. При обходе луча в любом порядке, таком, что каждый узел посещается непосредственно до или после своих потомков, все узлы с общим префиксом — т. е. все узлы подлуча — обходятся последовательно. Это делает облексный порядок весьма удобным, поскольку он соответствует рекурсивным схемам генерации. Многие алгоритмы генерации n-кортежей, с которыми мы встречались, выдают тот или иной облексный порядок. Подобным образом метод "простых изменений" (plain changes) (алгоритм 7.2.1.2P) посещает перестановки в облексном порядке соответствующей таблицы инверсий.

Метод двери-вертушки из алгоритма R представляет собой облексную подпрограмму, которая на каждом шаге изменяет только один элемент сочетания. Однако этот принцип не всегда выдерживается, так как зачастую приходится одновременно изменять два индекса c_j, чтобы сохранялось условие $c_t > \cdots > c_2 > c_1$. Например, алгоритм R изменяет 210 на 320, и в (27) можно найти девять таких "перекрестных" изменений.

Источник этого дефекта может быть отслежен в доказательстве того, что (25) удовлетворяет свойству двери-вертушки: мы заметили, что, когда $t \geq 2$, за строкой $010^{s-1}11^{t-2}$ следует строка $110^{s-1}01^{t-2}$. Следовательно, рекурсивное построение Γ_{st} включает переходы вида $110^a0 \leftrightarrow 010^a1$, когда подстрока наподобие 11000 сменяется подстрокой 01001 или наоборот — словом, когда две единицы пересекают одна другую.

Путь Грея для сочетаний называется *гомогенным*, или *однородным* (*homogeneous*), если на каждом шаге изменяется только один из индексов c_j. Гомогенная схема в виде битовой строки характеризуется тем, что содержит только переходы вида $10^a \leftrightarrow 0^a1$, $a \geq 1$ при переходе от одной строки к следующей. При гомоген-

ной схеме мы можем, например, сыграть все t-нотные аккорды на n-нотной клавиатуре, перемещая при переходе от одного аккорда к следующему только один палец.

Небольшая модификация (25) дает нам гомогенную облексную схему для (s,t)-сочетаний. Основная идея состоит в построении последовательности, которая начинается с 0^s1^t и заканчивается 1^t0^s — при этом тут же напрашивается следующая рекурсия: пусть $K_{s0} = 0^s$, $K_{0t} = 1^t$, $K_{s(-1)} = \emptyset$ и

$$K_{st} \;=\; 0K_{(s-1)t},\; 10K^R_{(s-1)(t-1)},\; 11K_{s(t-2)} \quad \text{для } st > 0. \tag{31}$$

В местах запятых в этой последовательности расположены строки 01^t0^{s-1}, за которой следует $101^{t-1}0^{s-1}$, и 10^s1^{t-1}, за которой следует 110^s1^{t-2}. Оба эти перехода однородны, хотя второй и требует переноса единицы через s нулей. Сочетания K_{33} при $s = t = 3$ представляют собой

$$\begin{array}{llll} 000111 & 010101 & 101100 & 100011 \\ 001011 & 010011 & 101001 & 110001 \\ 001101 & 011001 & 101010 & 110010 \\ 001110 & 011010 & 100110 & 110100 \\ 010110 & 011100 & 100101 & 111000 \end{array} \tag{32}$$

в форме битовых строк, а соответствующая схема “расположения пальцев” имеет вид

$$\begin{array}{llll} 210 & 420 & 532 & 510 \\ 310 & 410 & 530 & 540 \\ 320 & 430 & 531 & 541 \\ 321 & 431 & 521 & 542 \\ 421 & 432 & 520 & 543 \end{array}\;. \tag{33}$$

При преобразовании однородной схемы для обычных сочетаний $c_t \ldots c_1$ в соответствующую схему (6) для сочетаний с повторениями $d_t \ldots d_1$ сохраняется свойство, что на каждом шаге изменяется только один индекс d_j. При конвертации в соответствующие схемы (9) и (11) для композиций $p_t \ldots p_0$ и $q_t \ldots q_0$ при изменении c_j происходит изменение только двух (соседних) частей.

Схемы, близкие к идеальной. Однако можно поступить еще лучше! Все (s,t)-сочетания могут быть сгенерированы последовательностью строго гомогенных переходов, представляющих собой либо $01 \leftrightarrow 10$, либо $001 \leftrightarrow 100$. Другими словами, можно потребовать, чтобы на каждом шаге единственный индекс c_j изменялся не более чем на 2. Назовем такие схемы *близкими к идеальным* (*near-perfect*).

Столь строгое требование в действительности делает разработку близких к идеальной схем достаточно простой, поскольку доступным оказывается только сравнительно небольшое количество вариантов. Т. А. Дженкинс (T. A. Jenkyns) и Д. МакКарти (D. McCarthy) заметили, что если ограничиться облексными методами, близкими к идеальным на n-битовых строках, то такие методы можно легко охарактеризовать следующей теоремой [*Ars Combinatoria* **40** (1995), 153–159].

Теорема N. *Если $st > 0$, существует ровно $2s$ близких к идеальному способов перечисления всех (s,t)-сочетаний в облексном порядке. В действительности для каждого значения $1 \le a \le s$ имеется ровно одна такая последовательность N_{sta}, которая начинается с 1^t0^s и заканчивается $0^a1^t0^{s-a}$; прочие s возможностей являются обратными последовательностями N^R_{sta}.*

Таблица 2

Последовательности Чейза для $(3,3)$-сочетаний

$A_{33} = \widehat{C}^R_{33}$				$B_{33} = C_{33}$			
543	531	321	420	543	520	432	410
541	530	320	421	542	510	430	210
540	510	310	431	540	530	431	310
542	520	210	430	541	531	421	320
532	521	410	432	521	532	420	321

Доказательство. Теорема, определенно, справедлива при $s = t = 1$; для прочих случаев мы используем индукцию по $s + t$. Последовательность N_{sta}, если таковая существует, должна иметь вид $1X_{s(t-1)}$, $0Y_{(s-1)t}$ для некоторых близких к идеальным облексных последовательностей $X_{s(t-1)}$ и $Y_{(s-1)t}$. Если $t = 1$, то последовательность $X_{s(t-1)}$ представляет собой единственную строку 0^s; следовательно, $Y_{(s-1)t}$ должна представлять собой $N_{(s-1)1(a-1)}$ при $a > 1$ и $N^R_{(s-1)11}$ при $a = 1$. С другой стороны, если $t > 1$, из условия близости к идеалу вытекает, что последняя строка $X_{s(t-1)}$ не может начинаться с 1; следовательно, $X_{s(t-1)} = N_{s(t-1)b}$ для некоторого b. Если $a > 1$, $Y_{(s-1)t}$ должно представлять собой $N_{(s-1)t(a-1)}$, следовательно, b должно быть 1; аналогично b должно быть 1 при $s = 1$. В противном случае мы имеем $a = 1 < s$, и это приводит к $Y_{(s-1)t} = N^R_{(s-1)tc}$ для некоторого c. Переход от $10^b1^{t-1}0^{s-b}$ к $0^{c+1}1^t0^{s-1-c}$ является близким к идеальному, только если $c = 1$ и $b = 2$. ▮

Доказательство теоремы N дает следующую рекурсивную формулу при $st > 0$:

$$N_{sta} = \begin{cases} 1N_{s(t-1)1},\ 0N_{(s-1)t(a-1)}, & \text{если } 1 < a \le s; \\ 1N_{s(t-1)2},\ 0N^R_{(s-1)t1}, & \text{если } 1 = a < s; \\ 1N_{1(t-1)1},\ 01^t, & \text{если } 1 = a = s. \end{cases} \tag{34}$$

Конечно, также $N_{s0a} = 0^s$.

Пусть $A_{st} = N_{st1}$ и $B_{st} = N_{st2}$. Эти близкие к идеальным последовательности, открытые Филлипом Дж. Чейзом (Phillip J. Chase) в 1976 году, образуются в результате сдвига крайнего слева блока единиц вправо на одну или две позиции соответственно и удовлетворяют следующим взаимно рекуррентным соотношениям:

$$A_{st} = 1B_{s(t-1)},\ 0A^R_{(s-1)t}; \qquad B_{st} = 1A_{s(t-1)},\ 0A_{(s-1)t}. \tag{35}$$

"Чтобы сделать один шаг вперед, сделайте два шага вперед и один шаг назад; чтобы сделать два шага вперед, сделайте первый шаг, а затем другой". Эти уравнения выполняются для всех целых значений s и t, если мы определим A_{st} и B_{st} как $\emptyset$ при отрицательных s или t, за исключением $A_{00} = B_{00} = \epsilon$ (пустая строка). Таким образом, A_{st} в действительности выполняет $\min(s,1)$ шагов вперед, а B_{st} — $\min(s,2)$ шагов вперед. Например, в табл. 2 приведены листинги для $s = t = 3$ с применением эквивалентной записи в виде списка индексов $c_3c_2c_1$ вместо битовой строки $a_5a_4a_3a_2a_1a_0$.

Чейз заметил, что компьютерная реализация этих последовательностей становится проще, если определить

$$C_{st} = \begin{cases} A_{st}, & \text{если } s+t \text{ нечетно}; \\ B_{st}, & \text{если } s+t \text{ четно}; \end{cases} \qquad \widehat{C}_{st} = \begin{cases} A^R_{st}, & \text{если } s+t \text{ четно}; \\ B^R_{st}, & \text{если } s+t \text{ нечетно}. \end{cases} \tag{36}$$

[См. *Congressus Numerantium* **69** (1989), 215–242.] Тогда

$$C_{st} = \begin{cases} 1C_{s(t-1)},\ 0\widehat{C}_{(s-1)t}, & \text{если } s+t \text{ нечетно;} \\ 1C_{s(t-1)},\ 0C_{(s-1)t}, & \text{если } s+t \text{ четно;} \end{cases} \tag{37}$$

$$\widehat{C}_{st} = \begin{cases} 0C_{(s-1)t},\ 1\widehat{C}_{s(t-1)}, & \text{если } s+t \text{ четно;} \\ 0\widehat{C}_{(s-1)t},\ 1\widehat{C}_{s(t-1)}, & \text{если } s+t \text{ нечетно.} \end{cases} \tag{38}$$

Когда бит a_j готов к изменению, путем проверки j на четность можно сказать, в каком месте рекурсии мы находимся.

На самом деле последовательность C_{st} может быть сгенерирована удивительно простым алгоритмом, основанным на общих идеях, применимых к *любой* облексной схеме. Будем говорить, что бит a_j в облексном алгоритме *активный*, если его изменение предполагается до того, как будет изменено что-либо слева от него (другими словами, узел активного бита в соответствующем луче не является крайним справа дочерним узлом своего родителя). Предположим, что у нас есть вспомогательная таблица $w_n \ldots w_1 w_0$, где $w_j = 1$ тогда и только тогда, когда бит a_j активен или когда $j < r$, где r — наименьший индекс, такой, что $a_r \neq a_0$; мы также полагаем $w_n = 1$. Тогда для поиска следующей за $a_{n-1} \ldots a_1 a_0$ битовой строки можно использовать приведенный ниже метод.

> Установить $j \leftarrow r$. Если $w_j = 0$, установить $w_j \leftarrow 1$, $j \leftarrow j+1$ и повторять эти действия до тех пор, пока не будет достигнуто $w_j = 1$. Если $j = n$, завершить выполнение алгоритма; в противном случае установить $w_j \leftarrow 0$. Изменить a_j на $1 - a_j$ и выполнить все прочие изменения в $a_{j-1} \ldots a_0$ и r, которые применяются в конкретной используемой облексной схеме. (39)

Красота этого подхода основана на том факте, что цикл гарантированно эффективен: можно доказать, что операция $j \leftarrow j+1$ будет выполняться в среднем менее одного раза для каждого шага генерации (см. упр. 36).

Анализируя переходы при изменении битов в (37) и (38), можно легко конкретизировать остальные детали.

Алгоритм C (*Последовательность Чейза*). Этот алгоритм посещает все (s,t)-сочетания $a_{n-1} \ldots a_1 a_0$, где $n = s+t$, в близкой к идеальной последовательности Чейза C_{st}.

C1. [Инициализация.] Установить $a_j \leftarrow 0$ для $0 \le j < s$, $a_j \leftarrow 1$ для $s \le j < n$ и $w_j \leftarrow 1$ для $0 \le j \le n$. Если $s > 0$, установить $r \leftarrow s$; в противном случае установить $r \leftarrow t$.

C2. [Посещение.] Посетить сочетание $a_{n-1} \ldots a_1 a_0$.

C3. [Поиск j и ветвление.] Установить $j \leftarrow r$. Если $w_j = 0$, установить $w_j \leftarrow 1$ и $j \leftarrow j+1$ и повторять эти действия до тех пор, пока $w_j = 0$. Прекратить работу алгоритма, если $j = n$; в противном случае установить $w_j \leftarrow 0$ и выполнить ветвление: перейти к C4, если j нечетно и $a_j \neq 0$; к C5, если j четно и $a_j \neq 0$; к C6, если j четно и $a_j = 0$; и к C7, если j нечетно и $a_j = 0$.

C4. [Переход вправо на единицу.] Установить $a_{j-1} \leftarrow 1$, $a_j \leftarrow 0$. Если $r = j$ и $j > 1$, установить $r \leftarrow j-1$; в противном случае при $r = j-1$ установить $r \leftarrow j$. Вернуться к шагу C2.

C5. [Переход вправо на два.] Если $a_{j-2} \neq 0$, перейти к шагу C4. В противном случае установить $a_{j-2} \leftarrow 1$, $a_j \leftarrow 0$. Если $r = j$, установить $r \leftarrow \max(j-2, 1)$; в противном случае, если $r = j-2$, установить $r \leftarrow j-1$. Вернуться к шагу C2.

C6. [Переход влево на единицу.] Установить $a_j \leftarrow 1$, $a_{j-1} \leftarrow 0$. Если $r = j$ и $j > 1$, установить $r \leftarrow j-1$; в противном случае при $r = j-1$ установить $r \leftarrow j$. Вернуться к шагу C2.

C7. [Переход влево на два.] Если $a_{j-1} \neq 0$, перейти к шагу C6. В противном случае установить $a_j \leftarrow 1$, $a_{j-2} \leftarrow 0$. Если $r = j-2$, установить $r \leftarrow j$; в противном случае при $r = j-1$ установить $r \leftarrow j-2$. Вернуться к шагу C2. ▮

***Анализ последовательности Чейза.** Магические свойства алгоритма C требуют дальнейшего исследования, и более близкое рассмотрение оказывается весьма поучительным. Для данной битовой строки $a_{n-1} \ldots a_1 a_0$ определим $a_n = 1$, $u_n = n \bmod 2$ и

$$u_j = (1 - u_{j+1}) a_{j+1}, \quad v_j = (u_j + j) \bmod 2, \quad w_j = (v_j + a_j) \bmod 2 \tag{40}$$

для $n > j \geq 0$. Например, у нас может быть $n = 26$ и

$$\begin{aligned} a_{25} \ldots a_1 a_0 &= 11001001000011111101101010, \\ u_{25} \ldots u_1 u_0 &= 10100100100001010100100101, \\ v_{25} \ldots v_1 v_0 &= 00001110001011111110001111, \\ w_{25} \ldots w_1 w_0 &= 11000111001000000011100101. \end{aligned} \tag{41}$$

Используя эти определения, по индукции можно доказать, что $v_j = 0$ тогда и только тогда, когда бит a_j в рекуррентных соотношениях (37)–(38), которые генерируют $a_{n-1} \ldots a_1 a_0$, "управляется" C, а не $\widehat{C}$, за исключением случая, когда a_j является частью заключительной серии нулей или единиц с правого конца. Следовательно, w_j для $r \leq j < n$ согласуется со значением, вычисленным алгоритмом C в момент, когда посещается $a_{n-1} \ldots a_1 a_0$. Эти формулы могут использоваться для определения точного места появления данного сочетания в последовательности Чейза (см. упр. 39).

Если мы хотим работать со списком индексов $c_t \ldots c_2 c_1$, а не с битовыми строками $a_{n-1} \ldots a_1 a_0$, удобно слегка изменить обозначения, записав $C_t(n)$ вместо C_{st} и $\widehat{C}_t(n)$ вместо $\widehat{C}_{st}$, где $s + t = n$. Тогда $C_0(n) = \widehat{C}_0(n) = \epsilon$, а рекуррентные соотношения для $t \geq 0$ приобретают вид

$$C_{t+1}(n+1) = \begin{cases} nC_t(n),\ \widehat{C}_{t+1}(n), & \text{если } n \text{ четно;} \\ nC_t(n),\ C_{t+1}(n), & \text{если } n \text{ нечетно;} \end{cases} \tag{42}$$

$$\widehat{C}_{t+1}(n+1) = \begin{cases} C_{t+1}(n),\ n\widehat{C}_t(n), & \text{если } n \text{ нечетно;} \\ \widehat{C}_{t+1}(n),\ n\widehat{C}_t(n), & \text{если } n \text{ четно.} \end{cases} \tag{43}$$

Эти новые уравнения могут быть развернуты, что дает, например

$$\begin{aligned}
C_{t+1}(9) &= 8C_t(8),\ 6C_t(6),\ 4C_t(4),\ \ldots,\ 3\widehat{C}_t(3),\ 5\widehat{C}_t(5),\ 7\widehat{C}_t(7);\\
C_{t+1}(8) &= 7C_t(7),\ 6C_t(6),\ 4C_t(4),\ \ldots,\ 3\widehat{C}_t(3),\ 5\widehat{C}_t(5);\\
\widehat{C}_{t+1}(9) &= \qquad 6C_t(6),\ 4C_t(4),\ \ldots,\ 3\widehat{C}_t(3),\ 5\widehat{C}_t(5),\ 7\widehat{C}_t(7),\ 8\widehat{C}_t(8);\\
\widehat{C}_{t+1}(8) &= \qquad 6C_t(6),\ 4C_t(4),\ \ldots,\ 3\widehat{C}_t(3),\ 5\widehat{C}_t(5),\ 7\widehat{C}_t(7).
\end{aligned} \tag{44}$$

Обратите внимание на то, что одна и та же схема доминирует во всех четырех последовательностях. Смысл "..." в средине зависит от значения t: мы просто пропускаем все члены $nC_t(n)$ и $n\widehat{C}_t(n)$ при $n < t$.

За исключением краевых эффектов в самом начале и конце, все раскрытия формул в (44) основаны на бесконечной прогрессии

$$\ldots,\ 10,\ 8,\ 6,\ 4,\ 2,\ 0,\ 1,\ 3,\ 5,\ 7,\ 9,\ \ldots, \tag{45}$$

которая представляет собой естественный способ размещения неотрицательных чисел в бесконечной в обе стороны последовательности. Если опустить в (45) все члены, меньшие t, для некоторого заданного целого $t \geq 0$, то оставшиеся члены будут поддерживать то свойство, что соседние элементы отличаются либо на 1, либо на 2. Ричард Стенли (Richard Stanley) предложил для этой последовательности название *endo-order*, исходя из первых букв фразы "even numbers decreasing, odd..." ("четные числа уменьшаются, нечетные...”*). (Заметим, что если оставить только члены, меньшие N, и дополнить их до N, то эта последовательность превратится в последовательность "органных труб"; см. упр. 6.1–18.)

Можно запрограммировать рекурсии (42) и (43) непосредственно, но интереснее развернуть их с использованием (44), получив таким образом итеративный алгоритм, аналогичный алгоритму C. Такой алгоритм требует только $O(t)$ ячеек памяти и особенно эффективен, когда t мало́ по сравнению с n. Детальнее этот вопрос рассматривается в упр. 45.

***Перестановки мультимножеств, близкие к идеальным.** Последовательности Чейза естественным образом приводят к алгоритму, который будет генерировать перестановки любого мультимножества $\{s_0 \cdot 0, s_1 \cdot 1, \ldots, s_d \cdot d\}$ близким к идеальному способом, что означает следующее:

i) каждый переход имеет вид либо $a_{j+1}a_j \leftrightarrow a_ja_{j+1}$, либо $a_{j+1}a_ja_{j-1} \leftrightarrow a_{j-1}a_ja_{j+1}$;

ii) переходы второго вида обладают тем свойством, что $a_j = \min(a_{j-1}, a_{j+1})$.

Алгоритм C говорит нам, как добиться этого при $d = 1$, и мы можем распространить его на бо́льшие значения d с помощью следующей рекурсивной конструкции [*CACM* **13** (1970), 368–369, 376]. Предположим, что

$$\alpha_0,\ \alpha_1,\ \ldots,\ \alpha_{N-1}$$

представляет собой произвольный близкий к идеальному список перестановок $\{s_1 \cdot 1, \ldots, s_d \cdot d\}$. Затем алгоритм C при $s = s_0$ и $t = s_1 + \cdots + s_d$ говорит нам, каким образом сгенерировать список

$$\Lambda_j \ = \ \alpha_j 0^s,\ \ldots,\ 0^a\alpha_j 0^{s-a}\ , \tag{46}$$

* По всей видимости, наиболее адекватным переводом на русский язык могла бы быть "*чунупорядоченная последовательность*".— *Примеч. пер.*

в котором все переходы представляют собой либо $0x \leftrightarrow x0$, либо $00x \leftrightarrow x00$; последняя запись содержит $a = 1$ или 2 ведущих нуля, в зависимости от значений s и t. Таким образом, все переходы последовательности

$$\Lambda_0,\ \Lambda_1^R,\ \Lambda_2,\ \ldots,\ (\Lambda_{N-1} \text{ или } \Lambda_{N-1}^R) \tag{47}$$

близки к идеальным; ясно, что этот список содержит все перестановки.

Например, вот перестановки $\{0,0,0,1,1,2\}$, сгенерированные этим способом:

211000, 210100, 210001, 210010, 200110, 200101, 200011, 201001, 201010, 201100,
021100, 021001, 021010, 020110, 020101, 020011, 000211, 002011, 002101, 002110,
001120, 001102, 001012, 000112, 010012, 010102, 010120, 011020, 011002, 011200,
101200, 101020, 101002, 100012, 100102, 100120, 110020, 110002, 110200, 112000,
121000, 120100, 120001, 120010, 100210, 100201, 100021, 102001, 102010, 102100,
012100, 012001, 012010, 010210, 010201, 010021, 000121, 001021, 001201, 001210.

***Идеальные схемы.** Почему мы должны заниматься близкими к идеальным генераторами наподобие C_{st}, вместо того чтобы потребовать, чтобы все переходы имели наиболее простой вид $01 \leftrightarrow 10$?

Одна из причин в том, что идеальные схемы существуют не всегда. Например, в 7.2.1.2–(2) было показано, что нет способа сгенерировать все шесть перестановок мультимножества $\{1,1,2,2\}$ с помощью перестановок смежных элементов; таким образом, нет и идеальной схемы для $(2,2)$-сочетаний. В действительности наши шансы добиться идеала составляют около 1 к 4.

Теорема P. *Генерация всех (s,t)-сочетаний $a_{s+t-1}\ldots a_1a_0$ путем обмена смежных элементов $01 \leftrightarrow 10$ возможна тогда и только тогда, когда $s \le 1$ или $t \le 1$, или st нечетно.*

Доказательство. Рассмотрим все перестановки мультимножества $\{s\cdot 0, t\cdot 1\}$. Из упр. 5.1.2–16 мы знаем, что число таких перестановок m_k, имеющих k инверсий, представляет собой коэффициент при z^k в z-номиальном коэффициенте

$$\binom{s+t}{t}_z = \prod_{k=s+1}^{s+t}(1+z+\cdots+z^{k-1}) \Big/ \prod_{k=1}^{t}(1+z+\cdots+z^{k-1}). \tag{48}$$

Каждый обмен смежных элементов изменяет количество инверсий на ±1, так что идеальная схема генерации возможна, только если приблизительно половина всех перестановок имеет нечетное количество инверсий. Более точно значение $\binom{s+t}{t}_{-1} = m_0 - m_1 + m_2 - \cdots$ должно быть равно 0 или ±1. Но в упр. 49 показано, что

$$\binom{s+t}{t}_{-1} = \binom{\lfloor (s+t)/2 \rfloor}{\lfloor t/2 \rfloor}[st \text{ четно}], \tag{49}$$

и если не выполняется ни одно из условий $s \le 1$, $t \le 1$ или st нечетно, то эта величина превышает 1.

И наоборот: идеальные схемы легко построить при $s \le 1$ или $t \le 1$, и они также возможны при нечетном st. Первый нетривиальный случай осуществляется при $s = t = 3$, для которого имеется четыре существенно разных решения; наиболее

симметричное из них следующее:

$$210 — 310 — 410 — 510 — 520 — 521 — 531 — 532 — 432 — 431 —$$
$$421 — 321 — 320 — 420 — 430 — 530 — 540 — 541 — 542 — 543 \quad (50)$$

(см. упр. 51). Несколько авторов построили гамильтоновы цепи в соответствующих графах для произвольных нечетных чисел s и t; например, метод Идеса (Eades), Хикки (Hickey) и Рида (Read) [*JACM* **31** (1984), 19–29] представляет собой интересный пример программы с рекурсивными подпрограммами. Однако, к сожалению, ни одна из известных конструкций не является достаточно простой, чтобы ее можно было кратко описать или реализовать с приемлемой эффективностью. Таким образом, практическая важность идеальных генераторов сочетаний пока что не доказана. ▮

Резюмируя, мы видим, что изучение (s,t)-сочетаний приводит ко многим привлекательным схемам, одни из которых имеют большýю практическую ценность, а другие просто элегантны и красивы. На рис. 46 показаны основные схемы, доступные в случае $s = t = 5$, когда всего имеется $\binom{10}{5} = 252$ сочетаний. Лексикографический порядок (алгоритм L), код Грея двери-вертушки (алгоритм R), гомогенная схема K_{55} из (31) и близкая к идеальной схема Чейза (алгоритм C) показаны на рис. 46, (а)–(г) соответственно. На рис.46, (д) показана близкая к идеальной схема, которая настолько близка к идеальной, насколько это возможно при условии сохранения облексного порядка массива c (см. упр. 34). На рис.46, (е) показана идеальная схема Идеса–Хикки–Рида. Наконец, на рис. 46, (ж) и (з) показаны сочетания, полученные путем поворотов $a_j a_{j-1} \ldots a_0 \leftarrow a_{j-1} \ldots a_0 a_j$ или обменов $a_j \leftrightarrow a_0$, наподобие алгоритмов 7.2.1.2C и 7.2.1.2E (см. упр. 55 и 56).

***Сочетания мультимножества.** Если мультимножество может иметь перестановки, то оно может иметь и сочетания. Например, рассмотрим мультимножество $\{b,b,b,b,g,g,g,r,r,r,w,w\}$, представляющее корзину, в которой содержатся четыре синих мяча, по три зеленых и красных и два белых. Имеется 37 способов выбора пяти мячей из этой корзины; вот их перечисление в лексикографическом порядке:

$$\begin{aligned}&gbbbb,\ ggbbb,\ gggbb,\ rbbbb,\ rgbbb,\ rggbb,\ rgggb,\ rrbbb,\ rrgbb,\ rrggb,\\&rrggg,\ rrrbb,\ rrrgb,\ rrrgg,\ wbbbb,\ wgbbb,\ wggbb,\ wgggb,\ wrbbb,\ wrgbb,\\&wrggb,\ wrggg,\ wrrbb,\ wrrgb,\ wrrgg,\ wrrrb,\ wrrrg,\ wwbbb,\ wwgbb,\ wwggb,\\&wwggg,\ wwrbb,\ wwrgb,\ wwrgg,\ wwrrb,\ wwrrg,\ wwrrr.\end{aligned} \quad (51)$$

Этот факт может показаться незначительным и/или понятным лишь посвященным, но, как мы увидим ниже из теоремы W, лексикографическая генерация сочетаний мультимножества дает оптимальное решение важных комбинаторных задач.

Якоб Бернулли (James Bernoulli) заметил в своей работе *Ars Conjectandi* (1713), 119–123, что можно перечислить все такие сочетания, рассматривая коэффициент при z^5 в произведении $(1+z+z^2)(1+z+z^2+z^3)^2(1+z+z^2+z^3+z^4)$. В самом деле, это наблюдение легко понять, поскольку мы получим все возможные варианты выбора мячей из корзины, если перемножим полиномы

$$(1+w+ww)(1+r+rr+rrr)(1+g+gg+ggg)(1+b+bb+bbb+bbbb).$$

(а) (б) (в) (г) (д) (е) (ж) (з)

Рис. 46. Примеры схем $(5,5)$-сочетаний:

а) лексикографическая;
б) двери-вертушки;
в) гомогенная;
г) близкая к идеальной;
д) еще более близкая к идеальной;
е) идеальная;
ж) суффиксная;
з) с правым обменом.

Сочетаниям мультимножества можно поставить в соответствие *композиции с ограничениями на слагаемые*, или *ограниченные композиции* (*bounded compositions*), т. е. композиции, в которых отдельные части ограничены. Например, 37 мультисочетаний, перечисленных в (51), соответствуют 37 решениям

$$5 = r_3 + r_2 + r_1 + r_0, \quad 0 \le r_3 \le 2, \quad 0 \le r_2, r_1 \le 3, \quad 0 \le r_0 \le 4,$$

а именно $5 = 0{+}0{+}1{+}4 = 0{+}0{+}2{+}3 = 0{+}0{+}3{+}2 = 0{+}1{+}0{+}4 = \cdots = 2{+}3{+}0{+}0$.

Композиции с ограничениями на слагаемые, в свою очередь, являются частным случаем *факторных таблиц* (*contingency tables*), играющих важную роль в статистике. И все эти комбинаторные конфигурации могут быть сгенерированы как с помощью кодов наподобие кода Грея, так и в лексикографическом порядке. В упр. 60–63 раскрываются некоторые из используемых при этом идей.

***Тени.** В математике часто встречаются множества сочетаний. Например, множество 2-сочетаний (а именно множество неупорядоченных пар), по сути, представляет собой граф, а множество t-сочетаний для произвольного t называется однородным гиперграфом (uniform hypergraph). Если вершины выпуклого многогранника несколько смещены, так что никакие три из них не лежат на одной прямой, никакие четыре — в одной плоскости и, вообще, никакие $t+1$ вершин не лежат в $(t-1)$-мерной гиперплоскости, то получаемые $(t-1)$-мерные грани представляют собой "симплексы", вершины которых имеют большое значение в компьютерных приложениях. Исследователи выяснили, что такие множества сочетаний имеют важные свойства, связанные с лексикографической генерацией.

Если α — произвольное t-сочетание $c_t \ldots c_2 c_1$, его *тенью* $\partial\alpha$ является множество всех его $(t-1)$-элементных подмножеств $c_{t-1} \ldots c_2 c_1$, $\ldots$, $c_t \ldots c_3 c_1$, $c_t \ldots c_3 c_2$. Например, $\partial 5310 = \{310, 510, 530, 531\}$. Можно также представить t-сочетание в виде битовой строки $a_{n-1} \ldots a_1 a_0$; в этом случае $\partial\alpha$ представляет собой множество всех строк, получаемых заменой 1 на 0: $\partial 101011 = \{001011, 100011, 101001, 101010\}$. Если A — произвольное множество t-сочетаний, определим его тень

$$\partial A \;=\; \bigcup\{\, \partial\alpha \mid \alpha \in A \,\} \tag{52}$$

как множество всех $(t-1)$-сочетаний в тенях его членов. Например, $\partial\partial 5310 = \{10, 30, 31, 50, 51, 53\}$.

Эти определения применимы также к сочетаниям с повторениями, т. е. к мультисочетаниям: $\partial 5330 = \{330, 530, 533\}$ и $\partial\partial 5330 = \{30, 33, 50, 53\}$. В общем случае, если A — множество t-элементных мультимножеств, ∂A является множеством $(t-1)$-элементных мультимножеств. Заметим, однако, что сама тень ∂A никогда не содержит повторяющихся элементов.

Верхняя тень (*upper shadow*) $\varrho\alpha$ по отношению к универсуму U определяется аналогично, но идет от t-сочетаний к $(t+1)$-сочетаниям:

$$\varrho\alpha \;=\; \{\, \beta \subseteq U \mid \alpha \in \partial\beta \,\} \qquad \text{для } \alpha \in U; \tag{53}$$

$$\varrho A \;=\; \bigcup\{\, \varrho\alpha \mid \alpha \in A \,\} \qquad \text{для } A \subseteq U. \tag{54}$$

Если, например, $U = \{0, 1, 2, 3, 4, 5, 6\}$, то мы получим $\varrho 5310 = \{53210, 54310, 65310\}$; с другой стороны, если $U = \{\infty \cdot 0, \infty \cdot 1, \ldots, \infty \cdot 6\}$, то $\varrho 5310 = \{53100, 53110, 53210, 53310, 54310, 55310, 65310\}$.

Следующие фундаментальные теоремы, имеющие множество применений в различных отраслях математики и информатики, говорят нам, насколько малыми могут быть тени множеств.

Теорема K. *Если A — множество из N t-сочетаний элементов универсума $U = \{0, 1, \ldots, n-1\}$, то*

$$|\partial A| \geq |\partial P_{Nt}| \qquad \text{и} \qquad |\varrho A| \geq |\varrho Q_{Nnt}|, \tag{55}$$

где P_{Nt} обозначает первые N сочетаний, сгенерированных алгоритмом L, а именно N лексикографически наименьших сочетаний $c_t \ldots c_2 c_1$, удовлетворяющих условию (3), *а Q_{Nnt} обозначает последние N лексикографически наибольших таких сочетаний.* ▮

Теорема M. *Если A — множество из N t-мультисочетаний элементов универсума $U = \{\infty \cdot 0, \infty \cdot 1, \ldots, \infty \cdot s\}$, то*

$$|\partial A| \geq |\partial \widehat{P}_{Nt}| \qquad \text{и} \qquad |\varrho A| \geq |\varrho \widehat{Q}_{Nst}|, \tag{56}$$

где $\widehat{P}_{Nt}$ обозначает первые N лексикографически наименьших мультисочетаний $d_t \ldots d_2 d_1$, удовлетворяющих условию (6), *а $\widehat{Q}_{Nst}$ обозначает N лексикографически наибольших таких сочетаний.* ▮

Обе эти теоремы являются следствием более строгого результата, который будет доказан позже. Теорема K обычно называется теоремой Крускала–Катоны, так как она была открыта Д. Б. Крускалом (J. B. Kruskal) [*Math. Optimization Techniques*, ed. by R. Bellman (1963), 251–278], а позже заново открыта Г. Катоной (G. Katona) [*Theory of Graphs*, Tihany 1966, ed. by Erdős and Katona (Academic Press, 1968), 187–207]; ранее неполное доказательство было приведено М. П. Шютценбергером (M. P. Schützenberger) в его малоизвестной публикации [*RLE Quarterly Progress Report* **55** (1959), 117–118]. Теорема M была доказана Ф. С. Маколеем (F. S. Macaulay) многими годами ранее [*Proc. London Math. Soc.* (2) **26** (1927), 531–555].

Перед тем как приступить к доказательству (55) и (56), рассмотрим более внимательно, что означают эти формулы. Из теоремы L мы знаем, что первые N из всех t-сочетаний, посещенных алгоритмом L, — те, которые предшествуют $n_t \ldots n_2 n_1$, где

$$N = \binom{n_t}{t} + \cdots + \binom{n_2}{2} + \binom{n_1}{1}, \qquad n_t > \cdots > n_2 > n_1 \geq 0$$

является комбинаторным представлением N степени t. Иногда такое представление имеет меньше t ненулевых членов, потому что n_j может быть равно $j-1$; уберем нули и запишем

$$N = \binom{n_t}{t} + \binom{n_{t-1}}{t-1} + \cdots + \binom{n_v}{v}, \qquad n_t > n_{t-1} > \cdots > n_v \geq v \geq 1. \tag{57}$$

Теперь первые $\binom{n_t}{t}$ сочетаний $c_t \ldots c_1$ представляют собой t-сочетания $\{0, \ldots, n_t - 1\}$; следующие $\binom{n_{t-1}}{t-1}$ — те, в которых $c_t = n_t$, а $c_{t-1} \ldots c_1$ — $(t-1)$-сочетание элементов множества $\{0, \ldots, n_{t-1} - 1\}$ и т. д. Например, если $t = 5$ и $N = \binom{9}{5} + \binom{7}{4} + \binom{4}{3}$, то первые N сочетаний будут

$$P_{N5} = \{43210, \ldots, 87654\} \cup \{93210, \ldots, 96543\} \cup \{97210, \ldots, 97321\}. \tag{58}$$

Тень этого множества P_{N5}, к счастью, легко представить — это

$$\partial P_{N5} = \{3210, \ldots, 8765\} \cup \{9210, \ldots, 9654\} \cup \{9710, \ldots, 9732\}, \tag{59}$$

т. е. первые $\binom{9}{4} + \binom{7}{3} + \binom{4}{2}$ сочетаний в лексикографическом порядке для $t = 4$.

Другими словами, если определить функцию Крускала κ_t как

$$\kappa_t N = \binom{n_t}{t-1} + \binom{n_{t-1}}{t-2} + \cdots + \binom{n_v}{v-1}, \tag{60}$$

где N имеет единственное представление (57), а $\kappa_t 0 = 0$, то

$$\partial P_{Nt} \;=\; P_{(\kappa_t N)(t-1)}. \tag{61}$$

Теорема K говорит нам, например, что граф с миллионом ребер может содержать не более

$$\binom{1414}{3} + \binom{1009}{2} \;=\; 470\,700\,300$$

треугольников, т. е. не более 470 700 300 множеств вершин $\{u, v, w\}$ с ребрами u — v — w — u. Причина этого в том, что, согласно упр. 17, $1000000 = \binom{1414}{2} + \binom{1009}{1}$, и ребра $P_{(1000000)2}$ поддерживают $\binom{1414}{3} + \binom{1009}{2}$ треугольников; если бы треугольников было больше, граф должен был бы иметь как минимум $\kappa_3 470700301 = \binom{1414}{2} + \binom{1009}{1} + \binom{1}{0} = 1000001$ ребер в их тенях.

Крускал определил сопутствующую функцию

$$\lambda_t N = \binom{n_t}{t+1} + \binom{n_{t-1}}{t} + \cdots + \binom{n_v}{v+1} \tag{62}$$

для работы с задачами такого рода. Функции κ и λ связаны интересным соотношением, доказываемым в упр. 72:

$$\text{из} \quad M + N = \binom{s+t}{t} \quad \text{следует} \quad \kappa_s M + \lambda_t N \;=\; \binom{s+t}{t+1}, \quad \text{если } st > 0. \tag{63}$$

Возвращаясь к теореме M, размеры $\partial \widehat{P}_{Nt}$ и $\varrho \widehat{Q}_{Nst}$ оказываются равными

$$|\partial \widehat{P}_{Nt}| \;=\; \mu_t N \qquad \text{и} \qquad |\varrho \widehat{Q}_{Nst}| \;=\; N + \kappa_s N \tag{64}$$

(см. упражнение 81), где функция μ_t удовлетворяет уравнению

$$\mu_t N = \binom{n_t - 1}{t-1} + \binom{n_{t-1} - 1}{t-2} + \cdots + \binom{n_v - 1}{v-1} \tag{65}$$

при N, имеющем комбинаторное представление (57).

В табл. 3 показано, как ведут себя упомянутые функции $\kappa_t N$, $\lambda_t N$ и $\mu_t N$ для малых значений t и N. При больших значениях t и N эти функции могут быть хорошо аппроксимированы с помощью замечательной функции $\tau(x)$, введенной Теиджи Такаги (Teiji Takagi) в 1903 году; см. рис. 47 и упр 82–85.

Теоремы K и M являются следствиями существенно более общей теоремы дискретной геометрии, открытой Да-Лун Вангом (Da-Lun Wang) и Пинг Вангом (Ping Wang) [*SIAM J. Applied Math.* **33** (1977), 55–59], с которой мы сейчас познакомимся. Рассмотрим *дискретный n-мерный тор* $T(m_1, \ldots, m_n)$, элементы которого представляют собой целочисленные векторы $x = (x_1, \ldots, x_n)$, где $0 \le x_1 < m_1$,

Таблица 3

Примеры функций Крускала–Маколея κ, λ и μ

$N =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\kappa_1 N =$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\kappa_2 N =$	0	2	3	3	4	4	4	5	5	5	5	6	6	6	6	6	7	7	7	7	7
$\kappa_3 N =$	0	3	5	6	6	8	9	9	10	10	10	12	13	13	14	14	14	15	15	15	15
$\kappa_4 N =$	0	4	7	9	10	10	13	15	16	16	18	19	19	20	20	20	23	25	26	26	28
$\kappa_5 N =$	0	5	9	12	14	15	15	19	22	24	25	25	28	30	31	31	33	34	34	35	35
$\lambda_1 N =$	0	0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	153	171	190
$\lambda_2 N =$	0	0	0	1	1	2	4	4	5	7	10	10	11	13	16	20	20	21	23	26	30
$\lambda_3 N =$	0	0	0	0	1	1	1	2	2	3	5	5	5	6	6	7	9	9	10	12	15
$\lambda_4 N =$	0	0	0	0	0	1	1	1	1	2	2	2	3	3	4	6	6	6	6	7	7
$\lambda_5 N =$	0	0	0	0	0	0	1	1	1	1	1	2	2	2	2	3	3	3	4	4	5
$\mu_1 N =$	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\mu_2 N =$	0	1	2	2	3	3	3	4	4	4	4	5	5	5	5	5	6	6	6	6	6
$\mu_3 N =$	0	1	2	3	3	4	5	5	6	6	6	7	8	8	9	9	9	10	10	10	10
$\mu_4 N =$	0	1	2	3	4	4	5	6	7	7	8	9	9	10	10	10	11	12	13	13	14
$\mu_5 N =$	0	1	2	3	4	5	5	6	7	8	9	9	10	11	12	12	13	14	14	15	15

Рис. 47. Аппроксимация функции Крускала функцией Такаги (гладкая кривая на левом графике представляет собой нижнюю границу $\underline{\kappa}_5 N - N$ из упр. 80).

$\ldots$, $0 \le x_n < m_n$. Определим сумму и разность двух таких векторов x и y, как в формулах 4.3.2–(2) и 4.3.2–(3):

$$x + y \;=\; \bigl((x_1 + y_1) \bmod m_1, \ldots, (x_n + y_n) \bmod m_n\bigr), \tag{66}$$

$$x - y \;=\; \bigl((x_1 - y_1) \bmod m_1, \ldots, (x_n - y_n) \bmod m_n\bigr). \tag{67}$$

Определим также так называемый *перекрестный порядок* таких векторов, говоря, что $x \preceq y$ тогда и только тогда, когда

$$\nu x < \nu y \quad \text{или} \quad (\nu x = \nu y \text{ и } x \ge y \text{ лексикографически}). \tag{68}$$

Здесь, как обычно, $\nu(x_1, \ldots, x_n) = x_1 + \cdots + x_n$. Например, при $m_1 = m_2 = 2$ и $m_3 = 3$ двенадцать векторов $x_1 x_2 x_3$ в возрастающем перекрестном порядке представляют собой

$$000,\ 100,\ 010,\ 001,\ 110,\ 101,\ 011,\ 002,\ 111,\ 102,\ 012,\ 112 \tag{69}$$

(скобки и запятые для удобства опущены). *Дополнением* вектора в $T(m_1,\ldots,m_n)$ является вектор

$$\overline{x} \;=\; (m_1-1-x_1,\ldots,m_n-1-x_n). \tag{70}$$

Заметим, что $x \preceq y$ выполняется тогда и только тогда, когда $\overline{x} \succeq \overline{y}$. Следовательно, если $\mathrm{rank}(x)$ обозначает количество векторов, предшествующих x в перекрестном порядке, то

$$\mathrm{rank}(x)+\mathrm{rank}(\overline{x}) \;=\; T-1, \qquad \text{где } T=m_1\ldots m_n. \tag{71}$$

Удобно называть векторы "точками" и именовать их e_0, e_1, $\ldots$, e_{T-1} в возрастающем перекрестном порядке. Таким образом, мы имеем $e_7=002$ в (69), а в общем случае $\overline{e}_r = e_{T-1-r}$. Обратите внимание, что

$$e_1 = 100\ldots00, \quad e_2 = 010\ldots00, \quad \ldots, \quad e_n = 000\ldots01; \tag{72}$$

это так называемые *единичные векторы.* Множество

$$S_N \;=\; \{e_0,e_1,\ldots,e_{N-1}\}, \tag{73}$$

состоящее из наименьших N точек, называется *стандартным множеством*, и в частном случае $N=n+1$ будем писать

$$E=\{e_0,e_1,\ldots,e_n\}=\{000\ldots00,100\ldots00,010\ldots00,\ldots,000\ldots01\}. \tag{74}$$

Любое множество точек X имеет *размах* (*spread*) X^+, *ядро* (*core*) X° и *двойник* (*dual*) $X^\sim$, определяемые правилами

$$X^+ \;=\; \{\, x\in S_T \mid x\in X, \text{ или } x-e_1\in X, \text{ или } \cdots, \text{ или } x-e_n\in X \,\}; \tag{75}$$

$$X^\circ \;=\; \{\, x\in S_T \mid x\in X \text{ и } x+e_1\in X \text{ и } \cdots \text{ и } x+e_n\in X \,\}; \tag{76}$$

$$X^\sim \;=\; \{\, x\in S_T \mid \overline{x}\notin X \,\}. \tag{77}$$

Размах X можно определить алгебраически, записав

$$X^+ \;=\; X+E, \tag{78}$$

где $X+Y$ означает $\{\, x+y \mid x\in X \text{ и } y\in Y \,\}$. Ясно, что

$$X^+\subseteq Y \qquad \text{тогда и только тогда, когда} \qquad X\subseteq Y^\circ. \tag{79}$$

Эти понятия могут быть проиллюстрированы в двухмерном случае $m_1=4$, $m_2=6$ с помощью более или менее случайного тороидального размещения $X=\{00,12,13,14,15,21,22,25\}$, для которого мы имеем (графически)

	•	•	
	•		
	•		
	•	•	
		•	
•			

X

	•	•	+
	•	+	
	•	+	
	○	•	+
+		•	+
•	+	+	

X° и X^+

•	•	•	
•		•	•
•			•
•	•		•
•	•		•
•			•

$X^\sim$

○	•	•	+
•	+	○	•
•	+		○
○	•	+	○
○	•	+	○
•	+	+	○

$X^{\sim\circ}$ и $X^{\sim+}$. (80)

Здесь X на первых двух диаграммах состоит из точек, помеченных как • и ○, X° включает только ○, а X^+ состоит из +, • и ○. Заметим, что если мы повернем

диаграмму $X^{\sim\circ}$ и $X^{\sim+}$ на 180°, то получим диаграмму для X° и X^+, но с $(\bullet, \circ, +, \ )$, замененными соответственно на $(+, \ , \bullet, \circ)$. В действительности тождества

$$X^\circ \;=\; X^{\sim+\sim}, \qquad X^+ \;=\; X^{\sim\circ\sim} \tag{81}$$

выполняются в общем случае (см. упр. 86).

Теперь мы готовы к изложению теоремы Ванга и Ванга.

Теорема W. *Пусть X — произвольное множество, состоящее из N точек дискретного тора $T(m_1, \ldots, m_n)$, где $m_1 \le \cdots \le m_n$. Тогда $|X^+| \ge |S_N^+|$ и $|X^\circ| \le |S_N^\circ|$.*

Доказательство. Другими словами, стандартные множества S_N среди всех N-точечных множеств имеют наименьший размах и наибольшее ядро. Мы докажем этот результат с помощью следующего общего подхода, впервые примененного Ф. Д. В. Уипплом (F. J. W. Whipple) для доказательства теоремы M [*Proc. London Math. Soc.* (2) **28** (1928), 431–437]. Первый шаг состоит в доказательстве того, что размах и ядро стандартных множеств стандартны.

Лемма S. *Существуют функции α и β, такие, что $S_N^+ = S_{\alpha N}$ и $S_N^\circ = S_{\beta N}$.*

Доказательство. Мы можем считать, что $N > 0$. Пусть r — максимальное значение, для которого $e_r \in S_N^+$, и пусть $\alpha N = r+1$; мы должны доказать, что $e_q \in S_N^+$ для $0 \le q < r$. Допустим, что $e_q = x = (x_1, \ldots, x_n)$ и $e_r = y = (y_1, \ldots, y_n)$, и пусть k — наибольший индекс, для которого $x_k > 0$. Поскольку $y \in S_N^+$, существует индекс j, такой, что $y - e_j \in S_N$. Этого достаточно для доказательства того, что $x - e_k \preceq y - e_j$, и в упр. 88 это сделано.

Вторая часть следует из (81) при $\beta N = T - \alpha(T - N)$, поскольку $S_N^\sim = S_{T-N}$. ∎

Очевидно, что теорема W справедлива при $n = 1$, так что по индукции предположим, что она доказана для $n-1$ размерностей. Следующий шаг состоит в *сжатии* (*compress*) данного множества X в k-й координатной позиции путем разделения его на два непересекающихся множества

$$X_k(a) \;=\; \{\, x \in X \mid x_k = a \,\} \tag{82}$$

для $0 \le a < m_k$ и замены каждого $X_k(a)$ множеством

$$X'_k(a) \;=\; \{\, (s_1, \ldots, s_{k-1}, a, s_k, \ldots, s_{n-1}) \mid (s_1, \ldots, s_{n-1}) \in S_{|X_k(a)|} \,\} \tag{83}$$

с тем же количеством элементов. Множества S, использующиеся в (83), стандартны в $(n-1)$-мерном торе $T(m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_n)$. Заметим, что $(x_1, \ldots, x_{k-1}, a, x_{k+1}, \ldots, x_n) \preceq (y_1, \ldots, y_{k-1}, a, y_{k+1}, \ldots, y_n)$ тогда и только тогда, когда $(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n) \preceq (y_1, \ldots, y_{k-1}, y_{k+1}, \ldots, y_n)$; следовательно, $X'_k(a) = X_k(a)$ тогда и только тогда, когда $(n-1)$-мерные точки $(x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n)$, где $(x_1, \ldots, x_{k-1}, a, x_{k+1}, \ldots, x_n) \in X$, малы настолько, насколько это возможно при проекции на $(n-1)$-мерный тор. Пусть

$$C_k X \;=\; X'_k(0) \cup X'_k(1) \cup \cdots \cup X'_k(m_k - 1) \tag{84}$$

представляет собой сжатие X по k-й позиции. В упр. 90 доказывается тот фундаментальный факт, что сжатие не увеличивает размер размаха:

$$|X^+| \;\ge\; |(C_k X)^+| \qquad \text{для } 1 \le k \le n. \tag{85}$$

Кроме того, если сжатие изменяет X, то оно заменяет некоторые элементы другими элементами меньшего ранга. Таким образом, необходимо доказать теорему W только для полностью сжатых множеств X, у которых $X = C_k X$ для всех k.

Рассмотрим, например, случай $n = 2$. В полностью сжатом в двух измерениях множестве все точки передвинуты влево в своих строках и вниз — в своих столбцах, как в одиннадцатиточечных множествах

крайнее справа множество стандартное и имеет наименьший размах. В упр. 91 завершается доказательство теоремы W для двух размерностей.

Если $n > 2$, пусть $x = (x_1, \ldots, x_n) \in X$ и $x_j > 0$. Из условия $C_k X = X$ вытекает, что если $0 \le i < j$ и $i \ne k \ne j$, то $x + e_i - e_j \in X$. Применение этого факта к трем значениям k говорит нам, что $x + e_i - e_j \in X$ для любых $0 \le i < j$. Следовательно,

$$X_n(a) + E_n(0) \;\subseteq\; X_n(a-1) + e_n \quad \text{для } 0 < a < m, \tag{86}$$

где $m = m_n$ и $E_n(0)$ — сокращенная запись множества $\{e_0, \ldots, e_{n-1}\}$.

Пусть $X_n(a)$ содержит N_a элементов, так что $N = |X| = N_0 + N_1 + \cdots + N_{m-1}$, и пусть $Y = X^+$. Тогда

$$Y_n(a) \;=\; \bigl(X_n\bigl((a-1) \bmod m\bigr) + e_n\bigr) \cup \bigl(X_n(a) + E_n(0)\bigr)$$

представляет собой стандартное множество в $n-1$ размерностях, и (86) говорит нам, что

$$N_{m-1} \le \beta N_{m-2} \le N_{m-2} \le \cdots \le N_1 \le \beta N_0 \le N_0 \le \alpha N_0,$$

где α и β относятся к координатам от 1 до $n-1$. Таким образом,

$$\begin{aligned} |Y| &= |Y_n(0)| + |Y_n(1)| + |Y_n(2)| + \cdots + |Y_n(m-1)| \\ &= \alpha N_0 + N_0 + N_1 + \cdots + N_{m-2} = \alpha N_0 + N - N_{m-1}. \end{aligned}$$

Доказательство теоремы W теперь имеет красивое завершение. Пусть $Z = S_N$, и положим $|Z_n(a)| = M_a$. Мы хотим доказать, что $|X^+| \ge |Z^+|$, т. е. что

$$\alpha N_0 + N - N_{m-1} \;\ge\; \alpha M_0 + N - M_{m-1}, \tag{87}$$

поскольку аргументы из предыдущего абзаца применимы как к X, так и к Z. Мы докажем (87), показав, что $N_{m-1} \le M_{m-1}$ и $N_0 \ge M_0$.

Используя $(n-1)$-мерные функции α и β, определим

$$N'_{m-1} = N_{m-1},\; N'_{m-2} = \alpha N'_{m-1},\; \ldots,\; N'_1 = \alpha N'_2,\; N'_0 = \alpha N'_1; \tag{88}$$

$$N''_0 = N_0,\; N''_1 = \beta N''_0,\; N''_2 = \beta N''_1,\; \ldots,\; N''_{m-1} = \beta N''_{m-2}. \tag{89}$$

Тогда $N'_a \le N_a \le N''_a$ для $0 \le a < m$, и отсюда следует, что

$$N' = N'_0 + N'_1 + \cdots + N'_{m-1} \;\le\; N \;\le\; N'' = N''_0 + N''_1 + \cdots + N''_{m-1}. \tag{90}$$

В упр. 92 доказывается, что стандартное множество $Z' = S_{N'}$ имеет ровно N'_a элементов с n-й координатой, равной a, для каждого a. Из дуальности α и β

аналогично доказывается, что стандартное множество $Z'' = S_{N''}$ содержит ровно N''_a элементов с n-й координатой, равной a. Следовательно,

$$M_{m-1} = |Z_n(m-1)| \geq |Z'_n(m-1)| = N_{m-1},$$
$$M_0 = |Z_n(0)| \leq |Z''_n(0)| = N_0,$$

поскольку $Z' \subseteq Z \subseteq Z''$ согласно (90). Из (81) мы также имеем $|X^\circ| \leq |Z^\circ|$. ▮

Теперь мы готовы доказать теоремы K и M, которые в действительности являются частными случаями более общей теоремы Клементса (Clements) и Линдстрема (Lindström), которая применима к произвольным мультимножествам [*J. Combinatorial Theory* **7** (1969), 230–238].

Следствие C. *Если A — множество N t-мультисочетаний, содержащееся в мультимножестве $U = \{s_0 \cdot 0, s_1 \cdot 1, \ldots, s_d \cdot d\}$, где $s_0 \geq s_1 \geq \cdots \geq s_d$, то*

$$|\partial A| \geq |\partial P_{Nt}| \qquad \text{и} \qquad |\varrho A| \geq |\varrho Q_{Nt}|, \tag{91}$$

где P_{Nt} обозначает N лексикографически наименьших мультисочетаний $d_t \ldots d_2 d_1$ из U, а Q_{Nt} обозначает N лексикографически наибольших мультисочетаний.

Доказательство. Мультисочетания мультимножества U могут быть представлены как точки $x_1 \ldots x_n$ тора $T(m_1, \ldots, m_n)$, где $n = d + 1$ и $m_j = s_{n-j} + 1$; пусть x_j — количество вхождений $n - j$ в мультисочетание. Такое соответствие сохраняет лексикографический порядок. Например, если $U = \{0, 0, 0, 1, 1, 2, 3\}$, его 3-мультисочетаниями в лексикографическом порядке являются

$$000,\ 100,\ 110,\ 200,\ 210,\ 211,\ 300,\ 310,\ 311,\ 320,\ 321, \tag{92}$$

а соответствующие им точки $x_1x_2x_3x_4$ имеют вид

$$0003, 0012, 0021, 0102, 0111, 0120, 1002, 1011, 1020, 1101, 1110. \tag{93}$$

Пусть T_w — точки тора, вес которых $x_1 + \cdots + x_n = w$. Тогда каждое допустимое множество A t-мультисочетаний является подмножеством T_t. Более того, — и в этом главное — размах $T_0 \cup T_1 \cup \cdots \cup T_{t-1} \cup A$ равен

$$\begin{aligned}(T_0 \cup T_1 \cup \cdots \cup T_{t-1} \cup A)^+ &= T_0^+ \cup T_1^+ \cup \cdots \cup T_{t-1}^+ \cup A^+ \\ &= T_0 \cup T_1 \cup \cdots \cup T_t \cup \varrho A.\end{aligned} \tag{94}$$

Таким образом, верхняя тень ϱA просто равна $(T_0 \cup T_1 \cup \cdots \cup T_{t-1} \cup A)^+ \cap T_{t+1}$, и теорема W, по сути, говорит нам, что из $|A| = N$ следует $|\varrho A| \geq |\varrho(S_{M+N} \cap T_t)|$, где $M = |T_0 \cup \cdots \cup T_{t-1}|$. Следовательно, по определению перекрестного порядка $S_{M+N} \cap T_t$ состоит из N лексикографически наибольших t-мультисочетаний, а именно Q_{Nt}.

Доказательство того факта, что $|\partial A| \geq |\partial P_{Nt}|$, теперь вытекает из дополнительности (см. упр. 94). ▮

Упражнения

1. [*M23*] Поясните, почему правило Голомба (8) устанавливает однозначное соответствие между множествами $\{c_1, \ldots, c_t\} \subseteq \{0, \ldots, n-1\}$ и мультимножествами $\{e_1, \ldots, e_t\} \subseteq \{\infty \cdot 0, \ldots, \infty \cdot n - t\}$.

2. [*16*] Какой путь в сетке 11×13 соответствует битовой строке (13)?

▶ **3.** [*21*] (R. R. Fenichel, 1968.) Покажите, что композиции $q_t + \cdots + q_1 + q_0$ числа s из $t+1$ неотрицательных частей могут быть сгенерированы в лексикографическом порядке простым алгоритмом без циклов.

4. [*16*] Покажите, что любая композиция $q_t \ldots q_0$ числа s из $t+1$ неотрицательных частей соответствует композиции $r_s \ldots r_0$ числа t из $s+1$ неотрицательных частей. Какая композиция соответствует 10224000001010 при таком соответствии?

▶ **5.** [*20*] Какой хороший способ генерации всех целочисленных решений следующих систем неравенств можно предложить?

а) $n > x_t \geq x_{t-1} > x_{t-2} \geq x_{t-3} > \cdots > x_1 \geq 0$, где t нечетно.

б) $n \gg x_t \gg x_{t-1} \gg \cdots \gg x_2 \gg x_1 \gg 0$, где $a \gg b$ означает $a \geq b+2$.

6. [*M22*] Как часто выполняется каждый шаг алгоритма T?

7. [*22*] Разработайте алгоритм, который обходил бы "дуальные" сочетания $b_s \ldots b_2 b_1$ в *уменьшающемся* лексикографическом порядке (см. (5) и табл. 1). Подобно алгоритму T, ваш алгоритм должен избегать излишних присваиваний и ненужных поисков.

8. [*M23*] Разработайте алгоритм для генерации всех (s,t)-сочетаний $a_{n-1} \ldots a_1 a_0$ лексикографически в форме битовых строк. Общее время работы алгоритма должно составлять $O\bigl(\binom{n}{t}\bigr)$, в предположении, что $st > 0$.

9. [*M26*] Пусть все (s,t)-сочетания $a_{n-1} \ldots a_1 a_0$ перечислены в лексикографическом порядке и пусть $2A_{st}$ — общее количество изменений битов между соседними строками. Например, $A_{33} = 25$, поскольку между 20 строками в табл. 1 всего

$$2+2+2+4+2+2+4+2+2+6+2+2+4+2+2+4+2+2+2 = 50$$

изменений битов.

а) Покажите, что $A_{st} = \min(s,t) + A_{(s-1)t} + A_{s(t-1)}$ при $st > 0$; $A_{st} = 0$ при $st = 0$.

б) Докажите, что $A_{st} < 2\binom{s+t}{t}$.

▶ **10.** [*21*] Чемпионат США по бейсболу ("World Series") является состязанием, в котором чемпион Американской лиги (A) играет с чемпионом Национальной лиги (N), пока один из них не выиграет у другого четыре раза. Как лучше всего перечислить все возможные сценарии AAAA, AAANA, AAANNA, ..., NNNN? Каков простейший способ назначить этим сценариям последовательные целые числа?

11. [*19*] Какой из сценариев упр. 10 чаще всего встречался в XX веке? Какой не встречался ни разу? [*Указание*: результаты чемпионата World Series легко найти в Интернете.]

12. [*HM32*] Множество V n-битовых векторов, замкнутое относительно сложения по модулю 2, называется *бинарным векторным пространством* (*binary vector space*).

а) Докажите, что каждое такое множество V содержит 2^t элементов для некоторого целого t и может быть представлено как множество $\{x_1\alpha_1 \oplus \cdots \oplus x_t\alpha_t \mid 0 \leq x_1, \ldots, x_t \leq 1\}$, где векторы $\alpha_1, \ldots, \alpha_t$ образуют "канонический базис" со следующим свойством: существует t-сочетание $c_t \ldots c_2 c_1$ из $\{0, 1, \ldots, n-1\}$, такое, что если α_k — бинарный вектор $a_{k(n-1)} \ldots a_{k1} a_{k0}$, то мы имеем

$$a_{kc_j} = [j=k] \quad \text{для } 1 \leq j,k \leq t; \qquad\qquad a_{kl} = 0 \quad \text{для } 0 \leq l < c_k,\ 1 \leq k \leq t.$$

Например, канонические базисы при $n = 9$, $t = 4$ и $c_4c_3c_2c_1 = 7641$ имеют общий вид

$$\begin{aligned}
\alpha_1 &= *\,0\,0\,*\,0\,*\,*\,1\,0,\\
\alpha_2 &= *\,0\,0\,*\,1\,0\,0\,0\,0,\\
\alpha_3 &= *\,0\,1\,0\,0\,0\,0\,0\,0,\\
\alpha_4 &= *\,1\,0\,0\,0\,0\,0\,0\,0.
\end{aligned}$$

Существует 2^8 способов замены восьми звездочек нулями и/или единицами, и каждый из них образует канонический базис. Назовем t размерностью V.

б) Сколько может быть t-мерных пространств с n-битовыми векторами?

в) Разработайте алгоритм для генерации всех канонических базисов $(\alpha_1, \ldots, \alpha_t)$ размерностью t. *Указание*: соответствующие сочетания $c_t \ldots c_1$ должны лексикографически возрастать, как в алгоритме L.

г) Каким будет миллионный посещенный вашим алгоритмом базис при $n = 9$ и $t = 4$?

13. [*25*] Одномерная *конфигурация Изинга* длиной n, весом t и энергией r представляет собой бинарную строку $a_{n-1} \ldots a_0$, такую, что $\sum_{j=0}^{n-1} a_j = t$ и $\sum_{j=1}^{n-1} b_j = r$, где $b_j = a_j \oplus a_{j-1}$. Например, $a_{12} \ldots a_0 = 1100100100011$ имеет вес 6 и энергию 6, поскольку $b_{12} \ldots b_1 = 010110110010$.

Разработайте алгоритм для генерации всех таких конфигураций для данных n, t и r.

14. [*26*] При генерации бинарных строк $a_{n-1} \ldots a_1 a_0$ (s, t)-сочетаний в лексикографическом порядке нам иногда требуется изменить $2\min(s, t)$ бит при переходе от одного сочетания к другому. Например, в табл. 1 за 011100 следует 100011. Так что мы, по-видимому, не можем надеяться сгенерировать все сочетания с помощью алгоритма без циклов, если только не будем посещать их в некотором ином порядке.

Покажите, что в действительности имеется способ вычисления лексикографического преемника данного сочетания за $O(1)$ шагов, если каждое сочетание представлено в виде двухсвязного списка следующим образом: имеются массивы $l[0], \ldots, l[n]$ и $r[0], \ldots, r[n]$, такие, что $l[r[j]] = j$ для $0 \le j \le n$. Если $x_0 = l[0]$ и $x_j = l[x_{j-1}]$ для $0 < j < n$, то $a_j = [x_j > s]$ для $0 \le j < n$.

15. [*M22*] Воспользуйтесь тем фактом, что дуальные сочетания $b_s \ldots b_2 b_1$ располагаются в обратном лексикографическом порядке, для доказательства того, что сумма $\binom{b_s}{s} + \cdots + \binom{b_2}{2} + \binom{b_1}{1}$ связана с суммой $\binom{c_t}{t} + \cdots + \binom{c_2}{2} + \binom{c_1}{1}$ простым соотношением.

16. [*M21*] Каково миллионное сочетание, сгенерированное алгоритмом L, при t, равном (а) 2? (б) 3? (в) 4? (г) 5? (д) 1000000?

17. [*HM25*] Каким образом можно вычислить комбинаторное представление (20) для данных N и t?

▶ **18.** [*20*] Какое бинарное дерево мы получим при представлении биномиального дерева T_n с указателями на "правого ребенка" и "левого брата", как в упр. 2.3.2–5?

19. [*21*] Вместо того чтобы помечать ветви биномиального дерева T_4 так, как показано в (22), можно пометить каждый узел битовой строкой соответствующего сочетания.

Если пометить таким образом T_∞, опуская начальные нули, то прямой порядок обхода даст бинарные числа в обычном возрастающем порядке; так, миллионный узел превратится в 11110100001000111111. А каким будет миллионный узел T_∞ при обходе в *обратном порядке*?

20. [*M20*] Найдите производящие функции g и h, такие, что алгоритм F находит ровно $[z^N]\, g(z)$ допустимых сочетаний и устанавливает $t \leftarrow t + 1$ ровно $[z^N]\, h(z)$ раз.

21. [*M22*] (Joan E. Miller, 1971.) Докажите закон знакопеременных сочетаний (30).

22. [*M23*] Каково миллионное сочетание двери-вертушки, посещенное алгоритмом R при t, равном (а) 2? (б) 3? (в) 4? (г) 5? (д) 1000000?

23. [*M23*] Предположим, что мы изменили алгоритм R, устанавливая $j \leftarrow t+1$ на шаге R1 и устанавливая $j \leftarrow 1$, если из шага R3 выполняется переход к R2. Найдите распределение вероятности j и его среднее значение. Как эта величина связана со временем работы алгоритма?

▶ **24.** [*M25*] (В. Г. Пейн (W. H. Payne), 1974.) Продолжим выполнять предыдущее упражнение. Пусть j_k — значение j при k-м посещении в алгоритме R. Покажите, что $|j_{k+1}-j_k| \le 2$, и поясните, как отказаться от использования циклов на основе данного свойства.

25. [*M35*] Пусть $c_t \ldots c_2 c_1$ и $c'_t \ldots c'_2 c'_1$ — N-е и N'-е сочетания, сгенерированные методом двери-вертушки (алгоритм R). Докажите, что $|N - N'| > \sum_{k=1}^{m-1} \binom{2k}{k-1}$, если множество $C = \{c_t, \ldots, c_2, c_1\}$ имеет $m > 0$ элементов, не входящих в $C' = \{c'_t, \ldots, c'_2, c'_1\}$.

26. [*26*] Обладают ли элементы *тернарного* рефлексивного кода Грея свойствами, аналогичными свойствам кода Грея двери-вертушки Γ_{st}, если мы возьмем только те n-кортежи $a_{n-1} \ldots a_1 a_0$, у которых (а) $a_{n-1} + \cdots + a_1 + a_0 = t$? (б) $\{a_{n-1}, \ldots, a_1, a_0\} = \{r \cdot 0, s \cdot 1, t \cdot 2\}$?

▶ **27.** [*25*] Покажите, что существует простой способ генерации всех сочетаний *не более* t элементов из $\{0, 1, \ldots, n-1\}$ с использованием только переходов в духе кода Грея $0 \leftrightarrow 1$ и $01 \leftrightarrow 10$. (Другими словами, на каждом шаге должны выполняться вставка элемента, удаление элемента или сдвиг элемента на ± 1.) Примером такой последовательности является

$$0000,\ 0001,\ 0011,\ 0010,\ 0110,\ 0101,\ 0100,\ 1100,\ 1010,\ 1001,\ 1000$$

при $n = 4$ и $t = 2$. *Указание*: подумайте о китайских кольцах.

28. [*M21*] Истинно или ложно следующее утверждение? Последовательность всех (s,t)-сочетаний $a_{n-1} \ldots a_1 a_0$ в виде битовых строк находится в облексном порядке тогда и только тогда, когда соответствующие последовательности в индексном виде $b_s \ldots b_2 b_1$ (для нулей) и $c_t \ldots c_2 c_1$ (для единиц) находятся в облексном порядке.

▶ **29.** [*M28*] (Ф. Д. Чейз (P. J. Chase).) Для данной строки символов `+`, `-` и `0` назовем *R-блоком* подстроку вида $\texttt{-}^{k+1}$, которой предшествует `0`, и за которой не следует `-`. *L-блоком* назовем подстроку вида $\texttt{+-}^{k}$, за которой следует `0`; в обоих случаях $k \ge 0$. Например, строка `+00++-+++-000-` имеет два L-блока и один R-блок, заключенные в рамки. Заметим, что блоки не могут пересекаться.

Для таких строк при наличии хотя бы одного блока мы формируем строку-*преемник* следующим образом. Заменим крайнюю справа последовательность $\texttt{0-}^{k+1}$ на $\texttt{-+}^{k}\texttt{0}$, если крайний справа блок является R-блоком; в противном случае заменим крайнюю справа последовательность $\texttt{+-}^{k}\texttt{0}$ на $\texttt{0+}^{k+1}$. Кроме того, обращаем первый знак (если таковой имеется), который находится справа от измененного блока. Например,

$$\texttt{-+00++-} \to \texttt{-0+0-+-} \to \texttt{-0+-0--} \to \texttt{-0+--+0} \to \texttt{-0+--0+} \to \texttt{-00+++-},$$

где запись $\alpha \to \beta$ означает, что β является преемником α.

а) Какие строки не содержат блоков (а значит, не имеют преемников)?

б) Может ли существовать цикл строк $\alpha_0 \to \alpha_1 \to \cdots \to \alpha_{k-1} \to \alpha_0$?

в) Докажите, что если $\alpha \to \beta$, то $-\beta \to -\alpha$, где "$-$" означает "обратить все знаки" (а следовательно, каждая строка имеет не более одного предшественника).

г) Покажите, что если $\alpha_0 \to \alpha_1 \to \cdots \to \alpha_k$ и $k > 0$, то строки α_0 и α_k не содержат все их нули в одних и тех же позициях (таким образом, если α_0 имеет s знаков и t нулей, k должно быть меньше $\binom{s+t}{t}$).

д) Докажите, что каждая строка α с s знаками и t нулями принадлежит ровно одной цепочке $\alpha_0 \to \alpha_1 \to \cdots \to \alpha_{\binom{s+t}{t}-1}$.

30. [*M32*] В предыдущем упражнении определены 2^s способов генерации всех сочетаний из s нулей и t единиц посредством отображений `+` $\mapsto 0$, `-` $\mapsto 0$ и `0` $\mapsto 1$. Покажите, что каждый из этих способов дает однородную облексную последовательность, определяемую соответствующим рекуррентным соотношением. Является ли последовательность Чейза (37) частным случаем этой более общей конструкции?

31. [*M23*] Сколько всего облексных последовательностей (s,t)-сочетаний может быть при использовании представления (а) в виде битовых строк $a_{n-1}\ldots a_1a_0$? (б) в виде списка индексов $c_t\ldots c_2c_1$?

▶ **32.** [*M32*] Сколько всего облексных последовательностей строк (s,t)-сочетаний $a_{n-1}\ldots a_1a_0$ (а) обладают свойством двери-вертушки? (б) однородны?

33. [*HM33*] Сколько облексных последовательностей в упр. 31, (б) близки к идеальным?

34. [*M32*] Продолжая выполнять упр. 33, поясните, как при не слишком больших s и t найти такие схемы, которые как можно более близки к идеальным в том смысле, что количество "неидеальных" переходов $c_j \leftarrow c_j \pm 2$ минимально.

35. [*M26*] На скольких шагах последовательности Чейза C_{st} используются неидеальные переходы?

▶ **36.** [*M21*] Докажите, что метод (39) при генерации всех (s,t)-сочетаний $a_{n-1}\ldots a_1a_0$ выполняет операцию $j \leftarrow j+1$ ровно $\binom{s+t}{t}-1$ раз для любой облексной схемы сочетаний в виде битовых строк.

▶ **37.** [*27*] Какой алгоритм получается при использовании общего облексного метода (39) для получения (s,t)-сочетаний $a_{n-1}\ldots a_1a_0$: (а) в лексикографическом порядке? (б) в порядке двери-вертушки из алгоритма R? (в) в однородном порядке (31)?

38. [*26*] Разработайте облексный алгоритм наподобие алгоритма C для *обратной* последовательности C^R_{st}.

39. [*M21*] Сколько сочетаний предшествует битовой строке 110010010000111111101101010 в последовательности Чейза C_{st} при $s=12$ и $t=14$? (См. (41).)

40. [*M22*] Каково миллионное сочетание в последовательности Чейза C_{st} при $s=12$ и $t=14$?

41. [*M27*] Покажите, что имеется перестановка $c(0), c(1), c(2), \ldots$ неотрицательных целых чисел, такая, что элементы последовательности Чейза C_{st} получаются путем дополнения $s+t$ младших битов элементов $c(k)$ для $0 \le k < 2^{s+t}$, которые имеют вес $\nu(c(k)) = s$. (Таким образом, последовательность $\bar{c}(0), \ldots, \bar{c}(2^n-1)$ содержит в качестве подпоследовательностей все C_{st}, для которых $s+t=n$, так же как бинарный код Грея $g(0), \ldots, g(2^n-1)$ содержит все последовательности двери-вертушки Γ_{st}.) Поясните, как вычислить бинарное представление $c(k) = (\ldots a_2a_1a_0)_2$ на основе бинарного представления $k = (\ldots b_2b_1b_0)_2$.

42. [*HM34*] Используйте производящую функцию вида $\sum_{s,t} g_{st}w^sz^t$ для анализа каждого шага алгоритма C.

43. [*20*] Докажите или опровергните следующее утверждение: если $s(x)$ и $p(x)$ обозначают соответственно преемник и предшественник x в чун-порядке (см. с. 422), то $s(x+1) = p(x)+1$.

▶ **44.** [*M21*] Пусть $C_t(n)-1$ обозначает последовательность, получаемую из $C_t(n)$ путем исключения всех сочетаний с $c_1=0$ и последующей замены $c_t\ldots c_1$ на $(c_t-1)\ldots(c_1-1)$ в оставшихся сочетаниях. Покажите, что последовательность $C_t(n)-1$ близка к идеальной.

45. [*32*] Воспользуйтесь чун-порядком и наброском расширения из уравнения (44) для генерации сочетаний $c_t \ldots c_2 c_1$ последовательности Чейза $C_t(n)$ с помощью нерекурсивной процедуры.

▶ **46.** [*33*] Разработайте нерекурсивный алгоритм для дуальных сочетаний $b_s \ldots b_2 b_1$ последовательности Чейза C_{st}, а именно для позиций нулей в $a_{n-1} \ldots a_1 a_0$.

47. [*26*] Реализуйте близкий к идеальному метод генерации перестановок мультимножества из (46) и (47).

48. [*M21*] Предположим, что $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}$ — любая последовательность перестановок мультимножества $\{s_1 \cdot 1, \ldots, s_d \cdot d\}$, где α_k отличается от α_{k+1} обменом двух элементов. Пусть $\beta_0, \ldots, \beta_{M-1}$ — любая последовательность (s,t)-сочетаний двери-вертушки, где $s = s_0$, $t = s_1 + \cdots + s_d$, а $M = \binom{s+t}{t}$. Далее, пусть Λ_j — последовательность из M элементов, полученная путем применения обменов двери-вертушки к начальной строке $\alpha_j \uparrow \beta_0$; здесь $\alpha \uparrow \beta$ означает строку, полученную заменой элементами α единиц в β с сохранением порядка слева направо. Например, если $\beta_0, \ldots, \beta_{M-1}$ представляют собой 0110, 0101, 1100, 1001, 0011, 1010, а $\alpha_j = 12$, то Λ_j представляет собой 0120, 0102, 1200, 1002, 0012, 1020. (Последовательность двери-вертушки *не* обязательно однородна.)

Докажите, что последовательность (47) содержит все перестановки $\{s_0 \cdot 0, s_1 \cdot 1, \ldots, s_d \cdot d\}$ и что соседние перестановки отличаются одна от другой обменом двух элементов.

49. [*HM23*] Пусть q — первообразный m-й корень единицы, такой, как $e^{2\pi i/m}$. Покажите, что

$$\binom{n}{k}_q = \binom{\lfloor n/m \rfloor}{\lfloor k/m \rfloor} \binom{n \bmod m}{k \bmod m}_q.$$

▶ **50.** [*HM25*] Распространите формулу из предыдущего упражнения на *q-мультиномиальные* (*q-multinomial*) коэффициенты

$$\binom{n_1 + \cdots + n_t}{n_1, \ldots, n_t}_q.$$

51. [*25*] Найдите все гамильтоновы пути в графе, вершины которого представляют собой перестановки $\{0,0,0,1,1,1\}$, причем дуги соединяют вершины, получающиеся при перестановке двух соседних элементов. Какие из этих путей эквивалентны относительно операции замены нулей единицами и/или лево-правого отражения?

52. [*M37*] Обобщая теорему P, найдите необходимое и достаточное условие того, что все перестановки мультимножества $\{s_0 \cdot 0, \ldots, s_d \cdot d\}$ могут быть сгенерированы с помощью перестановок соседних элементов $a_j a_{j-1} \leftrightarrow a_{j-1} a_j$.

53. [*M46*] (Д. Г. Лемер (D. H. Lehmer), 1965.) Предположим, что N перестановок $\{s_0 \cdot 0, \ldots, s_d \cdot d\}$ не могут быть сгенерированы с помощью идеальной схемы, поскольку $(N+x)/2$ из них имеют четное количество инверсий, где $x \geq 2$. Можно ли сгенерировать их все с помощью последовательности $N + x - 2$ обменов соседних элементов $a_{\delta_k} \leftrightarrow a_{\delta_k - 1}$ для $1 \leq k < N + x - 1$, где имеется $x - 1$ случаев "шпор" с $\delta_k = \delta_{k-1}$, возвращающих нас к перестановкам, с которыми мы только что сталкивались? Например, подходящая последовательность $\delta_1 \ldots \delta_{94}$ для 90 перестановок $\{0,0,1,1,2,2\}$, где $x = \binom{2+2+2}{2,2,2}_{-1} = 6$, имеет вид $234535432523451\alpha42\alpha^R51\alpha42\alpha^R51\alpha4$, где $\alpha = 45352542345355$, если начать с $a_5 a_4 a_3 a_2 a_1 a_0 = 221100$.

54. [*M40*] Для каких значений s и t могут быть сгенерированы все (s,t)-сочетания, если в дополнение к обменам смежных элементов $a_j \leftrightarrow a_{j-1}$ разрешить обмен конечных элементов $a_{n-1} \leftrightarrow a_0$?

▶ **55.** [*33*] (Фрэнк Раски (Frank Ruskey), 2004.) (a) Покажите, что все (s,t)-сочетания $a_{s+t-1} \ldots a_1 a_0$ можно эффективно сгенерировать путем последовательных циклических сдвигов

$a_j a_{j-1} \ldots a_0 \leftarrow a_{j-1} \ldots a_0 a_j$. (б) Какие команды MMIX превращают $(a_{s+t-1} \ldots a_1 a_0)_2$ в следующий за ним элемент при $s + t < 64$?

56. [*M49*] (Бак (Buck) и Видеманн (Wiedemann), 1984.) Можно ли сгенерировать все (t, t)-сочетания $a_{2t-1} \ldots a_1 a_0$ путем повторных обменов a_0 с некоторым другим элементом?

▶ **57.** [*22*] (Фрэнк Раски (Frank Ruskey).) Может ли пианист проиграть все возможные четырехнотные аккорды, размах которых не превышает одной октавы, перемещая всякий раз при переходе к новому аккорду только один палец? Это задача генерации всех сочетаний $c_t \ldots c_1$, таких, что $n > c_t > \cdots > c_1 \geq 0$ и $c_t - c_1 < m$, где $t = 4$ и (а) $m = 8$, $n = 52$, если рассматривать только белые клавиши пианино; (б) $m = 13$, $n = 88$, если рассматривать и черные клавиши пианино.

58. [*20*] Рассмотрите задачу о пианисте из упр. 57 с дополнительным условием, что аккорды не содержат соседних нот. (Другими словами, $c_{j+1} > c_j + 1$ для $t > j \geq 1$. Такие аккорды обычно более гармоничны.)

59. [*M25*] Имеется ли *идеальное* решение задачи о пианисте с четырьмя нотами, в которой на каждом шаге палец перемещался бы на *соседнюю* клавишу?

60. [*23*] Разработайте алгоритм для генерации всех *ограниченных* композиций

$$t = r_s + \cdots + r_1 + r_0, \qquad \text{где } 0 \leq r_j \leq m_j \text{ для } s \geq j \geq 0.$$

61. [*32*] Покажите, что все ограниченные композиции могут быть сгенерированы путем изменения на каждом шаге только двух из частей.

▶ **62.** [*M27*] *Факторная таблица* представляет собой матрицу неотрицательных целых чисел (a_{ij}) размером $m \times n$, обладающую заданными суммами строк $r_i = \sum_{j=1}^{n} a_{ij}$ и столбцов $c_j = \sum_{i=1}^{m} a_{ij}$, где $r_1 + \cdots + r_m = c_1 + \cdots + c_n$.

а) Покажите, что факторные таблицы размером $2 \times n$ эквивалентны ограниченным композициям.

б) Какая факторная таблица является лексикографически наибольшей для $(r_1, \ldots, r_m; c_1, \ldots, c_n)$, если считывать элементы матрицы построчно слева направо и сверху вниз в порядке $(a_{11}, a_{12}, \ldots, a_{1n}, a_{21}, a_{22}, \ldots, a_{2n}, \ldots, a_{m1}, a_{m2}, \ldots, a_{mn})$?

в) Какая факторная таблица является лексикографически наибольшей для $(r_1, \ldots, r_m; c_1, \ldots, c_n)$, если считывать элементы матрицы по столбцам сверху вниз и по строкам слева направо в порядке $(a_{11}, a_{21}, \ldots, a_{m1}, a_{12}, a_{22}, \ldots, a_{m2}, \ldots, a_{1n}, a_{2n}, \ldots, a_{mn})$?

г) Какая факторная таблица является лексикографически наименьшей для $(r_1, \ldots, r_m; c_1, \ldots, c_n)$ при построчном и постолбцовом считывании?

д) Поясните, как сгенерировать все факторные таблицы для $(r_1, \ldots, r_m; c_1, \ldots, c_n)$ в лексикографическом порядке.

63. [*M41*] Покажите, что все факторные таблицы для $(r_1, \ldots, r_m; c_1, \ldots, c_n)$ могут быть сгенерированы с помощью изменения на каждом шаге ровно четырех элементов матрицы.

▶ **64.** [*M30*] Постройте облексный цикл Грея для всех из $2^s \binom{s+t}{t}$ *подкубов*, которые имеют s цифр и t звездочек, используя только следующие преобразования: $*0 \leftrightarrow 0*$, $*1 \leftrightarrow 1*$, $0 \leftrightarrow 1$. Например, одним таким циклом при $s = t = 2$ является

$$(00{**}, 01{**}, 0{*}1{*}, 0{**}1, 0{**}0, 0{*}0{*}, {*}00{*}, {*}01{*}, {*}0{*}1, {*}0{*}0, {**}00, {**}01,$$
$${**}11, {**}10, {*}1{*}0, {*}1{*}1, {*}11{*}, {*}10{*}, 1{*}0{*}, 1{**}0, 1{**}1, 1{*}1{*}, 11{**}, 10{**}).$$

65. [*M40*] Подсчитайте общее количество облексных путей Грея в подкубах, в которых используются только разрешенные в упр. 64 преобразования. Сколько из этих путей являются циклами?

▶ **66.** [*22*] Покажите, что для данных $n \geq t \geq 0$ существует путь Грея через все канонические базисы $(\alpha_1, \ldots, \alpha_t)$ из упр. 12, в котором на каждом шаге выполняется изменение только одного бита. Например, приведем один такой путь для $n = 3$ и $t = 2$:

$$\begin{matrix}001\\010\end{matrix}, \quad \begin{matrix}101\\010\end{matrix}, \quad \begin{matrix}101\\110\end{matrix}, \quad \begin{matrix}001\\110\end{matrix}, \quad \begin{matrix}001\\100\end{matrix}, \quad \begin{matrix}011\\100\end{matrix}, \quad \begin{matrix}010\\100\end{matrix}.$$

67. [*46*] Рассмотрим конфигурации Изинга из упр. 13, для которых $a_0 = 0$. Существует ли цикл Грея для этих конфигураций при заданных n, t и r, в котором все переходы имеют вид $0^k1 \leftrightarrow 10^k$ или $01^k \leftrightarrow 1^k0$? Например, в случае $n = 9$, $t = 5$, $r = 6$ имеется единственный цикл

$$(010101110,\ 010110110,\ 011010110,\ 011011010,\ 011101010,\ 010111010).$$

68. [*M01*] Если α является t-сочетанием, то что собой представляет (а) $\partial^t\alpha$? (б) $\partial^{t+1}\alpha$?

▶ **69.** [*M22*] Насколько велико наименьшее множество t-сочетаний A, для которого выполняется соотношение $|\partial A| < |A|$?

70. [*M25*] Чему равно максимальное значение $\kappa_t N - N$ при $N \geq 0$?

71. [*M20*] Сколько t-клик может иметь граф с миллионом ребер?

▶ **72.** [*M22*] Покажите, что если N имеет комбинаторное представление степени t (57), то существует простой способ нахождения комбинаторного представления степени s дополняющего числа $M = \binom{s+t}{t} - N$ для $N < \binom{s+t}{t}$. Выведите (63) в качестве следствия.

73. [*M23*] (Э. Д. В. Хилтон (A. J. W. Hilton), 1976.) Пусть A — множество s-сочетаний, а B — множество t-сочетаний, и оба содержатся в $U = \{0, \ldots, n-1\}$, где $n \geq s + t$. Покажите, что если A и B *перекрестно пересекающиеся множества* (*cross-intersecting*) в том смысле, что $\alpha \cap \beta \neq \emptyset$ для всех $\alpha \in A$ и $\beta \in B$, то такими же являются и множества Q_{Mns} и Q_{Nnt}, определенные в теореме K, где $M = |A|$ и $N = |B|$.

74. [*M21*] Чему равны $|\varrho P_{Nt}|$ и $|\varrho Q_{Nnt}|$ в теореме K?

75. [*M20*] Правая часть (60) не всегда является комбинаторным представлением степени $(t-1)$ для $\kappa_t N$, поскольку $v - 1$ может быть равно нулю. Покажите, однако, что положительное целое N имеет не больше двух представлений, если в (57) допустить $v = 0$, и оба они дают одно и то же значение $\kappa_t N$ согласно (60). Следовательно,

$$\kappa_k\kappa_{k+1}\ldots\kappa_t N = \binom{n_t}{k-1} + \binom{n_{t-1}}{k-2} + \cdots + \binom{n_v}{k-1+v-t} \qquad \text{для } 1 \leq k \leq t.$$

76. [*M20*] Найдите простую формулу для $\kappa_t(N+1) - \kappa_t N$.

▶ **77.** [*M26*] Докажите следующие свойства функции κ, работая с биномиальными коэффициентами, без учета теоремы K.

а) $\kappa_t(M+N) \leq \kappa_t M + \kappa_t N$.

б) $\kappa_t(M+N) \leq \max(\kappa_t M, N) + \kappa_{t-1} N$.

Указание: $\binom{m_t}{t} + \cdots + \binom{m_1}{1} + \binom{n_t}{t} + \cdots + \binom{n_1}{1}$ равно $\binom{m_t \vee n_t}{t} + \cdots + \binom{m_1 \vee n_1}{1} + \binom{m_t \wedge n_t}{t} + \cdots + \binom{m_1 \wedge n_1}{1}$, где $\vee$ и $\wedge$ обозначают максимум и минимум.

78. [*M22*] Покажите, что теорема K легко следует из неравенства (б) в предыдущем упражнении. И наоборот: оба неравенства являются следствиями теоремы K. *Указание*: любое множество t-сочетаний A может быть записано как $A = A_1 + A_0 0$, где $A_1 = \{\alpha \in A \mid 0 \notin \alpha\}$.

79. [*M23*] Докажите, что для $t \geq 2$ соотношение $M \geq \mu_t N$ справедливо тогда и только тогда, когда $M + \lambda_{t-1} M \geq N$.

80. [*HM26*] (Л. Ловас (L. Lovász), 1979.) Функция $\binom{x}{t}$ монотонно возрастает от 0 до ∞ при увеличении x от $t-1$ до ∞; следовательно, можно определить

$$\underline{\kappa}_t N = \binom{x}{t-1}, \qquad \text{если } N = \binom{x}{t} \text{ и } x \ge t-1.$$

Докажите, что $\kappa_t N \ge \underline{\kappa}_t N$ для всех целых $t \ge 1$ и $N \ge 0$. *Указание*: неравенство становится равенством при целых значениях x.

▶ **81.** [*M27*] Покажите, что формула (64) дает минимальный размер тени в теореме M.

82. [*HM31*] Функция Такаги на рис. 47 определяется для $0 \le x \le 1$ формулой

$$\tau(x) = \sum_{k=1}^{\infty} \int_0^x r_k(t)\,dt,$$

где $r_k(t) = (-1)^{\lfloor 2^k t \rfloor}$ — функция Радемахера из уравнения 7.2.1.1–(16).

а) Докажите, что $\tau(x)$ непрерывна на отрезке $[0\,..\,1]$, но ее производная не существует ни в одной точке.

б) Покажите, что $\tau(x)$ — единственная непрерывная функция, удовлетворяющая уравнению

$$\tau(\tfrac{1}{2}x) \;=\; \tau(1-\tfrac{1}{2}x) \;=\; \tfrac{1}{2}x + \tfrac{1}{2}\tau(x) \qquad \text{при } 0 \le x \le 1.$$

в) Чему равно асимптотическое значение $\tau(\epsilon)$ при малых ϵ?

г) Докажите, что значение $\tau(x)$ при рациональном x рационально.

д) Найдите все корни уравнения $\tau(x) = 1/2$.

е) Найдите все корни уравнения $\tau(x) = \max_{0\le x\le 1} \tau(x)$.

83. [*HM46*] Определите множество R всех рациональных чисел r, таких, что уравнение $\tau(x) = r$ имеет несчетное количество решений. Если $\tau(x)$ рационально, а x — иррационально, верно ли, что $\tau(x) \in R$? (*Предупреждение*: эта задача может оказаться настоящим наркотиком.)

84. [*HM27*] Докажите для $T = \binom{2t-1}{t}$ асимптотическую формулу

$$\kappa_t N - N \;=\; \frac{T}{t}\Bigl(\tau\Bigl(\frac{N}{T}\Bigr) + O\Bigl(\frac{(\log t)^3}{t}\Bigr)\Bigr) \qquad \text{при } 0 \le N \le T.$$

85. [*HM21*] Свяжите функции $\lambda_t N$ и $\mu_t N$ с функцией Такаги $\tau(x)$.

86. [*M20*] Докажите закон дуальности размаха/ядра $X^{\sim+} = X^{\circ\sim}$.

87. [*M21*] Истинны или ложны следующие утверждения? (а) $X \subseteq Y^\circ$ тогда и только тогда, когда $Y^\sim \subseteq X^{\sim\circ}$; (б) $X^{\circ+\circ} = X^\circ$; (в) $\alpha M \le N$ тогда и только тогда, когда $M \le \beta N$.

88. [*M20*] Поясните, в чем полезность перекрестного порядка, завершив доказательство леммы S.

89. [*16*] Вычислите функции α и β для тора $2 \times 2 \times 3$ (69).

90. [*M22*] Докажите фундаментальную лемму о сжатии (85).

91. [*M24*] Докажите теорему W для двумерных торов $T(l, m)$, $l \le m$.

92. [*M28*] Пусть $x = x_1 \ldots x_{n-1}$ — N-й элемент тора $T(m_1, \ldots, m_{n-1})$ и пусть S — множество всех элементов $T(m_1, \ldots, m_{n-1}, m)$, которые $\preceq x_1 \ldots x_{n-1}(m-1)$ в перекрестном порядке. Пусть N_a элементов S имеют последним компонентом a $(0 \le a < m)$. Докажите, что $N_{m-1} = N$ и $N_{a-1} = \alpha N_a$ для $1 \le a < m$, где α — функция размаха для стандартных множеств в $T(m_1, \ldots, m_{n-1})$.

93. [*M25*] (а) Найдите N, для которого результат теоремы W ложен, если параметры m_1, m_2, …, m_n не располагаются в неубывающем порядке. (б) Где в доказательстве этой теоремы используется гипотеза о том, что $m_1 \le m_2 \le \cdots \le m_n$?

94. [*M20*] Покажите, что ∂-часть следствия C вытекает из ϱ-части. *Указание*: дополнениями мультисочетаний (92) по отношению к U являются 3211, 3210, 3200, 3110, 3100, 3000, 2110, 2100, 2000, 1100, 1000.

95. [*17*] Объясните, почему теоремы K и M вытекают из следствия C.

▶ **96.** [*M22*] Пусть S — бесконечная последовательность $(s_0, s_1, s_2, \ldots)$ положительных целых чисел и пусть

$$\binom{S(n)}{k} = [z^k] \prod_{j=0}^{n-1} (1 + z + \cdots + z^{s_j});$$

таким образом, $\binom{S(n)}{k}$ при $s_0 = s_1 = s_2 = \cdots = 1$ представляет собой обычный биномиальный коэффициент $\binom{n}{k}$.

Обобщая комбинаторную систему счисления, покажите, что каждое неотрицательное целое N имеет единственное представление

$$N = \binom{S(n_t)}{t} + \binom{S(n_{t-1})}{t-1} + \cdots + \binom{S(n_1)}{1},$$

где $n_t \geq n_{t-1} \geq \cdots \geq n_1 \geq 0$ и $\{n_t, n_{t-1}, \ldots, n_1\} \subseteq \{s_0 \cdot 0, s_1 \cdot 1, s_2 \cdot 2, \ldots\}$. Воспользуйтесь этим представлением для получения простой формулы для чисел $|\partial P_{Nt}|$ в следствии C.

▶ **97.** [*M26*] В разделе было замечено, что вершины выпуклого многогранника могут быть слегка смещены таким образом, что все его грани будут представлять собой симплексы. В общем случае любое множество сочетаний, которое содержит тени всех своих элементов, называется *симплициальным комплексом* (*simplicial complex*); таким образом, C является симплициальным комплексом тогда и только тогда, когда $\alpha \subseteq \beta$, а из $\beta \in C$ вытекает $\alpha \in C$ тогда и только тогда, когда C представляет собой порядковый идеал по отношению к включению множеств.

Если C содержит в точности N_t сочетаний размером t, то $(N_0, N_1, \ldots, N_n)$ представляет собой *вектор размеров* симплициального комплекса из n вершин.

а) Чему равны векторы размеров пяти правильных многогранников (тетраэдра, куба, октаэдра, додекаэдра и икосаэдра) при небольшой подстройке их вершин?

б) Постройте симплициальный комплекс с вектором размеров $(1, 4, 5, 2, 0)$.

в) Найдите необходимое и достаточное условие допустимости заданного вектора $(N_0, N_1, \ldots, N_n)$.

г) Докажите, что $(N_0, \ldots, N_n)$ является допустимым тогда и только тогда, когда допустим его "дуальный" вектор $(\overline{N}_0, \ldots, \overline{N}_n)$, где $\overline{N}_t = \binom{n}{t} - N_{n-t}$.

д) Перечислите все допустимые векторы размеров $(N_0, N_1, N_2, N_3, N_4)$ и дуальные им. Какие из этих векторов самодуальны?

98. [*30*] Продолжая выполнять упр. 97, найдите эффективный способ подсчета количества допустимых векторов размеров $(N_0, N_1, \ldots, N_n)$ при $n \leq 100$.

99. [*M25*] *Клаттером* (*clutter*)* называется множество C сочетаний, несравнимых в том смысле, что из $\alpha \subseteq \beta$ и $\alpha, \beta \in C$ вытекает $\alpha = \beta$. Вектор размеров клаттера определяется так же, как в упр. 97.

а) Найдите необходимое и достаточное условие того, что $(M_0, M_1, \ldots, M_n)$ является вектором размеров клаттера.

б) Перечислите все такие векторы размеров для $n = 4$.

▶ **100.** [*M30*] (Клементс (Clements) и Линдстрем (Lindström).) Пусть A — "симплициальный мультикомплекс", множество подмультимножеств мультимножества U в следствии C, обладающее тем свойством, что $\partial A \subseteq A$. Насколько больши́м может быть общий вес $\nu A = \sum\{|\alpha| \mid \alpha \in A\}$ при $|A| = N$?

* Дословно — "беспорядком". — *Примеч. пер.*

101. [*M25*] Если $f(x_1,\ldots,x_n)$ — булева формула, обозначим через $F(p)$ вероятность того, что $f(x_1,\ldots,x_n)=1$, если каждая переменная x_j независимо равна 1 с вероятностью p.

а) Вычислите $G(p)$ и $H(p)$ для булевых формул $g(w,x,y,z)=wxz\vee wyz\vee xy\bar z$, $h(w,x,y,z)=\bar wyz\vee xyz$.

б) Покажите, что существует *монотонная* булева функция $f(w,x,y,z)$, такая, что $F(p)=G(p)$, но не существует такой функции, что $F(p)=H(p)$. Поясните, как проверить это условие в общем случае.

102. [*HM35*] (Ф. С. Маколей (F. S. Macaulay), 1927.) *Полиномиальным идеалом* (*polynomial ideal*) I для переменных $\{x_1\ldots,x_s\}$ является множество полиномов, замкнутое по отношению к операциям сложения, умножения на константу и умножения на любую из переменных. Он называется *однородным* (*homogeneous*), если состоит из линейных комбинаций множества однородных полиномов, т. е. полиномов наподобие $xy+z^2$, все члены которых имеют одну и ту же степень. Пусть N_t — максимальное число линейно независимых элементов степени t в I. Например, при $s=2$ множество всех $\alpha(x_0,x_1,x_2)(x_0x_1^2-2x_1x_2^2)+\beta(x_0,x_1,x_2)x_0x_1x_2^2$, где α и β пробегают по всем возможным полиномам от $\{x_0,x_1,x_2\}$, представляет собой однородный полиномиальный идеал с $N_0=N_1=N_2=0$, $N_3=1$, $N_4=4$, $N_5=9$, $N_6=15$, $\ldots$.

а) Докажите, что для любого такого идеала I существует другой идеал I', в котором все однородные полиномы степени t представляют собой линейные комбинации N_t независимых *одночленов* (*monomial*). (Одночлен представляет собой произведение переменных наподобие $x_1^3x_2x_5^4$.)

б) Используйте теорему M и (64) для того, чтобы доказать, что $N_{t+1}\ge N_t+\kappa_sN_t$ для всех $t\ge 0$.

в) Покажите, что неравенство $N_{t+1}>N_t+\kappa_sN_t$ выполняется только для конечного количества t. (Это утверждение эквивалентно "основной теореме Гильберта", доказанной Д. Гильбертом (D. Hilbert) в *Göttinger Nachrichten* (1888), 450–457; *Math. Annalen* **36** (1890), 473–534.)

▶ **103.** [*M38*] Тень подкуба $a_1\ldots a_n$, где каждое a_j является 0, 1 либо $*$, получается путем замены некоторых $*$ на 0 или 1, например:

$$\partial 0{*}11{*}0 \;=\; \{0011{*}0,\ 0111{*}0,\ 0{*}1100,\ 0{*}1110\}.$$

Пусть A — произвольное множество N подкубов $a_1\ldots a_n$ с s цифрами и t звездочками. Найдите множество P_{Nst}, такое, что $|\partial A|\ge|P_{Nst}|$.

104. [*M41*] Тень бинарной строки $a_1\ldots a_n$ получается путем удаления одного из ее битов, например:

$$\partial 110010010 \;=\; \{10010010,\ 11010010,\ 11000010,\ 11001000,\ 11001010,\ 11001001\}.$$

Найдите множество P_{Nn}, такое, что если A — произвольное множество N бинарных строк $a_1\ldots a_n$, то $|\partial A|\ge|P_{Nn}|$.

105. [*M20*] *Универсальным циклом t-сочетаний* $\{0,1,\ldots,n-1\}$ является цикл из $\binom{n}{t}$ чисел, блоки которых из t последовательных элементов пробегают все t-сочетания $\{c_1,\ldots,c_t\}$. Например,

$$(0214506132051624315263042536410354 6)$$

является универсальным циклом для $t=3$ и $n=7$.

Докажите, что такие циклы невозможны, если только $\binom{n}{t}$ не кратно n.

106. [*M21*] (Л. Пуансо (L. Poinsot), 1809.) Найдите "красивый" универсальный цикл 2-сочетаний $\{0,1,\ldots,2m\}$. *Указание*: рассмотрите разности последовательных элементов по модулю $(2m+1)$.

107. [*22*] (О. Терквем (O. Terquem), 1849.) Из теоремы Пуансо следует, что все 28 костей домино из традиционного набора "дубль-шесть" можно расположить в виде цикла так, что очки на соприкасающихся домино будут одинаковы.

Сколько всего может быть таких циклов?

108. [*M31*] Найдите универсальные циклы 3-сочетаний множеств $\{0, \ldots, n-1\}$, где $n \bmod 3 \neq 0$.

109. [*M31*] Найдите универсальные циклы 3-*мультисочетаний* множеств $\{0, 1, \ldots, n-1\}$, для которых $n \bmod 3 \neq 0$ (т. е. сочетаний $d_1 d_2 d_3$, в которых разрешены повторения). Например, при $n = 5$ таким циклом является

$$(0001224111233022234413334002444011 3).$$

▶ **110.** [*26*] *Криббедж* (*cribbage*) — карточная игра для колоды из 52 карт, каждая из которых имеет масть (♣, ♢, ♡ или ♠) и достоинство (A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q или K). Игроки в криббедж должны быть экспертами в вычислении счета для сочетаний 5 карт $C = \{c_1, c_2, c_3, c_4, c_5\}$, одна из которых, c_k, называется *стартером*. Счет представляет собой сумму очков, вычисляемую следующим образом для каждого подмножества S множества C и каждого выбора k. Пусть $|S| = s$.

i) Пятнадцать: если $\sum\{v(c) \mid c \in S\} = 15$, где $(v(\mathtt{A}), v(\mathtt{2}), v(\mathtt{3}), \ldots, v(\mathtt{9}), v(\mathtt{10}), v(\mathtt{J}), v(\mathtt{Q}), v(\mathtt{K})) = (1, 2, 3, \ldots, 9, 10, 10, 10, 10)$, засчитывается два очка.

ii) Пары: если $s = 2$ и обе карты одного и того же достоинства, засчитывается два очка.

iii) "Тропа" (run): если $s \geq 3$ и достоинства карт следуют друг за другом и если C не содержит тропы длиной $s + 1$, засчитывается s очков.

iv) Масть (flush): если $s = 4$ и все карты S имеют одну и ту же масть и если $c_k \notin S$, засчитывается $4 +$ [c_k имеет ту же масть, что и другие карты] очков.

v) Козырный валет (nob): если $s = 1$ и $c_k \notin S$, засчитывается одно очко, если карта — валет (J) той же масти, что и c_k.

Например, если у вас на руках $\{\mathtt{J}\clubsuit, \mathtt{5}\clubsuit, \mathtt{5}\diamondsuit, \mathtt{6}\heartsuit\}$ и стартером является 4♣, то ваш счет составляет 4×2 за пятнадцать, 2 — за пару, 2×3 — за тропы и 1 — за козырного валета, итого — 17 очков.

Сколько в точности вариантов сочетаний карт и стартера приводят к счету, равному x очкам ($x = 0, 1, 2, \ldots$)?

▶ **111.** [*M26*] (П. Эрдеш (P. Erdős), Ч. Ко (C. Ko) и Р. Радо (R. Rado).) Предположим, что A — множество r-сочетаний n-множества, причем $\alpha \cap \beta \neq \emptyset$ для любых $\alpha, \beta \in A$. Покажите, что $|A| \leq \binom{n-1}{r-1}$, если $r \leq n/2$. *Указание*: рассмотрите $\partial^{n-2r} B$, где B — множество дополнений A.

7.2.1.4. Генерация всех разбиений. Великолепная книга Ричарда Стенли (Richard Stanley) *Перечислительная комбинаторика* (*Enumerative Combinatorics*, 1986) начинается с рассмотрения "двенадцатизадачия" — массива размером $2 \times 2 \times 3$ фундаментальных комбинаторных задач, часто возникающих на практике (см. табл. 1), который базируется на ряде лекций Ж.-К. Роты (G.-C. Rota). Все эти двенадцать задач можно описать в терминах распределения заданного количества шаров по

заданному количеству урн. Например, имеется девять способов поместить два шара в три урны, если и шары, и урны помечены.

(Порядок шаров *внутри* урны игнорируется.) Но если шары не помечены, то некоторые из размещений становятся неразличимыми, и становятся возможными только шесть различных способов размещения.

(1)

Если урны не помечены, то размещения наподобие и , по сути, одинаковы, следовательно, различимы только два из девяти исходных размещений. Если же у нас имеется три помеченных шара, то вот все различные способы их размещения в трех непомеченных урнах.

(2)

Наконец, если не помечены ни шары, ни урны, то показанные пять способов размещения сводятся только к трем.

(3)

В двенадцатизадачии рассматриваются все размещения для помеченных и непомеченных урн и шаров, а также необязательное требование, чтобы в каждой урне содержалось не менее или не более одного шара.

Таблица 1

Двенадцатизадачие

Шаров в урне	Ограничений нет	≤ 1	≥ 1
n помеченных шаров, m помеченных урн	n-кортежи из m объектов	n-перестановки из m объектов	Разбиения $\{1,\ldots,n\}$ на m упорядоченных частей
n непомеченных шаров, m помеченных урн	n-мультисочетания из m объектов	n-сочетания из m объектов	Композиции n из m частей
n помеченных шаров, m непомеченных урн	Разбиения $\{1,\ldots,n\}$ на $\le m$ частей	n объектов в m карманах	Разбиения $\{1,\ldots,n\}$ на m частей
n непомеченных шаров, m непомеченных урн	Разбиения n на $\le m$ частей	n объектов в m карманах	Разбиения n на m частей

Мы уже рассматривали n-кортежи, перестановки, сочетания и композиции в предыдущих разделах данной главы; две из двенадцати записей табл. 1 тривиальны — размещение n объектов по m приемным карманам. Таким образом, мы можем завершить наше изучение комбинаторной математики рассмотрением оставшихся пяти записей таблицы, которые включают *разбиения* (*partitions*).

Начнем с того, что слово "разбиение" в математике имеет множество значений. Оно возникает всякий раз, когда выполняется разделение некоторого объекта на подобъекты.

— ДЖОРДЖ ЭНДРЮС (GEORGE ANDREWS),
Теория разбиений (*The Theory of Partitions*) (1976)

Одно и то же имя имеют две достаточно разные концепции. *Разбиения множества* представляют собой способы его разделения на непересекающиеся подмножества; таким образом, (2) иллюстрирует пять разбиений $\{1,2,3\}$, а именно

$$\{1,2,3\}, \qquad \{1,2\}\{3\}, \qquad \{1,3\}\{2\}, \qquad \{1\}\{2,3\}, \qquad \{1\}\{2\}\{3\}. \tag{4}$$

Разбиения целого числа представляют собой способы его записи в виде суммы натуральных чисел, независимо от их порядка; таким образом, в (3) иллюстрируются три разбиения числа 3, а именно

$$3, \qquad 2+1, \qquad 1+1+1. \tag{5}$$

Мы будем следовать распространенной практике и называть разбиения целых чисел просто разбиениями, не указывая их объекта. Другой вид будем называть разбиениями множеств, чтобы ясно указывать различия между указанными видами разбиений. Важны оба вида разбиений, так что мы изучим их по очереди.

Генерация всех разбиений целого числа. Разбиение целого числа n формально можно определить как последовательность неотрицательных целых чисел $a_1 \ge a_2 \ge \cdots$, такую, что $n = a_1 + a_2 + \cdots$; например, одно из разбиений числа 7 — $a_1 = a_2 = 3$, $a_3 = 1$ и $a_4 = a_5 = \cdots = 0$. Количество ненулевых членов называется количеством *частей*, а нулевые члены обычно просто опускаются. Таким образом, мы записываем $7 = 3+3+1$, или просто 331, чтобы сэкономить место, если контекст очевиден.

Простейший (и один из быстрейших) способ генерации всех разбиений — посетить их в обратном лексикографическом порядке, начиная с 'n' и заканчивая '$11\ldots1$.' Например, разбиения 8 в указанном порядке представляют собой

$$\begin{gathered}8,\ 71,\ 62,\ 611,\ 53,\ 521,\ 5111,\ 44,\ 431,\ 422,\ 4211,\ 41111,\ 332,\ 3311,\\ 3221,\ 32111,\ 311111,\ 2222,\ 22211,\ 221111,\ 2111111,\ 11111111.\end{gathered} \tag{6}$$

Если разбиение состоит не из одних единиц, оно завершается некоторым числом $(x{+}1)$, где $x \ge 1$, за которым следует нуль или несколько единиц; следовательно, очередное наименьшее разбиение в лексикографическом порядке получается путем замены суффикса $(x{+}1)1\ldots1$ на $x\ldots xr$ для некоторого соответствующего остатка $r \le x$. Процесс достаточно эффективен, если отслеживать наибольший индекс q, такой, что $a_q \ne 1$, как предложено Д. К. С. Мак-Кеем (J. K. S. McKay) [*CACM* **13** (1970), 52], и заполнять массив единицами, как предложено А. Зогби (A. Zoghbi) и И. Стойменовичем (I. Stojmenović) [*International Journal of Computer Math.* **70** (1998), 319–332].

Алгоритм P (*Разбиения n в обратном лексикографическом порядке*). Этот алгоритм генерирует все разбиения $a_1 \ge a_2 \ge \cdots \ge a_m \ge 1$ с $a_1 + a_2 + \cdots + a_m = n$ и $1 \le m \le n$ для $n \ge 1$. Значение a_0 устанавливается равным нулю.

P1. [Инициализация.] Установить $a_m \leftarrow 1$ для $n \geq m > 1$. Затем установить $m \leftarrow 1$ и $a_0 \leftarrow 0$.

P2. [Сохранение заключительной части.] Установить $a_m \leftarrow n$ и $q \leftarrow m - [n=1]$.

P3. [Посещение.] Посетить разбиение $a_1 a_2 \ldots a_m$. Затем, если $a_q \neq 2$, перейти к шагу P5.

P4. [Замена 2 на 1+1.] Установить $a_q \leftarrow 1$, $q \leftarrow q-1$, $m \leftarrow m+1$ и вернуться к шагу P3. (В этот момент $a_k = 1$ для $q < k \leq n$.)

P5. [Уменьшение a_q.] Завершить работу, если $q = 0$. В противном случае установить $x \leftarrow a_q - 1$, $a_q \leftarrow x$, $n \leftarrow m - q + 1$ и $m \leftarrow q + 1$.

P6. [Копирование x при необходимости.] Если $n \leq x$, вернуться к шагу P2. В противном случае установить $a_m \leftarrow x$, $m \leftarrow m + 1$, $n \leftarrow n - x$ и повторить данный шаг. ▮

Обратите внимание, что переход от одного разбиения к следующему выполняется особенно просто, если имеется двойка; в этом случае шаг P4 просто заменяет крайнюю справа двойку единицей и добавляет единицу справа. К счастью, эта удачная ситуация наиболее распространена. Так, для $n = 100$ двойку содержат около 79% всех разбиений.

Если мы хотим генерировать все разбиения числа n на фиксированное количество частей, к нашим услугам другой простой алгоритм. Приведенный далее метод, разработанный в диссертации К. Ф. Гинденбурга (C. F. Hindenburg) в XVIII веке [*Infinitinomii Dignitatum Exponentis Indeterminati* (Göttingen, 1779), 73–91], посещает все разбиения в *солексном* (*colex*) порядке, т. е. в лексикографическом порядке "отраженных" последовательностей $a_m \ldots a_2 a_1$.

Алгоритм H (*Разбиение n на m частей*). Этот алгоритм генерирует все целые m-кортежи $a_1 \ldots a_m$, такие, что $a_1 \geq \cdots \geq a_m \geq 1$ и $a_1 + \cdots + a_m = n$, в предположении, что $n \geq m \geq 2$. Значение-ограничитель хранится в a_{m+1}.

H1. [Инициализация.] Установить $a_1 \leftarrow n - m + 1$ и $a_j \leftarrow 1$ для $1 < j \leq m$. Установить также $a_{m+1} \leftarrow -1$.

H2. [Посещение.] Посетить разбиение $a_1 \ldots a_m$. Затем, если $a_2 \geq a_1 - 1$, перейти к шагу H4.

H3. [Настройка a_1 и a_2.] Установить $a_1 \leftarrow a_1 - 1$, $a_2 \leftarrow a_2 + 1$ и вернуться к шагу H2.

H4. [Поиск j.] Установить $j \leftarrow 3$ и $s \leftarrow a_1 + a_2 - 1$. Затем, если $a_j \geq a_1 - 1$, установить $s \leftarrow s + a_j$ и $j \leftarrow j + 1$ и повторять эти действия до тех пор, пока не будет выполнено условие $a_j < a_1 - 1$. (Теперь $s = a_1 + \cdots + a_{j-1} - 1$ и $a_j < a_1 - 1$.)

H5. [Увеличение a_j.] Завершить работу алгоритма, если $j > m$. В противном случае установить $x \leftarrow a_j + 1$, $a_j \leftarrow x$, $j \leftarrow j - 1$.

H6. [Настройка $a_1 \ldots a_j$.] Пока $j > 1$, устанавливать $a_j \leftarrow x$, $s \leftarrow s - x$ и $j \leftarrow j - 1$. В конце установить $a_1 \leftarrow s$ и вернуться к шагу H2. ▮

Например, при $n = 11$ и $m = 4$ последовательные посещаемые алгоритмом разбиения представляют собой

$$8111,\ 7211,\ 6311,\ 5411,\ 6221,\ 5321,\ 4421,\ 4331,\ 5222,\ 4322,\ 3332. \tag{7}$$

Основная идея состоит в том, что солексный порядок переходит от одного разбиения $a_1 \ldots a_m$ к другому путем поиска наименьшего j, такого, что a_j можно увеличить без изменения $a_{j+1} \ldots a_m$. Новое разбиение $a'_1 \ldots a'_m$ будет иметь $a'_1 \geq \cdots \geq a'_j = a_j + 1$ и $a'_1 + \cdots + a'_j = a_1 + \cdots + a_j$, и эти условия достижимы тогда и только тогда, когда $a_j < a_1 - 1$. Кроме того, наименьшее такое разбиение $a'_1 \ldots a'_m$ в солексном порядке имеет $a'_2 = \cdots = a'_j = a_j + 1$.

Шаг H3 обрабатывает простой случай $j = 2$, который наиболее распространен. Следует отметить, что значение j почти всегда достаточно мало́; позже мы докажем, что общее время работы алгоритма H не превышает количества посещенных разбиений, умноженного на небольшую константу, плюс $O(m)$.

Другие представления разбиений. Мы определили разбиение как последовательность неотрицательных целых чисел $a_1 a_2 \ldots$, обладающую теми свойствами, что $a_1 \geq a_2 \geq \cdots$ и $a_1 + a_2 + \cdots = n$, но его можно рассматривать как n-кортеж неотрицательных целых чисел $c_1 c_2 \ldots c_n$, таких, что

$$c_1 + 2c_2 + \cdots + nc_n \;=\; n. \tag{8}$$

Здесь c_j — количество появлений целого числа j в последовательности $a_1 a_2 \ldots$; например, разбиение 331 соответствует количествам $c_1 = 1$, $c_2 = 0$, $c_3 = 2$, $c_4 = c_5 = c_6 = c_7 = 0$. Число частей в таком представлении равно $c_1 + c_2 + \cdots + c_n$. Можно легко разработать процедуру, аналогичную алгоритму P, для генерации разбиений в форме количества частей (см. упр. 5).

Мы уже встречались с неявным представлением в виде количества частей в формулах наподобие 1.2.9–(38), которая выражает симметричную функцию

$$h_n = \sum_{N \geq d_n \geq \cdots \geq d_2 \geq d_1 \geq 1} x_{d_1} x_{d_2} \ldots x_{d_n} \tag{9}$$

как

$$\sum_{\substack{c_1, c_2, \ldots, c_n \geq 0 \\ c_1 + 2c_2 + \cdots + nc_n = n}} \frac{S_1^{c_1}}{1^{c_1} c_1!} \frac{S_2^{c_2}}{2^{c_2} c_2!} \cdots \frac{S_n^{c_n}}{n^{c_n} c_n!}, \tag{10}$$

где S_j — симметричная функция $x_1^j + x_2^j + \cdots + x_N^j$. Сумма в (9), по сути, берется по всем n-мультисочетаниям из N объектов, в то время как сумма в (10) берется по всем разбиениям n. Таким образом, например, $h_3 = \frac{1}{6} S_1^3 + \frac{1}{2} S_1 S_2 + \frac{1}{3} S_3$, и при $N = 2$ мы имеем

$$x^3 + x^2 y + x y^2 + y^3 = \tfrac{1}{6}(x+y)^3 + \tfrac{1}{2}(x+y)(x^2+y^2) + \tfrac{1}{3}(x^3+y^3).$$

Другие суммы по всем разбиениям имеются в упр. 1.2.5–21, 1.2.9–10, 1.2.9–11, 1.2.10–12 и т. д.; по этой причине разбиения занимают важное место в изучении симметричных функций — класса функций, широко распространенного в математике. [Глава 7 книги Ричарда Стенли (Richard Stanley) *Enumerative Combinatorics* **2** (1999) представляет собой превосходное введение в современные аспекты теории симметричных функций.]

Разбиения можно весьма привлекательно визуализировать, рассмотрев массив из n точек, имеющий a_1 точек в верхней строке, a_2 в следующей строке и т. д. Такое размещение точек называется *диаграммой Ферररса* в честь Н. М. Ферररса (N. M. Ferrers) [см. *Philosophical Mag.* **5** (1853), 199–202]; наибольший квадратный

(а) 8887211 (б) 75444443

Рис. 48. Диаграммы Феррерса и квадраты Дюрфи для двух сопряженных разбиений.

подмассив точек, содержащийся в такой диаграмме, называется *квадратом Дюрфи* по имени В. П. Дюрфи (W. P. Durfee) [см. *Johns Hopkins Univ. Circular* **2** (December 1882), 23]. Например, на рис. 48, (а) показана диаграмма Феррерса для разбиения 8887211 с квадратом Дюрфи размером 4×4. Квадрат Дюрфи содержит k^2 точек, где k — наибольший индекс, такой, что $a_k \geq k$; можно называть k *следом* (*trace*) разбиения.

Для произвольного разбиения α вида $a_1 a_2 \ldots$ *сопряженным* (*conjugate*) с ним является разбиение $\alpha^T = b_1 b_2 \ldots$, которое получается путем транспонирования строк и столбцов соответствующей диаграммы Феррерса, т. е. путем отражения диаграммы относительно главной диагонали. Например, на рис. 48, (б) показано, что $(8887211)^T = 75444443$. Если $\beta = \alpha^T$, то, очевидно, $\alpha = \beta^T$; разбиение β имеет a_1 частей, а α — b_1 частей. Имеется простое соотношение между представлениями в виде количества частей $c_1 \ldots c_n$ разбиения α и сопряженного разбиения $b_1 b_2 \ldots$, а именно

$$b_j - b_{j+1} \;=\; c_j \qquad \text{для всех } j \geq 1. \tag{11}$$

Это соотношение позволяет легко вычислить разбиение, сопряженное данному (см. упр. 6).

Понятие сопряженности часто поясняет свойства разбиений, которые иначе выглядели бы весьма загадочными. Например, теперь, зная определение α^T, можно легко увидеть, что значение $j-1$ на шаге H5 алгоритма H представляет собой вторую наименьшую часть сопряженного разбиения $(a_1 \ldots a_m)^T$, где $m < n$. Следовательно, среднее количество работы, которую необходимо выполнить на шагах H4 и H6, по сути, пропорционально среднему размеру второй наименьшей части случайного разбиения, наибольшая часть которого равна m. Ниже мы увидим, что эта вторая наименьшая часть почти всегда достаточно мала.

Кроме того, *алгоритм H выводит разбиения в лексикографическом порядке сопряженных с ними.* Например, соответствующими сопряженными с (7) разбиениями являются

$$\begin{gathered} 41111111,\ 4211111,\ 422111,\ 42221,\ 431111, \\ 43211,\ 4322,\ 4331,\ 44111,\ 4421,\ 443; \end{gathered} \tag{12}$$

они представляют собой разбиения $n = 11$ с наибольшей частью, равной 4. Один из способов генерации всех разбиений n состоит в том, чтобы начать с тривиального разбиения 'n', а затем поочередно применить алгоритм H для $m = 2, 3, \ldots, n$; такой процесс даст все α в лексикографическом порядке α^T (см. упр. 7). Таким образом, алгоритм H можно рассматривать как дуальный к алгоритму P.

Имеется как минимум еще один способ представления разбиений, называемый *краевым представлением* (*rim representation*) [см. S. Comét, *Numer. Math.* **1** (1959),

90–109]. Предположим, что мы заменили точки на диаграмме Феррерса квадратами, получив таким образом подобие таблицы, как делалось в разделе 5.1.4; например, разбиение 8887211 с рис. 48, (a) превращается в

. (13)

Правая граница этого образа может рассматриваться как путь длиной $2n$ из нижнего левого угла в правый верхний угол квадрата размером $n \times n$, а из табл. 7.2.1.3–1 мы знаем, что такой путь соответствует (n, n)-сочетанию.

Например, (13) соответствует 70-битовой строке

$$0\ldots01001011111010001\ldots1 \;=\; 0^{28}1^{1}0^{2}1^{1}0^{1}1^{5}0^{1}1^{1}0^{3}1^{27}, \tag{14}$$

в начале которой мы размещаем достаточное количество нулей, а в конце — единиц, чтобы получить в точности по n каждых из них. Нули представляют шаги вверх, а единицы — вправо. Легко видеть, что определенная таким образом битовая строка имеет ровно n инверсий; и наоборот, каждая перестановка мультимножества $\{n{\cdot}0,\ n{\cdot}1\}$, которая содержит ровно n инверсий, соответствует разбиению n. Если разбиение имеет t различных частей, его битовая строка может быть записана в виде

$$0^{n-q_1-q_2-\cdots-q_t}1^{p_1}0^{q_1}1^{p_2}0^{q_2}\ldots1^{p_t}0^{q_t}1^{n-p_1-p_2-\cdots-p_t}, \tag{15}$$

где показатели степени p_j и q_j представляют собой положительные целые числа. В таком случае стандартным представлением разбиения является

$$a_1a_2\ldots \;=\; (p_1+\cdots+p_t)^{q_t}(p_1+\cdots+p_{t-1})^{q_{t-1}}\ldots(p_1)^{q_1}, \tag{16}$$

т. е. в нашем примере $(1{+}1{+}5{+}1)^3(1{+}1{+}5)^1(1{+}1)^1(1)^2 = 8887211$.

Количество разбиений. Вопрос, заданный в 1740 году Филиппом Ноде (Philippe Naudé), побудил Леонарда Эйлера (Leonhard Euler) написать две фундаментальные статьи, в которых он подсчитывал количества разбиений различных видов, изучая их производящие функции [*Commentarii Academiæ Scientiarum Petropolitanæ* **13** (1741), 64–93; *Novi Comment. Acad. Sci. Pet.* **3** (1750), 125–169]. Он заметил, что коэффициент при z^n в бесконечном произведении

$$(1+z+z^2+\cdots+z^j+\cdots)(1+z^2+z^4+\cdots+z^{2k}+\cdots)(1+z^3+z^6+\cdots+z^{3l}+\cdots)\ldots$$

равен количеству неотрицательных целых решений уравнения $j+2k+3l+\cdots=n$; а $1+z^m+z^{2m}+\cdots$ равно $1/(1-z^m)$. Следовательно, если мы запишем

$$P(z) \;=\; \prod_{m=1}^{\infty}\frac{1}{1-z^m} \;=\; \sum_{n=0}^{\infty}p(n)\,z^n, \tag{17}$$

то количество разбиений n будет равно $p(n)$. Функция $P(z)$, в свою очередь, обладает рядом поразительных математических свойств.

Например, Эйлер открыл наличие большого количества сокращений при раскрытии скобок знаменателя $P(z)$:

$$\begin{aligned}(1-z)(1-z^2)(1-z^3)\ldots &= 1 - z - z^2 + z^5 + z^7 - z^{12} - z^{15} + z^{22} + z^{26} - \cdots \\ &= \sum_{-\infty<n<\infty} (-1)^n z^{(3n^2+n)/2}. \qquad (18)\end{aligned}$$

Комбинаторное доказательство этого замечательного тождества, основанное на диаграммах Феррерса, имеется в упр. 5.1.1–14; его можно также доказать, положив $u = z$ и $v = z^2$ в еще более замечательном тождестве, опубликованном Якоби (Jacobi) в 1829 году

$$\prod_{k=1}^{\infty}(1 - u^k v^{k-1})(1 - u^{k-1}v^k)(1 - u^k v^k) \;=\; \sum_{n=-\infty}^{\infty} (-1)^n u^{\binom{n}{2}} v^{\binom{-n}{2}}, \qquad (19)$$

поскольку левая часть при этом становится равной $\prod_{k=1}^{\infty}(1 - z^{3k-2})(1 - z^{3k-1})(1 - z^{3k})$; см. упр. 5.1.1–20. Из тождества Эйлера (18) вытекает, что количество разбиений для $n > 0$ удовлетворяет рекуррентному соотношению

$$p(n) = p(n-1) + p(n-2) - p(n-5) - p(n-7) + p(n-12) + p(n-15) - \cdots, \qquad (20)$$

где $p(k) = 0$ при $k < 0$; это рекуррентное соотношение позволяет вычислить значения быстрее, чем выполняя вычисление степенных рядов (17):

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$p(n) =$	1	1	2	3	5	7	11	15	22	30	42	56	77	101	135	176

Из раздела 1.2.8 мы знаем, что решение рекуррентного соотношения Фибоначчи $f(n) = f(n-1) + f(n-2)$ при положительных $f(0)$ и $f(1)$ растет экспоненциально: $f(n) = \Theta(\phi^n)$. Дополнительные члены '$- p(n-5) - p(n-7)$' в (20) оказывают подавляющее влияние на количество разбиений; более того, если ограничиться в рекуррентном соотношении только этими членами, полученный в результате ряд осциллирует между положительными и отрицательными значениями. Следующие члены, '$+ p(n-12) + p(n-15)$', восстанавливают экспоненциальный рост.

Реальная скорость роста $p(n)$ имеет порядок $A^{\sqrt{n}}/n$ для некоторой константы A. Например, в упр. 33 доказывается, что $p(n)$ растет как минимум со скоростью $e^{2\sqrt{n}}/n$. Еще один простой способ получить неплохую *верхнюю* границу состоит в логарифмировании (17):

$$\ln P(z) \;=\; \sum_{m=1}^{\infty} \ln \frac{1}{1 - z^m} \;=\; \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{z^{mn}}{n}; \qquad (21)$$

затем следует рассмотреть поведение вблизи $z = 1$, положив $z = e^{-t}$ при $t > 0$:

$$\ln P(e^{-t}) = \sum_{m,n\ge1} \frac{e^{-mnt}}{n} = \sum_{n\ge1} \frac{1}{n}\,\frac{1}{e^{tn} - 1} < \sum_{n\ge1} \frac{1}{n^2 t} = \frac{\zeta(2)}{t}. \qquad (22)$$

Далее, поскольку $p(n) \le p(n+1) < p(n+2) < \cdots$ и $e^t > 1$, получаем

$$\frac{p(n)}{1 - e^{-t}} = \sum_{k=n}^{\infty} p(n) e^{(n-k)t} < \sum_{k=0}^{\infty} p(k) e^{(n-k)t} = e^{nt} P(e^{-t}) < e^{nt+\zeta(2)/t} \qquad (23)$$

для всех $t > 0$. Приравнивание $t = \sqrt{\zeta(2)/n}$ дает

$$p(n) \;<\; Ce^{2C\sqrt{n}}/\sqrt{n}, \qquad \text{где } C = \sqrt{\zeta(2)} = \pi/\sqrt{6}. \tag{24}$$

Более точную информацию о величине $\ln P(e^{-t})$ можно получить, воспользовавшись формулой суммирования Эйлера (раздел 1.2.11.2) или преобразованием Меллина (раздел 5.2.2); см. упр. 25. Однако рассмотренные методы недостаточно мощны для определения точного поведения $P(e^{-t})$, так что самое время добавить в наш арсенал новые методы.

Производящая функция Эйлера $P(z)$ идеально подходит для *формулы суммирования Пуассона* [*J. École Royale Polytechnique* **12** (1823), 404–509, §63], согласно которой

$$\sum_{n=-\infty}^{\infty} f(n+\theta) \;=\; \lim_{M\to\infty} \sum_{m=-M}^{M} e^{2\pi m i\theta} \int_{-\infty}^{\infty} e^{-2\pi m i y} f(y)\,dy, \tag{25}$$

где f — "хорошо себя ведущая" функция. Эта формула основана на том факте, что в левой части находится периодическая функция от θ, а в правой — ее разложение в ряд Фурье. Функция f достаточно хорошая, если, например, $\int_{-\infty}^{\infty} \bigl|f(y)\bigr|\,dy < \infty$ и

i) либо $f(n+\theta)$ является аналитической функцией комплексной переменной θ в области $|\Im\theta| \le \epsilon$ для некоторого $\epsilon > 0$ и $0 \le \Re\theta \le 1$ и любого n, и левая часть (25) равномерно сходится в $|\Im\theta| \le \epsilon$;

ii) либо $f(\theta) = \frac{1}{2}\lim_{\epsilon\to 0}\bigl(f(\theta-\epsilon) + f(\theta+\epsilon)\bigr) = g(\theta) - h(\theta)$ для всех действительных чисел θ, где g и h — монотонно возрастающие, а $g(\pm\infty)$, $h(\pm\infty)$ конечны.

[См. Peter Henrici, *Applied and Computational Complex Analysis* **2** (New York: Wiley, 1977), Theorem 10.6e.] Формула Пуассона — не панацея для задач суммирования любого вида, но если она применима, то ее результаты, как мы увидим, могут быть весьма впечатляющими.

Умножим формулу Эйлера (18) на $z^{1/24}$ для "завершения квадрата":

$$\frac{z^{1/24}}{P(z)} \;=\; \sum_{n=-\infty}^{\infty} (-1)^n\, z^{\frac{3}{2}(n+\frac{1}{6})^2}. \tag{26}$$

Тогда для всех $t > 0$ имеем $e^{-t/24}/P(e^{-t}) = \sum_{n=-\infty}^{\infty} f(n)$, где

$$f(y) \;=\; e^{-\frac{3}{2}t(y+\frac{1}{6})^2 + \pi i y}; \tag{27}$$

и эта функция f подходит для применения формулы суммирования по обоим приведенным выше критериям (i) и (ii). Таким образом, можно попытаться проинтегрировать $e^{-2\pi m i y} f(y)$, и этот интеграл оказывается очень простым для всех m (см. упр. 27):

$$\int_{-\infty}^{\infty} e^{-a(y+b)^2 + 2ciy}\,dy \;=\; \sqrt{\frac{\pi}{a}}\, e^{-c^2/a - 2bci} \qquad \text{при } a > 0. \tag{28}$$

Подстановка в (25) при $\theta = 0$, $a = \frac{3}{2}t$, $b = \frac{1}{6}$ и $c = (\frac{1}{2} - m)\pi$ дает

$$\sum_{n=-\infty}^{\infty} f(n) = \sum_{m=-\infty}^{\infty} g(m), \qquad g(m) = \sqrt{\frac{2\pi}{3t}}\, e^{-2(m-\frac{1}{2})^2\pi^2/(3t) + \frac{1-2m}{6}\pi i}. \tag{29}$$

Эти члены объединяются и сокращаются, как показано в упр. 27, давая в результате

$$\frac{e^{-t/24}}{P(e^{-t})} = \sqrt{\frac{2\pi}{t}} \sum_{n=-\infty}^{\infty} (-1)^n e^{-6\pi^2(n+\frac{1}{6})^2/t} = \sqrt{\frac{2\pi}{t}} \frac{e^{-\zeta(2)/t}}{P(e^{-4\pi^2/t})}. \tag{30}$$

Небольшой сюрприз: мы только что доказали еще один замечательный факт о $P(z)$ — приведенную далее теорему.

Теорема D. *Производящая функция* (17) *для разбиений удовлетворяет функциональному соотношению*

$$\ln P(e^{-t}) = \frac{\zeta(2)}{t} + \frac{1}{2}\ln\frac{t}{2\pi} - \frac{t}{24} + \ln P(e^{-4\pi^2/t}) \tag{31}$$

при $\Re t > 0$. ▮

Эта теорема была открыта Рихардом Дедекиндом (Richard Dedekind) [*Crelle* **83** (1877), 265–292, §6], который для функции $z^{1/24}/P(z)$ использовал запись $\eta(\tau)$, где $z = e^{2\pi i\tau}$; его доказательство основано на существенно более сложной теории эллиптических функций. Заметим, что, когда t — малое положительное число, значение $\ln P(e^{-4\pi^2/t})$ *крайне* мало́; например, при $t = 0.1$ получаем $\exp(-4\pi^2/t) \approx 3.5 \times 10^{-172}$. Таким образом, теорема D дает нам практически все, что необходимо знать о значении $P(z)$ при z, близком к 1.

Г. Г. Харди (G. H. Hardy) и С. Рамануджан (S. Ramanujan) использовали это знание для вывода асимптотического поведения $p(n)$ при больши́х n, и их работа была расширена много лет спустя Гансом Радемахером (Hans Rademacher), открывшим ряд, который не только асимптотичен, но и сходится [*Proc. London Math. Soc.* (2) **17** (1918), 75–115; **43** (1937), 241–254]. Формула Харди–Рамануджана–Радемахера для $p(n)$ — поистине одно из наиболее изумительных тождеств; оно гласит, что

$$p(n) = \frac{\pi}{2^{5/4}3^{3/4}(n-1/24)^{3/4}} \sum_{k=1}^{\infty} \frac{A_k(n)}{k} I_{3/2}\left(\sqrt{\frac{2}{3}}\frac{\pi}{k}\sqrt{n-1/24}\right). \tag{32}$$

Здесь $I_{3/2}$ означает модифицированную сферическую функцию Бесселя

$$I_{3/2}(z) = \left(\frac{z}{2}\right)^{3/2} \sum_{k=0}^{\infty} \frac{1}{\Gamma(k+5/2)} \frac{(z^2/4)^k}{k!} = \sqrt{\frac{2z}{\pi}}\left(\frac{\cosh z}{z} - \frac{\sinh z}{z^2}\right); \tag{33}$$

а коэффициент $A_k(n)$ определяется формулой

$$A_k(n) = \sum_{h=0}^{k-1} [h \perp k] \exp\left(2\pi i\left(\frac{\sigma(h,k,0)}{24} - \frac{nh}{k}\right)\right), \tag{34}$$

где $\sigma(h,k,0)$ — сумма Дедекинда, определенная в уравнении 3.3.3–(16). В результате

$$A_1(n) = 1, \qquad A_2(n) = (-1)^n, \qquad A_3(n) = 2\cos\frac{(24n+1)\pi}{18}, \tag{35}$$

и в общем случае $A_k(n)$ лежит между $-k$ и k.

Доказательство (32) может увести нас слишком далеко, но основная идея заключается в использовании “метода седловой точки”, рассматриваемого в разделе 7.2.1.5. Член для $k = 1$ выводится из поведения $P(z)$ при z, близком к 1; следующий

член выводится из поведения при z, близком к -1, когда можно применить преобразование, аналогичное (31). В общем случае k-й член (32) учитывает поведение $P(z)$ при z, стремящемся к $e^{2\pi ih/k}$, для несократимых дробей h/k со знаменателем k; каждый k-й корень единицы представляет собой полюс для каждой из дробей $1/(1-z^k)$, $1/(1-z^{2k})$, $1/(1-z^{3k})$, ... в бесконечном произведении $P(z)$.

Старший член (32) может быть существенно упрощен, если нам достаточно грубого приближения:

$$p(n) \;=\; \frac{e^{\pi\sqrt{2n/3}}}{4n\sqrt{3}}\bigl(1+O(n^{-1/2})\bigr). \tag{36}$$

Или, если такой точности недостаточно, можно воспользоваться более детальным приближением

$$p(n) \;=\; \frac{e^{\pi\sqrt{2n'/3}}}{4n'\sqrt{3}}\left(1-\frac{1}{\pi}\sqrt{\frac{3}{2n'}}\right)\Bigl(1+O\bigl(e^{-\pi\sqrt{n/6}}\bigr)\Bigr), \quad n'=n-\frac{1}{24}. \tag{37}$$

Например, $p(100)$ имеет точное значение 190 569 292; формула (36) дает нам $p(100)\approx 1.993\times 10^8$, в то время как (37) дает гораздо лучшую оценку: 190 568 944.783.

Э. Одлызко (A. Odlyzko) заметил, что при больши́х n формула Харди–Рамануджана–Радемахера в действительности дает близкий к оптимальному способ вычисления точного значения $p(n)$, поскольку арифметические операции могут быть выполнены примерно за $O\bigl(\log p(n)\bigr)=O(n^{1/2})$ шагов. [См. *Handbook of Combinatorics* **2** (MIT Press, 1995), 1068–1069.] Основной вклад дают несколько первых членов (32); затем ряд стремится к членам, не превышающим порядка $k^{-3/2}$ и обычно имеющим порядок k^{-2}. Кроме того, около половины коэффициентов $A_k(n)$ оказываются нулевыми (см. упр. 28). Например, при $n=10^6$ члены для $k=1$, 2 и 3 приблизительно равны $\approx 1.47\times 10^{1107}$, 1.23×10^{550} и -1.23×10^{364} соответственно. Сумма первых 250 членов приближенно равна $\approx 1471684986\ldots 73818.01$, в то время как истинное значение представляет собой $1471684986\ldots 73818$; при этом 123 из 250 членов равны нулю.

Количество частей. Удобно ввести обозначение

$$\left|{n \atop m}\right| \tag{38}$$

для числа разбиений n, состоящих ровно из m частей. Тогда для всех целых m и n выполняется рекуррентное соотношение

$$\left|{n \atop m}\right| \;=\; \left|{n-1 \atop m-1}\right| + \left|{n-m \atop m}\right|, \tag{39}$$

поскольку $\left|{n-1 \atop m-1}\right|$ подсчитывает разбиения, наименьшая часть которых равна 1, а $\left|{n-m \atop m}\right|$ — прочие разбиения. (Если наименьшая часть равна 2 или больше, можно вычесть из каждой части по 1 и получить разбиение $n-m$ на m частей.) Аналогичные рассуждения позволяют заключить, что $\left|{m+n \atop m}\right|$ — количество разбиений n на *не более* m частей, а именно на m неотрицательных слагаемых. Мы также знаем из транспонирования диаграмм Феррерса, что $\left|{n \atop m}\right|$ представляет собой количество

Таблица 2

Количества разбиений

n	$\left\vert {n \atop 0} \right\vert$	$\left\vert {n \atop 1} \right\vert$	$\left\vert {n \atop 2} \right\vert$	$\left\vert {n \atop 3} \right\vert$	$\left\vert {n \atop 4} \right\vert$	$\left\vert {n \atop 5} \right\vert$	$\left\vert {n \atop 6} \right\vert$	$\left\vert {n \atop 7} \right\vert$	$\left\vert {n \atop 8} \right\vert$	$\left\vert {n \atop 9} \right\vert$	$\left\vert {n \atop 10} \right\vert$	$\left\vert {n \atop 11} \right\vert$
0	1	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0	0	0
3	0	1	1	1	0	0	0	0	0	0	0	0
4	0	1	2	1	1	0	0	0	0	0	0	0
5	0	1	2	2	1	1	0	0	0	0	0	0
6	0	1	3	3	2	1	1	0	0	0	0	0
7	0	1	3	4	3	2	1	1	0	0	0	0
8	0	1	4	5	5	3	2	1	1	0	0	0
9	0	1	4	7	6	5	3	2	1	1	0	0
10	0	1	5	8	9	7	5	3	2	1	1	0
11	0	1	5	10	11	10	7	5	3	2	1	1

разбиений n, у которых *наибольшая* часть равна m. Таким образом, $\left\vert {n \atop m} \right\vert$ — число, которое сто́ит того, чтобы его изучить. Граничные условия

$$\left\vert {n \atop 0} \right\vert = \delta_{n0} \qquad \text{и} \qquad \left\vert {n \atop m} \right\vert = 0 \quad \text{для } m < 0 \text{ или } n < 0 \tag{40}$$

делают простой табуляцию $\left\vert {n \atop m} \right\vert$ для небольших значений параметров, и мы можем получить массив чисел аналогично знакомым треугольникам для $\binom{n}{m}$, $\left[{n \atop m}\right]$, $\left\{{n \atop m}\right\}$ и $\left\langle {n \atop m} \right\rangle$, с которыми мы встречались ранее (табл. 2). Производящая функция имеет вид

$$\sum_n \left\vert {n \atop m} \right\vert z^n \;=\; \frac{z^m}{(1-z)(1-z^2)\ldots(1-z^m)}. \tag{41}$$

Почти все разбиения n состоят из $\Theta(\sqrt{n}\log n)$ частей. Этот факт, открытый П. Эрдешем (P. Erdös) и Д. Ленером (J. Lehner) [*Duke Math. J.* **8** (1941), 335–345], имеет весьма поучительное доказательство.

Теорема E. *Пусть* $C = \pi/\sqrt{6}$ *и* $m = \frac{1}{2C}\sqrt{n}\ln n + x\sqrt{n} + O(1)$. *Тогда*

$$\frac{1}{p(n)}\left\vert {m+n \atop m} \right\vert \;=\; F(x)\bigl(1 + O(n^{-1/2+\epsilon})\bigr) \tag{42}$$

для всех $\epsilon > 0$ *и всех фиксированных* x *при* $n \to \infty$, *где*

$$F(x) \;=\; e^{-e^{-Cx}/C}. \tag{43}$$

Функция $F(x)$ в (43) достаточно быстро стремится к 0 при $x \to -\infty$ и быстро возрастает до 1 при $x \to +\infty$; таким образом, это функция распределения вероятностей. На рис. 49, (б) показана соответствующая функция плотности распределения вероятностей $f(x) = F'(x)$, в основном сосредоточенная в области $-2 \le x \le 4$. (См. упр. 35.)

Для сравнения на рис. 49, (а) показаны значения $\left\vert {n \atop m} \right\vert = \left\vert {m+n \atop m} \right\vert - \left\vert {m-1+n \atop m-1} \right\vert$ для $n = 100$; в этом случае $\frac{1}{2C}\sqrt{n}\ln n \approx 18$.

Рис. 49. Разбиения n на m частей для (а) $n = 100$; (б) $n \to \infty$. (См. теорему E.)

Доказательство. Воспользуемся тем фактом, что $\left|{m+n \atop m}\right|$ — количество разбиений n, у которых бóльшая часть $\le m$. Тогда, согласно принципу включения и исключения (формула 1.3.3–(29)), имеем

$$\left|{m+n \atop m}\right| = p(n) - \sum_{j>m} p(n-j) + \sum_{j_2>j_1>m} p(n-j_1-j_2) - \sum_{j_3>j_2>j_1>m} p(n-j_1-j_2-j_3) + \cdots,$$

поскольку $p(n-j_1-\cdots-j_r)$ — количество разбиений n, которые используют каждую из частей $\{j_1,\ldots,j_r\}$ хотя бы один раз. Запишем это как

$$\frac{1}{p(n)}\left|{m+n \atop m}\right| = 1 - \Sigma_1 + \Sigma_2 - \Sigma_3 + \cdots, \quad \Sigma_r = \sum_{j_r>\cdots>j_1>m} \frac{p(n-j_1-\cdots-j_r)}{p(n)}. \tag{44}$$

Чтобы вычислить Σ_r, нам нужна хорошая оценка отношения $p(n-t)/p(n)$. Нам везет, поскольку из (36) вытекает, что

$$\begin{aligned}\frac{p(n-t)}{p(n)} &= \exp\bigl(2C\sqrt{n-t} - \ln(n-t) + O\bigl((n-t)^{-1/2}\bigr) - 2C\sqrt{n} + \ln n\bigr)\\ &= \exp\bigl(-Ctn^{-1/2} + O(n^{-1/2+2\epsilon})\bigr) \qquad \text{при } 0 \le t \le n^{1/2+\epsilon}.\end{aligned} \tag{45}$$

Далее, если $t \ge n^{1/2+\epsilon}$, мы имеем $p(n-t)/p(n) \le p(n-n^{1/2+\epsilon})/p(n) \approx \exp(-Cn^{\epsilon})$, значение, которое асимптотически меньше, чем любая степень n. Следовательно, можно безопасно использовать приближение

$$\frac{p(n-t)}{p(n)} \approx \alpha^t, \qquad \alpha = \exp(-Cn^{-1/2}), \tag{46}$$

для всех значений $t \ge 0$. Например, мы имеем

$$\begin{aligned}\Sigma_1 = \sum_{j>m} \frac{p(n-j)}{p(n)} &= \frac{\alpha^{m+1}}{1-\alpha}\bigl(1 + O(n^{-1/2+2\epsilon})\bigr) + \sum_{n\ge j>n^{1/2+\epsilon}} \frac{p(n-j)}{p(n)}\\ &= \frac{e^{-Cx}}{C}\bigl(1 + O(n^{-1/2+2\epsilon})\bigr) + O(ne^{-Cn^{\epsilon}}),\end{aligned}$$

Рис. 50. Кривая Темперли (49) для ограничивающей формы случайного разбиения.

поскольку $\alpha/(1-\alpha) = n^{1/2}/C + O(1)$ и $\alpha^m = n^{-1/2}e^{-Cx} + O(n^{-1})$. Аналогичные доводы (см. упр. 36) доказывают, что при $r = O(\log n)$

$$\Sigma_r \;=\; \frac{e^{-Crx}}{C^r r!}\bigl(1 + O(n^{-1/2+2\epsilon})\bigr) + O(e^{-n^{\epsilon/2}}). \tag{47}$$

Наконец — и это замечательное свойство принципа включения и исключения — частичные суммы (44) всегда "берут в вилку" истинное значение, в том смысле, что

$$1 - \Sigma_1 + \Sigma_2 - \cdots - \Sigma_{2r-1} \le \frac{1}{p(n)}\left|\begin{matrix} m+n \\ m \end{matrix}\right| \le 1 - \Sigma_1 + \Sigma_2 - \cdots - \Sigma_{2r-1} + \Sigma_{2r} \tag{48}$$

для всех r. (См. упр. 37.) Когда $2r$ близко к $\ln n$ и n велико, член Σ_{2r} чрезвычайно мал; таким образом, мы получаем (42), за исключением того, что вместо ϵ используется 2ϵ. ▮

Теорема E говорит о том, что наибольшая часть случайного разбиения почти всегда представляет собой $\frac{1}{2C}\sqrt{n}\ln n + O(\sqrt{n}\log\log\log n)$, и при достаточно большом n остальные части также имеют тенденцию к предсказуемости. Предположим, например, что мы берем все разбиения числа 25 и накладываем одну на другую их диаграммы Феррерса, заменяя точки квадратами, как в случае краевого представления. Какие ячейки будут заниматься особенно часто? На рис. 50 показан результат: случайное разбиение стремится к типичному виду, приближающемуся к предельной кривой при $n \to \infty$.

Г. Н. В. Темперли (H. N. V. Temperley) [*Proc. Cambridge Philos. Soc.* **48** (1952), 683–697] привел эвристические обоснования того, что основные части a_k большого случайного разбиения $a_1 \ldots a_m$ будут удовлетворять приближенному закону

$$e^{-Ck/\sqrt{n}} + e^{-Ca_k/\sqrt{n}} \;\approx\; 1, \tag{49}$$

и его формула была впоследствии подтверждена в строгом виде. Например, теорема Б. Питтеля [*Advances in Applied Math.* **18** (1997), 432–488] позволяет заключить, что след случайного разбиения почти всегда равен $\frac{\ln 2}{C}\sqrt{n} \approx 0.54\sqrt{n}$, что согласуется с (49) с ошибкой не более $O(\sqrt{n}\ln n)^{1/2}$; таким образом, около 29% всех точек Феррерса лежат в пределах квадрата Дюрфи.

С другой стороны, если рассмотреть только разбиения n на m частей, где m — фиксированная величина, ограничивающая кривая будет иной: почти все разбиения имеют

$$a_k \approx \frac{n}{m} \ln \frac{m}{k} \tag{50}$$

при достаточно больших m и n. На рис. 51 показан случай $n = 50$, $m = 5$. В действительности тот же предел справедлив при m, растущем с ростом n, но со скоростью, меньшей $\sqrt{n}$ [см. А. Вершик и Ю. Якубович, *Moscow Math. J.* **1** (2001), 457–468].

Рис. 51. Ограничивающая форма (50) для m частей.

Краевое представление дает нам дальнейшую информацию о *дважды* ограниченных разбиениях, когда ограничено не только количество частей, но и размер каждой части. Разбиение, имеющее не более m частей, каждая из которых не превышает l, находится внутри прямоугольника $m \times l$. Все такие разбиения соответствуют перестановкам мультимножества $\{m \cdot 0, l \cdot 1\}$, которые имеют ровно n инверсий (инверсии мультимножеств мы уже изучали в упр. 5.1.2–16). В частности, в упомянутом упражнении выводится неочевидная формула для количества способов получения n инверсий.

Теорема C. *Количество разбиений n, которые состоят не более чем из m частей, не превышающих по размеру l, составляет*

$$[z^n] \binom{l+m}{m}_z = [z^n] \frac{(1-z^{l+1})}{(1-z)} \frac{(1-z^{l+2})}{(1-z^2)} \cdots \frac{(1-z^{l+m})}{(1-z^m)}. \tag{51}$$

Этот результат получен А. Коши (A. Cauchy), *Comptes Rendus Acad. Sci.* **17** (Paris, 1843), 523–531. Обратите внимание, что при $l \to \infty$ числитель становится равным просто 1. Интересное комбинаторное доказательство более общего результата имеется ниже, в упр. 40. ▮

Анализ алгоритмов. Теперь мы более чем достаточно знаем о количественных аспектах разбиений, чтобы с высокой степенью точности проанализировать поведение алгоритма P. Предположим, что шаги P1, ..., P6 этого алгоритма выполняются соответственно $T_1(n)$, ..., $T_6(n)$ раз. Очевидно, что $T_1(n) = 1$ и $T_3(n) = p(n)$; кроме того, закон Кирхгофа говорит нам, что $T_2(n) = T_5(n)$ и $T_4(n) + T_5(n) = T_3(n)$. Мы переходим к шагу P4 по одному разу для каждого разбиения, содержащего 2; ясно, что это происходит $p(n-2)$ раз.

Таким образом, единственная возможная загадка алгоритма P — это количество выполнений шага P6, в котором возможен переход к самому себе. Однако небольшое размышление открывает нам, что алгоритм сохраняет значение ≥ 2 в массиве $a_1 a_2 \ldots$ только на шаге P2 или перед проверкой $n \leq x$ на шаге P6; и каждое такое значение в конечном итоге уменьшается на 1 либо на шаге P4, либо на шаге P5. Следовательно,

$$T_2''(n) + T_6(n) = p(n) - 1, \tag{52}$$

где $T_2''(n)$ — количество раз, когда шаг P2 устанавливает a_m равным значению ≥ 2. Пусть $T_2(n) = T_2'(n) + T_2''(n)$, так что $T_2'(n)$ — количество раз, когда шаг P2 устанавливает $a_m \leftarrow 1$. Тогда $T_2'(n) + T_4(n)$ — количество разбиений, заканчивающихся на 1, следовательно,

$$T_2'(n) \;+\; T_4(n) = p(n-1). \tag{53}$$

Итак, мы получили достаточное количество уравнений, чтобы определить все искомые величины:

$$\bigl(T_1(n), \ldots, T_6(n)\bigr) \;= \\ \bigl(1,\; p(n)-p(n{-}2),\; p(n),\; p(n{-}2),\; p(n)-p(n{-}2),\; p(n{-}1)-1\bigr). \tag{54}$$

Из асимптотики $p(n)$ мы также знаем среднее количество вычислений на одно разбиение:

$$\Bigl(\frac{T_1(n)}{p(n)}, \ldots, \frac{T_6(n)}{p(n)}\Bigr) \;=\; \Bigl(0,\; \frac{2C}{\sqrt{n}},\; 1,\; 1-\frac{2C}{\sqrt{n}},\; \frac{2C}{\sqrt{n}},\; 1-\frac{C}{\sqrt{n}}\Bigr) \;+\; O\Bigl(\frac{1}{n}\Bigr), \tag{55}$$

где $C = \pi/\sqrt{6} \approx 1.283$. (См. упражнение 45.) Общее количество обращений к памяти на одно разбиение составляет, таким образом, $3 + C/\sqrt{n} + O(1/n)$.

Если кто-то захочет сгенерировать все разбиения, он должен не только выполнить непомерную работу, но и быть безмерно внимательным, чтобы не ошибиться.

— ЛЕОНАРД ЭЙЛЕР (LEONHARD EULER),
Разбиение чисел (*De Partitione Numerorum*) (1750)

Алгоритм H более сложен для анализа, но можно доказать как минимум верхнюю границу времени его работы. Ключевой величиной является значение j, наименьшего индекса, для которого $a_j < a_1 - 1$. Последовательные значения j для $m = 4$ и $n = 11$ равны $(2, 2, 2, 3, 2, 2, 3, 4, 2, 3, 5)$, и мы замечаем, что $j = b_{l-1} + 1$, где $b_1 \ldots b_l$ — сопряженное разбиение $(a_1 \ldots a_m)^T$ и $m < n$. (См. (7) и (12).) На шаге H3 выделяется случай $j = 2$, поскольку он не только наиболее часто встречается, но и очень легко обрабатывается.

Пусть $c_m(n)$ — накопленное общее значение $j-1$, просуммированное по всем $\left|{n \atop m}\right|$ разбиениям, сгенерированным алгоритмом H. Например, $c_4(11) = 1+1+1+2+1+1+2+3+1+2+4 = 19$. Можно рассматривать $c_m(n)/\left|{n \atop m}\right|$ как хороший показатель времени работы на одно разбиение, поскольку время выполнения наиболее затратных шагов, H4 и H6, грубо пропорционально $j-2$. Это отношение $c_m(n)/\left|{n \atop m}\right|$ *не* ограничено, поскольку $c_m(m) = m$ при $\left|{m \atop m}\right| = 1$. Однако следующая теорема показывает, что, тем не менее, алгоритм H вполне эффективен.

Теорема H. *Мера стоимости $c_m(n)$ алгоритма H не превышает $3\left|{n \atop m}\right| + m$.*

Доказательство. Можно легко проверить, что $c_m(n)$ удовлетворяет тому же рекуррентному соотношению, что и $\left|{n \atop m}\right|$, а именно

$$c_m(n) \;=\; c_{m-1}(n-1) \;+\; c_m(n-m) \qquad \text{для } m, n \geq 1, \tag{56}$$

если искусственно определить $c_m(n) = 1$ при $1 \leq n < m$; см. (39). Однако граничные условия теперь иные:

$$c_m(0) = [m > 0]; \qquad\qquad c_0(n) = 0. \tag{57}$$

Таблица 3

Стоимости алгоритма H

n	$c_0(n)$	$c_1(n)$	$c_2(n)$	$c_3(n)$	$c_4(n)$	$c_5(n)$	$c_6(n)$	$c_7(n)$	$c_8(n)$	$c_9(n)$	$c_{10}(n)$	$c_{11}(n)$
0	0	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1
2	0	1	2	1	1	1	1	1	1	1	1	1
3	0	1	2	3	1	1	1	1	1	1	1	1
4	0	1	3	3	4	1	1	1	1	1	1	1
5	0	1	3	4	4	5	1	1	1	1	1	1
6	0	1	4	6	5	5	6	1	1	1	1	1
7	0	1	4	7	7	6	6	7	1	1	1	1
8	0	1	5	8	11	8	7	7	8	1	1	1
9	0	1	5	11	12	12	9	8	8	9	1	1
10	0	1	6	12	16	17	13	10	9	9	10	1
11	0	1	6	14	19	21	18	14	11	10	10	11

В табл. 3 показано поведение $c_m(n)$ при малых m и n.

Для доказательства теоремы докажем на самом деле более строгий результат:

$$c_m(n) \;\le\; 3\left|{n \atop m}\right| + 2m - n - 1 \qquad \text{для } n \ge m \ge 2. \tag{58}$$

В упр. 50 показано, что это неравенство выполняется при $m \le n \le 2m$, так что доказательство будет завершено, если доказать его для $n > 2m$. В этом случае по индукции имеем

$$\begin{aligned} c_m(n) &= c_1(n-m) + c_2(n-m) + c_3(n-m) + \cdots + c_m(n-m) \\ &\le 1 + \left(3\left|{n-m \atop 2}\right| + 3 - n + m\right) + \left(3\left|{n-m \atop 3}\right| + 5 - n + m\right) + \cdots \\ &\qquad\qquad + \left(3\left|{n-m \atop m}\right| + 2m - 1 - n + m\right) \\ &= 3\left|{n-m \atop 1}\right| + 3\left|{n-m \atop 2}\right| + \cdots + 3\left|{n-m \atop m}\right| - 3 + m^2 - (m-1)(n-m) \\ &= 3\left|{n \atop m}\right| + 2m^2 - m - (m-1)n - 3; \end{aligned}$$

а $2m^2 - m - (m-1)n - 3 \le 2m - n - 1$, так как $n \ge 2m + 1$. ▮

***Код Грея для разбиений.** При генерации разбиений в виде количества частей $c_1 \ldots c_n$ (как в упр. 5) на каждом шаге изменяется не более четырех значений c_j. Однако можно предпочесть минимизировать изменения отдельных частей, генерируя разбиения таким образом, чтобы следующее за $a_1 a_2 \ldots$ разбиение получалось простой установкой $a_j \leftarrow a_j + 1$ и $a_k \leftarrow a_k - 1$ для некоторых j и k, как в алгоритме двери-вертушки из раздела 7.2.1.3. Это оказывается всегда возможным; для $n = 6$ имеется единственный способ такой генерации:

$$111111,\ 21111,\ 3111,\ 2211,\ 222,\ 321,\ 33,\ 42,\ 411,\ 51,\ 6. \tag{59}$$

В общем случае $\left|{m+n \atop m}\right|$ разбиений n на не более m частей всегда могут быть сгенерированы подходящим путем Грея.

Заметим, что $\alpha \to \beta$ представляет собой допустимый переход от одного разбиения к другому тогда и только тогда, когда мы получаем диаграмму Ферररса

для β путем перемещения только одной точки диаграммы Ферररса для α. Таким образом, $\alpha^T \to \beta^T$ также является допустимым переходом. Отсюда следует, что каждый код Грея для разбиения не более чем на m частей соответствует коду Грея для разбиений на части, не превосходящие m. Мы будем работать именно с этим последним ограничением.

Общее количество кодов Грея для разбиений очень велико: для $n = 7$ их 52; для $n = 8$ — 652; для $n = 9$ — 298 896; а для $n = 10$ — 2 291 100 484. Однако реально простое их построение неизвестно. Вероятно, причина в том, что некоторые разбиения имеют только двух соседей, а именно разбиения $d^{n/d}$ при $1 < d < n$ и n, кратном d. Для таких разбиений предшествующими и последующими должны быть разбиения $\{(d{+}1)d^{n/d-2}(d{-}1),\ d^{n/d-1}(d{-}1)1\}$, и это требование, похоже, исключает любой простой рекурсивный подход.

Карла Д. Сэведж (Carla D. Savage) [*J. Algorithms* **10** (1989), 577–595] нашла способ преодоления этих трудностей ценой всего лишь небольшой сложности. Пусть

$$\mu(m,n) \;=\; \overbrace{m\ \ m\ \ \ldots\ \ m}^{\lfloor n/m \rfloor}\ (n \bmod m) \tag{60}$$

представляет собой лексикографически наибольшее разбиение n с частями $\le m$. Наша цель состоит в построении рекурсивно определенных путей Грея $L(m,n)$ и $M(m,n)$ от разбиения 1^n до $\mu(m,n)$, где $L(m,n)$ проходит по всем разбиениям, части которых ограничены m, в то время как $M(m,n)$ не только проходит по всем этим разбиениям, но и включает разбиения, наибольшая часть которых равна $m{+}1$, при условии, что все прочие части строго меньше m. Например, $L(3,8)$ — 11111111, 2111111, 311111, 221111, 22211, 2222, 3221, 32111, 3311, 332, в то время как $M(3,8)$ —

$$\begin{gathered}11111111,\ 2111111,\ 221111,\ 22211,\ 2222,\ 3221,\\ 3311,\ 32111,\ 311111,\ 41111,\ 4211,\ 422,\ 332;\end{gathered} \tag{61}$$

дополнительные разбиения, начинающиеся с 4, дадут нам “пространство подгонки” в других частях рекурсии. Мы определим $L(m,n)$ для всех $n \ge 0$, а $M(m,n)$ — только для $n > 2m$.

Следующее построение, проиллюстрированное для простоты при $m = 5$, *почти* работоспособно:

$$L(5) = \begin{Bmatrix} L(3) \\ 4L(\infty)^R \\ 5L(\infty) \end{Bmatrix} \text{ при } n \le 7; \quad \begin{Bmatrix} L(3) \\ 4L(2)^R \\ 5L(2) \\ 4\,3\,1 \\ 4\,4 \\ 5\,3 \end{Bmatrix} \text{ при } n = 8; \quad \begin{Bmatrix} M(4) \\ 5\,4L(4)^R \\ 5\,5L(5) \end{Bmatrix} \text{ при } n \ge 9; \tag{62}$$

$$M(5) = \begin{Bmatrix} L(4) \\ 5L(4)^R \\ 6L(3) \\ 6\,4L(\infty)^R \\ 5\,5L(\infty) \end{Bmatrix} \text{ при } 11 \le n \le 13; \qquad \begin{Bmatrix} L(4) \\ 5M(4)^R \\ 6L(4) \\ 5\,5\,4L(4)^R \\ 5\,5\,5L(5) \end{Bmatrix} \text{ при } n \ge 14. \tag{63}$$

Здесь параметр n в $L(m,n)$ и $M(m,n)$ опущен, поскольку он может быть выведен из контекста; предполагается, что каждые L и M генерируют разбиения для любого

количества, остающегося после вычитания предыдущих частей. Так, например, (63) указывает, что

$$M(5,14) \;=\; L(4,14),\; 5M(4,9)^R,\; 6L(4,8),\; 554L(4,0)^R,\; 555L(5,-1);$$

последовательность $L(5,-1)$ в действительности пустая, а $L(4,0)$ представляет собой пустую строку, так что последнее разбиение $M(5,14)$ представляет собой $554 = \mu(5,14)$, как и должно быть. Запись $L(\infty)$ означает $L(\infty,n) = L(n,n)$, путь Грея для всех разбиений n, начинающихся с 1^n и заканчивающихся n^1.

В общем случае $L(m)$ и $M(m)$ определяются для всех $m \ge 3$, по сути, одними и теми же правилами, если цифры 2, 3, 4, 5 и 6 в (62) и (63) заменить соответственно на $m-3$, $m-2$, $m-1$, m и $m+1$. Диапазоны $n \le 7$, $n = 8$, $n \ge 9$ превращаются в $n \le 2m-3$, $n = 2m-2$, $n \ge 2m-1$; диапазоны $11 \le n \le 13$ и $n \ge 14$ становятся диапазонами $2m+1 \le n \le 3m-2$ и $n \ge 3m-1$. Последовательности $L(0)$, $L(1)$, $L(2)$ имеют очевидные определения, поскольку эти пути единственны при $m \le 2$. Последовательность $M(2)$ представляет собой 1^n, 21^{n-2}, 31^{n-3}, 221^{n-4}, 2221^{n-6}, $\ldots$, $\mu(2,n)$ для $n \ge 5$.

Теорема S. *Пути Грея $L'(m,n)$ для $m,n \ge 0$ и $M'(m,n)$ для $n \ge 2m+1 \ge 5$ существуют для всех разбиений с описанными выше свойствами, за исключением случая $L'(4,6)$. Кроме того, L' и M' удовлетворяют взаимным рекурсиям* (62) *и* (63), *за исключением нескольких случаев.*

Доказательство. Выше мы отмечали, что (62) и (63) *почти* работоспособны; читатель может убедиться, что затруднения возникают только в случае $L(4,6)$, когда (62) дает

$$\begin{aligned} L(4,6) &= L(2,6),\; 3L(1,3)^R,\; 4L(1,2),\; 321,\; 33,\; 42 \\ &= 111111,\; 21111,\; 2211,\; 222,\; 3111,\; 411,\; 321,\; 33,\; 42. \end{aligned} \tag{64}$$

Если $m > 4$, все в порядке, поскольку переход от конца $L(m-2,2m-2)$ к началу $(m-1)L(m-3,m-1)^R$ является переходом от $(m-2)(m-2)2$ к $(m-1)(m-3)2$. Удовлетворительного пути $L(4,6)$ не существует, поскольку все коды Грея через эти девять разбиений должны оканчиваться одним из следующих разбиений: 411, 33, 3111, 222 или 2211.

Для нейтрализации этой аномалии необходимо исправить определения $L(m,n)$ и $M(m,n)$ в восьми местах, где вызывается "некорректная подпрограмма" $L(4,6)$. Один простой путь состоит в следующих определениях:

$$\begin{aligned} L'(4,6) &= 111111, 21111, 3111, 411, 321, 33, 42; \\ L'(3,5) &= 11111, 2111, 221, 311, 32. \end{aligned} \tag{65}$$

Таким образом, в $L(4,6)$ опущены 222 и 2211; мы также перепрограммируем $L(3,5)$ так, чтобы 2111 было по соседству с 221. В этом случае, как показано в упр. 60, всегда легко "состыковать" два разбиения, отсутствующие в $L(4,6)$. ▌

Упражнения

▸ **1.** [*M21*] Найдите формулу для общего количества размещений в каждой из задач двенадцатизадачия. Например, количество n-кортежей из m вещей равно m^n. (Где это можно, воспользуйтесь обозначением из (38) и будьте аккуратны, чтобы ваши формулы работали даже при значениях $m = 0$ и $n = 0$.)

▶ **2.** [*20*] Покажите, что небольшое изменение шага H1 дает алгоритм, который генерирует все разбиения n *не более* чем на m частей.

3. [*M17*] Разбиение $a_1+\cdots+a_m$ числа n на m частей $a_1 \geq \cdots \geq a_m$ является *оптимально сбалансированным*, если $|a_i - a_j| \leq 1$ для $1 \leq i, j \leq m$. Докажите, что имеется ровно одно такое разбиение, какими бы ни были $n \geq m \geq 1$, и приведите простую формулу для выражения j-й части a_j в виде функции от j, m и n.

4. [*M22*] (Гидеон Эрлих (Gideon Ehrlich), 1974.) Каково лексикографически наименьшее разбиение n, в котором все части $\geq r$? Например, для $n = 19$ и $r = 5$ ответом является 766.

▶ **5.** [*23*] Разработайте алгоритм, который генерирует все разбиения n в виде количества частей $c_1 \ldots c_n$ из (8). Сгенерируйте их в солексном порядке, т. е. лексикографическом порядке $c_n \ldots c_1$, который эквивалентен лексикографическому порядку соответствующих разбиений $a_1 a_2 \ldots$. Для эффективности поддерживайте таблицу связей $l_0 l_1 \ldots l_n$, такую, что если различные значения k, для которых $c_k > 0$, представляют собой $k_1 < \cdots < k_t$, то

$$l_0 = k_1, \quad l_{k_1} = k_2, \quad \ldots, \quad l_{k_{t-1}} = k_t, \quad l_{k_t} = 0.$$

(Так, разбиение 331 будет представлено как $c_1 \ldots c_7 = 1020000$, $l_0 = 1$, $l_1 = 3$ и $l_3 = 0$; прочие связи l_2, l_4, l_5, l_6, l_7 могут принимать любые значения.)

6. [*20*] Разработайте алгоритм вычисления $b_1 b_2 \ldots = (a_1 a_2 \ldots)^T$ для заданного $a_1 a_2 \ldots$.

7. [*M20*] Предположим, что $a_1 \ldots a_n$ и $a'_1 \ldots a'_n$ — разбиения n, такие, что $a_1 \geq \cdots \geq a_n \geq 0$ и $a'_1 \geq \cdots \geq a'_n \geq 0$, и пусть им соответствуют сопряженные разбиения $b_1 \ldots b_n = (a_1 \ldots a_n)^T$ и $b'_1 \ldots b'_n = (a'_1 \ldots a'_n)^T$. Покажите, что $b_1 \ldots b_n < b'_1 \ldots b'_n$ тогда и только тогда, когда $a_n \ldots a_1 < a'_n \ldots a'_1$.

8. [*15*] Пусть $(p_1 \ldots p_t, q_1 \ldots q_t)$ — краевое представление разбиения $a_1 a_2 \ldots$, как в (15) и (16). Какой вид имеет в этом случае сопряженное разбиение $(a_1 a_2 \ldots)^T$?

9. [*22*] Пусть $a_1 a_2 \ldots a_m$ и $b_1 b_2 \ldots b_m = (a_1 a_2 \ldots a_m)^T$ — сопряженные разбиения. Покажите, что мультимножества $\{a_1 + 1, a_2 + 2, \ldots, a_m + m\}$ и $\{b_1 + 1, b_2 + 2, \ldots, b_m + m\}$ равны.

10. [*21*] При рассмотрении разбиений зачастую полезными оказываются бинарные деревья двух простых видов: (а) дерево, которое включает все разбиения всех целых чисел; (б) дерево, которое включает все разбиения данного целого n (на рисунке показано дерево для $n = 8$).

Выведите общее правило, лежащее в основе этих конструкций. Какой порядок обхода дерева соответствует лексикографическому порядку разбиений?

11. [*M22*] Сколько имеется способов заплатить один евро монетами номиналом 1, 2, 5, 10, 20, 50 и/или 100 центов? А если можно использовать не более двух монет каждого номинала?

▶ **12.** [*M21*] (Л. Эйлер (L. Euler), 1750.) Воспользуйтесь производящей функцией для доказательства того, что количество способов разбиения n на *различные* части равно количеству способов разбиения n на *нечетные* части. Например, $5 = 4+1 = 3+2$; $5 = 3+1+1 = 1+1+1+1+1$.

[*Примечание*: в двух следующих упражнениях использованы комбинаторные методы для доказательства расширений этой знаменитой теоремы.]

▶ **13.** [*M23*] (Ф. Франклин (F. Franklin), 1882.) Найдите взаимно однозначное соответствие $\alpha \leftrightarrow \beta$ между разбиениями n, такими, что α содержит ровно k повторяющихся частей тогда и только тогда, когда β содержит ровно k четных частей. (Например, разбиение 64421111 имеет две повторяющиеся части $\{4, 1\}$ и три четные части $\{6, 4, 2\}$. Случай $k = 0$ соответствует результату, полученному Эйлером.)

▶ **14.** [*M28*] (Д. Д. Сильвестер (J. J. Sylvester), 1882.) Найдите взаимно однозначное соответствие между разбиениями n на различные части $a_1 > a_2 > \cdots > a_m$ с ровно k "просветами", в которых $a_j > a_{j+1} + 1$, и разбиениями n на нечетные части с ровно $k+1$ различными значениями. (Например, при $k = 0$ это построение доказывает, что количество способов записать n как сумму последовательных целых чисел равно количеству нечетных делителей n.)

15. [*M20*] (Д. Д. Сильвестер (J. J. Sylvester).) Найдите производящую функцию для количества *самосопряженных* разбиений, т. е. таких разбиений, у которых $\alpha = \alpha^T$.

16. [*M21*] Найдите формулу для $\sum_{m,n} p(k,m,n)\, w^m z^n$, где $p(k,m,n)$ — количество разбиений n, состоящих из m частей и имеющих след k. Просуммируйте ее по k для получения нетривиального тождества.

17. [*M26*] *Объединенным разбиением* (*joint partition*) n называется пара последовательностей $(a_1, \ldots, a_r;\ b_1, \ldots, b_s)$ положительных целых чисел, у которых

$$a_1 \geq \cdots \geq a_r, \quad b_1 > \cdots > b_s, \quad \text{и} \quad a_1 + \cdots + a_r + b_1 + \cdots + b_s = n.$$

Таким образом, при $s = 0$ получается обычное разбиение, а при $r = 0$ — разбиение на различные части.

а) Найдите простую формулу для производящей функции $\sum u^{r+s} v^s z^n$, где суммирование выполняется по всем объединенным разбиениям n с r обычных частей a_i и s различных частей b_j.

б) Аналогично найдите простую формулу для $\sum v^s z^n$, где суммирование выполняется по всем объединенным разбиениям, состоящим ровно из $r + s = t$ частей, для заданного значения t. Например, ответом для $t = 2$ является $(1+v)(1+vz)z^2/((1-z)(1-z^2))$.

в) Какое тождество вы вывели?

▶ **18.** [*M23*] (Дорон Зайльбергер (Doron Zeilberger).) Покажите, что существует взаимно однозначное соответствие между парами целочисленных последовательностей $(a_1, a_2, \ldots, a_r;\ b_1, b_2, \ldots, b_s)$, таких, что

$$a_1 \geq a_2 \geq \cdots \geq a_r, \qquad b_1 > b_2 > \cdots > b_s,$$

и парами целочисленных последовательностей $(c_1, c_2, \ldots, c_{r+s};\ d_1, d_2, \ldots, d_{r+s})$, таких, что

$$c_1 \geq c_2 \geq \cdots \geq c_{r+s}, \qquad d_j \in \{0, 1\} \quad \text{для } 1 \leq j \leq r+s,$$

связанных уравнениями для мультимножеств

$$\{a_1, a_2, \ldots, a_r\} = \{c_j \mid d_j = 0\} \quad \text{и} \quad \{b_1, b_2, \ldots, b_s\} = \{c_j + r + s - j \mid d_j = 1\}.$$

В результате получаем интересное тождество

$$\sum_{\substack{a_1\ge\cdots\ge a_r>0,\,r\ge0\\ b_1>\cdots>b_s>0,\,s\ge0}} u^{r+s}v^s z^{a_1+\cdots+a_r+b_1+\cdots+b_s} = \sum_{\substack{c_1\ge\cdots\ge c_t>0,\,t\ge0\\ d_1,\ldots,d_t\in\{0,1\}}} u^t v^{d_1+\cdots+d_t} z^{c_1+\cdots+c_t+(t-1)d_1+\cdots+d_{t-1}}.$$

19. [*M22*] (Э. Гейне (E. Heine), 1847.) Докажите четырехпараметрическое тождество

$$\prod_{m=1}^{\infty}\frac{(1-wxz^m)(1-wyz^m)}{(1-wz^m)(1-wxyz^m)}=\sum_{k=0}^{\infty}\frac{w^k(x-1)(x-z)\ldots(x-z^{k-1})(y-1)(y-z)\ldots(y-z^{k-1})z^k}{(1-z)(1-z^2)\ldots(1-z^k)(1-wz)(1-wz^2)\ldots(1-wz^k)}.$$

Указание: выполните суммирование по k или l в формуле

$$\sum_{k,l\ge0} u^k v^l z^{kl}\frac{(z-az)(z-az^2)\ldots(z-az^k)}{(1-z)(1-z^2)\ldots(1-z^k)}\,\frac{(z-bz)(z-bz^2)\ldots(z-bz^l)}{(1-z)(1-z^2)\ldots(1-z^l)}$$

и рассмотрите упрощение, возникающее при $b=auz$.

▶ **20.** [*M21*] Сколько приблизительно времени займет вычисление таблицы числа разбиений $p(n)$ для $1\le n\le N$ с использованием рекуррентного соотношения Эйлера (20)?

21. [*M21*] (Л. Эйлер (L. Euler).) Пусть $q(n)$ — количество разбиений n на различные части. Как лучше вычислить $q(n)$, зная $p(1)$, ..., $p(n)$?

22. [*HM21*] (Л. Эйлер (L. Euler).) Пусть $\sigma(n)$ — сумма всех положительных делителей положительного целого числа n. Тогда $\sigma(n)=n+1$ для простого n и может оказаться существенно больше n для "высокосоставного" n. Докажите, что, несмотря на достаточно хаотичное поведение, $\sigma(n)$ удовлетворяет почти такому же рекуррентному соотношению (20), как и количество разбиений:

$$\sigma(n)=\sigma(n-1)+\sigma(n-2)-\sigma(n-5)-\sigma(n-7)+\sigma(n-12)+\sigma(n-15)-\cdots$$

для $n\ge1$, с тем отличием, что, когда в правой части встречается член '$\sigma(0)$', вместо него используется значение 'n'. Например, $\sigma(11)=1+11=\sigma(10)+\sigma(9)-\sigma(6)-\sigma(4)=18+13-12-7$; $\sigma(12)=1+2+3+4+6+12=\sigma(11)+\sigma(10)-\sigma(7)-\sigma(5)+12=12+18-8-6+12$.

23. [*HM25*] Воспользуйтесь тождеством тройного произведения Якоби (19) для доказательства еще одной открытой им формулы:

$$\prod_{k=1}^{\infty}(1-z^k)^3 \;=\; 1-3z+5z^3-7z^6+9z^{10}-\cdots \;=\; \sum_{n=0}^{\infty}(-1)^n(2n+1)z^{\binom{n+1}{2}}.$$

24. [*M26*] (С. Рамануджан (S. Ramanujan), 1919.) Пусть $A(z)=\prod_{k=1}^{\infty}(1-z^k)^4$.

а) Докажите, что $[z^n]\,A(z)$ кратно 5, если $n \bmod 5=4$.

б) Докажите, что $[z^n]\,A(z)B(z)^5$ обладает тем же свойством, где B — произвольный степенной ряд с целыми коэффициентами.

в) Следовательно, $p(n)$ кратно 5, если $n \bmod 5=4$.

25. [*HM27*] Улучшите оценку $\ln P(e^{-t})$ (22) с помощью (а) формулы суммирования Эйлера и (б) преобразований Меллина. *Указание*: билогарифмическая функция $\mathrm{Li}_2(x)=x/1^2+x^2/2^2+x^3/3^2+\cdots$ удовлетворяет соотношению $\mathrm{Li}_2(x)+\mathrm{Li}_2(1-x)=\zeta(2)-(\ln x)\ln(1-x)$.

26. [*HM22*] В упр. 5.2.2–44 и 5.2.2–51 мы изучили два способа доказательства того, что

$$\sum_{k=1}^{\infty}e^{-k^2/n}\;=\frac{1}{2}(\sqrt{\pi n}-1)+O(n^{-M})\qquad\text{для всех } M>0.$$

Покажите, что формула суммирования Пуассона дает более строгий результат.

27. [*HM21*] Докажите (28) и завершите вычисления, приводящие к теореме D.

28. [*HM42*] (Д. Г. Лемер (D. H. Lehmer).) Покажите, что коэффициенты Харди–Рамануджана–Радемахера $A_k(n)$, определенные в (34), должны обладать следующими замечательными свойствами.

а) Если k нечетно, то $A_{2k}(km + 4n + (k^2 - 1)/8) = A_2(m)A_k(n)$.

б) Если p — простое число, $p^e > 2$ и $k \perp 2p$, то

$$A_{p^e k}(k^2 m + p^{2e} n - (k^2 + p^{2e} - 1)/24) \;=\; (-1)^{[p^e=4]} A_{p^e}(m) A_k(n).$$

В этой формуле $k^2 + p^{2e} - 1$ кратно 24, если p или k делятся на 2 или 3; в противном случае деление на 24 должно выполняться по модулю $p^e k$.

в) Если p — простое число, $|A_{p^e}(n)| < 2^{[p>2]} p^{e/2}$.

г) Если p — простое число, $A_{p^e}(n) \neq 0$ тогда и только тогда, когда $1 - 24n$ — квадратичный вычет по модулю p и либо $e = 1$, либо $24n \bmod p \neq 1$.

д) Вероятность того, что $A_k(n) = 0$, если k делится ровно на t простых чисел ≥ 5, а n — случайное целое число, приблизительно равна $1 - 2^{-t}$.

▶ **29.** [*M16*] Обобщая (41), вычислите сумму $\sum_{a_1 \geq a_2 \geq \cdots \geq a_m \geq 1} z_1^{a_1} z_2^{a_2} \ldots z_m^{a_m}$.

30. [*M17*] Найдите суммы

$$\text{(а)} \quad \sum_{k \geq 0} \left| {n - km \atop m - 1} \right| \qquad \text{и} \qquad \text{(б)} \quad \sum_{k \geq 0} \left| {n \atop m - k} \right|$$

в аналитическом виде (эти суммы конечны, поскольку при больших k суммируемые члены равны 0).

31. [*M24*] (А. де Морган (A. de Morgan), 1843.) Покажите, что $\left|{n \atop 2}\right| = \lfloor n/2 \rfloor$ и $\left|{n \atop 3}\right| = \lfloor (n^2 + 6)/12 \rfloor$; найдите аналогичную формулу для $\left|{n \atop 4}\right|$.

32. [*M15*] Докажите, что $\left|{n \atop m}\right| \leq p(n - m)$ для всех $m, n \geq 0$. Когда неравенство превращается в равенство?

33. [*HM20*] Воспользуйтесь тем фактом, что имеется ровно $\binom{n-1}{m-1}$ *композиций* n из m частей (см. 7.2.1.3–(9)), для доказательства нижней границы $\left|{n \atop m}\right|$. Затем приравняйте $m = \lfloor \sqrt{n} \rfloor$ для получения элементарной нижней границы $p(n)$.

▶ **34.** [*HM21*] Покажите, что $\left|{n - m(m-1)/2 \atop m}\right|$ — количество разбиений n на m различных частей. Следовательно,

$$\left| {n \atop m} \right| \;=\; \frac{n^{m-1}}{m!\,(m-1)!} \left(1 + O\left(\frac{m^3}{n}\right)\right) \qquad \text{при } m \leq n^{1/3}.$$

35. [*HM21*] Рассмотрим распределение вероятностей Эрдеша–Ленера (43). (а) Какое значение x наиболее вероятно? (б) Какое значение x является медианой? (в) Какое значение x является средним значением? (г) Чему равно стандартное отклонение?

36. [*HM24*] Докажите ключевую оценку (47), необходимую для теоремы E.

37. [*M22*] Докажите лемму (48) о том, что при использовании принципа включения и исключения частичные суммы "берут в вилку" истинное значение, проанализировав, сколько раз разбиение ровно с q различными частями, превосходящими m, учитывается в r-й частичной сумме.

38. [*M20*] Какова для заданных положительных целых чисел l и m производящая функция, перечисляющая разбиения из m частей с наибольшей частью l? (См. (51).)

39. [*M20*] (А. Коши (A. Cauchy).) Продолжая выполнять упр. 38, укажите производящую функцию для количества разбиений на m *различных* частей, причем все они меньше l?

▶ **40.** [*M25*] (Ф. Франклин (F. Franklin).) Обобщая теорему C, покажите, что при $0 \le k \le m$

$$[z^n]\,\frac{(1-z^{l+1})\ldots(1-z^{l+k})}{(1-z)(1-z^2)\ldots(1-z^m)}$$

представляет собой количество разбиений $a_1a_2\ldots$ числа n на m или меньшее количество частей, обладающих тем свойством, что $a_1 \le a_{k+1}+l$.

41. [*HM42*] Расширьте формулу Харди–Рамануджана–Радемахера (32) для получения сходящегося ряда для разбиений n не более чем на m частей, ни одна из которых не превосходит l.

42. [*HM42*] Найдите ограниченную область, аналогичную (49), для случайных разбиений n на не более $\theta\sqrt{n}$ частей, ни одна из которых не превосходит $\varphi\sqrt{n}$, в предположении, что $\theta\varphi > 1$.

43. [*M18*] Сколько для заданных n и k имеется разбиений n, таких, что $a_1 > a_2 > \cdots > a_k$?

▶ **44.** [*M22*] У скольких разбиений n две наименьшие части равны?

45. [*HM21*] Вычислите асимптотическое значение $p(n-1)/p(n)$ с относительной ошибкой $O(n^{-2})$.

46. [*M20*] Что больше — $T_2'(n)$ или $T_2''(n)$ — в анализе алгоритма P в разделе?

▶ **47.** [*HM22*] (А. Ньенхьюз (A. Nijenhuis) и Г. С. Вильф (H. S. Wilf), 1975.) Приведенный далее простой алгоритм, основываясь на таблице количеств разбиений $p(0)$, $p(1)$, $\ldots$, $p(n)$, генерирует случайное разбиение числа n с использованием представления в виде количества частей $c_1\ldots c_n$ (8). Докажите, что он генерирует все разбиения с одной и той же вероятностью.

N1. [Инициализация.] Установить $m \leftarrow n$ и $c_1\ldots c_n \leftarrow 0\ldots 0$.

N2. [Выполнено?] Завершить работу, если $m = 0$.

N3. [Генерация.] Сгенерировать случайное целое число M из диапазона $0 \le M < mp(m)$.

N4. [Выбор частей.] Установить $s \leftarrow 0$. Затем для $j = 1, 2, \ldots$ и для $k = 1, 2, \ldots, \lfloor m/j \rfloor$ многократно устанавливать $s \leftarrow s+kp(m-jk)$, пока не будет выполнено условие $s > M$.

N5. [Обновление.] Установить $c_k \leftarrow c_k + j$, $m \leftarrow m - jk$ и вернуться к шагу N2. ▮

Указание: на шаге N4, основанном на тождестве

$$\sum_{j=1}^{\infty}\sum_{k=1}^{\lfloor m/j\rfloor} kp(m-jk) \;=\; mp(m),$$

выбирается пара значений (j,k) с вероятностью $kp(m-jk)/(mp(m))$.

48. [*HM40*] Проанализируйте время работы алгоритма из предыдущего упражнения.

▶ **49.** [*HM26*] (а) Какой вид имеет производящая функция $F(z)$ для суммы наименьших частей всех разбиений n? (Ряд начинается с $z + 3z^2 + 5z^3 + 9z^4 + 12z^5 + \cdots$.)

(б) Найдите асимптотическое значение $[z^n]\,F(z)$ с относительной ошибкой $O(n^{-1})$.

50. [*HM33*] Обозначим $c(m) = c_m(2m)$ в рекуррентных соотношениях (56) и (57).

а) Докажите, что $c_m(m+k) = m-k+c(k)$ для $0 \le k \le m$.

б) Следовательно, (58) выполняется для $m \le n \le 2m$, если $c(m) < 3p(m)$ для всех $m \ge 0$.

в) Покажите, что $c(m) - m$ равно сумме вторых наименьших частей всех разбиений m.

г) Найдите взаимно однозначное соответствие между всеми разбиениями n, вторая наименьшая часть которых равна k, и всеми разбиениями чисел $\le n$, наименьшая часть которых равна $k+1$.

д) Опишите производящую функцию $\sum_{m\ge 0} c(m)z^m$.

е) Выведите, что $c(m) < 3p(m)$ для всех $m \ge 0$.

51. [*M46*] Выполните детальный анализ алгоритма H.

▶ **52.** [*M21*] Каково миллионное разбиение, сгенерированное алгоритмом P для $n = 64$? *Указание*: $p(64) = 1741630 = 1000000 + \binom{77}{13} + \binom{60}{10} + \binom{47}{8} + \binom{35}{5} + \binom{27}{3} + \binom{22}{2} + \binom{18}{1} + \binom{15}{0}$.

▶ **53.** [*M21*] Каково миллионное разбиение, сгенерированное алгоритмом H для $m = 32$ и $n = 100$? *Указание*: $999999 = \binom{80}{12} + \binom{66}{11} + \binom{50}{7} + \binom{41}{6} + \binom{33}{5} + \binom{26}{4} + \binom{21}{4}$.

▶ **54.** [*M30*] Пусть $\alpha = a_1a_2\ldots$ и $\beta = b_1b_2\ldots$ — разбиения n. Мы говорим, что α *мажоризирует* β, что записывается как $\alpha \succeq \beta$ или $\beta \preceq \alpha$, если $a_1 + \cdots + a_k \ge b_1 + \cdots + b_k$ для всех $k \ge 0$.

а) Истинно или ложно следующее утверждение? Из $\alpha \succeq \beta$ вытекает $\alpha \ge \beta$ (лексикографически).

б) Истинно или ложно следующее утверждение? Из $\alpha \succeq \beta$ вытекает $\beta^T \succeq \alpha^T$.

в) Покажите, что любые два разбиения n имеют наибольшую нижнюю границу $\alpha \wedge \beta$, такую, что $\alpha \succeq \gamma$ и $\beta \succeq \gamma$ тогда и только тогда, когда $\alpha \wedge \beta \succeq \gamma$. Поясните, как вычислить $\alpha \wedge \beta$.

г) Аналогично поясните, как вычислить наименьшую верхнюю границу $\alpha \vee \beta$, такую, что $\gamma \succeq \alpha$ и $\gamma \succeq \beta$ тогда и только тогда, когда $\gamma \succeq \alpha \vee \beta$.

д) Если α имеет l частей, а β — m частей, то сколько частей имеют $\alpha \wedge \beta$ и $\alpha \vee \beta$?

е) Истинно или ложно следующее утверждение? Если α состоит из различных частей и β состоит из различных частей, то же самое можно сказать и о $\alpha \wedge \beta$ и $\alpha \vee \beta$.

▶ **55.** [*M37*] Продолжая выполнять предыдущее упражнения, назовем α *покрытием* β, если $\alpha \succeq \beta$ и $\alpha \ne \beta$ и если из $\alpha \succeq \gamma \succeq \beta$ вытекает $\gamma = \alpha$ или $\gamma = \beta$. Например, на рис. 52 показано отношение покрытия между разбиениями числа 12.

а) Будем записывать $\alpha \rhd \beta$, если $\alpha = a_1a_2\ldots$ и $\beta = b_1b_2\ldots$ представляют собой разбиения, для которых $b_k = a_k - [k=l] + [k=l+1]$ для всех $k \ge 1$ и некоторого $l \ge 1$. Докажите, что α покрывает β тогда и только тогда, когда $\alpha \rhd \beta$ или $\beta^T \rhd \alpha^T$.

б) Покажите, что существует простой способ выяснить, верно ли, что α покрывает β, путем расмотрения краевого представления α и β.

в) Пусть $n = \binom{n_2}{2} + \binom{n_1}{1}$, где $n_2 > n_1 \ge 0$ и $n_2 > 2$. Покажите, что ни одно разбиение n не покрывает больше $n_2 - 2$ разбиений.

г) Будем называть разбиение μ *минимальным*, если не существует разбиения λ, такого, что $\mu \rhd \lambda$. Докажите, что μ минимально тогда и только тогда, когда μ^T состоит из различных частей.

д) Предположим, что $\alpha = \alpha_0 \rhd \alpha_1 \rhd \cdots \rhd \alpha_k$ и $\alpha = \alpha'_0 \rhd \alpha'_1 \rhd \cdots \rhd \alpha'_{k'}$, где α_k и $\alpha'_{k'}$ — минимальные разбиения. Докажите, что $k = k'$ и $\alpha_k = \alpha'_{k'}$.

е) Поясните, как вычислить лексикографически наименьшее разбиение на различные части, которое мажоризирует данное разбиение α.

ж) Опишите лексикографически наименьшее разбиение λ_n числа n на различные части. Чему равна длина всех путей $n^1 = \alpha_0 \rhd \alpha_1 \rhd \cdots \rhd \lambda_n^T$?

з) Чему равна длина длиннейшего и кратчайшего путей вида $n^1 = \alpha_0, \alpha_1, \ldots, \alpha_l = 1^n$, где α_j покрывает α_{j+1} для $0 \le j < l$?

▶ **56.** [*M32*] Разработайте алгоритм для генерации всех разбиений α, таких, что $\lambda \preceq \alpha \preceq \mu$, для данных разбиений λ и μ, $\lambda \preceq \mu$.

Рис. 52. Решетка мажоризации для разбиений 12. (См. упр. 54–58.)

Примечание. Такой алгоритм имеет множество применений. Например, для генерации всех разбиений, которые имеют m частей, причем ни одна из них не превышает l, можно положить λ равным минимальному такому разбиению, т. е., как в упр. 3, $\lceil n/m\rceil \ldots \lfloor n/m\rfloor$, а μ — наибольшим, т. е. $\bigl((n{-}m{+}1)1^{m-1}\bigr) \wedge \bigl(l^{\lfloor n/l\rfloor}(n \bmod l)\bigr)$. Аналогично, согласно известной теореме Х. Г. Ландау (H. G. Landau) [*Bull. Math. Biophysics* **15** (1953), 143–148], разбиения $\binom{m}{2}$, такие, что

$$\left\lfloor\frac{m}{2}\right\rfloor^{\lfloor m/2\rfloor}\left\lfloor\frac{m-1}{2}\right\rfloor^{\lceil m/2\rceil} \preceq \alpha \preceq (m{-}1)(m{-}2)\ldots 21,$$

представляют собой “векторы счетов” в круговом турнире, т. е. разбиения $a_1\ldots a_m$, когда игрок на j-м месте побеждает в a_j играх.

57. [*M22*] Предположим, что матрица (a_{ij}) из нулей и единиц имеет суммы элементов строк $r_i = \sum_j a_{ij}$ и суммы элементов столбцов $c_j = \sum_i a_{ij}$. Путем перестановки строк и столбцов можно добиться того, что $r_1 \ge r_2 \ge \cdots$ и $c_1 \ge c_2 \ge \cdots$. В таком случае $\lambda = r_1r_2\ldots$ и $\mu = c_1c_2\ldots$ представляют собой разбиения $n = \sum_{i,j} a_{ij}$. Докажите, что такая матрица существует тогда и только тогда, когда $\lambda \preceq \mu^T$.

58. [*M23*] (*Симметричные средние.*) Пусть $\alpha = a_1 \ldots a_m$ и $\beta = b_1 \ldots b_m$ — разбиения n. Докажите, что неравенство

$$\frac{1}{m!}\sum x_{p_1}^{a_1} \ldots x_{p_m}^{a_m} \;\geq\; \frac{1}{m!}\sum x_{p_1}^{b_1} \ldots x_{p_m}^{b_m}$$

выполняется для всех неотрицательных значений переменных $(x_1, \ldots, x_m)$, где суммы берутся по всем $m!$ перестановкам $\{1, \ldots, m\}$, тогда и только тогда, когда $\alpha \succeq \beta$. (Например, это неравенство сводится к $(y_1 + \cdots + y_n)/n \geq (y_1 \ldots y_n)^{1/n}$ в частном случае $m = n$, $\alpha = n0\ldots0$, $\beta = 11\ldots1$, $x_j = y_j^{1/n}$.)

59. [*M22*] Путь Грея (59) симметричен в том смысле, что обращенная последовательность 6, 51, ..., 111111 имеет тот же вид, что и сопряженная последовательность $(111111)^T$, $(21111)^T$, ..., $(6)^T$. Найдите все пути Грея α_1, ..., $\alpha_{p(n)}$, симметричные в данном смысле.

60. [*23*] Завершите доказательство теоремы S, изменяя определения $L(m,n)$ и $M(m,n)$ во всех местах, где в (62) и (63) вызывается $L(4,6)$.

61. [*26*] Реализуйте схему генерации разбиений на основе теоремы S, всегда указывая две части, которые должны изменяться между посещениями.

62. [*46*] Докажите или опровергните следующее утверждение: для всех достаточно больших целых чисел n и $3 \leq m < n$, таких, что $n \bmod m \neq 0$, и для всех разбиений α числа n, у которых $a_1 \leq m$, существует путь Грея для всех разбиений с частями $\leq m$, начинающийся с 1^n и заканчивающийся α, за исключением $\alpha = 1^n$ и $\alpha = 21^{n-2}$.

63. [*47*] Для каких разбиений λ и μ имеется код Грея по всем разбиениям α, таким, что $\lambda \preceq \alpha \preceq \mu$?

▶ **64.** [*32*] (*Бинарные разбиения.*) Разработайте алгоритм, не содержащий циклов, который посещает все разбиения n на степени 2, где каждый шаг заменяет $2^k + 2^k$ на 2^{k+1} или наоборот.

65. [*23*] Хорошо известно, что каждая коммутативная группа из m элементов может быть представлена в виде дискретного тора $T(m_1, \ldots, m_n)$ с операцией сложения 7.2.1.3–(66), где $m = m_1 \ldots m_n$, а m_j кратно m_{j+1} при $1 \leq j < n$. Например, для $m = 360 = 2^3 \cdot 3^2 \cdot 5^1$ имеется шесть таких групп, соответствующих разложениям $(m_1, m_2, m_3) = (30, 6, 2)$, $(60, 6, 1)$, $(90, 2, 2)$, $(120, 3, 1)$, $(180, 2, 1)$ и $(360, 1, 1)$.

Поясните, как систематически сгенерировать все такие разложения с помощью алгоритма, который на каждом шаге изменяет ровно два множителя m_j.

▶ **66.** [*M25*] (*P-разбиения.*) Предположим, что вместо требования $a_1 \geq a_2 \geq \cdots$ мы рассматриваем все неотрицательные композиции n, удовлетворяющие некоторому заданному *частичному* порядку. Например, П. А. Мак-Мэган (P. A. MacMahon) заметил, что все решения "холмистых" неравенств $a_4 \leq a_2 \geq a_3 \leq a_1$ можно разделить на пять неперекрывающихся типов:

$$a_1 \geq a_2 \geq a_3 \geq a_4; \quad a_1 \geq a_2 \geq a_4 > a_3;$$
$$a_2 > a_1 \geq a_3 \geq a_4; \quad a_2 > a_1 \geq a_4 > a_3; \quad a_2 \geq a_4 > a_1 \geq a_3.$$

Все эти типы легко перечислимы, поскольку, например, $a_2 > a_1 \geq a_4 > a_3$ эквивалентно $a_2 - 2 \geq a_1 - 1 \geq a_4 - 1 \geq a_3$; количество решений при $a_3 \geq 0$ и $a_1 + a_2 + a_3 + a_4 = n$ равно количеству разбиений $n - 1 - 2 - 0 - 1$ не более чем на четыре части.

Поясните, как решить общую задачу следующего вида. Дано отношение частичного порядка $\prec$ для m элементов. Рассмотрим все m-кортежи $a_1 \ldots a_m$, обладающие тем свойством, что $a_j \geq a_k$ при $j \prec k$. Полагая, что индексы выбраны таким образом, что из $j \prec k$ вытекает $j \leq k$, покажите, что все соответствующие m-кортежи делятся на N классов, по одному для каждого из выходов алгоритма топологической сортировки 7.2.1.2V. Какой вид

имеет производящая функция для всех неотрицательных $a_1 \ldots a_m$, сумма которых равна n? Каким образом можно сгенерировать их всех?

67. [*M25*] (П. А. Мак-Мэган (P. A. MacMahon), 1886.) *Идеальное разбиение* n представляет собой мультимножество, которое имеет ровно $n+1$ подмультимножеств, и эти мультимножества представляют собой разбиения целых чисел 0, 1, ..., n. Например, мультимножества $\{1,1,1,1,1\}$, $\{2,2,1\}$ и $\{3,1,1\}$ являются идеальными разбиениями 5.

Поясните, как построить идеальные разбиения n, имеющие наименьшее количество элементов.

68. [*M23*] Какое разбиение n на m частей имеет наибольшее произведение $a_1 \ldots a_m$ при (а) заданном m; (б) произвольном m?

69. [*M30*] Найдите все $n < 10^9$, такие, что уравнение $x_1 + x_2 + \cdots + x_n = x_1 x_2 \ldots x_n$ имеет единственное решение в натуральных числах $x_1 \geq x_2 \geq \cdots \geq x_n$. (Имеются, например, единственные решения для $n = 2$, 3 и 4; однако $5+2+1+1+1 = 5\cdot2\cdot1\cdot1\cdot1$, $3+3+1+1+1 = 3\cdot3\cdot1\cdot1\cdot1$ и $2+2+2+1+1 = 2\cdot2\cdot2\cdot1\cdot1$.)

70. [*M30*] ("Болгарский пасьянс.") Даны n карт, разделенные произвольным образом на одну или несколько стопок. Затем многократно выполняются следующие действия — из каждой стопки берется по одной карте, и они образуют новую стопку.

Покажите, что если $n = 1+2+\cdots+m$, то этот процесс всегда достигает самовоспроизводящегося состояния со стопками размером $\{m, m-1, \ldots, 1\}$. Например, если $n = 10$ и мы начнем со стопок размером $\{3,3,2,2\}$, то получим такую последовательность разбиений:

$$3322 \to 42211 \to 5311 \to 442 \to 3331 \to 4222 \to 43111 \to 532 \to 4321 \to 4321 \to \cdots.$$

Какие циклы состояний возможны для других значений n?

71. [*M46*] Продолжая выполнять предыдущее упражнение, выясните, чему равно максимальное количество шагов, которые могут быть выполнены до того, как Болгарский пасьянс с n картами достигнет циклического состояния.

72. [*M30*] Сколько разбиений n не имеют предшественника в Болгарском пасьянсе?

73. [*M25*] Предположим, мы записали все разбиения n, например

$$6,\ 51,\ 42,\ 411,\ 33,\ 321,\ 3111,\ 222,\ 2211,\ 21111,\ 111111$$

для $n = 6$, и заменили все j-е появления числа k на j:

$$1,\ 11,\ 11,\ 112,\ 12,\ 111,\ 1123,\ 123,\ 1212,\ 11234,\ 123456.$$

а) Докажите, что эта операция порождает перестановку отдельных элементов.
б) Сколько всего раз встречается элемент k?

7.2.1.5. Генерация всех разбиений множеств. Перейдем теперь к рассмотрению другого вида разбиений. *Разбиения множеств* представляют собой способы рассмотрения множества как объединения непустых, непересекающихся множеств, именуемых *блоками*. Например, в начале предыдущего раздела были перечислены пять существенно различных разбиений множества $\{1,2,3\}$ (7.2.1.4–(2) и 7.2.1.4–(4)). Эти пять разбиений более компактно можно записать в виде

$$123, \quad 12|3, \quad 13|2, \quad 1|23, \quad 1|2|3 \tag{1}$$

с применением вертикальной черты для отделения одного блока от другого. В этом списке элементы каждого блока могут быть записаны в любом порядке, как и сами блоки, так что '13|2', '31|2', '2|13' и '2|31' — это представления одного и того же разбиения. Можно стандартизировать представления путем соглашения, например,

о перечислении элементов каждого блока в возрастающем порядке, а сами блоки располагать в порядке возрастания их наименьших элементов. При этих соглашениях разбиениями множества $\{1,2,3,4\}$ являются

$$\begin{gathered}1234,\ 123|4,\ 124|3,\ 12|34,\ 12|3|4,\ 134|2,\ 13|24,\ 13|2|4,\\ 14|23,\ 1|234,\ 1|23|4,\ 14|2|3,\ 1|24|3,\ 1|2|34,\ 1|2|3|4,\end{gathered} \tag{2}$$

получаемые путем добавления 4 к блокам (1) всеми возможными способами.

Разбиения множеств появляются в самых разных контекстах. Политики и экономисты, например, часто говорят о "коалициях", разработчики вычислительных систем — о "шаблонах обращения к кэш-памяти", поэты — о "схемах рифм" (см. упр. 34–37). В разделе 2.3.3 мы встречались с *отношением эквивалентности* между объектами, а именно с бинарным отношением, обладающим свойствами рефлексивности, симметричности и транзитивности, и определяющим разбиение этих объектов на так называемые "классы эквивалентности". Верно и обратное: каждое разбиение множества определяет отношение эквивалентности; в частности, если Π является разбиением множества $\{1,2,\ldots,n\}$, можно записать

$$j \equiv k \ \text{(по модулю } \Pi), \tag{3}$$

если j и k принадлежат одному и тому же блоку Π.

Один из наиболее удобных способов представления разбиения множества в компьютере состоит в кодировании его *ограниченно растущей строкой* (*restricted growth string*), т. е. строкой $a_1a_2\ldots a_n$ неотрицательных целых чисел, в которой

$$a_1 = 0 \quad \text{и} \quad a_{j+1} \le 1 + \max(a_1,\ldots,a_j) \text{ для } 1 \le j < n. \tag{4}$$

Идея заключается в установке $a_j = a_k$ тогда и только тогда, когда $j \equiv k$, причем выбирать при этом для a_j, где j — наименьшее число в своем блоке, наименьшее доступное число. Например, ограниченно растущие строки для пятнадцати разбиений (2) представляют собой соответственно

$$\begin{gathered}0000,\ 0001,\ 0010,\ 0011,\ 0012,\ 0100,\ 0101,\ 0102,\\ 0110,\ 0111,\ 0112,\ 0120,\ 0121,\ 0122,\ 0123.\end{gathered} \tag{5}$$

Такое соглашение наводит на мысль о следующей простой схеме генерации разбиений, предложенной Джорджем Хатчинсоном (George Hutchinson) [*CACM* **6** (1963), 613–614].

Алгоритм H (*Ограниченно растущие строки в лексикографическом порядке*). Для заданного $n \ge 2$ данный алгоритм генерирует все разбиения $\{1,2,\ldots,n\}$ путем посещения всех строк $a_1a_2\ldots a_n$, удовлетворяющих условию ограниченного роста (4). Поддерживается вспомогательный массив $b_1b_2\ldots b_n$, где $b_{j+1} = 1 + \max(a_1,\ldots,a_j)$; значение b_n из соображений эффективности реально хранится в отдельной переменной m.

H1. [Инициализация.] Установить $a_1\ldots a_n \leftarrow 0\ldots 0$, $b_1\ldots b_{n-1} \leftarrow 1\ldots 1$ и $m \leftarrow 1$.

H2. [Посещение.] Посетить ограниченно растущую строку $a_1\ldots a_n$, которая представляет разбиение на $m + [a_n = m]$ блоков. Затем перейти к шагу H4, если $a_n = m$.

H3. [Увеличение a_n.] Установить $a_n \leftarrow a_n + 1$ и вернуться к шагу H2.

H4. [Поиск j.] Установить $j \leftarrow n-1$; затем, пока $a_j = b_j$, устанавливать $j \leftarrow j-1$.

H5. [Увеличение a_j.] Завершить работу алгоритма, если $j = 1$. В противном случае установить $a_j \leftarrow a_j + 1$.

H6. [Обнуление $a_{j+1} \ldots a_n$.] Установить $m \leftarrow b_j + [a_j = b_j]$ и $j \leftarrow j + 1$. Затем, пока $j < n$, устанавливать $a_j \leftarrow 0$, $b_j \leftarrow m$ и $j \leftarrow j + 1$. Наконец установить $a_n \leftarrow 0$ и вернуться к шагу H2. ▮

В упр. 47 доказывается, что шаги H4–H6 выполняются редко, а циклы на шагах H4 и H6 почти всегда короткие. Вариант этого алгоритма с использованием связанного списка имеется в упр. 2.

Коды Грея для разбиений множеств. Один из способов быстрого прохода по всем разбиениям множества состоит в изменении на каждом шаге только одной цифры ограниченно растущей строки $a_1 \ldots a_n$, поскольку изменение a_j просто означает перемещение элемента j из одного блока в другой. Элегантный способ построения такого списка был предложен Гидеоном Эрлихом (Gideon Ehrlich) [*JACM* **20** (1973), 507–508]. Можно последовательно добавлять цифры

$$0,\ m,\ m-1,\ \ldots,\ 1 \qquad \text{или} \qquad 1,\ \ldots,\ m-1,\ m,\ 0 \tag{6}$$

к каждой строке $a_1 \ldots a_{n-1}$ в списке разбиений $n-1$ элементов, где $m = 1 + \max(a_1, \ldots, a_{n-1})$, чередуя оба варианта. Таким образом, список '00, 01' для $n = 2$ превращается в '000, 001, 011, 012, 010' для $n = 3$, который, в свою очередь, для $n = 4$ превращается в

$$\begin{gathered}0000,\ 0001,\ 0011,\ 0012,\ 0010,\ 0110,\ 0112,\ 0111,\\ 0121,\ 0122,\ 0123,\ 0120,\ 0100,\ 0102,\ 0101.\end{gathered} \tag{7}$$

В упр. 14 показано, что схема Эрлиха приводит к простому алгоритму, который дает порядок кода Грея без особых дополнительных усилий по сравнению с алгоритмом H.

Предположим, однако, что нас не интересуют *все* разбиения; мы можем захотеть рассмотреть только те из них, которые состоят ровно из m блоков. Можно ли пройти по этой меньшей коллекции ограниченно растущих строк так, чтобы на каждом шаге изменялась только одна цифра строки? Да; очень красивый способ генерации такого списка был открыт Фрэнком Раски (Frank Ruskey) [*Lecture Notes in Comp. Sci.* **762** (1993), 205–206]. Он определил две такие последовательности, A_{mn} и A'_{mn}, которые начинаются с лексикографически наименьшей m-блочной строки $0^{n-m}01\ldots(m-1)$. Различие между ними в том, что если $n > m+1$, то A_{mn} заканчивается $01\ldots(m-1)0^{n-m}$, в то время как A'_{mn} заканчивается $0^{n-m-1}01\ldots(m-1)0$. Вот рекурсивные соотношения Раски для $1 < m < n$:

$$A_{m(n+1)} = \begin{cases} A_{(m-1)n}(m-1), A^R_{mn}(m-1), \ldots, A^R_{mn}1, A_{mn}0, & \text{если } m \text{ четно;} \\ A'_{(m-1)n}(m-1), A_{mn}(m-1), \ldots, A^R_{mn}1, A_{mn}0, & \text{если } m \text{ нечетно;} \end{cases} \tag{8}$$

$$A'_{m(n+1)} = \begin{cases} A'_{(m-1)n}(m-1), A_{mn}(m-1), \ldots, A_{mn}1, A^R_{mn}0, & \text{если } m \text{ четно;} \\ A_{(m-1)n}(m-1), A^R_{mn}(m-1), \ldots, A_{mn}1, A^R_{mn}0, & \text{если } m \text{ нечетно.} \end{cases} \tag{9}$$

(Другими словами, мы начинаем либо с $A_{(m-1)n}(m-1)$, либо с $A'_{(m-1)n}(m-1)$, а затем по очереди используем либо $A^R_{mn}j$, либо $A_{mn}j$ при уменьшении j от $m-1$ до 0.)

Конечно, базовые случаи представляют собой простые одноэлементные списки

$$A_{1n} = A'_{1n} = \{0^n\} \qquad \text{и} \qquad A_{nn} = \{01\ldots(n{-}1)\}. \tag{10}$$

При таких определениях $\left\{{5 \atop 3}\right\} = 25$ разбиений $\{1,2,3,4,5\}$ на три блока представляют собой

$$\begin{gathered}00012,\ 00112,\ 01112,\ 01012,\ 01002,\ 01102,\ 00102,\\ 00122,\ 01122,\ 01022,\ 01222,\ 01212,\ 01202,\\ 01201,\ 01211,\ 01221,\ 01021,\ 01121,\ 00121,\\ 00120,\ 01120,\ 01020,\ 01220,\ 01210,\ 01200.\end{gathered} \tag{11}$$

(См. эффективную реализацию этого метода в упр. 17.)

В схеме Эрлиха (7) наиболее часто изменяются крайние справа цифры строки $a_1\ldots a_n$, но в схеме Раски основные изменения происходят ближе к левому краю. Однако в обоих случаях на каждом шаге изменяется только один элемент a_j, и это очень простое изменение: либо a_j изменяется на ±1, либо происходит переход между двумя крайними значениями 0 и $1+\max(a_1,\ldots,a_{j-1})$. При тех же ограничениях последовательность A'_{1n}, A'_{2n}, …, A'_{nn} проходит по *всем* разбиениям в возрастающем порядке количества блоков.

Количество разбиений множеств. Мы видели, что имеется 5 разбиений множества $\{1,2,3\}$ и 15 разбиений множества $\{1,2,3,4\}$. Быстрый способ вычисления этих значений был открыт Ч. С. Пирсом (C. S. Peirce), который представил в [*American Journal of Mathematics* **3** (1880), page 48] следующий числовой треугольник.

$$\begin{array}{cccccc} 1 & & & & & \\ 2 & 1 & & & & \\ 5 & 3 & 2 & & & \\ 15 & 10 & 7 & 5 & & \\ 52 & 37 & 27 & 20 & 15 & \\ 203 & 151 & 114 & 87 & 67 & 52 \end{array} \tag{12}$$

Здесь элементы ϖ_{n1}, ϖ_{n2}, …, ϖ_{nn} n-й строки подчиняются простому рекуррентному соотношению

$$\varpi_{nk} = \varpi_{(n-1)k} + \varpi_{n(k+1)} \text{ для } 1 \le k < n; \qquad \varpi_{nn} = \varpi_{(n-1)1} \text{ при } n > 1; \tag{13}$$

а $\varpi_{11} = 1$. Треугольник Пирса обладает многими замечательными свойствами, некоторые из них рассматриваются в упр. 26–31. Например, ϖ_{nk} представляет собой количество разбиений $\{1,2,\ldots,n\}$, у которых k — наименьший из блоков.

Элементы диагонали и первого столбца треугольника Пирса, которые говорят нам об общем количестве разбиений, широко известны как *числа Белла*, поскольку Э. Т. Белл (E. T. Bell) написал о них ряд важных статей [*AMM* **41** (1934), 411–419; *Annals of Math.* (2) **35** (1934), 258–277; **39** (1938), 539–557]. Следуя Луи Комте (Louis Comtet), мы будем обозначать числа Белла как ϖ_n, чтобы не путать их с числами Бернулли B_n. Вот несколько первых чисел Белла.

$$\begin{array}{rcccccccccccccc} n & = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ \varpi_n & = & 1 & 1 & 2 & 5 & 15 & 52 & 203 & 877 & 4140 & 21147 & 115975 & 678570 & 4213597 \end{array}$$

Обратите внимание на быстрый рост последовательности, но не столь быстрый, как $n!$; ниже мы докажем, что $\varpi_n = \Theta(n/\log n)^n$.

Числа Белла $\varpi_n = \varpi_{n1}$ при $n \geq 0$ должны удовлетворять рекуррентной формуле

$$\varpi_{n+1} \;=\; \varpi_n + \binom{n}{1}\varpi_{n-1} + \binom{n}{2}\varpi_{n-2} + \cdots \;=\; \sum_k \binom{n}{k}\varpi_{n-k}, \tag{14}$$

поскольку каждое разбиение множества $\{1, \ldots, n+1\}$ получается для некоторого k путем выбора k элементов из $\{1, \ldots, n\}$ для размещения в блоке, содержащем элемент $n+1$, с последующим разбиением оставшихся элементов ϖ_{n-k} способами. Это рекуррентное соотношение, найденное Ёшисуке Мацунагой (Yoshisuke Matsunaga) в XVIII веке (см. раздел 7.2.1.7), приводит к красивой производящей функции

$$\Pi(z) \;=\; \sum_{n=0}^{\infty} \varpi_n \frac{z^n}{n!} \;=\; e^{e^z-1}, \tag{15}$$

открытой В. А. Уитвортом (W. A. Whitworth) [*Choice and Chance*, 3rd edition (1878), 3.XXIV]. Если умножить обе части (14) на $z^n/n!$ и просуммировать по n, получится

$$\Pi'(z) = \sum_{n=0}^{\infty} \varpi_{n+1}\frac{z^n}{n!} = \Bigl(\sum_{k=0}^{\infty}\frac{z^k}{k!}\Bigr)\Bigl(\sum_{m=0}^{\infty}\varpi_m\frac{z^m}{m!}\Bigr) = e^z\Pi(z);$$

решением этого дифференциального уравнения при $\Pi(0) = 1$ является (15).

Числа ϖ_n много лет изучались в связи с их любопытными свойствами, связанными с данной формулой, задолго до того, как Уитворт указал их комбинаторную связь с разбиениями множеств. Например,

$$\varpi_n = \frac{n!}{e}\,[z^n]\,e^{e^z} = \frac{n!}{e}\,[z^n]\sum_{k=0}^{\infty}\frac{e^{kz}}{k!} = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^n}{k!} \tag{16}$$

[*Mat. Sbornik* **3** (1868), 62; **4** (1869), 39; G. Dobiński, *Archiv der Math. und Physik* **61** (1877), 333–336; **63** (1879), 108–110]. Кристиан Крамп (Christian Kramp), рассматривая разложение e^{e^z} в [*Der polynomische Lehrsatz*, ed. by C. F. Hindenburg (Leipzig: 1796), 112–113], привел два способа вычисления коэффициентов, а именно — либо с использованием (14), либо суммируя $p(n)$ членов, по одному для каждого обычного разбиения n. (См. формулу Арбогаста в упр. 1.2.5–21. Крамп, близко подошедший к открытию этой формулы, похоже, предпочитал свой метод, основанный на разбиениях, не понимая, что он требует более чем полиномиального времени при все бóльших и бóльших n; для коэффициента при z^{10} он вычислил значение 116015 вместо верного 115975.)

***Асимптотические оценки.** Изучить, насколько быстро с ростом n увеличивается значение ϖ_n, можно с использованием основных принципов теории комплексных вычетов: если степенной ряд $\sum_{k=0}^{\infty} a_k z^k$ сходится везде в области $|z| < r$, то

$$a_{n-1} \;=\; \frac{1}{2\pi i}\oint \frac{a_0 + a_1 z + a_2 z^2 + \cdots}{z^n}\,dz, \tag{17}$$

где интеграл берется по простому замкнутому пути, который обходит начало координат против часовой стрелки и остается внутри окружности $|z| = r$. Пусть $f(z) = \sum_{k=0}^{\infty} a_k z^{k-n}$ — подынтегральная функция. Мы можем выбрать любой описанный путь, но зачастую имеются специальные методы, применимые, когда путь проходит через точку z_0, в которой производная $f'(z_0)$ равна нулю, поскольку поблизости

Рис. 53. Поведение аналитической функции вблизи седловой точки.

этой точки

$$f(z_0+\epsilon e^{i\theta}) \;=\; f(z_0)+\frac{f''(z_0)}{2}\epsilon^2 e^{2i\theta}+O(\epsilon^3). \tag{18}$$

Если, например, $f(z_0)$ и $f''(z_0)$ действительны и положительны, скажем, $f(z_0)=u$ и $f''(z_0)=2v$, то эта формула гласит, что значение $f(z_0\pm\epsilon)$ приближенно равно $u+v\epsilon^2$, в то время как $f(z_0\pm i\epsilon)$ приближенно равно $u-v\epsilon^2$. Если z проходит от $z_0-i\epsilon$ до $z_0+i\epsilon$, то значение $f(z)$ возрастает до максимального значения u, после чего вновь снижается; однако большее значение $u+v\epsilon^2$ достигается функцией слева и справа от этого пути. Другими словами, альпинист, путешествующий по комплексной плоскости с рельефом, высота которого в точке z равна $\Re f(z)$, обнаружит "перевал" в точке z_0; рельеф здесь имеет вид седла (рис. 53). Полный интеграл от $f(z)$ одинаков для любого пути, но путь, не проходящий через седловую точку, будет не столь приятен, как проходящий через нее, так как потребует сокращения некоторых больши́х значений $f(z)$, чего можно избежать. Таким образом, лучше всего выбрать путь, проходящий через седловую точку z_0 в направлении возрастания мнимой части. Этот важный метод, разработанный П. Дебаем (P. Debye) [*Math. Annalen* **67** (1909), 535–558], так и называется — "метод седловой точки".

Познакомимся с методом седловой точки на примере, ответ к которому нам известен:

$$\frac{1}{(n-1)!} \;=\; \frac{1}{2\pi i}\oint \frac{e^z}{z^n}\,dz. \tag{19}$$

Наша цель — найти хорошее приближение значения интеграла в правой части при больши́х n. Работать с функцией $f(z)=e^z/z^n$ будет удобнее, если переписать ее как $e^{g(z)}$, где $g(z)=z-n\ln z$; тогда седловая точка достигается там, где значение $g'(z_0)=1-n/z_0$ равно нулю, т. е. в точке $z_0=n$. Если $z=n+it$, мы получим

$$g(z)=g(n)+\sum_{k=2}^{\infty}\frac{g^{(k)}(n)}{k!}(it)^k=n-n\ln n-\frac{t^2}{2n}+\frac{it^3}{3n^2}+\frac{t^4}{4n^3}-\frac{it^5}{5n^4}+\cdots,$$

поскольку $g^{(k)}(z)=(-1)^k(k-1)!\,n/z^k$ при $k\ge 2$. Проинтегрируем $f(z)$ по прямоугольному пути от $n-im$ через $n+im$, $-n+im$ и $-n-im$ назад к $n-im$:

$$\frac{1}{2\pi i}\oint\frac{e^z}{z^n}\,dz=\frac{1}{2\pi}\int_{-m}^{m}f(n+it)\,dt+\frac{1}{2\pi i}\int_{n}^{-n}f(t+im)\,dt$$
$$+\frac{1}{2\pi}\int_{m}^{-m}f(-n+it)\,dt+\frac{1}{2\pi i}\int_{-n}^{n}f(t-im)\,dt.$$

Ясно, что если выбрать $m = 2n$, то на трех последних участках пути $|f(z)| \le 2^{-n} f(n)$, поскольку $|e^z| = e^{\Re z}$ и $|z| \ge \max\bigl(|\Re z|, |\Im z|\bigr)$; так что мы остаемся с

$$\frac{1}{2\pi i}\oint \frac{e^z}{z^n}\,dz \;=\; \frac{1}{2\pi}\int_{-m}^{m} e^{g(n+it)}\,dt + O\Bigl(\frac{ne^n}{2^n n^n}\Bigr).$$

Теперь вернемся к методике, которой мы неоднократно пользовались ранее, например, для вывода формулы 5.1.4–(53): если $\hat f(t)$ — хорошее приближение $f(t)$ при $t \in A$ и если суммы $\sum_{t\in B}|f(t)|$ и $\sum_{t\in C}|\hat f(t)|$ малы, то $\sum_{t\in A\cup C}\hat f(t)$ представляет собой хорошее приближение для $\sum_{t\in A\cup B} f(t)$. Эта идея применима как к суммам, так и к интегралам. [Этот общий метод, разработанный Лапласом (Laplace) в 1782 году, часто называется “trading tails”*; см. *CMath* §9.4.] Если $|t| \le n^{1/2+\epsilon}$, имеем

$$\begin{aligned} e^{g(n+it)} &= \exp\Bigl(g(n) - \frac{t^2}{2n} + \frac{it^3}{3n^2} + \cdots\Bigr) \\ &= \frac{e^n}{n^n}\exp\Bigl(-\frac{t^2}{2n} + \frac{it^3}{3n^2} + \frac{t^4}{4n^3} + O(n^{5\epsilon-3/2})\Bigr) \\ &= \frac{e^n}{n^n}e^{-t^2/(2n)}\Bigl(1 + \frac{it^3}{3n^2} + \frac{t^4}{4n^3} - \frac{t^6}{18n^4} + O(n^{9\epsilon-3/2})\Bigr). \end{aligned}$$

При $|t| > n^{1/2+\epsilon}$ мы получаем

$$|e^{g(n+it)}| < |f(n + in^{1/2+\epsilon})| = \frac{e^n}{n^n}\exp\Bigl(-\frac{n}{2}\ln(1+n^{2\epsilon-1})\Bigr) = O\Bigl(\frac{e^{n-n^{2\epsilon}/2}}{n^n}\Bigr).$$

Кроме того, можно пренебречь неполной гамма-функцией

$$\int_{n^{1/2+\epsilon}}^{\infty} e^{-t^2/(2n)}t^k\,dt = 2^{(k-1)/2}n^{(k+1)/2}\,\Gamma\Bigl(\frac{k+1}{2}, \frac{n^{2\epsilon}}{2}\Bigr) = O(n^{O(1)}e^{-n^{2\epsilon}/2}).$$

Таким образом, можно применить метод Лапласа и получить приближение

$$\begin{aligned} \frac{1}{2\pi i}\oint \frac{e^z}{z^n}\,dz &= \frac{e^n}{2\pi n^n}\int_{-\infty}^{\infty} e^{-t^2/(2n)}\Bigl(1 + \frac{it^3}{3n^2} + \frac{t^4}{4n^3} - \frac{t^6}{18n^4} + O(n^{9\epsilon-3/2})\Bigr)dt \\ &= \frac{e^n}{2\pi n^n}\Bigl(I_0 + \frac{i}{3n^2}I_3 + \frac{1}{4n^3}I_4 - \frac{1}{18n^4}I_6 + O(n^{9\epsilon-3/2})\Bigr), \end{aligned}$$

где $I_k = \int_{-\infty}^{\infty} e^{-t^2/(2n)}t^k\,dt$. Конечно, $I_k = 0$ при нечетном k. В противном случае можно оценить значение I_k с помощью широко известного факта

$$\int_{-\infty}^{\infty} e^{-at^2}t^{2l}\,dt = \frac{\Gamma\bigl((2l+1)/2\bigr)}{a^{(2l+1)/2}} = \frac{\sqrt{2\pi}}{(2a)^{(2l+1)/2}}\prod_{j=1}^{l}(2j-1) \tag{20}$$

при $a > 0$; см. упр. 39. Собирая воедино все рассмотренные результаты, получаем для всех $\epsilon > 0$ асимптотическую оценку

$$\frac{1}{(n-1)!} = \frac{e^n}{\sqrt{2\pi}n^{n-1/2}}\Bigl(1 + 0 + \frac{3}{4n} - \frac{15}{18n} + O(n^{9\epsilon-2})\Bigr), \tag{21}$$

полностью согласующуюся с приближением Стирлинга, выведенным совершенно другим способом в 1.2.11.2–(19). Прочие члены в разложении $g(n+it)$ позволяют

* Дословно — “торговля хвостами”. — *Примеч. пер.*

доказать, что истинная ошибка в (21) составляет всего лишь $O(n^{-2})$, поскольку та же процедура дает асимптотический ряд общего вида $e^n/(\sqrt{2\pi}n^{n-1/2})\bigl(1+c_1/n+c_2/n^2+\cdots+c_m/n^m+O(n^{-m-1})\bigr)$ для всех m.

То, что мы занялись выводом данного результата, оправдывается важной технической деталью: функция $\ln z$ не является однозначной на пути интегрирования, поскольку возрастает на $2\pi i$ при движении вокруг начала координат. На самом деле этот факт лежит в основе механизма, который заставляет работать теорему о вычетах. Однако наш вывод корректен, поскольку неоднозначность логарифма не влияет на подынтегральную функцию $f(z)=e^z/z^n$ при целом n. Кроме того, если n не целое, можно немного изменить вывод, оставив его совершенно строгим, путем взятия интеграла (19) по пути, начинающемуся в $-\infty$, обходящему начало координат против часовой стрелки и возвращающемуся в $-\infty$. Это даст нам интеграл Ганкеля для гамма-функции 1.2.5–(17); таким образом можно вывести асимптотическую формулу

$$\frac{1}{\Gamma(x)} \;=\; \frac{1}{2\pi i}\oint \frac{e^z}{z^x}\,dz \;=\; \frac{e^x}{\sqrt{2\pi}x^{x-1/2}}\Bigl(1-\frac{1}{12x}+O(x^{-2})\Bigr), \tag{22}$$

справедливую для всех действительных чисел x при $x\to\infty$.

Итак, метод седловой точки, похоже, работает, хотя это не самый простой способ получения данного конкретного результата. Применим его теперь к выводу приближения чисел Белла:

$$\frac{\varpi_{n-1}}{(n-1)!}=\frac{1}{2\pi ie}\oint e^{g(z)}\,dz, \qquad g(z)=e^z-n\ln z. \tag{23}$$

Седловая точка нового подынтегрального выражения теперь находится в точке $z_0=\xi>0$, где

$$\xi e^{\xi} \;=\; n. \tag{24}$$

(В действительности следует писать $\xi(n)$, чтобы указать, что ξ зависит от n; но это излишне загромоздило бы следующие далее формулы.) Предположим на минутку, что добрый волшебник сказал нам значение ξ. Тогда мы могли бы взять интеграл по пути $z=\xi+it$ и получить

$$g(\xi+it)=e^{\xi}-n\Bigl(\ln\xi-\frac{(it)^2}{2!}\frac{\xi+1}{\xi^2}-\frac{(it)^3}{3!}\frac{\xi^2-2!}{\xi^3}-\frac{(it)^4}{4!}\frac{\xi^3+3!}{\xi^4}+\cdots\Bigr).$$

Путем интегрирования по соответствующему прямоугольному пути можно доказать, как мы делали это выше, что хорошим приближением интеграла в (23) является

$$\int_{-n^{\epsilon-1/2}}^{n^{\epsilon-1/2}} e^{g(\xi)-na_2t^2-nia_3t^3+na_4t^4+\cdots}\,dt, \qquad a_k=\frac{\xi^{k-1}+(-1)^k(k-1)!}{k!\,\xi^k}; \tag{25}$$

см. упр. 43. Замечая, что a_kt^k под знаком интеграла представляет собой $O(n^{k\epsilon-k/2})$, мы получаем асимптотическое разложение в виде

$$\varpi_{n-1}=\frac{e^{e^{\xi}-1}(n-1)!}{\xi^{n-1}\sqrt{2\pi n(\xi+1)}}\Bigl(1+\frac{b_1}{n}+\frac{b_2}{n^2}+\cdots+\frac{b_m}{n^m}+O\Bigl(\frac{\log n}{n}\Bigr)^{m+1}\Bigr), \tag{26}$$

где $(\xi+1)^{3k}b_k$ — полином степени $4k$ от ξ (см. упр. 44). Например,

$$b_1 = -\frac{2\xi^4-3\xi^3-20\xi^2-18\xi+2}{24(\xi+1)^3}; \tag{27}$$

$$b_2 = \frac{4\xi^8-156\xi^7-695\xi^6-696\xi^5+1092\xi^4+2916\xi^3+1972\xi^2-72\xi+4}{1152(\xi+1)^6}. \tag{28}$$

В (26) можно воспользоваться приближением Стирлинга (21) для доказательства того, что

$$\varpi_{n-1} = \exp\Bigl(n\Bigl(\xi-1+\frac{1}{\xi}\Bigr)-\xi-\frac{1}{2}\ln(\xi+1)-1-\frac{\xi}{12n}+O\Bigl(\frac{\log n}{n}\Bigr)^2\Bigr); \tag{29}$$

а в упр. 45 доказывается аналогичная формула:

$$\varpi_n = \exp\Bigl(n\Bigl(\xi-1+\frac{1}{\xi}\Bigr)-\frac{1}{2}\ln(\xi+1)-1-\frac{\xi}{12n}+O\Bigl(\frac{\log n}{n}\Bigr)^2\Bigr). \tag{30}$$

Следовательно, $\varpi_n/\varpi_{n-1} \approx e^\xi = n/\xi$. Точнее,

$$\frac{\varpi_{n-1}}{\varpi_n} = \frac{\xi}{n}\Bigl(1+O\Bigl(\frac{1}{n}\Bigr)\Bigr). \tag{31}$$

Но чему же равно асимптотическое значение ξ? Из определения (24) вытекает, что

$$\begin{aligned}\xi = \ln n - \ln\xi &= \ln n - \ln(\ln n - \ln\xi)\\ &= \ln n - \ln\ln n + O\Bigl(\frac{\log\log n}{\log n}\Bigr);\end{aligned} \tag{32}$$

мы можем идти по этому пути, как показано в упр. 49. Однако асимптотический ряд для ξ, полученный таким способом, никогда не даст точность лучше, чем $O(1/(\log n)^m)$ для все большего и большего значения m; таким образом, при умножении на n в формуле (29) для ϖ_{n-1} или в формуле (30) для ϖ_n мы получим огромную неточность.

Вот почему, если мы хотим использовать для вычисления хорошего приближения чисел Белла (29) или (30), наилучшая стратегия — начать с вычисления точного значения ξ, не прибегая к плохо сходящимся рядам. Метод Ньютона, рассматривавшийся в примечаниях перед описанием алгоритма 4.7N, дает эффективную итеративную схему

$$\xi_0 = \ln n, \qquad \xi_{k+1} = \frac{\xi_k}{\xi_k+1}(1+\xi_0-\ln\xi_k), \tag{33}$$

которая быстро сходится к точному значению. Например, для $n = 100$ пятая итерация дает значение

$$\xi_5 = 3.38563\,01402\,90050\,18488\,82443\,64529\,72686\,74917-, \tag{34}$$

верное до сорокового знака. Применение этого значения в (29) дает последовательные приближения

$$(1.6176088053\ldots, 1.6187421339\ldots, 1.6187065391\ldots, 1.6187060254\ldots)\times 10^{114}$$

Рис. 54. Числа Стирлинга $\left\{ {100 \atop m} \right\}$ имеют наибольшие значения при $m = 28$ и $m = 29$.

при учете соответственно членов 1, b_1/n, b_2/n^2, b_3/n^3; истинное значение ϖ_{99} представляет собой 115-значное целое число 16187060274460...20741.

Теперь, когда мы знаем количества разбиений множества ϖ_n, попробуем выяснить, сколько из них состоят ровно из m блоков. Оказывается, что почти все разбиения $\{1, \ldots, n\}$ имеют порядка $n/\xi = e^\xi$ блоков примерно с ξ элементами в блоке. Например, на рис. 54 показана гистограмма чисел Стирлинга $\left\{ {n \atop m} \right\}$ для $n = 100$; в этом случае $e^\xi \approx 29.54$.

Величину $\left\{ {n \atop m} \right\}$ можно исследовать, применив метод седловой точки к формуле 1.2.9–(23), которая гласит, что

$$\left\{ {n \atop m} \right\} = \frac{n!}{m!} [z^n] (e^z - 1)^m = \frac{n!}{m!} \frac{1}{2\pi i} \oint e^{m \ln(e^z - 1) - (n+1) \ln z} \, dz. \tag{35}$$

Пусть $\alpha = (n+1)/m$. Функция $g(z) = \alpha^{-1} \ln(e^z - 1) - \ln z$ имеет седловую точку $\sigma > 0$ при

$$\frac{\sigma}{1 - e^{-\sigma}} = \alpha. \tag{36}$$

Заметим, что $\alpha > 1$ при $1 \le m \le n$. Это значение σ можно получить как

$$\sigma = \alpha - \beta, \qquad \beta = T(\alpha e^{-\alpha}), \tag{37}$$

где T — "функция дерева" из 2.3.4.4–(30). Фактически β — значение в диапазоне от 0 до 1, для которого

$$\beta e^{-\beta} = \alpha e^{-\alpha}; \tag{38}$$

функция xe^{-x} возрастает от 0 до e^{-1} при увеличении x от 0 до 1, а затем вновь уменьшается до 0. Таким образом, β определяется однозначно, и мы имеем

$$e^\sigma = \frac{\alpha}{\beta}. \tag{39}$$

Все такие пары α и β можно получить с использованием обратных формул

$$\alpha = \frac{\sigma e^\sigma}{e^\sigma - 1}, \qquad \beta = \frac{\sigma}{e^\sigma - 1}; \tag{40}$$

например, значения $\alpha = \ln 4$ и $\beta = \ln 2$ соответствуют $\sigma = \ln 2$.

Можно, как ранее, показать, что интеграл в (35) асимптотически равен интегралу $e^{(n+1)g(z)} \, dz$ по пути $z = \sigma + it$. (См. упр. 58.) В упр. 56 доказывается, что

ряд Тейлора вблизи $z = \sigma$

$$g(\sigma + it) \;=\; g(\sigma) - \frac{t^2(1-\beta)}{2\sigma^2} - \sum_{k=3}^{\infty} \frac{(it)^k}{k!}\, g^{(k)}(\sigma) \tag{41}$$

обладает тем свойством, что

$$|g^{(k)}(\sigma)| < 2(k-1)!\,(1-\beta)/\sigma^k \qquad \text{для всех } k > 0. \tag{42}$$

Таким образом, можно легко убрать множитель $N = (n+1)(1-\beta)$ из степенного ряда $(n+1)g(z)$, и метод седловой точки приведет к формуле

$$\left\{ {n \atop m} \right\} = \frac{n!}{m!} \frac{1}{(\alpha-\beta)^{n-m}\beta^m\sqrt{2\pi N}} \left(1 + \frac{b_1}{N} + \frac{b_2}{N^2} + \cdots + \frac{b_l}{N^l} + O\left(\frac{1}{N^{l+1}}\right)\right) \tag{43}$$

при $N \to \infty$, где $(1-\beta)^{2k} b_k$ — полином от α и β. (Величина $(\alpha-\beta)^{n-m}\beta^m$ в знаменателе проистекает из того факта, что $(e^\sigma - 1)^m/\sigma^n = (\alpha/\beta - 1)^m/(\alpha-\beta)^n$, в соответствии с (37) и (39).) Например,

$$b_1 \;=\; \frac{6 - \beta^3 - 4\alpha\beta^2 - \alpha^2\beta}{8(1-\beta)} - \frac{5(2-\beta^2-\alpha\beta)^2}{24(1-\beta)^2}. \tag{44}$$

В упр. 57 доказывается, что $N \to \infty$ тогда и только тогда, когда $n - m \to \infty$. Асимптотическое разложение для $\left\{ {n \atop m} \right\}$, аналогичное (43), но несколько более сложное, было впервые получено Лео Мозером (Leo Moser) и Максом Виманом (Max Wyman), *Duke Math. J.* **25** (1957), 29–43.

Формула (43) выглядит немного страшновато из-за того, что она создана с учетом применимости для всего диапазона количества блоков m. Значительное упрощение формулы возможно при относительно малых и относительно больших значениях m (см. упр. 60 и 61); однако эти упрощения не дают точных результатов в важных случаях, когда $\left\{ {n \atop m} \right\}$ имеет наибольшее значение. Рассмотрим подробнее эти важные случаи, определяющие форму пика на рис. 54.

Пусть $\xi e^\xi = n$, как в (24), и положим $m = \exp(\xi + r/\sqrt{n}) = ne^{r/\sqrt{n}}/\xi$; мы будем считать, что $|r| \le n^\epsilon$, а значит, m близко к e^ξ. Старший член (43) можно переписать как

$$\frac{n!}{m!} \frac{1}{(\alpha-\beta)^{n-m}\beta^m\sqrt{2\pi(n+1)(1-\beta)}} \;=$$
$$\frac{m^n}{m!} \frac{(n+1)!}{(n+1)^{n+1}} \frac{e^{n+1}}{\sqrt{2\pi(n+1)}} \left(1 - \frac{\beta}{\alpha}\right)^{m-n} \frac{e^{-\beta m}}{\sqrt{1-\beta}}, \tag{45}$$

и приближение Стирлинга для $(n+1)!$ приводит к очевидным сокращениям в этом выражении. С помощью компьютерной алгебры находим

$$\frac{m^n}{m!} \;=\; \frac{1}{\sqrt{2\pi}} \exp\left(n\left(\xi - 1 + \frac{1}{\xi}\right) - \frac{1}{2}\left(\xi + r^2 + \frac{r^2}{\xi}\right)\right.$$
$$\left. - \left(\frac{r}{2} + \frac{r^3}{6} + \frac{r^3}{3\xi}\right)\frac{1}{\sqrt{n}} + O(n^{4\epsilon-1})\right);$$

а соответствующие величины, связанные с α и β, представляют собой

$$\frac{\beta}{\alpha} = \frac{\xi}{n} + \frac{r\xi^2}{n\sqrt{n}} + O(\xi^3 n^{2\epsilon-2});$$

$$e^{-\beta m} = \exp\Bigl(-\xi - \frac{r\xi^2}{\sqrt{n}} + O(\xi^3 n^{2\epsilon-1})\Bigr);$$

$$\Bigl(1-\frac{\beta}{\alpha}\Bigr)^{m-n} = \exp\Bigl(\xi - 1 + \frac{r(\xi^2-\xi-1)}{\sqrt{n}} + O(\xi^3 n^{2\epsilon-1})\Bigr).$$

Таким образом, окончательный результат имеет вид

$$\left\{ {n \atop e^{\xi+r/\sqrt{n}}} \right\} = \frac{1}{\sqrt{2\pi}} \exp\Bigl(n\Bigl(\xi - 1 + \frac{1}{\xi}\Bigr) - \frac{\xi}{2} - 1 - \frac{\xi+1}{2\xi}\Bigl(r + \frac{3\xi(2\xi+3)+(\xi+2)r^2}{6(\xi+1)\sqrt{n}}\Bigr)^2 + O(\xi^3 n^{4\epsilon-1})\Bigr). \quad (46)$$

Возведенное в квадрат выражение в последней строке равно нулю, когда

$$r = -\frac{\xi(2\xi+3)}{2(\xi+1)\sqrt{n}} + O(\xi^2 n^{-3/2});$$

таким образом, максимум достигается, когда количество блоков равно

$$m = \frac{n}{\xi} - \frac{3+2\xi}{2+2\xi} + O\Bigl(\frac{\xi}{n}\Bigr). \quad (47)$$

Сравнивая (46) и (30), мы видим, что наибольшее число Стирлинга $\left\{ {n \atop m} \right\}$ для заданного значения n приблизительно равно $\xi\varpi_n/\sqrt{2\pi n}$.

Метод седловой точки применим к существенно более сложным задачам, чем рассмотренные в этом разделе. Превосходное описание ряда эффективных методов можно найти в книгах N. G. de Bruijn, *Asymptotic Methods in Analysis* (1958), главы 5 и 6; F. W. J. Olver, *Asymptotics and Special Functions* (1974), глава 4; R. Wong, *Asymptotic Approximations of Integrals* (2001), главы 2 и 7.

***Случайные разбиения множеств.** Размеры блоков в разбиении $\{1,\ldots,n\}$ представляют собой обычное разбиение числа n. Следовательно, нас может заинтересовать, к какому виду разбиений оно может относиться. На рис. 50 в разделе 7.2.1.4 показан результат суперпозиции диаграмм Феррерса для всех $p(25) = 1958$ разбиений числа 25; эти разбиения стремятся к симметричной кривой из формулы 7.2.1.4–(49). В отличие от этого на рис. 55 показан результат суперпозиции соответствующих диаграмм для всех $\varpi_{25} \approx 4.6386 \times 10^{18}$ разбиений множества $\{1,\ldots,25\}$. Очевидно, что "форма" случайного разбиения множества существенно отличается от таковой для случайного разбиения целого числа.

Это изменение связано с тем, что некоторые разбиения целого числа появляются в виде размеров блоков разбиения множества только в редких случаях, в то время как другие весьма распространены. Например, разбиение $n = 1+1+\cdots+1$ возможно только единственным способом, в то время как для четного числа n разбиение $n = 2+2+\cdots+2$ может реализоваться $(n-1)(n-3)\ldots(1)$ способами. Для $n = 25$ разбиение целого числа

$$25 = 4+4+3+3+3+2+2+2+1+1$$

Рис. 55. Форма случайного разбиения множества при $n = 25$.

в действительности возникает более чем в 2% случаев от общего количества разбиений множества. (Это конкретное разбиение наиболее частое в случае $n = 25$. Ответ к упр. 1.2.5–21 гласит, что ровно

$$\frac{n!}{c_1!\,1!^{c_1}\,c_2!\,2!^{c_2}\ldots c_n!\,n!^{c_n}} \tag{48}$$

разбиений множества соответствуют разбиению целого числа $n = c_1 \cdot 1 + c_2 \cdot 2 + \cdots + c_n \cdot n$.)

Можно легко определить среднее количество k-блоков в случайном разбиении множества $\{1, \ldots, n\}$: если записать все ϖ_n возможных разбиений, каждый конкретный k-элементный блок встретится ровно ϖ_{n-k} раз. Следовательно, среднее значение составляет

$$\binom{n}{k}\frac{\varpi_{n-k}}{\varpi_n}\,. \tag{49}$$

Расширение (31), как доказывается в упр. 64, дает

$$\frac{\varpi_{n-k}}{\varpi_n} \;=\; \left(\frac{\xi}{n}\right)^k\left(1 + \frac{k\xi(k\xi+k+1)}{2(\xi+1)^2 n} + O\left(\frac{k^3}{n^2}\right)\right) \qquad \text{если } k \le n^{2/3}, \tag{50}$$

где ξ определено в (24). Следовательно, если, скажем, $k \le n^\epsilon$, формула (49) упрощается до

$$\frac{n^{\underline{k}}}{k!}\left(\frac{\xi}{n}\right)^k\left(1 + O\left(\frac{1}{n}\right)\right) \;=\; \frac{\xi^k}{k!}\left(1 + O(n^{2\epsilon-1})\right). \tag{51}$$

В среднем имеется около ξ блоков размером 1, $\xi^2/2!$ блоков размером 2 и т. д.

Дисперсия этих величин мала (см. упр. 65), и оказывается, что случайное разбиение ведет себя, по сути, так, как если бы количество k-блоков подчинялось распределению Пуассона со средним $\xi^k/k!$. Гладкая кривая на рис. 55 проходит через точки $\bigl(f(k), k\bigr)$ в координатах диаграммы Феррерса, где

$$f(k) \;=\; \xi^{k+1}/(k+1)! + \xi^{k+2}/(k+2)! + \xi^{k+3}/(k+3)! + \cdots \tag{52}$$

представляет собой приближенное расстояние от верхней линии, соответствующей размеру блока $k \ge 0$. (Эта кривая при бóльших n становится все более близкой к вертикали.)

Наибольший блок имеет тенденцию содержать примерно $e\xi$ элементов. Кроме того, вероятность того, что блок, содержащий элемент 1, имеет размер менее $\xi+a\sqrt{\xi}$, приближается к вероятности того, что нормальное отклонение не превышает a. [См. John Haigh, *J. Combinatorial Theory* **A13** (1972), 287–295; V. N. Sachkov, *Probabilistic Methods in Combinatorial Analysis* (1997), глава 4 (перевод книги на русском языке, изданной в 1978 г.); Yu. Yakubovich, *J. Mathematical Sciences* **87** (1997), 4124–4137, перевод статьи на русском языке, опубликованной в 1995 г.; B. Pittel, *J. Combinatorial Theory* **A79** (1997), 326–359.]

Красивый способ генерации случайного разбиения множества $\{1,2,\ldots,n\}$ был предложен А. Й. Стамом (A. J. Stam) в *Journal of Combinatorial Theory* **A35** (1983), 231–240. Пусть M — случайное целое число, принимающее значение m с вероятностью

$$p_m = \frac{m^n}{e\,m!\,\varpi_n}; \tag{53}$$

в силу (16) сумма всех этих вероятностей равна 1. После того как M выбрано, генерируем случайный n-кортеж $X_1X_2\ldots X_n$, где каждое X_j независимо равномерно распределено между 0 и $M-1$. Тогда в разбиении $i\equiv j$ тогда и только тогда, когда $X_i=X_j$. Эта процедура работает корректно, поскольку каждое разбиение множества с k блоками получается с вероятностью $\sum_{m\ge0}(m^{\underline{k}}/m^n)p_m=1/\varpi_n$.

Например, при $n=25$ имеем

$p_4\approx.00000372$	$p_9\approx.15689865$	$p_{14}\approx.04093663$	$p_{19}\approx.00006068$
$p_5\approx.00019696$	$p_{10}\approx.21855285$	$p_{15}\approx.01531445$	$p_{20}\approx.00001094$
$p_6\approx.00313161$	$p_{11}\approx.21526871$	$p_{16}\approx.00480507$	$p_{21}\approx.00000176$
$p_7\approx.02110279$	$p_{12}\approx.15794784$	$p_{17}\approx.00128669$	$p_{22}\approx.00000026$
$p_8\approx.07431024$	$p_{13}\approx.08987171$	$p_{18}\approx.00029839$	$p_{23}\approx.00000003$

Всеми остальными вероятностями можно пренебречь. Так, в большинстве случаев можно получить случайное разбиение 25 элементов, рассматривая случайное 25-значное число в системе счисления с основанием 9, 10, 11 или 12. Число M можно сгенерировать с помощью метода 3.4.1–(3); оно имеет тенденцию быть близким к $n/\xi=e^{\xi}$ (см. упр. 67).

***Разбиения мультимножеств.** Разбиения целых чисел и разбиения множеств представляют собой крайние случаи существенно более общей задачи — разбиений мультимножеств. Действительно, разбиение n, по сути, — не что иное, как разбиение мультимножества $\{1,1,\ldots,1\}$, состоящего из n единиц.

С этой точки зрения имеется, по сути, $p(n)$ типов мультимножеств с n элементами. Например, при $n=4$ имеется пять различных случаев разбиения мультимножества:

$$\begin{gathered}
1234,\ 123|4,\ 124|3,\ 12|34,\ 12|3|4,\ 134|2,\ 13|24,\ 13|2|4,\\
14|23,\ 14|2|3,\ 1|234,\ 1|23|4,\ 1|24|3,\ 1|2|34,\ 1|2|3|4;\\
1123,\ 112|3,\ 113|2,\ 11|23,\ 11|2|3,\ 123|1,\ 12|13,\ 12|1|3,\ 13|1|2,\ 1|1|23,\ 1|1|2|3;\\
1122,\ 112|2,\ 11|22,\ 11|2|2,\ 122|1,\ 12|12,\ 12|1|2,\ 1|1|22,\ 1|1|2|2;\\
1112,\ 111|2,\ 112|1,\ 11|12,\ 11|1|2,\ 12|1|1,\ 1|1|1|2;\\
1111,\ 111|1,\ 11|11,\ 11|1|1,\ 1|1|1|1.
\end{gathered} \tag{54}$$

Когда мультимножество содержит m различных элементов, n_1 первого вида, n_2 второго, $\ldots$, n_m последнего, общее количество разбиений мы обозначаем как $p(n_1, n_2, \ldots, n_m)$. Примеры в (54) показывают, что

$$p(1,1,1,1) = 15, \quad p(2,1,1) = 11, \quad p(2,2) = 9, \quad p(3,1) = 7, \quad p(4) = 5. \tag{55}$$

Разбиения с $m = 2$ часто называют биразбиениями, с $m = 3$ — триразбиениями, а в общем случае эти комбинаторные объекты известны как *мультиразбиения*. Изучение мультиразбиений было начато много лет назад П. А. Мак-Мэганом (P. A. MacMahon) [*Philosophical Transactions* **181** (1890), 481–536; **217** (1917), 81–113; *Proc. Cambridge Philos. Soc.* **22** (1925), 951–963]; однако предмет изучения настолько обширен, что в нем и поныне остается множество нерешенных вопросов. В оставшейся части данного раздела и упражнениях к нему мы бегло ознакомимся с некоторыми наиболее интересными и поучительными аспектами этой теории.

Прежде всего стоит отметить, что мультиразбиения, по сути, являются разбиениями *векторов* с неотрицательными целыми компонентами, а именно способами разложения такого вектора на сумму векторов с теми же свойствами. Например, девять разбиений $\{1,1,2,2\}$, перечисленных в (54), представляют собой не что иное, как девять разбиений двухкомпонентного вектора-столбца $\binom{2}{2}$, а именно

$$\begin{matrix}2\\2\end{matrix},\quad \begin{matrix}2\,0\\1\,1\end{matrix},\quad \begin{matrix}2\,0\\0\,2\end{matrix},\quad \begin{matrix}2\,0\,0\\0\,1\,1\end{matrix},\quad \begin{matrix}1\,1\\2\,0\end{matrix},\quad \begin{matrix}1\,1\\1\,1\end{matrix},\quad \begin{matrix}1\,1\,0\\1\,0\,1\end{matrix},\quad \begin{matrix}1\,1\,0\\0\,0\,2\end{matrix},\quad \begin{matrix}1\,1\,0\,0\\0\,0\,1\,1\end{matrix}. \tag{56}$$

Для краткости здесь опущены знаки $+$, как и в случае одномерных разбиений целых чисел. Каждое разбиение можно записать в каноническом виде, если перечислить его части в невозрастающем лексикографическом порядке.

Для генерации разбиений любого заданного мультимножества достаточно простого алгоритма. В приведенной далее процедуре мы представляем разбиения в стеке, который содержит тройки элементов (c, u, v), где c обозначает номер компонента, $u > 0$ — пока еще не разбитый остаток в компоненте c, а v — c-й компонент текущей части, где $0 \le v \le u$. В действительности тройки для удобства располагаются в трех массивах, $(c_0, c_1, \ldots)$, $(u_0, u_1, \ldots)$ и $(v_0, v_1, \ldots)$, а кроме того, поддерживается массив "кадров стека" $(f_0, f_1, \ldots)$, так что $(l+1)$-й вектор разбиения состоит из элементов от f_l до $f_{l+1} - 1$ в массивах c, u и v. Например, биразбиение $\begin{smallmatrix}3&2&2&1&1&0&0\\1&2&0&1&1&3&1\end{smallmatrix}$ представлено следующими массивами.

j	0	1	2	3	4	5	6	7	8	9	10	11	
c_j	1	2	1	2	1	2	1	2	1	2	2	2	
u_j	9	9	6	8	4	6	2	6	1	5	4	1	
v_j	3	1	2	2	2	0	1	1	1	1	3	1	
	$f_0 = 0$		$f_1 = 2$		$f_2 = 4$		$f_3 = 6$		$f_4 = 8$		$f_5 = 10$	$f_6 = 11$	$f_7 = 12$

(57)

Алгоритм M (*Мультиразбиения в убывающем лексикографическом порядке*). Для данного мультимножества $\{n_1 \cdot 1, \ldots, n_m \cdot m\}$ этот алгоритм посещает все его разбиения с использованием описанных выше массивов $f_0 f_1 \ldots f_n$, $c_0 c_1 \ldots c_{mn}$, $u_0 u_1 \ldots u_{mn}$ и $v_0 v_1 \ldots v_{mn}$, где $n = n_1 + \cdots + n_m$. Полагаем, что $m > 0$ и $n_1, \ldots, n_m > 0$.

M1. [Инициализация.] Установить $c_j \leftarrow j+1$ и $u_j \leftarrow v_j \leftarrow n_{j+1}$ для $0 \le j < m$; установить также $f_0 \leftarrow a \leftarrow l \leftarrow 0$ и $f_1 \leftarrow b \leftarrow m$. (На последующих шагах текущий кадр стека пробегает от a до $b-1$ включительно.)

M2. [Вычитание v из u.] (Здесь мы хотим найти все разбиения вектора u в текущем кадре на части, лексикографически $\le v$. Сначала мы используем само v.) Установить $j \leftarrow a$, $k \leftarrow b$ и $x \leftarrow 0$. Затем, пока $j < b$, выполнять следующее: установить $u_k \leftarrow u_j - v_j$. Если $u_k = 0$, установить $x \leftarrow 1$ и $j \leftarrow j+1$. В противном случае при $x = 0$ установить $c_k \leftarrow c_j$, $v_k \leftarrow \min(v_j, u_k)$, $x \leftarrow [u_k < v_j]$, $k \leftarrow k+1$, $j \leftarrow j+1$. В противном случае установить $c_k \leftarrow c_j$, $v_k \leftarrow u_k$, $k \leftarrow k+1$, $j \leftarrow j+1$. (Заметим, что $x = [v$ изменено$]$.)

M3. [Внесение в стек, если не нуль.] Если $k > b$, установить $a \leftarrow b$, $b \leftarrow k$, $l \leftarrow l+1$, $f_{l+1} \leftarrow b$ и вернуться к шагу M2.

M4. [Посещение разбиения.] Посетить разбиение, представленное $l+1$ векторами, находящимися в стеке. (Для $0 \le k \le l$ вектор имеет v_j в компоненте c_j, $f_k \le j < f_{k+1}$.)

M5. [Уменьшение v.] Установить $j \leftarrow b-1$; пока $v_j = 0$, устанавливать $j \leftarrow j-1$. Затем, если $j = a$ и $v_j = 1$, перейти к шагу M6. В противном случае установить $v_j \leftarrow v_j - 1$ и $v_k \leftarrow u_k$ для $j < k < b$. Вернуться к шагу M2.

M6. [Возврат.] Завершить работу алгоритма, если $l = 0$. В противном случае установить $l \leftarrow l-1$, $b \leftarrow a$, $a \leftarrow f_l$ и вернуться к шагу M5. ▮

Ключевым в этом алгоритме является шаг M2, который уменьшает текущий остаточный вектор u на наибольшую разрешенную часть, v; этот шаг также при необходимости уменьшает v до лексикографически наибольшего вектора $\le v$, который меньше или равен новому остаточному значению в каждом компоненте. (См. упр. 68.)

Завершим этот раздел рассмотрением интересной связи между мультиразбиениями и процедурой сортировки начиная с младших цифр из алгоритма поразрядной сортировки (алгоритм 5.2.5R). Легче всего понять идею путем рассмотрения конкретного примера. Взгляните на табл. 1, где шаг (0) представляет собой девять четырехкомпонентных векторов-столбцов в лексикографическом порядке. Для идентификации в строке ниже приведены их порядковые номера ①–⑨. Шаг (1) выполняет устойчивую сортировку векторов, начиная с расположения четвертого (младшего) компонента в порядке уменьшения; аналогично шаги (2)–(4) выполняют устойчивую сортировку третьей, второй и первой строк. Теория поразрядной сортировки гласит, что таким образом восстанавливается исходный лексикографический порядок.

Предположим, что последовательности порядковых номеров после этих операций устойчивой сортировки представляют собой соответственно α_4, $\alpha_3\alpha_4$, $\alpha_2\alpha_3\alpha_4$ и $\alpha_1\alpha_2\alpha_3\alpha_4$, где все α являются перестановками. В табл. 1 значения α_4, α_3, α_2 и α_1 показаны в скобках. А теперь перейдем к главному: где бы перестановка α_j не имела спуск, числа в строке j после сортировки также должны иметь спуск, поскольку сортировка устойчива. (В таблице эти спуски помечены символами "$\wedge$".) Например, там, где в α_3 за 8 идет 7, в строке 3 за 5 идет 3. Следовательно, элементы $a_1 \ldots a_9$ в строке 3 после шага (2) не являются произвольным разбиением их суммы;

Таблица 1

Поразрядная сортировка и мультиразбиения

Шаг (0): исходное разбиение	Шаг (1): сортировка строки 4	Шаг (2): сортировка строки 3
6 5 5 4 3 2 1 0 0	0 6 4 3 5 0 5 2 1	0 6 5 2 5 1 4 3 0
3 2 1 0 4 5 6 4 2	2 3 0 4 2 4 1 5 6	2 3 2 5 1 6 0 4 4
6 6 3 1 1 5 2 0 7	7 6 1 1 6 0 3 5 2	7 6 6 5$_\wedge$3 2$_\wedge$1 1 0
4 2 1 3 3 1 1 2 5	5$_\wedge$4 3 3$_\wedge$2 2$_\wedge$1 1 1	5 4 2 1 1 1 3 3 2
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨	⑨ ① ④ ⑤ ② ⑧ ③ ⑥ ⑦	⑨ ① ② ⑥ ③ ⑦ ④ ⑤ ⑧
	$\alpha_4 = (\,9_\wedge 1\ \ 4\ \ 5_\wedge 2\ \ 8_\wedge 3\ \ 6\ \ 7\,)$	$\alpha_3 = (\,1\ \ 2\ \ 5\ \ 8_\wedge 7\ \ 9_\wedge 3\ \ 4\ \ 6\,)$

Шаг (3): сортировка строки 2	Шаг (4): сортировка строки 1
1 2 3 0 6 0 5 5 4	6 5 5 4$_\wedge$3$_\wedge$2$_\wedge$1 0 0
6$_\wedge$5 4 4$_\wedge$3$_\wedge$2 2 1 0	3 2 1 0 4 5 6 4 2
2 5 1 0 6 7 6 3 1	6 6 3 1 1 5 2 0 7
1 1 3 2 4 5 2 1 3	4 2 1 3 3 1 1 2 5
⑦ ⑥ ⑤ ⑧ ① ⑨ ② ③ ④	① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨
$\alpha_2 = (\,6_\wedge 4\ \ 8\ \ 9_\wedge 2_\wedge 1\ \ 3\ \ 5\ \ 7\,)$	$\alpha_1 = (\,5\ \ 7\ \ 8\ \ 9_\wedge 3_\wedge 2_\wedge 1\ \ 4\ \ 6\,)$

они должны удовлетворять условиям

$$a_1 \ge a_2 \ge a_3 \ge a_4 > a_5 \ge a_6 > a_7 \ge a_8 \ge a_9. \tag{58}$$

Однако числа $(a_1-2, a_2-2, a_3-2, a_4-2, a_5-1, a_6-1, a_7, a_8, a_9)$ образуют, по сути, произвольное разбиение исходной суммы минус $(4+6)$. Величина уменьшения, $4+6$, представляет собой сумму индексов, где наблюдается спуск; это число является тем, что в разделе 5.1.1 называлось индексом α_3 и обозначалось $\operatorname{ind}\alpha_3$.

Итак, мы видим, что любое разбиение m-компонентного числа не более чем на r частей (с дополнительными нулями, добавленными для того, чтобы количество столбцов было равно r) можно закодировать как последовательность перестановок $\alpha_1, \ldots, \alpha_m$ множества $\{1, \ldots, r\}$, такую, что произведение $\alpha_1 \ldots \alpha_m$ представляет собой тождественную перестановку, вместе с последовательностью обычных одномерных разбиений чисел $(n_1 - \operatorname{ind}\alpha_1, \ldots, n_m - \operatorname{ind}\alpha_m)$ не более чем на r частей. Например, векторы в табл. 1 представляют разбиение $(26, 27, 31, 22)$ на 9 частей; перестановки $\alpha_1, \ldots, \alpha_4$ приведены в таблице, и мы имеем $(\operatorname{ind}\alpha_1, \ldots, \operatorname{ind}\alpha_4) = (15, 10, 10, 11)$; соответственно разбиения представляют собой

$$26-15 = (322111100), \qquad 27-10 = (332222210),$$
$$31-10 = (544321110), \qquad 22-11 = (221111111).$$

И обратно: любые такие перестановки и разбиения дают мультиразбиения $(n_1, \ldots, n_m)$. Если r и m малы, при перечислении или доказательствах свойств мультиразбиений, в особенности биразбиений, может оказаться полезным рассмотрение $r!^{m-1}$ последовательностей одномерных разбиений. [См. Basil Gordon, *J. London Math. Soc.* **38** (1963), 459–464.]

Неплохой обзор ранних работ, посвященных изучению разбиений на различные части и/или на строго положительные части, можно найти в статье M. S. Cheema and T. S. Motzkin, *Proc. Symp. Pure Math.* **19** (Amer. Math. Soc., 1971), 39–70.

Упражнения

1. [*20*] (Д. Хатчинсон (G. Hutchinson).) Покажите, что простое изменение алгоритма H позволяет генерировать все разбиения $\{1, \ldots, n\}$ на *не более* чем r блоков для заданных n и $r \geq 2$.

▶ **2.** [*22*] При использовании разбиений множества на практике нам часто требуется связать вместе элементы каждого блока. Соответственно, удобно иметь массив связей $l_1 \ldots l_n$ и массив заголовков $h_1 \ldots h_t$, так что элементами j-го блока t-блочного разбиения являются $i_1 > \cdots > i_k$, где

$$i_1 = h_j, \quad i_2 = l_{i_1}, \quad \ldots, \quad i_k = l_{i_{k-1}} \quad \text{и} \quad l_{i_k} = 0.$$

Например, представление 137|25|489|6 должно иметь $t = 4$, $l_1 \ldots l_9 = 001020348$ и $h_1 \ldots h_4 = 7596$.

Разработайте вариацию алгоритма H, которая генерирует разбиения с использованием описанного представления.

3. [*M23*] Каково миллионное разбиение множества $\{1, \ldots, 12\}$, сгенерированное алгоритмом H?

▶ **4.** [*21*] Обозначим через $\rho(x_1 \ldots x_n)$ для произвольной строки $x_1 \ldots x_n$ ограниченно растущую строку, которая соответствует отношению эквивалентности $j \equiv k \iff x_j = x_k$. Классифицируем каждое пятибуквенное английское слово из Stanford GraphBase путем применения функции ρ; например, $\rho(\texttt{tooth}) = 01102$. Сколько из 52 разбиений пятиэлементного множества представимо таким способом английскими словами? Каковы наиболее распространенные слова каждого типа?

5. [*22*] Угадайте очередные элементы двух последовательностей: (а) 0, 1, 1, 1, 12, 12, 12, 12, 12, 12, 100, 121, 122, 123, 123, ...; (б) 0, 1, 12, 100, 112, 121, 122, 123, ...

▶ **6.** [*25*] Предложите алгоритм генерации всех разбиений множества $\{1, \ldots, n\}$, в которых имеется ровно c_1 блоков размера 1, c_2 блоков размера 2 и т. д.

7. [*M20*] Сколько перестановок $a_1 \ldots a_n$ множества $\{1, \ldots, n\}$ обладают тем свойством, что из $a_{k-1} > a_k > a_j$ вытекает $j > k$?

8. [*20*] Предложите способ генерации всех перестановок множества $\{1, \ldots, n\}$, которые при проходе слева направо имеют ровно m минимумов.

9. [*M20*] Сколько ограниченно растущих строк $a_1 \ldots a_n$ содержат в точности k_j вхождений j для заданных целых значений k_0, k_1, ..., k_{n-1}?

10. [*25*] *Полупомеченным* (*semilabeled*) называется ориентированное дерево, листья которого помечены целыми числами $\{1, \ldots, k\}$, но прочие узлы остаются непомеченными. Так, имеется 15 полупомеченных деревьев с 5 вершинами.

Найдите взаимно однозначное соответствие между разбиениями множества $\{1, \ldots, n\}$ и полупомеченными деревьями с $n + 1$ вершинами.

▶ **11.** [*28*] Из раздела 7.2.1.2 мы узнали, что знаменитая головоломка Дьюдени `send+more` = `money` представляет собой "чистый" буквометик, т. е. буквометик с единственным решением. Эта головоломка соответствует разбиению множества на 13 позиций цифр, ограниченно растущая строка которого $\rho(\texttt{sendmoremoney})$ представляет собой 0123456145217. Можно заинтересоваться вопросом, насколько везучим нужно быть, чтобы встретиться с такой конструкцией? Сколько ограниченно растущих строк длиной 13 определяют чистые буквометики вида $a_1a_2a_3a_4 + a_5a_6a_7a_8 = a_9a_{10}a_{11}a_{12}a_{13}$?

12. [*M31*] (*Решетка разбиений.*) Если Π и Π' представляют собой разбиения одного и того же множества, мы записываем $\Pi \preceq \Pi'$, если $x \equiv y$ (по модулю Π) всегда, когда $x \equiv y$ (по модулю Π'). Другими словами, $\Pi \preceq \Pi'$ означает, что Π' представляет собой "утончение" Π, получаемое путем разбиения нескольких (возможно, ни одного) блоков последнего; Π, в свою очередь, является "огрублением" или *коалесценцией* Π', получаемой путем объединения вместе нескольких (возможно, ни одного) блоков. Это частичное упорядочение, как легко видеть, представляет собой решетку, причем $\Pi \vee \Pi'$ является наибольшим общим утончением Π и Π', а $\Pi \wedge \Pi'$ — наименьшим общим огрублением. Например, если представить разбиения ограниченно растущими строками $a_1a_2a_3a_4$, то соответствующая решетка для разбиений множества $\{1, 2, 3, 4\}$ имеет следующий вид.

Пути, ведущие на этой диаграмме вверх, идут от разбиения к его утончениям. Разбиения на t блоков находятся на уровне t снизу, а их потомки образуют решетку разбиений множества $\{1, \ldots, t\}$.

а) Поясните, как вычислить $\Pi \vee \Pi'$ для заданных $a_1 \ldots a_n$ и $a'_1 \ldots a'_n$.
б) Поясните, как вычислить $\Pi \wedge \Pi'$ для заданных $a_1 \ldots a_n$ и $a'_1 \ldots a'_n$.
в) Когда в этой решетке Π' покрывает Π? (См. упр. 7.2.1.4–55.)
г) Если Π имеет t блоков с размерами $s_1, \ldots, s_t$, то сколько разбиений оно покрывает?
д) Если Π имеет t блоков с размерами $s_1, \ldots, s_t$, то сколькими разбиениями оно покрывается?
е) Истинно или ложно следующее утверждение? Если $\Pi \vee \Pi'$ покрывает Π, то Π' покрывает $\Pi \wedge \Pi'$.
ж) Истинно или ложно следующее утверждение? Если Π' покрывает $\Pi \wedge \Pi'$, то $\Pi \vee \Pi'$ покрывает Π.
з) Пусть $b(\Pi)$ обозначает количество блоков Π. Докажите, что

$$b(\Pi) + b(\Pi') \;\le\; b(\Pi \vee \Pi') + b(\Pi \wedge \Pi').$$

13. [*M28*] (Стефен К. Милн (Stephen C. Milne), 1977.) Если A — множество разбиений $\{1, \ldots, n\}$, его *тенью* ∂A является множество всех разбиений Π', такое, что Π покрывает Π' для некоторого $\Pi \in A$ (подобная концепция рассматривалась в 7.2.1.3–(54)).

Пусть $\Pi_1, \Pi_2, \ldots$ — разбиения множества $\{1, \ldots, n\}$ на t блоков в лексикографическом порядке их ограниченно растущих строк. Пусть также $\Pi'_1, \Pi'_2, \ldots$ — $(t-1)$-блочные разбиения, также находящиеся в лексикографическом порядке. Докажите, что существует функция $f_{nt}(N)$, такая, что

$$\partial\{\Pi_1, \ldots, \Pi_N\} = \{\Pi'_1, \ldots, \Pi'_{f_{nt}(N)}\} \qquad \text{для } 0 \le N \le \left\{ {n \atop t} \right\}.$$

Указание: диаграмма из упр. 12 демонстрирует, что $(f_{43}(0), \ldots, f_{43}(6)) = (0, 3, 5, 7, 7, 7, 7)$.

14. [*23*] Разработайте алгоритм для генерации разбиений множества в порядке кода Грея наподобие (7).

15. [*M21*] Каково последнее разбиение, сгенерированное алгоритмом из упр. 14?

16. [*16*] Список (11) представляет собой последовательность Раски A_{35}. Какой вид имеет A'_{35}?

17. [*26*] Реализуйте код Грея для всех m-блочных разбиений множества $\{1,\ldots,n\}$, определяемый рекурсивными соотношениями Раски.

18. [*M46*] Для каких n можно сгенерировать все ограниченно растущие строки $a_1\ldots a_n$ так, чтобы на каждом шаге некоторое a_j изменялось на ±1?

19. [*28*] Докажите, что существует код Грея для ограниченно растущих строк, в котором на каждом шаге некоторое a_j изменяется либо на ±1, либо на ±2, когда мы хотим сгенерировать (а) все ϖ_n строк $a_1\ldots a_n$; (б) только $\left\{{n \atop m}\right\}$ случаев, удовлетворяющих условию $\max(a_1,\ldots,a_n)=m-1$.

20. [*17*] Если Π — разбиение множества $\{1,\ldots,n\}$, сопряженное к нему разбиение Π^T определяется правилом

$$j\equiv k \ (\text{по модулю } \Pi^T) \qquad\Longleftrightarrow\qquad n+1-j\equiv n+1-k \ (\text{по модулю } \Pi).$$

Предположим, Π имеет ограниченно растущую строку 001010202013; какой будет ограниченно растущая строка Π^T?

21. [*M27*] Сколько разбиений множества $\{1,\ldots,n\}$ являются самосопряженными?

22. [*M23*] Если X — случайная переменная с заданным распределением, то математическое ожидание X^n называется n-м *моментом* этого распределения. Чему равен n-й момент, если X представляет собой (а) распределение Пуассона со средним значением 1 (формула 3.4.1–(40))? (б) количество фиксированных элементов в случайной перестановке множества $\{1,\ldots,m\}$, где $m\ge n$ (формула 1.3.3–(27))?

23. [*HM30*] Пусть $f(x)=\sum a_kx^k$ и пусть $f(\varpi)$ означает $\sum a_k\varpi_k$.

а) Докажите символьную формулу $f(\varpi+1)=\varpi f(\varpi)$. (Например, если $f(x)$ — полином x^2, то эта формула гласит, что $\varpi_2+2\varpi_1+\varpi_0=\varpi_3$.)

б) Аналогично докажите, что $f(\varpi+k)=\varpi^{\underline{k}}f(\varpi)$ для всех положительных целых k.

в) Докажите, что если p — простое, то $\varpi_{n+p}\equiv\varpi_n+\varpi_{n+1}$ (по модулю p). *Указание*: сначала покажите, что $x^{\underline{p}}\equiv x^p-x$.

г) Следовательно, если $N=p^{p-1}+p^{p-2}+\cdots+p+1$, то $\varpi_{n+N}\equiv\varpi_n$ (по модулю p).

24. [*HM35*] Продолжая выполнять упражнение, докажите, что числа Белла удовлетворяют периодическому закону $\varpi_{n+p^{e-1}N}\equiv\varpi_n$ (по модулю p^e), если p — нечетное простое число. *Указание*: Покажите, что

$$x^{\underline{p^e}}\equiv g_e(x)+1 \ \big(\text{по модулю } p^e,\ p^{e-1}g_1(x),\ \ldots \text{ и } pg_{e-1}(x)\big),\ \text{где } g_j(x)=(x^p-x-1)^{p^j}.$$

25. [*M27*] Докажите, что $\varpi_n/\varpi_{n-1}\le\varpi_{n+1}/\varpi_n\le\varpi_n/\varpi_{n-1}+1$.

▶ **26.** [*M22*] В соответствии с рекуррентными уравнениями (13) числа ϖ_{nk} в треугольнике Пирса подсчитывают пути от nk до 11 в бесконечном ориентированном графе

Поясните, почему каждый путь от $n1$ до 11 соответствует разбиению множества $\{1,\ldots,n\}$.

▶ **27.** [*M35*] "Циклом колеблющейся диаграммы" (vacillating tableau loop) порядка n является последовательность разбиений целых чисел $\lambda_k = a_{k1}a_{k2}a_{k3}\ldots$, у которых $a_{k1} \ge a_{k2} \ge a_{k3} \ge \cdots$ для $0 \le k \le 2n$, такая, что $\lambda_0 = \lambda_{2n} = e_0$ и $\lambda_k = \lambda_{k-1} + (-1)^k e_{t_k}$ для $1 \le k \le 2n$ и для некоторого $0 \le t_k \le n$; здесь e_t обозначает единичный вектор $0^{t-1}10^{n-t}$ с $0 < t \le n$, а e_0 состоит из нулей.

а) Перечислите все такие циклы порядка 4. [*Указание*: их всего 15.]

б) Докажите, что $t_{2k-1} = 0$ ровно у ϖ_{nk} циклов порядка n.

▶ **28.** [*M25*] (*Обобщенные ладейные полиномы.*) Рассмотрим расстановку $a_1 + \cdots + a_m$ квадратных ячеек по строкам и столбцам, где строка k содержит ячейки в столбцах 1, ..., a_k. Поместим нуль или более "ладей" в ячейки так, чтобы в каждой строке и в каждом столбце было не более одной ладьи. Пустую ячейку будем называть свободной, если ни справа, ни снизу от нее нет ладьи. Например, на рис. 56 показаны два таких размещения. Одно — с четырьмя ладьями и строками длиной $(3, 1, 4, 1, 5, 9, 2, 6, 5)$; второе — с девятью ладьями на квадратной доске размером 9×9. Ладьи на рисунке показаны черными кружками; белые кружки размещаются слева и над ладьями, так что свободные ячейки не содержат никаких кружков.

Рис. 56. Размещение ладей и свободные ячейки.

Пусть $R(a_1, \ldots, a_m)$ — полином от x и y, полученный суммированием $x^r y^f$ по всем корректным расстановкам ладей, где r — количество ладей, а f — количество свободных ячеек; например, левое размещение на рис. 56 добавляет x^4y^{17} в полином $R(3, 1, 4, 1, 5, 9, 2, 6, 5)$.

а) Докажите, что $R(a_1, \ldots, a_m) = R(a_1, \ldots, a_{j-1}, a_{j+1}, a_j, a_{j+2}, \ldots, a_m)$; другими словами, порядок длин сторон значения не имеет, и можно считать, что $a_1 \ge \cdots \ge a_m$, как и в диаграмме Феррерса наподобие 7.2.1.4–(13).

б) Докажите, что если $a_1 \ge \cdots \ge a_m$ и если $b_1 \ldots b_n = (a_1 \ldots a_m)^T$ — сопряженное разбиение, то $R(a_1, \ldots, a_m) = R(b_1, \ldots, b_n)$.

в) Найдите рекуррентное соотношение для вычисления $R(a_1, \ldots, a_m)$ и воспользуйтесь им для вычисления $R(3, 2, 1)$.

г) Обобщите треугольник Пирса (12) путем замены правил сложения (13) на

$$\varpi_{nk}(x, y) \;=\; x\varpi_{(n-1)k}(x, y) + y\varpi_{n(k+1)}(x, y), \qquad 1 \le k < n.$$

Таким образом, $\varpi_{21}(x, y) = x + y$, $\varpi_{32}(x, y) = x + xy + y^2$, $\varpi_{31}(x, y) = x^2 + 2xy + xy^2 + y^3$ и т. д. Докажите, что получающееся таким образом значение $\varpi_{nk}(x, y)$ представляет собой ладейный полином $R(a_1, \ldots, a_{n-1})$, где $a_j = n - j - [j < k]$.

д) Полином $\varpi_{n1}(x, y)$ из п. (г) можно рассматривать как обобщенное число Белла $\varpi_n(x, y)$, представляющее пути от $n1$ до 11 в ориентированном графе из упр. 26, которые имеют данное количество "шагов x" на северо-восток и данное количество "шагов y" на восток. Докажите, что

$$\varpi_n(x, y) \;=\; \sum_{a_1 \ldots a_n} x^{n-1-\max(a_1,\ldots,a_n)} y^{a_1+\cdots+a_n},$$

где сумма берется по всем ограниченно растущим строкам $a_1 \ldots a_n$ длиной n.

29. [*M26*] Продолжая выполнять предыдущее упражнение, используем обозначение $R_r(a_1,\ldots,a_m) = [x^r]\,R(a_1,\ldots,a_m)$ для полинома от y, который перечисляет все свободные ячейки при расстановке r ладей.

а) Покажите, что количество способов расстановки n ладей на доске размером $n\times n$, таких, что при этом остается f свободных ячеек, представляет собой количество перестановок $\{1,\ldots,n\}$, в которых имеется f инверсий. Таким образом, в соответствии с формулой 5.1.1–(8) и упр. 5.1.2–16

$$R_n(\overbrace{n,\ldots,n}^{n}) \;=\; n!_y \;=\; \prod_{k=1}^{n}(1+y+\cdots+y^{k-1}).$$

б) Что собой представляет $R_r(\overbrace{n,\ldots,n}^{m})$ — производящая функция для r ладей на доске размером $m\times n$?

в) Докажите обобщенную формулу

$$\prod_{j=1}^{m}\frac{1-y^{a_j+j-m+t}}{1-y} \;=\; \sum_{k=0}^{m}\frac{t!_y}{(t-k)!_y}\,R_{m-k}(a_1,\ldots,a_m),$$

где $a_1\ge\cdots\ge a_m\ge 0$, а t — неотрицательное целое число. [*Примечание.* Величина $t!_y/(t-k)!_y = \prod_{j=0}^{k-1}((1-y^{t-j})/(1-y))$ равна нулю при $k>t\ge 0$. Таким образом, например, при $t=0$ правая часть сводится к $R_m(a_1,\ldots,a_m)$. Можно вычислить R_m, R_{m-1}, …, R_0, последовательно устанавливая $t=0$, 1, …, m.]

г) Покажите, что если $a_1\ge a_2\ge\cdots\ge a_m\ge 0$ и $a'_1\ge a'_2\ge\cdots\ge a'_m\ge 0$, то $R(a_1,a_2\ldots,a_m) = R(a'_1,a'_2,\ldots,a'_m)$ тогда и только тогда, когда соответствующие мультимножества $\{a_1{+}1,a_2{+}2,\ldots,a_m{+}m\}$ и $\{a'_1{+}1,a'_2{+}2,\ldots,a'_m{+}m\}$ одинаковы.

30. [*HM30*] Обобщенные числа Стирлинга $\left\{{n \atop m}\right\}_q$ определяются рекуррентным соотношением

$$\left\{{n+1 \atop m}\right\}_q \;=\; (1+q+\cdots+q^{m-1})\left\{{n \atop m}\right\}_q + \left\{{n \atop m-1}\right\}_q; \qquad \left\{{0 \atop m}\right\}_q \;=\; \delta_{m0}.$$

Таким образом, $\left\{{n \atop m}\right\}_q$ представляет собой полином от q, а $\left\{{n \atop m}\right\}_1$ является обычным числом Стирлинга $\left\{{n \atop m}\right\}$, поскольку оно удовлетворяет рекуррентному соотношению 1.2.6–(46).

а) Докажите, что обобщенное число Белла $\varpi_n(x,y) = R(n-1,\ldots,1)$ из упр. 28, (д) имеет явный вид

$$\varpi_n(x,y) \;=\; \sum_{m=0}^{n} x^{n-m}y^{\binom{m}{2}}\left\{{n \atop m}\right\}_y.$$

б) Покажите, что обобщенные числа Стирлинга подчиняются рекуррентному соотношению

$$q^m\left\{{n+1 \atop m+1}\right\}_q \;=\; q^n\left\{{n \atop m}\right\}_q + \binom{n}{1}q^{n-1}\left\{{n-1 \atop m}\right\}_q + \cdots \;=\; \sum_k\binom{n}{k}q^k\left\{{k \atop m}\right\}_q.$$

в) Найдите производящую функцию для $\left\{{n \atop m}\right\}_q$, обобщающую 1.2.9–(23) и 1.2.9–(28).

31. [*HM23*] Обобщая (15), покажите, что элементы треугольника Пирса имеют простую производящую функцию, если вычислить сумму

$$\sum_{n,k}\varpi_{nk}\,\frac{w^{n-k}}{(n-k)!}\,\frac{z^{k-1}}{(k-1)!}.$$

32. [*M22*] Пусть δ_n — количество ограниченно растущих строк $a_1 \ldots a_n$, сумма которых $a_1 + \cdots + a_n$ четна, минус количество строк, сумма которых $a_1 + \cdots + a_n$ нечетна. Докажите, что

$$\delta_n = (1, 0, -1, -1, 0, 1) \qquad \text{при} \qquad n \bmod 6 = (1, 2, 3, 4, 5, 0).$$

Указание: см. упр. 28, (д).

33. [*M21*] Сколько имеется разбиений множества $\{1, 2, \ldots, n\}$, у которых $1 \not\equiv 2$, $2 \not\equiv 3$, $\ldots$, $k - 1 \not\equiv k$?

34. [*14*] Поэтические произведения характеризуются *схемами рифм*, которые представляют собой разбиения строк строфы, обладающие теми свойствами, что $j \equiv k$ тогда и только тогда, когда строка j рифмуется со строкой k. Например, "лимерик" обычно представляет собой пятистрочное стихотворение с определенными ритмическими ограничениями и со схемой рифм, описываемой ограниченно растущей строкой 00110*.

Какие схемы рифм используются в классических *сонетах* (а) Гвиттоне д'Ареццо (ок. 1270)? (б) Петрарки (ок. 1350)? (в) Спенсера (1595)? (г) Шекспира (1609)? (д) Элизабет Браунинг (1850)?

35. [*M21*] Пусть ϖ'_n — количество схем n-строчных стихотворений, которые являются "полностью рифмованными", в том смысле, что каждая строка рифмуется как минимум еще с одной. Таким образом, мы имеем $\langle \varpi'_0, \varpi'_1, \varpi'_2, \ldots \rangle = \langle 1, 0, 1, 1, 4, 11, 41, \ldots \rangle$. Дайте комбинаторное доказательство того, что $\varpi'_n + \varpi'_{n+1} = \varpi_n$.

36. [*M22*] Продолжая выполнять упр. 35, выясните, что собой представляет производящая функция $\sum_n \varpi'_n z^n / n!$?

37. [*M18*] А. С. Пушкин в своем романе в стихах *Евгений Онегин* (1833) использовал тщательно разработанную структуру клаузул (т. е. окончаний стихотворных строк, состоящих из последнего ударного слога и следующих за ним безударных), основанную не только на "мужских" окончаниях, состоящих из одного ударного слога (стро́й-водо́й, старожи́л-дави́л), но и на "женских" окончаниях, состоящих из ударного и следующего за ним безударного слога (пра́вил-заста́вил, расхо́да-го́да).

Он в том покое поселился, / Где деревенский старожил
Лет сорок с ключницей бранился, / В окно смотрел и мух давил.
Все было просто: пол дубовый, / Два шкафа, стол, диван пуховый,
Нигде ни пятнышка чернил. / Онегин шкафы отворил:
В одном нашел тетрадь расхода, / В другом наливок целый строй,
Кувшины с яблочной водой, / И календарь осьмого года:
Старик, имея много дел, / В иные книги не глядел.

— А. С. ПУШКИН, *Евгений Онегин* (1833)

Каждая так называемая онегинская строфа представляет собой сонет со строгой схемой 01012233455477, причем рифмы являются мужскими или женскими в соответствии с тем, четна ли соответствующая цифра или нечетна. Несколько современных переводчиков Пушкина сумели выдержать эти правила при переводе на английский и немецкий языки.

---

* Кроме того, в каноническом лимерике конец последней строки повторяет конец первой. — *Примеч. пер.*

How do I justify this stanza? / These feminine rhymes? My wrinkled muse?
This whole passé extravaganza? / How can I (careless of time) use
The dusty bread molds of Onegin / In the brave bakery of Reagan?
The loaves will surely fail to rise / Or else go stale before my eyes.
The truth is, I can't justify it. / But as no shroud of critical terms
Can save my corpse from boring worms, / I may as well have fun and try it.
If it works, good; and if not, well, / A theory won't postpone its knell.

— ВИКРАМ СЕТ (VIKRAM SETH),
Золотые ворота (The Golden Gate) (1986)

Согласно упр. 35, 14-строчный стих может иметь любую из $\varpi'_{14} = 24\,011\,157$ полных схем рифм. Но сколько имеется схем, если мы получим возможность наложить на каждый блок условие использования женской или мужской рифмы?

▶ **38.** [*M30*] Пусть σ_k — циклическая перестановка $(1, 2, \ldots, k)$. Темой данного упражнения является изучение последовательностей $k_1 k_2 \ldots k_n$, именуемых *σ-циклами*, для которых $\sigma_{k_1}\sigma_{k_2}\ldots\sigma_{k_n}$ представляет собой тождественную перестановку. Например, при $n = 4$ имеется 15 σ-циклов, а именно

$$1111, 1122, 1212, 1221, 1333, 2112, 2121, 2211, 2222, 2323, 3133, 3232, 3313, 3331, 4444.$$

а) Найдите взаимно однозначное соответствие между разбиениями $\{1, 2, \ldots, n\}$ и σ-циклами длиной n.
б) Сколько σ-циклов длиной n обладают тем свойством, что $1 \le k_1, \ldots, k_n \le m$ для заданных m и n?
в) Сколько σ-циклов длиной n обладают тем свойством, что $k_i = j$ для заданных i, j и n?
г) Сколько σ-циклов длиной n обладают тем свойством, что $k_1, \ldots, k_n \ge 2$?
д) Сколько разбиений $\{1, \ldots, n\}$ обладают тем свойством, что $1 \not\equiv 2$, $2 \not\equiv 3$, $\ldots$, $n-1 \not\equiv n$ и $n \not\equiv 1$?

39. [*HM16*] Вычислите $\int_0^\infty e^{-t^{p+1}} t^q\,dt$, где p и q — неотрицательные целые числа. *Указание*: см. упр. 1.2.5–20.

40. [*HM20*] Предположим, что метод седловой точки используется для оценки $[z^{n-1}]\,e^{cz}$. В разделе при выводе (21) из (19) работа велась с $c = 1$. Как должен измениться вывод, если c — произвольная положительная константа?

41. [*HM21*] Решите предыдущее упражнение при $c = -1$.

42. [*HM23*] Воспользуйтесь методом седловой точки для оценки $[z^{n-1}]\,e^{z^2}$ с относительной ошибкой $O(1/n^2)$.

43. [*HM22*] Обоснуйте замену интеграла в (23) на (25).

44. [*HM22*] Поясните, как вычислить b_1, b_2, $\ldots$ в (26) на основании a_2, a_3, $\ldots$ в (25).

▶ **45.** [*HM23*] Покажите, что в дополнение к (26) имеется также разложение

$$\varpi_n = \frac{e^{e^{\xi}-1}n!}{\xi^n\sqrt{2\pi n(\xi+1)}}\Bigl(1 + \frac{b'_1}{n} + \frac{b'_2}{n^2} + \cdots + \frac{b'_m}{n^m} + O\Bigl(\frac{1}{n^{m+1}}\Bigr)\Bigr),$$

где $b'_1 = -(2\xi^4 + 9\xi^3 + 16\xi^2 + 6\xi + 2)/(24(\xi+1)^3)$.

46. [*HM25*] Оцените значение ϖ_{nk} в треугольнике Пирса при $n \to \infty$.

47. [*M21*] Проанализируйте время работы алгоритма H.

48. [*HM25*] Если n не является целым числом, интеграл в (23) может быть взят по контуру Ганкеля для определения обобщенного числа Белла ϖ_x для всех действительных $x > 0$. Покажите, что, как в (16),

$$\varpi_x = \frac{1}{e}\sum_{k=0}^{\infty}\frac{k^x}{k!}.$$

▶ **49.** [*HM35*] Докажите, что для больших n число ξ, определенное в (24), равно

$$\ln n - \ln\ln n + \sum_{j,k\ge 0}\begin{bmatrix} j+k \\ j+1\end{bmatrix}\alpha^j\frac{\beta^k}{k!}, \qquad \alpha = -\frac{1}{\ln n}, \quad \beta = \frac{\ln\ln n}{\ln n}.$$

▶ **50.** [*HM21*] Если $\xi(n)e^{\xi(n)} = n$ и $\xi(n) > 0$, то как между собой соотносятся $\xi(n+k)$ и $\xi(n)$?

51. [*HM27*] Воспользуйтесь методом седловой точки для оценки $t_n = n!\,[z^n]\,e^{z+z^2/2}$, количества *инволюций* на n элементах (или разбиений множества $\{1,\ldots,n\}$ на блоки размером ≤ 2).

52. [*HM22*] *Семиинварианты*, или *кумулянты* распределения вероятностей определяются правилом 1.2.10–(23). Чему равны кумулянты, если вероятность того, что целое число равно k, составляет (а) $e^{1-e^\xi}\varpi_k\xi^k/k!$? (б) $\sum_j \left\{{k \atop j}\right\} e^{e^{-1}-1-j}/k!$?

▶ **53.** [*HM30*] Пусть $G(z) = \sum_{k=0}^{\infty} p_k z^k$ — производящая функция для дискретного распределения вероятностей, сходящаяся при $|z| < 1+\delta$; итак, коэффициенты p_k неотрицательны, $G(1) = 1$, а среднее и дисперсия равны соответственно $\mu = G'(1)$ и $\sigma^2 = G''(1) + G'(1) - G'(1)^2$. Если $X_1, \ldots, X_n$ — независимые случайные переменные с указанным распределением, то вероятность того, что $X_1 + \cdots + X_n = m$ равна $[z^m]\,G(z)^n$, и часто возникает необходимость оценить эту вероятность при m, близком к среднему значению μn.

Будем считать, что $p_0 \ne 0$ и что не существует целого $d > 1$, которое является общим делителем всех индексов k, для которых $p_k \ne 0$; эти предположения означают, что m не обязано удовлетворять никаким специальным условиям конгруэнтности по модулю d при больших n. Докажите, что

$$[z^{\mu n+r}]\,G(z)^n = \frac{e^{-r^2/(2\sigma^2 n)}}{\sigma\sqrt{2\pi n}} + O\Bigl(\frac{1}{n}\Bigr) \qquad \text{при } n\to\infty,$$

где $\mu n + r$ является целым числом. *Указание*: проинтегрируйте $G(z)^n/z^{\mu n+r}$ по окружности $|z| = 1$.

54. [*HM20*] Пусть α и β определены формулами (40). Покажите, что их арифметическое и геометрическое средние равны соответственно $\frac{\alpha+\beta}{2} = s\coth s$ и $\sqrt{\alpha\beta} = s\operatorname{csch} s$, где $s = \sigma/2$.

55. [*HM20*] Предложите способ вычисления числа β, требующегося в формуле (43).

▶ **56.** [*HM26*] Пусть $g(z) = \alpha^{-1}\ln(e^z - 1) - \ln z$ и $\sigma = \alpha - \beta$, как в (37).

а) Докажите, что $(-\sigma)^{n+1}g^{(n+1)}(\sigma) = n! - \sum_{k=0}^{n}\left\langle {n \atop k}\right\rangle \alpha^k\beta^{n-k}$; числа Эйлера $\left\langle {n \atop k}\right\rangle$ определены в разделе 5.1.3.

б) Докажите, что $\frac{\beta}{\alpha}n! < \sum_{k=0}^{n}\left\langle {n \atop k}\right\rangle\alpha^k\beta^{n-k} < n!$ для всех $\sigma > 0$. *Указание*: см. упр. 5.1.3–25.

в) Проверьте неравенство (42).

57. [*HM22*] С использованием обозначений из (43) докажите, что (а) $n+1-m < 2N$; (б) $N < 2(n+1-m)$.

58. [*HM31*] Завершите доказательство (43) следующим образом.

а) Покажите, что для всех $\sigma > 0$ существует число $\tau \ge 2\sigma$, такое, что τ кратно 2π и $|e^{\sigma+it} - 1|/|\sigma + it|$ монотонно убывает при $0 \le t \le \tau$.

б) Докажите, что $\int_{-\tau}^{\tau} \exp\bigl((n+1)g(\sigma+it)\bigr)\,dt$ приводит к (43).

в) Покажите, что соответствующими интегралами по прямолинейным путям $z = t \pm i\tau$, где $-n \le t \le \sigma$, и $z = -n \pm it$, где $-\tau \le t \le \tau$, можно пренебречь.

▶ **59.** [*HM23*] Какое предсказание дает (43) для приближенного значения $\left\{ {n \atop n} \right\}$?

60. [*HM25*] (а) Покажите, что частичные суммы в тождестве

$$\left\{ {n \atop m} \right\} = \frac{m^n}{m!} - \frac{(m-1)^n}{1!\,(m-1)!} + \frac{(m-2)^n}{2!\,(m-2)!} - \cdots + (-1)^m \frac{0^n}{m!\,0!}$$

поочередно приближаются к конечному значению сверху и снизу. (б) Сделайте вывод о том, что

$$\left\{ {n \atop m} \right\} = \frac{m^n}{m!}\bigl(1 - O(ne^{-n^\epsilon})\bigr) \qquad \text{при } m \le n^{1-\epsilon}.$$

(в) Выведите аналогичный результат из (43).

61. [*HM26*] Докажите, что, если $m = n - r$, где $r \le n^\epsilon$ и $\epsilon \le n^{1/2}$, формула (43) дает

$$\left\{ {n \atop n-r} \right\} = \frac{n^{2r}}{2^r r!}\Bigl(1 + O(n^{2\epsilon-1}) + O\Bigl(\frac{1}{r}\Bigr)\Bigr).$$

62. [*HM40*] Докажите строго, что если $\xi e^\xi = n$, то максимум $\left\{ {n \atop m} \right\}$ достигается либо при $m = \lfloor e^\xi - 1 \rfloor$, либо при $m = \lceil e^\xi - 1 \rceil$.

▶ **63.** [*M35*] (Д. Питман (J. Pitman).) Докажите, что имеется следующий элементарный способ определить максимум как чисел Стирлинга, так и многих других подобных величин. Предположим, что $0 \le p_j \le 1$.

а) Пусть $f(z) = (1 + p_1(z-1)) \ldots (1 + p_n(z-1))$ и $a_k = [z^k]\,f(z)$; таким образом, a_k представляет собой вероятность получить k "орлов" при n независимых подбрасываниях монет с вероятностями выпадения орла соответственно $p_1, \ldots, p_n$. Докажите, что $a_{k-1} < a_k$ при $k \le \mu = p_1 + \cdots + p_n$, $a_k \ne 0$.

б) Аналогично докажите, что $a_{k+1} < a_k$ при $k \ge \mu$ и $a_k \ne 0$.

в) Если $f(x) = a_0 + a_1 x + \cdots + a_n x^n$ — некоторый ненулевой полином с неотрицательными коэффициентами и n действительными корнями, докажите, что $a_{k-1} < a_k$ при $k \le \mu$ и $a_{k+1} < a_k$ при $k \ge \mu$, где $\mu = f'(1)/f(1)$. Следовательно, если $a_m = \max(a_0, \ldots, a_n)$, то либо $m = \lfloor \mu \rfloor$, либо $m = \lceil \mu \rceil$.

г) Используя гипотезу (в) и полагая $a_j = 0$ при $j < 0$ или $j > n$, покажите, что существуют индексы $s \le t$, такие, что $a_{k+1} - a_k < a_k - a_{k-1}$ тогда и только тогда, когда $s \le k \le t$ (т. е. гистограмма последовательности $(a_0, a_1, \ldots, a_n)$ всегда имеет колоколообразную форму).

д) Что говорят полученные результаты о числах Стирлинга?

64. [*HM21*] Докажите приближенное отношение (50) с использованием (30) и упр. 50.

▶ **65.** [*HM22*] Чему равна дисперсия количества блоков размером k в случайном разбиении множества $\{1, \ldots, n\}$?

66. [*M46*] Какое разбиение числа n дает больше всего разбиений множества $\{1, \ldots, n\}$?

67. [*HM20*] Чему равны среднее и дисперсия M в методе Стама (53)?

68. [*21*] До каких значений могут вырасти переменные l и b в алгоритме M при генерации всех $p(n_1, \ldots, n_m)$ разбиений мультимножества $\{n_1 \cdot 1, \ldots, n_m \cdot m\}$?

▶ **69.** [*22*] Модифицируйте алгоритм M так, чтобы он выдавал разбиения не более чем на r частей.

▶ **70.** [*M22*] Проанализируйте количество возможных r-блочных разбиений в n-элементных мультимножествах (а) $\{0, \ldots, 0, 1\}$; (б) $\{1, 2, \ldots, n-1, n-1\}$. Чему равно общее количество разбиений, просуммированное по всем r?

71. [*M20*] Сколько разбиений мультимножества $\{n_1 \cdot 1, \ldots, n_m \cdot m\}$ состоит ровно из двух частей?

72. [*M26*] Можно ли вычислить $p(n, n)$ за полиномиальное время?

▶ **73.** [*M32*] Можно ли вычислить за полиномиальное время $p(2, \ldots, 2)$ (в скобках всего n двоек)?

74. [*M46*] Можно ли вычислить за полиномиальное время $p(n, \ldots, n)$ (в скобках всего n значений n)?

75. [*HM41*] Найдите асимптотическое значение $p(n, n)$.

76. [*HM36*] Найдите асимптотическое значение $p(2, \ldots, 2)$ (в скобках всего n двоек).

77. [*HM46*] Найдите асимптотическое значение $p(n, \ldots, n)$ (в скобках всего n значений n).

78. [*20*] Какие разбиения $(15, 10, 10, 11)$ приводят к перестановкам α_1, α_2, α_3 и α_4, показанным в табл. 1?

79. [*22*] Последовательность u_1, u_2, u_3, ... называется *универсальной* для разбиений множества $\{1, \ldots, n\}$, если ее подпоследовательности $(u_{m+1}, u_{m+2}, \ldots, u_{m+n})$ при $0 \le m < \varpi_n$ представляют все возможные разбиения множества с выполнением соглашения "$j \equiv k$ тогда и только тогда, когда $u_{m+j} = u_{m+k}$". Например, $(0, 0, 0, 1, 0, 2, 2)$ является универсальной последовательностью для разбиений множества $\{1, 2, 3\}$.

Напишите программу для поиска всех универсальных последовательностей разбиений множества $\{1, 2, 3, 4\}$, обладающих следующими свойствами: (i) $u_1 = u_2 = u_3 = u_4 = 0$; (ii) последовательность обладает свойством ограниченного роста; (iii) $0 \le u_j \le 3$; и (iv) $u_{16} = u_{17} = u_{18} = 0$ (так что последовательность, по сути, *циклична*).

80. [*M28*] Докажите, что универсальные циклы для разбиений $\{1, 2, \ldots, n\}$ существуют в смысле предыдущего упражнения для всех $n \ge 4$.

81. [*29*] Найдите способ так сложить обычную колоду из 52 игральных карт, чтобы был возможен следующий фокус. Пять игроков по очереди снимают колоду (выполняя циклическую перестановку) столько раз, сколько хотят. Затем каждый игрок берет по одной карте с вершины колоды. Фокусник просит игроков посмотреть на свои карты и образовать аффинные группы, объединив карты одинаковой масти (валет пик, однако, рассматривается как "джокер" — он остается в одиночестве).

Увидев аффинные группы, и ничего не зная об их масти, фокусник, тем не менее, может назвать все пять карт (если изначально они были сложены в колоду соответствующим образом).

82. [*22*] Сколькими способами можно разделить следующие 15 костей домино (возможно, перевернув некоторые из них) на три части по пять костей, чтобы суммы частей (рассматривая кости как дроби) были одинаковы?

Сократ. Подобно тому как в едином от природы человеческом теле имеется две одноименные части, лишь с обозначением "левая" или "правая", так обстоит дело и с состоянием безумия, которое обе наши речи признали составляющим в нас от природы единый вид, но одна речь выделила из него часть, обращенную налево, и не остановилась на этом делении, пока не нашла там некую так называемую левую любовь, которую вполне справедливо и осудила; другая же наша речь ведет нас к правой части неистовства, одноименной с первой, и находит там некую божественную любовь, которой отдает предпочтение, и восхваляет ее как причину величайших для нас благ.

— ПЛАТОН, Федр (*Ph?drus*) *266A (ок. 370 г. до н. э.)*

7.2.1.6. Генерация всех деревьев. К этому моменту мы уже завершили изучение классических концепций комбинаторики: кортежей, перестановок, сочетаний и разбиений. Однако информатика внесла свой вклад, добавив еще один фундаментальный класс к традиционному репертуару, а именно иерархические конструкции, известные как деревья. Деревья встречаются в информатике повсюду, как видно из раздела 2.3 и практически каждого из последующих разделов *Искусства программирования*. Поэтому сейчас мы обратимся к изучению простых алгоритмов, с помощью которых можно исчерпывающе исследовать деревья различного вида.

Сначала давайте рассмотрим фундаментальную связь между вложенными скобками и лесами деревьев. Например, строка

(1)

содержит 15 левых скобок '(' с метками `1`, `2`, ..., `f` и 15 правых скобок ')', также с метками от `1` до `f`. Серые линии под строками показывают, как соответствующие одна другой скобки образуют 15 пар: `12`, `21`, `3f`, `44`, `53`, `6a`, `78`, `85`, `97`, `a6`, `b9`, `ce`, `db`, `ed` и `fc`. Эта строка соответствует лесу

, (2)

узлами которого в прямом порядке обхода являются 12, 21, 3f, ..., fc (отсортированные по первой координате) и 21, 12, 53, ..., 3f в обратном порядке обхода (отсортированные по второй координате). Если мы представим червяка, который

проползает по краю леса

(3)

и видит, проползая по левой дуге узла, открывающую (левую) скобку '(', а проползая по правой — закрывающую (правую) скобку ')', то его путь реконструирует исходную строку (1). Лес (2), в свою очередь, соответствует бинарному дереву

(4)

посредством "естественного соответствия", рассмотренного в разделе 2.3.2; в данном случае список узлов в *симметричном* порядке обхода представляет собой (21), (12), (53), ..., (3f). Левое поддерево узла (x) в бинарном дереве — это крайний слева потомок (x) в лесу, или "внешний узел" □, если (x) не имеет потомков. Правое поддерево (x) в бинарном дереве представляет собой потомок правого "брата" в лесу, или "внешний узел" □, если (x) — крайний справа потомок своего родителя. Корни деревьев в лесу рассматриваются как братья, и крайний слева корень в лесу является корнем бинарного дерева.

Строка скобок $a_1a_2\ldots a_{2n}$ имеет корректную вложенность тогда и только тогда, когда она содержит n символов '(' и n символов ')', и k-й символ '(' предшествует k-му символу ')' для $1 \le k \le n$. Простейший способ получить все строки вложенных скобок — посетить их в лексикографическом порядке. Далее приведен алгоритм, который рассматривает ')' как лексикографически меньший символ по сравнению с '(', и включает некоторые усовершенствования для повышения эффективности по сравнению с оригинальным алгоритмом И. Семба (I. Semba) [*Inf. Processing Letters* **12** (1981), 188–192].

Алгоритм P (*Вложенные скобки в лексикографическом порядке*). Для заданного целого числа $n \ge 2$ этот алгоритм генерирует все строки $a_1a_2\ldots a_{2n}$ вложенных скобок.

P1. [Инициализация.] Установить $a_{2k-1} \leftarrow$ '(' и $a_{2k} \leftarrow$ ')' для $1 \le k \le n$; установить также $a_0 \leftarrow$ ')' и $m \leftarrow 2n-1$. (Мы используем a_0 в качестве ограничителя на шаге P4.)

Таблица 1

Вложенные скобки и сопутствующие объекты для $n = 4$

$a_1a_2\ldots a_8$	Лес	Бинарное дерево	$d_1d_2d_3d_4$	$z_1z_2z_3z_4$	$p_1p_2p_3p_4$	$c_1c_2c_3c_4$	Соответствие
()()()()			1111	1357	1234	0000	
()()(())			1102	1356	1243	0001	
()(())()			1021	1347	1324	0010	
()(()())			1012	1346	1342	0011	
()((()))			1003	1345	1432	0012	
(())()()			0211	1257	2134	0100	
(())(())			0202	1256	2143	0101	
(()())()			0121	1247	2314	0110	
(()()())			0112	1246	2341	0111	
(()(()))			0103	1245	2431	0112	
((()))()			0031	1237	3214	0120	
((())())			0022	1236	3241	0121	
(((())))			0013	1235	3421	0122	
(((())))			0004	1234	4321	0123	

P2. [Посещение.] Посетить строку вложенных скобок $a_1a_2\ldots a_{2n}$. (В этот момент $a_m =$ '(' и $a_k =$ ')' для $m < k \le 2n$.)

P3. [Простой случай?] Установить $a_m \leftarrow$ ')'. Затем, если $a_{m-1} =$ ')', установить $a_{m-1} \leftarrow$ '(', $m \leftarrow m-1$ и вернуться к шагу P2.

P4. [Поиск j.] Установить $j \leftarrow m-1$ и $k \leftarrow 2n-1$. Пока $a_j =$ '(', устанавливать $a_j \leftarrow$ ')', $a_k \leftarrow$ '(', $j \leftarrow j-1$ и $k \leftarrow k-2$.

P5. [Увеличение a_j.] Завершить алгоритм, если $j = 0$. В противном случае установить $a_j \leftarrow$ '(', $m \leftarrow 2n - 1$ и вернуться к шагу P2. ▮

Позже мы увидим, что цикл на шаге P4 почти всегда очень короткий: операция $a_j \leftarrow$ ')' выполняется в среднем только около $\frac{1}{3}$ раза на одно посещение строки.

Почему работает алгоритм P? Пусть A_{pq} — последовательность всех строк α, содержащих p левых скобок и $q \geq p$ правых скобок (причем строка $(^{q-p}\alpha$ обладает корректной вложенностью), перечисленных в лексикографическом порядке. Тогда алгоритм P обязан генерировать A_{nn}, где, как легко видеть, A_{pq} подчиняется рекуррентному соотношению

$$A_{pq} \;=\;)\,A_{p(q-1)},\;\; (\,A_{(p-1)q}, \quad \text{при } 0 \leq p \leq q \neq 0; \qquad A_{00} = \epsilon; \tag{5}$$

кроме того, строка A_{pq} пуста при $p < 0$ или $p > q$. Первым элементом A_{pq} является $)^{q-p}()\ldots()$, где имеется p пар '()'; последний элемент равен $(^p)^q$. Таким образом, процесс лексикографической генерации состоит из сканирования справа в поисках завершающей строки вида $a_j \ldots a_{2n} =)\,(^{p+1})^q$ и замене ее строкой $()^{q+1-p}()\ldots()$. Шаги P4 и P5 эффективно выполняют данную задачу, а шаг P3 предназначен для обработки простого случая $p = 0$.

В табл. 1 проиллюстрирован выход алгоритма P при $n = 4$ вместе с соответствующими лесами и бинарными деревьями, как в (2) и (4). В таблице приведены и некоторые другие эквивалентные комбинаторные объекты. Например, строка вложенных скобок может быть закодирована с применением кодирования длин серий как

$$()^{d_1}()^{d_2}\ldots()^{d_n}, \tag{6}$$

где неотрицательные целые числа $d_1 d_2 \ldots d_n$ характеризуются ограничениями

$$d_1 + d_2 + \cdots + d_k \leq k \quad \text{для } 1 \leq k < n; \qquad d_1 + d_2 + \cdots + d_n = n. \tag{7}$$

Вложенные скобки могут также быть представлены последовательностью $z_1 z_2 \ldots z_n$, которая определяет индексы появлений левых скобок. По сути, $z_1 z_2 \ldots z_n$ представляет собой одно из $\binom{2n}{n}$ сочетаний n элементов из множества $\{1, 2, \ldots, 2n\}$, отвечающих определенным ограничениям

$$z_{k-1} \;<\; z_k \;<\; 2k \qquad \text{для } 1 \leq k \leq n, \tag{8}$$

если считать, что $z_0 = 0$. Разумеется, все z связаны с d:

$$d_k \;=\; z_{k+1} - z_k - 1 \qquad \text{для } 1 \leq k < n. \tag{9}$$

Алгоритм P становится особенно простым, если переписать его для генерации сочетаний $z_1 z_2 \ldots z_n$ вместо строк $a_1 a_2 \ldots a_{2n}$ (см. упр. 2).

Строка скобок может также быть представлена перестановкой $p_1 p_2 \ldots p_n$, где k-я правая скобка соответствует p_k-й левой скобке; другими словами, k-й узел связанного леса в обратном порядке обхода является p_k-м узлом в прямом порядке обхода. Согласно упр. 2.3.2–20 узел j является (истинным) потомком узла k в лесу тогда и только тогда, когда $j < k$ и $p_j > p_k$, если помечать узлы в обратном порядке обхода. Таблица инверсий $c_1 c_2 \ldots c_n$ характеризует эту перестановку в соответствии с правилом, согласно которому справа от k имеется ровно c_k элементов, меньших k

(см. упр. 5.1.1–7); у допустимых таблиц инверсий $c_1 = 0$ и

$$0 \le c_{k+1} \le c_k + 1 \qquad \text{для } 1 \le k < n. \tag{10}$$

Более того, в упр. 3 доказывается, что c_k представляет собой уровень, на котором в лесу находится k-й в прямом порядке обхода узел (глубина k-й левой скобки); этот факт эквивалентен формуле

$$c_k = 2k - 1 - z_k. \tag{11}$$

В табл. 1 и упр. 6 показан также специальный вид *паросочетания*, при котором $2n$ человек за круглым столом могут одновременно пожать друг другу руки, причем без перекрещивания.

Таким образом, алгоритм P может оказаться весьма полезным. Но если наша цель — генерация всех бинарных деревьев, представленных левыми связями $l_1 l_2 \ldots l_n$ и правыми связями $r_1 r_2 \ldots r_n$, то лексикографическая последовательность из табл. 1 весьма неудобна; данные, которые требуются для перехода от одного дерева к следующему за ним, не будут легкодоступными. К счастью, имеется остроумный альтернативный алгоритм для непосредственной генерации всех связанных бинарных деревьев.

Алгоритм B (*Бинарные деревья*). Для заданного $n \ge 1$ этот алгоритм генерирует все бинарные деревья с n внутренними узлами, представленными с помощью левых связей $l_1 l_2 \ldots l_n$ и правых связей $r_1 r_2 \ldots r_n$, с узлами, помеченными в прямом порядке. (Таким образом, например, узел 1 всегда корневой, а l_k всегда равно либо $k+1$, либо 0; если $l_1 = 0$ и $n > 1$, то $r_1 = 2$.)

B1. [Инициализация.] Установить $l_k \leftarrow k+1$ и $r_k \leftarrow 0$ для $1 \le k < n$; установить также $l_n \leftarrow r_n \leftarrow 0$ и $l_{n+1} \leftarrow 1$ (для удобства на шаге B3).

B2. [Посещение.] Посетить бинарное дерево, представленное связями $l_1 l_2 \ldots l_n$ и $r_1 r_2 \ldots r_n$.

B3. [Поиск j.] Установить $j \leftarrow 1$. Пока $l_j = 0$, устанавливать $r_j \leftarrow 0$, $l_j \leftarrow j+1$ и $j \leftarrow j+1$. Затем завершить работу алгоритма, если $j > n$.

B4. [Поиск k и y.] Установить $y \leftarrow l_j$ и $k \leftarrow 0$. Пока $r_y > 0$, устанавливать $k \leftarrow y$ и $y \leftarrow r_y$.

B5. [Продвижение y.] Если $k > 0$, установить $r_k \leftarrow 0$; в противном случае установить $l_j \leftarrow 0$. Затем установить $r_y \leftarrow r_j$, $r_j \leftarrow y$ и вернуться к шагу B2. ▮

[См. W. Skarbek, *Theoretical Computer Science* **57** (1988), 153–159; на шаге B3 используется идея Д. Корша (J. Korsh).] В упр. 44 доказывается, что циклы на шагах B3 и B4 обычно достаточно коротки. Фактически в среднем требуется менее девяти обращений к памяти для преобразования связного бинарного дерева в следующее за ним.

В табл. 2 показаны 14 бинарных деревьев, сгенерированных при $n = 4$, вместе с соответствующими лесами и двумя связанными с ними последовательностями — массивами $e_1 e_2 \ldots e_n$ и $s_1 s_2 \ldots s_n$, которые определяются тем свойством, что узел k в прямом порядке обхода имеет e_k дочерних узлов и s_k потомков в связанном лесу. (Таким образом, s_k — размер k-го левого поддерева в бинарном дереве, а $s_k + 1$ — величина поля `SCOPE` в смысле 2.3.3–(5).) В следующем столбце 14 лесов из табл. 1 повторяются в лексикографическом порядке алгоритма P, но зеркально

Таблица 2

Связанные бинарные деревья и сопутствующие объекты при $n = 4$

$l_1l_2l_3l_4$	$r_1r_2r_3r_4$	Бинарное дерево	Лес	$e_1e_2e_3e_4$	$s_1s_2s_3s_4$	Солексный лес	Левый брат/правый потомок
2340	0000			1110	3210		
0340	2000			0110	0210		
2040	0300			2010	3010		
2040	3000			1010	1010		
0040	2300			0010	0010		
2300	0040			1200	3200		
0300	2040			0200	0200		
2300	0400			2100	3100		
2300	4000			1100	2100		
0300	2400			0100	0100		
2000	0340			3000	3000		
2000	4300			2000	2000		
2000	3040			1000	1000		
0000	2340			0000	0000		

отраженными по вертикали. В последнем столбце показаны бинарные деревья, которые представляют солексный лес. Они также представляют лес из столбца 4, но с использованием связей с левым братом и правым потомком вместо левого потомка и правого брата. Последний столбец представляет интересную связь между вложенными скобками и бинарными деревьями, так что вносит определенный вклад в понимание того, почему корректно работает алгоритм B (см. упр. 19).

***Коды Грея для деревьев.** Наш предыдущий опыт работы с другими комбинаторными объектами говорит нам, что, вероятно, мы можем генерировать строки скобок и деревья путем минимальных изменений между соседними экземплярами. И действительно, существует как минимум три очень красивых способа добиться поставленной цели.

Рассмотрим сначала случай вложенных скобок, которые могут быть представлены последовательностью $z_1 z_2 \ldots z_n$, удовлетворяющей условию (8). "Близкий к идеальному" способ генерации всех таких сочетаний в смысле раздела 7.2.1.3 представляет собой метод, при котором мы проходим по всем возможным этапам так, что на каждом шаге изменяется только один компонент z_j, причем его изменение равно ± 1 или ± 2; это означает, что из каждой строки скобок мы получаем следующую за ней путем простых изменений — либо `()` $\leftrightarrow$ `)(`, либо `())` $\leftrightarrow$ `))(` вблизи j-й левой скобки. Вот один из вариантов для $n = 4$:

$$1357, 1356, 1346, 1345, 1347, 1247, 1245, 1246, 1236, 1234, 1235, 1237, 1257, 1256.$$

Любое решение для $n-1$ можно расширить для случая n, беря каждый шаблон $z_1 z_2 \ldots z_{n-1}$ и позволяя z_n пройти по всем допустимым значениям с использованием *чун-порядка* (*endo-order*) или обратного к нему (см. 7.2.1.3–(45)), двигаясь вниз от $2n-2$, а затем вверх до $2n-1$ (или наоборот) и опуская все элементы, которые $\le z_{n-1}$.

Алгоритм N (*Вложенные скобки, близкие к идеальным*). Этот алгоритм посещает все n-сочетания $z_1 \ldots z_n$ элементов $\{1, \ldots, 2n\}$, которые представляют собой индексы левых скобок в строке вложенных скобок, подчиняющиеся условию изменения индексов по одному. Процессом управляет вспомогательный массив $g_1 \ldots g_n$, который представляет временные цели алгоритма.

N1. [Инициализация.] Установить $z_j \leftarrow 2j-1$ и $g_j \leftarrow 2j-2$ для $1 \le j \le n$.

N2. [Посещение.] Посетить n-сочетание $z_1 \ldots z_n$. Затем установить $j \leftarrow n$.

N3. [Поиск j.] Если $z_j = g_j$, установить $g_j \leftarrow g_j \oplus 1$ (тем самым обращая младший бит), $j \leftarrow j-1$ и повторить этот шаг.

N4. [Финишная прямая?] Если $g_j - z_j$ четно, установить $z_j \leftarrow z_j + 2$ и вернуться к шагу N2.

N5. [Уменьшение.] Установить $t \leftarrow z_j - 2$. Если $t < 0$, завершить работу алгоритма. В противном случае при $t \le z_{j-1}$, установить $t \leftarrow t + 2[t < z_{j-1}] + 1$. Наконец установить $z_j \leftarrow t$ и вернуться к шагу N2. ▮

[Похожий алгоритм был описан в работе D. R?lants van Baronaigien, *J. Algorithms* **35** (2000), 100–107; см. также Xiang, Ushijima, and Tang, *Inf. Proc. Letters* **76** (2000), 169–174. Ф. Раски (F. Ruskey) и А. Проскуровски (A. Proskurowski) ранее в *J. Algorithms* **11** (1990), 68–84, показали, как построить *идеальные* коды Грея для всех таблиц $z_1 \ldots z_n$ для четных $n \ge 4$ таким образом, чтобы на каждом шаге изменялось на ± 1 только одно значение z_j; однако их построение весьма сложное, и неизвестна ни одна идеальная схема, достаточно простая для практического использования. В упр. 48 показано, что идеальная схема невозможна для нечетных $n \ge 5$.]

Если наша цель состоит в генерации связанных древовидных структур, а не строк скобок, "идеальности" с точки зрения изменений z-индексов недостаточно,

поскольку простые обмены наподобие () ↔)(не обязательно соответствуют простым действиям со связями. Существенно лучший подход может быть основан на алгоритмах "поворотов", с помощью которых мы поддерживали сбалансированность деревьев поиска в разделе 6.2.3. *Поворот влево* изменяет бинарное дерево

; (12)

таким образом, соответствующий лес изменяется

с

на

. (13)

"Узел (A) становится крайним слева дочерним узлом своего правого брата". *Поворот вправо*, конечно же, представляет собой обратное преобразование: "крайний слева дочерний узел узла (B) становится его левым братом". Вертикальная линия на рисунках (12) означает соединение с общим контекстом — левую или правую связь либо указатель на корень. Любое из поддеревьев α, μ или ω может быть пустым. Троеточие '$\cdots$' в (13), представляющее дополнительных братьев в левой части семейства, содержащего (B), также может быть пустым.

Приятная информация о поворотах заключается в том, что при этом изменяются только три связи: правая связь из (A), левая связь из (B) и указатель сверху. Повороты сохраняют порядок симметричного обхода бинарного дерева и обратный порядок обхода леса. (Заметим также, что повороты бинарного дерева естественным образом соответствуют применению *ассоциативного закона*

$$(\alpha\mu)\omega \;=\; \alpha(\mu\omega) \tag{14}$$

в алгебраической формуле.)

Для генерации всех бинарных деревьев или лесов посредством поворотов можно воспользоваться простой схемой, очень похожей на классический рефлексивный код Грея для n-кортежей (алгоритм 7.2.1.1H) и метод простых изменений для перестановок (алгоритм 7.2.1.2P). Рассмотрим лес из $n-1$ узлов с k корнями (A_1), ..., (A_k). В таком случае существует $k+1$ лесов из n узлов, которые имеют одну и ту же последовательность первых $n-1$ узлов в обратном порядке обхода, но с последней вершиной (n); например, при $k=3$ это

полученные последовательными поворотами (A_3), (A_2) и (A_1) влево. Кроме того, в крайних случаях, когда (n) находится либо справа, либо вверху, можно выполнить

любой требующийся поворот над остальными $n-1$ узлами, поскольку узел (n) не находится на указанном пути. Таким образом, как замечено в работе J. M. Lucas, D. R?lants van Baronaigien, and F. Ruskey, *J. Algorithms* **15** (1993), 343–366, можно расширить любой список $(n-1)$-узловых деревьев до списка всех n-узловых деревьев, просто позволив узлу (n) "прогуливаться" назад и вперед. Уделив внимание низкоуровневым деталям, можно выполнить указанную работу с замечательной эффективностью.

Алгоритм L (*Генерация связанных бинарных деревьев путем поворотов*). Этот алгоритм генерирует все пары массивов $l_0 l_1 \ldots l_n$ и $r_1 \ldots r_n$, представляющих левые и правые связи n-узловых бинарных деревьев, где l_0 является корнем дерева, а связи (l_k, r_k) указывают соответственно на левое и правое поддеревья k-го в симметричном порядке обхода узла. Каждое дерево получается из предшественника путем простого поворота. Для управления процессом используются два вспомогательных массива, $k_1 \ldots k_n$ и $o_0 o_1 \ldots o_n$, представляющие обратные указатели и направления.

L1. [Инициализация.] Установить $l_j \leftarrow 0$, $r_j \leftarrow j+1$, $k_j \leftarrow j-1$ и $o_j \leftarrow -1$ для $1 \le j < n$; установить также $l_0 \leftarrow o_0 \leftarrow 1$, $l_n \leftarrow r_n \leftarrow 0$, $k_n \leftarrow n-1$ и $o_n \leftarrow -1$.

L2. [Посещение.] Посетить бинарное дерево (или лес), представленное массивами $l_0 l_1 \ldots l_n$ и $r_1 \ldots r_n$. Затем установить $j \leftarrow n$ и $p \leftarrow 0$.

L3. [Поиск j.] Если $o_j > 0$, установить $m \leftarrow l_j$ и перейти к шагу L5, если $m \ne 0$. Если $o_j < 0$, установить $m \leftarrow k_j$; затем перейти к шагу L4, если $m \ne 0$, в противном случае установить $p \leftarrow j$. Если $m = 0$, в любом случае установить $o_j \leftarrow -o_j$, $j \leftarrow j-1$ и повторить этот шаг.

L4. [Поворот влево.] Установить $r_m \leftarrow l_j$, $l_j \leftarrow m$, $x \leftarrow k_m$ и $k_j \leftarrow x$. Если $x = 0$, установить $l_p \leftarrow j$, в противном случае установить $r_x \leftarrow j$. Вернуться к шагу L2.

L5. [Поворот вправо.] Завершить работу алгоритма, если $j = 0$. В противном случае установить $l_j \leftarrow r_m$, $r_m \leftarrow j$, $k_j \leftarrow m$, $x \leftarrow k_m$. Если $x = 0$, установить $l_p \leftarrow m$, в противном случае установить $r_x \leftarrow m$. Вернуться к шагу L2. ▮

В упр. 38 доказывается, что алгоритм L требует только около девяти обращений к памяти на генерацию одного дерева; таким образом, он почти такой же быстрый, как и алгоритм B. (Фактически можно сэкономить два обращения к памяти на шаг, если хранить значения o_n, l_n и k_n в регистрах. Но, конечно же, точно так можно ускорить и алгоритм B.)

В табл. 3 показана последовательность бинарных деревьев и лесов, посещенных алгоритмом L при $n = 4$, с некоторыми вспомогательными данными, способствующими дальнейшему пониманию процесса. Перестановка $q_1 q_2 q_3 q_4$ перечисляет узлы в прямом порядке обхода, при том что они пронумерованы в обратном порядке обхода в лесу (в симметричном порядке в бинарном дереве); обратной к данной перестановке является перестановка $p_1 p_2 p_3 p_4$ из табл. 1. "Со-лес" представляет собой сопряжение (отражение слева направо) леса; а числа $u_1 u_2 u_3 u_4$ аналогичны числам $s_1 s_2 s_3 s_4$ из табл. 2. В последнем столбце показан так называемый "дуальный лес". Важность всех этих величин поясняется в упр. 11–13, 19, 24, 26 и 27.

Связи $l_0 l_1 \ldots l_n$ и $r_1 \ldots r_n$ в алгоритме L и табл. 3 *не* сравнимы со связями $l_1 \ldots l_n$ и $r_1 \ldots r_n$ из алгоритма B и табл. 2, поскольку алгоритм L сохраняет симметричный/обратный порядок обхода, в то время как алгоритм B сохраняет прямой

Таблица 3

Бинарные деревья и леса, сгенерированные с помощью поворотов для $n = 4$

$l_0l_1l_2l_3l_4$	$r_1r_2r_3r_4$	$k_1k_2k_3k_4$	Бинарное дерево	Лес	$q_1q_2q_3q_4$	Со-лес	$u_1u_2u_3u_4$	Дуальный лес
10000	2340	0123			1234		0000	
10003	2400	0122			1243		1000	
10002	4300	0121			1423		2000	
40001	2300	0120			4123		3000	
40021	3000	0110			4132		3100	
10023	4000	0111			1432		2100	
10020	3040	0113			1324		0100	
30010	2040	0103			3124		0200	
40013	2000	0100			4312		3200	
40123	0000	0000			4321		3210	
30120	0040	0003			3214		0210	
20100	0340	0023			2134		0010	
20103	0400	0022			2143		1010	
40102	0300	0020			4213		3010	

порядок обхода. Узел k в алгоритме L представляет собой k-й узел слева направо в бинарном дереве, так что для идентификации корня требуется значение l_0; но в алгоритме B узел k является k-м узлом в прямом порядке обхода, так что в этом случае корнем всегда является узел 1.

Алгоритм L обладает тем интересующим нас свойством, что на каждом шаге изменяются только три связи; но в действительности в этом отношении его можно улучшить, если придерживаться соглашения о прямом порядке обхода из алгорит-

ма B. В упр. 25 представлен алгоритм, который генерирует все связанные деревья или леса путем изменения на каждом шаге только двух связей, сохраняя прямой порядок обхода. Одна связь становится нулевой, в то время как вторая — ненулевой. Этот алгоритм "обрезки и прививки", последний из трех обещанных "красивых алгоритмов для кодов Грея", имеет единственный недостаток: его управляющий механизм существенно сложнее, чем у алгоритма L, так что ему требуется примерно на 40% больше времени на вычисления.

Количество деревьев. Существует простая формула для общего количества генерируемых алгоритмами P, B, N и L деревьев, а именно

$$C_n \;=\; \frac{1}{n+1}\binom{2n}{n} \;=\; \binom{2n}{n} - \binom{2n}{n-1}; \tag{15}$$

этот факт был доказан в 2.3.4.4–(14). Первыми несколькими значениями являются

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13
C_n =	1	1	2	5	14	42	132	429	1430	4862	16796	58786	208012	742900

Они называются *числами Каталана* по имени Эжена Каталана (Eugène Catalan), который написал о них несколько важных статей [*Journal de math.* **3** (1838), 508–516; **4** (1839), 95–99]. Приближение Стирлинга дает нам асимптотическое значение

$$C_n \;=\; \frac{4^n}{\sqrt{\pi}\,n^{3/2}}\Bigl(1 - \frac{9}{8n} + \frac{145}{128n^2} - \frac{1155}{1024n^3} + \frac{36939}{32768n^4} + O(n^{-5})\Bigr); \tag{16}$$

в частности, можно заключить, что

$$\frac{C_{n-k}}{C_n} \;=\; \frac{1}{4^k}\Bigl(1 + \frac{3k}{2n} + O\Bigl(\frac{k^2}{n^2}\Bigr)\Bigr) \qquad \text{при } |k| \le \frac{n}{2}. \tag{17}$$

(Само собой, C_{n-1}/C_n в точности равно $(n+1)/(4n-2)$ согласно формуле (15).) В разделе 2.3.4.4 была также выведена производящая функция

$$C(z) \;=\; C_0 + C_1 z + C_2 z^2 + C_3 z^3 + \cdots \;=\; \frac{1-\sqrt{1-4z}}{2z} \tag{18}$$

и доказана важная формула

$$[z^n]\,C(z)^r \;=\; \frac{r}{n+r}\binom{2n+r-1}{n} \;=\; \binom{2n+r-1}{n} - \binom{2n+r-1}{n-1}; \tag{19}$$

см. ответ к упр. 2.3.4.4–33 и уравнение (5.70) в *CMath*.

Эти факты дают нам более чем достаточно информации для анализа алгоритма P для лексикографической генерации вложенных скобок. Очевидно, что шаг P2 выполняется C_n раз; затем шаг P3 чаще всего вносит простое изменение и возвращается к шагу P2. Как часто происходит переход к шагу P4? На этот вопрос легко дать ответ: это количество раз, когда на шаге P2 $m = 2n-1$. А m — это положение крайней справа скобки '(', так что $m = 2n-1$ выполняется ровно C_{n-1} раз. Таким образом, вероятность того, что на шаге P3 устанавливается $m \leftarrow m-1$ и происходит возврат к шагу P2, равна $(C_n - C_{n-1})/C_n \approx 3/4$ согласно формуле (17). С другой стороны, пусть при переходе к шагу P4 нам нужно установить $a_j \leftarrow$ ')' и $a_k \leftarrow$ '(' на этом шаге ровно $h-1$ раз. Количество случаев, когда $h > x$, равно количеству вложенных строк длиной $2n$, заканчивающихся тривиальными парами

$()\ldots()$ в количестве x штук, а именно C_{n-x}. Таким образом, общее количество изменений a_j и a_k на шаге P4 с учетом (17) составляет

$$\begin{aligned} C_{n-1}+C_{n-2}+\cdots+C_1 &= C_n\Bigl(\frac{C_{n-1}}{C_n}+\frac{C_{n-2}}{C_n}+\cdots+\frac{C_1}{C_n}\Bigr) \\ &= \frac{1}{3}C_n\Bigl(1+\frac{2}{n}+O\Bigl(\frac{1}{n^2}\Bigr)\Bigr); \end{aligned} \tag{20}$$

итак, сделанное ранее утверждение об эффективности алгоритма доказано.

Более глубокому пониманию способствует изучение рекурсивной структуры (5), лежащей в основе алгоритма P. Последовательности A_{pq} в этой формуле имеют C_{pq} элементов, где

$$C_{pq}=C_{p(q-1)}+C_{(p-1)q} \quad \text{при } 0\le p\le q\ne 0; \qquad C_{00}=1; \tag{21}$$

а при $p<0$ или $p>q$ значение $C_{pq}=0$. Так можно сформировать следующий треугольный массив.

$$\begin{array}{lllllll} C_{00} & & & & & & \\ C_{01} & C_{11} & & & & & \\ C_{02} & C_{12} & C_{22} & & & & \\ C_{03} & C_{13} & C_{23} & C_{33} & & & \\ C_{04} & C_{14} & C_{24} & C_{34} & C_{44} & & \\ C_{05} & C_{15} & C_{25} & C_{35} & C_{45} & C_{55} & \\ C_{06} & C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66} \end{array} = \begin{array}{lllllll} 1 & & & & & & \\ 1 & 1 & & & & & \\ 1 & 2 & 2 & & & & \\ 1 & 3 & 5 & 5 & & & \\ 1 & 4 & 9 & 14 & 14 & & \\ 1 & 5 & 14 & 28 & 42 & 42 & \\ 1 & 6 & 20 & 48 & 90 & 132 & 132 \end{array} \tag{22}$$

Каждый его элемент представляет собой сумму ближайших соседей сверху и слева; на диагонали в нем находятся числа Каталана $C_n=C_{nn}$. Элементы этого треугольника имеют древнее происхождение, ведущее отсчет от работы де Муавра (de Moivre) 1711 года, и называются баллотировочными числами (ballot numbers), поскольку представляют последовательности $p+q$ бюллетеней, для которых подсчет голосов ни в какой момент не покажет перевеса кандидата с p голосами над оппонентом с q голосами. Общую формулу

$$C_{pq} = \frac{q-p+1}{q+1}\binom{p+q}{p} = \binom{p+q}{p}-\binom{p+q}{p-1} \tag{23}$$

можно доказать как по индукции, так и многими другими способами (см. упр. 39 и ответ к упр. 2.2.1–4). Заметим, что согласно (19)

$$C_{pq} = [z^p]\,C(z)^{q-p+1}. \tag{24}$$

При $n=4$ алгоритм P, по сути, описывает дерево рекурсии

, (25)

поскольку из спецификации (5) вытекает, что $A_{nn}=(\,A_{(n-1)n}$ и что

$$\begin{aligned} A_{pq} = {}&)^{q-p}(\,A_{(p-1)p},\ )^{q-p-1}(\,A_{(p-1)(p+1)},\ )^{q-p-2}(\,A_{(p-1)(p+2)}, \\ &\ldots,\ (\,A_{(p-1)q} \qquad \text{при } 0\le p<q. \end{aligned} \tag{26}$$

Количество листьев ниже узла pq в этом дереве рекурсии равно C_{pq}, а узел pq встречается на уровне $n-1-p$ ровно $C_{(n-q)(n-1-p)}$ раз; таким образом, мы должны получить

$$\sum_q C_{(n-q)(n-1-p)} C_{pq} \;=\; C_n \qquad \text{для } 0 \le p < n. \tag{27}$$

Четырнадцать листьев (25) слева направо соответствуют четырнадцати строкам табл. 1 сверху вниз. Заметим, что записи в столбце $c_1c_2c_3c_4$ этой таблицы назначают листьям дерева (25) числа 0000, 0001, 0010, ..., 0123 в соответствии с "десятичной записью Дьюи" для узлов дерева (но с индексами, начинающимися с нуля, а не с единицы, и с дополнительным нулем в начале).

Червяк, который ползет от одного листа к следующему по низу дерева рекурсии, будет подниматься и опускаться на h уровней при изменении h координат $c_1 \ldots c_n$, т. е. когда алгоритм P сбрасывает значения h '(' и h ')'. Это наблюдение облегчает понимание предыдущего вывода о том, что условие $h > x$ выполняется ровно C_{n-x} раз в процессе движения червяка.

Еще один способ понимания алгоритма P возникает при рассмотрении бесконечного ориентированного графа, определяемого рекурсией (21).

(28)

Очевидно, что C_{pq} в силу (21) представляет собой количество путей в этом ориентированном графе от pq к 00. И в самом деле, каждая строка скобок в A_{pq} непосредственно соответствует такому пути, где '(' означает шаг влево, а ')' — шаг вверх. Алгоритм P систематически исследует все такие пути, пытаясь при продолжении частичного пути сначала перейти вверх.

Таким образом, легко определить N-ю строку вложенных скобок, посещаемую алгоритмом P, начиная с узла nn и выполняя следующее, находясь в узле pq: если $p = q = 0$, завершить работу; в противном случае, если $N \le C_{p(q-1)}$, вывести ')', установить $q \leftarrow q-1$ и продолжить работу; в противном случае установить $N \leftarrow N - C_{p(q-1)}$, вывести '(', установить $p \leftarrow p-1$ и продолжить работу. Приведенный далее алгоритм [Frank Ruskey, Ph.D. thesis (University of California at San Diego, 1978), 16–24] избегает предвычисления треугольника Каталана путем вычисления C_{pq} "на лету".

Алгоритм U (*Строка вложенных скобок по ее рангу*). Для заданных n и N, где $1 \le N \le C_n$, данный алгоритм вычисляет N-й вывод $a_1 \ldots a_{2n}$ алгоритма P.

U1. [Инициализация.] Установить $q \leftarrow n$ и $m \leftarrow p \leftarrow c \leftarrow 1$. Пока $p < n$, устанавливать $p \leftarrow p+1$ и $c \leftarrow ((4p-2)c)/(p+1)$.

U2. [Выполнено?] Завершить работу алгоритма, если $q = 0$.

U3. [Вверх?] Установить $c' \leftarrow ((q+1)(q-p)c)/((q+p)(q-p+1))$. (В этот момент $1 \le N \le c = C_{pq}$ и $c' = C_{p(q-1)}$.) Если $N \le c'$, установить $q \leftarrow q-1$, $c \leftarrow c'$, $a_m \leftarrow$ ‘)’, $m \leftarrow m+1$ и вернуться к шагу U2.

U4. [Влево.] Установить $p \leftarrow p-1$, $c \leftarrow c-c'$, $N \leftarrow N-c'$, $a_m \leftarrow$ ‘(’, $m \leftarrow m+1$ и вернуться к шагу U3. ▮

Случайные деревья. Строку вложенных скобок $a_1a_2\ldots a_{2n}$ можно получить случайным образом, просто применив алгоритм U к случайному целому числу N между 1 и C_n. Однако эта идея не слишком хороша при n, превышающем 32 или около того, поскольку C_n может быть очень большим. Более простой и лучший способ предложен в работе D. B. Arnold and M. R. Sleep, *ACM Trans. Prog. Languages and Systems* **2** (1980), 122–128, и заключается в генерации случайного “путешествия червяка”, начиная с узла $\textcircled{nn}$ в графе (28) и выполняя ветвления влево и вверх с соответствующими вероятностями. Полученный алгоритм практически такой же, как и алгоритм U, но работает только с неотрицательными целыми числами, меньшими n^2+n+1.

Алгоритм W (*Равномерно распределенные случайные строки вложенных скобок*). Этот алгоритм генерирует случайную строку $a_1a_2\ldots a_{2n}$ корректно вложенных скобок (и).

W1. [Инициализация.] Установить $p \leftarrow q \leftarrow n$ и $m \leftarrow 1$.

W2. [Выполнено?] Завершить работу алгоритма, если $q = 0$.

W3. [Вверх?] Пусть X — случайное целое число в диапазоне $0 \le X < (q+p)(q-p+1)$. Если $X < (q+1)(q-p)$, установить $q \leftarrow q-1$, $a_m \leftarrow$ ‘)’, $m \leftarrow m+1$ и вернуться к шагу W2.

W4. [Влево.] Установить $p \leftarrow p-1$, $a_m \leftarrow$ ‘(’, $m \leftarrow m+1$, и вернуться к шагу W3. ▮

Путь червяка можно рассматривать как последовательность $w_0w_1\ldots w_{2n}$, где w_m — текущая глубина червяка после m шагов. Таким образом, $w_0 = 0$; $w_m = w_{m-1}+1$, если $a_m =$ ‘(’; и $w_m = w_{m-1}-1$, если $a_m =$ ‘)’; а кроме того, $w_m \ge 0$, $w_{2n} = 0$. Последовательность $w_0w_1\ldots w_{30}$, соответствующая (1) и (2), представляет собой 0121012321234345432321232343210. На шаге W3 алгоритма W имеем $q+p = 2n+1-m$ и $q-p = w_{m-1}$.

Назовем *контуром* (*outline*) леса путь, который проходит через точки плоскости $(m, -w_m)$, где $0 \le m \le 2n$, а $w_0w_1\ldots w_{2n}$ — маршрут червяка, соответствующий связанной строке $a_1\ldots a_{2n}$ вложенных скобок. На рис. 57 показано, что будет, если мы изобразим контуры всех 50-узловых лесов, причем степень затенения каждой точки соответствует количеству лесов, лежащих над ней. Например, w_1 всегда равно единице, так что треугольная область вверху слева на рис. 57 имеет максимально черный цвет. w_2 может принимать значения 0 или 2, причем 0 встречается в $C_{49} \approx C_{50}/4$ случаях; так что соответствующая ромбовидная область леса затенена на 75%. Таким образом, рис. 57 иллюстрирует форму случайного леса, аналогичную формам случайных разбиений, которые мы видели на рис. 50, 51 и 55 разделов 7.2.1.4 и 7.2.1.5.

Конечно, в действительности мы не в состоянии изобразить контуры всех этих лесов, поскольку их общее количество — $C_{50} = 1\,978\,261\,657\,756\,160\,653\,623\,774\,456$. Но с помощью математики можно имитировать такое действие. Вероятность того,

Рис. 57. Форма случайного 50-узлового леса.

Рис. 58. Положение внутренних узлов случайного 50-узлового бинарного дерева.

что $w_{2m} = 2k$, составляет $C_{(m-k)(m+k)}C_{(n-m-k)(n-m+k)}/C_n$, поскольку всего имеется $C_{(m-k)(m+k)}$ способов начать с $m+k$ левых скобок '(' и $m-k$ правых скобок ')' и $C_{(n-m-k)(n-m+k)}$ способов закончить $n-(m+k)$ левыми скобками '(' и $n-(m-k)$ правыми скобками ')'. С использованием формулы (23) и приближения Стирлинга получаем, что эта вероятность равна

$$\frac{(2k+1)^2(n+1)}{(m+k+1)(n-m+k+1)}\binom{2m}{m-k}\binom{2n-2m}{n-m+k}\Big/\binom{2n}{n}$$
$$= \frac{(2k+1)^2}{\sqrt{\pi}\,\bigl(\theta(1-\theta)n\bigr)^{3/2}}\,e^{-k^2/(\theta(1-\theta)n)}\Bigl(1+O\Bigl(\frac{k+1}{n}\Bigr)+O\Bigl(\frac{k^3}{n^2}\Bigr)\Bigr)\,, \qquad (29)$$

где $m = \theta n$ и $n \to \infty$ для $0 < \theta < 1$. Среднее значение w_{2m} выводится в упр. 57; оно равно

$$\frac{(4m(n-m)+n)\binom{2m}{m}\binom{2n-2m}{n-m}}{n\binom{2n}{n}} - 1 \;=\; 4\sqrt{\frac{\theta(1-\theta)n}{\pi}} - 1 + O\Bigl(\frac{1}{\sqrt{n}}\Bigr) \qquad (30)$$

и показано для $n = 50$ на рис. 57 в виде кривой.

Когда n велико, маршрут червяка приближается к так называемому броуновскому движению, представляющему собой важную концепцию в теории вероятно-

Рис. 59. Положение внешних узлов случайного 50-узлового бинарного дерева.

стей. См., например, Paul Lévy, *Processus Stochastiques et Mouvement Brownien* (1948), 225–237; Guy Louchard, *J. Applied Prob.* **21** (1984), 479–499, и *BIT* **26** (1986), 17–34; David Aldous, *Electronic Communications in Probability* **3** (1998), 79–90; Jon Warren, *Electronic Communications in Probability* **4** (1999), 25–29; J.-F. Marckert, *Random Structures & Algorithms* **24** (2004), 118–132.

Какой вид имеет случайное *бинарное* дерево? Этот вопрос был исследован Фрэнком Раски (Frank Ruskey) в *SIAM J. Algebraic and Discrete Methods* **1** (1980), 43–50, и ответ на него достаточно интересен. Предположим, что мы изобразили бинарное дерево, как в (4), с m-м внутренним узлом в горизонтальной позиции m, где узлы пронумерованы в симметричном порядке обхода. Если изобразить все 50-узловые бинарные деревья таким образом и наложить их одно на другое, то получится распределение положений узлов, показанное на рис. 58. Аналогично, если пронумеровать *внешние* узлы от 0 до n в симметричном порядке обхода и разместить их в горизонтальных позициях .5, 1.5, $\ldots$, $n + .5$, то "бахрома" от всех 50-узловых деревьев образует распределение, показанное на рис. 59. Обратите внимание, что корневой узел с наибольшей вероятностью имеет номер 1 или n, становясь крайним слева или справа; менее всего вероятно, что он находится посредине, в позиции $\lfloor (n+1)/2 \rfloor$ или $\lceil (n+1)/2 \rceil$.

Как и на рис. 57, гладкие кривые на рис. 58 и 59 показывают среднюю глубину узла; точные формулы выводятся в упр. 58 и 59. Асимптотически средняя глубина внешнего узла m равна

$$8\sqrt{\frac{\theta(1-\theta)n}{\pi}} - 1 + O\Big(\frac{1}{\sqrt{n}}\Big) \qquad \text{при } m = \theta n \text{ и } n \to \infty \tag{31}$$

для всех фиксированных отношений $0 < \theta < 1$, что удивительно похоже на формулу (30); средняя глубина *внутреннего* узла m асимптотически та же, но с '-1', замененным на '-3'. Таким образом, можно сказать, что *в среднем форма бинарного*

дерева приближенно представляет собой нижнюю половину эллипса шириной n единиц и глубиной $4\sqrt{n/\pi}$ уровней.

Три других заслуживающих упоминания способа генерации случайных закодированных лесов рассматриваются в упр. 60–62. Они менее непосредственны, чем алгоритм W, но представляют собой значительный комбинаторный интерес. Первый начинается с произвольной случайной строки, содержащей n левых скобок '(' и n правых скобок ')', не обязательно вложенных; каждая из $\binom{2n}{n}$ возможных строк равновероятна. Затем строка обрабатывается для преобразования в последовательность корректно вложенных скобок таким образом, что к каждому окончательному результату приводят ровно $n+1$ исходных строк. Второй метод аналогичен, но начинается с последовательности $n+1$ нулей и n двоек, отображая их таким образом, что к каждому конечному результату приводят ровно $2n+1$ исходных строк. Третий метод дает конечный результат ровно из n битовых строк, содержащих $n-1$ единиц и $n+1$ нулей. Другими словами, эти три метода предоставляют комбинаторное доказательство того, что C_n одновременно равно $\binom{2n}{n}/(n+1)$, $\binom{2n+1}{n}/(2n+1)$ и $\binom{2n}{n-1}/n$. Например, при $n=4$ мы получаем $14 = 70/5 = 126/9 = 56/4$.

Если мы хотим генерировать случайные бинарные деревья непосредственно в связанном виде, можно воспользоваться красивым методом, предложенным Ж.-Л. Реми (J.-L. Rémy) [*RAIRO Informatique Théorique* **19** (1985), 179–195]. Его подход особенно поучителен, поскольку показывает, как случайные деревья Каталана могут возникнуть "в природе", с использованием удивительно простого механизма, основанного на классической идее Олинде Родригеса (Olinde Rodrigues) [*J. de Math.* **3** (1838), 549]. Предположим, что наша цель — получение не просто обычного n-узлового бинарного дерева, но *декорированного* бинарного дерева, т. е. расширенного бинарного дерева, в котором внешние узлы помечены числами от 0 до n в некотором порядке. Всего имеется $(n+1)!$ способов декорировать любое заданное бинарное дерево; так что общее количество декорированных бинарных деревьев с n внутренними узлами составляет

$$D_n = (n+1)!\,C_n = \frac{(2n)!}{n!} = (4n-2)\,D_{n-1}. \tag{32}$$

Реми обнаружил, что существует $4n-2$ простых пути построения декорированного дерева порядка n из данного декорированного дерева порядка $n-1$: мы просто выбираем любой узел из $2n-1$ узлов (внутренних и внешних) данного дерева, скажем, узел x, и заменяем его

либо на

, либо на

, (33)

таким образом вставляя новый внутренний узел и новый лист, перемещая тем самым x вместе с его потомками (если таковые имеются) на один уровень вниз.

Например, вот один из вариантов построения декорированного дерева порядка 6:

. (34)

Обратите внимание, что каждое декорированное дерево получается с помощью данного процесса единственным способом, поскольку предшественник каждого дерева должен быть деревом, которое получается путем вычеркивания листа с максимальным номером. Таким образом, построение Реми дает равномерно распределенные декорированные деревья; и если проигнорировать внешние узлы, то мы получим случайные бинарные деревья обычного, недекорированного, вида.

Один привлекательный способ реализации процедуры Реми состоит в использовании таблицы связей $L_0L_1\ldots L_{2n}$, в которой внешние узлы (листья) имеют четные номера, а внутренние (ветвления) — нечетные. Корневым является узел L_0; левым и правым дочерними узлами внутреннего узла $2k-1$ являются соответственно узлы L_{2k-1} и L_{2k} для $1 \le k \le n$. В этом случае программа оказывается короткой и приятной.

Алгоритм R (*Растущее случайное бинарное дерево*). Этот алгоритм строит представление с использованием связей $L_0L_1\ldots L_{2N}$ (с указанными выше соглашениями) равномерно распределенного случайного дерева с N внутренними узлами.

R1. [Инициализация.] Установить $n \leftarrow 0$ и $L_0 \leftarrow 0$.

R2. [Выполнено?] (В этот момент связи $L_0L_1\ldots L_{2n}$ представляют собой случайное n-узловое бинарное дерево.) Завершить работу алгоритма, если $n = N$.

R3. [Продвижение n.] Пусть X — случайное целое число от 0 до $4n+1$ включительно. Установить $n \leftarrow n+1$, $b \leftarrow X \bmod 2$, $k \leftarrow \lfloor X/2 \rfloor$, $L_{2n-b} \leftarrow 2n$, $L_{2n-1+b} \leftarrow L_k$, $L_k \leftarrow 2n-1$ и вернуться к шагу R2. ▮

***Цепочки подмножеств.** Теперь, когда мы прочно усвоили получение деревьев и строк скобок, самое время перейти к рассмотрению *шаблона рождественской елки*,* который представляет собой замечательный способ размещения всех 2^n битовых строк длиной n в $\binom{n}{\lfloor n/2 \rfloor}$ строках и $n+1$ столбцах, открытый Н. Г. де Брейном (N. G. de Bruijn), К. ван Эббенхорст Тенгберген (C. van Ebbenhorst Tengbergen) и Д. Круйсвейком (D. Kruyswijk) [*Nieuw Archief voor Wiskunde* (2) **23** (1951), 191–193].

Шаблон рождественской елки порядка 1 представляет собой единственную строку '0 1'; шаблон порядка 2 имеет вид

$$\begin{array}{ccc} & 10 & \\ 00 & 01 & 11 \end{array}. \tag{35}$$

В общем случае шаблон рождественской елки порядка $n+1$ получается, если взять каждую строку '$\sigma_1\ \sigma_2\ \ldots\ \sigma_s$' шаблона рождественской елки порядка n и заменить ее двумя строками:

$$\begin{array}{ccccc} & \sigma_2 0 & \ldots & \sigma_s 0 & \\ \sigma_1 0 & \sigma_1 1 & \ldots & \sigma_{s-1} 1 & \sigma_s 1 \end{array}. \tag{36}$$

(При $s = 1$ первая из этих строк опускается.)

Действуя таким образом, мы получим пример шаблона порядка 8, который приведен в табл. 4. По индукции легко убедиться в следующем:

i) каждая из 2^n битовых строк появляется в шаблоне ровно один раз;

ii) битовые строки с k единицами находятся в одном и том же столбце;

* Это название выбрано из сентиментальных соображений, поскольку внешний вид шаблона напоминает елку, а кроме того, эта тема была предметом девятой ежегодной "лекции рождественской елки" автора в Станфордском университете в декабре 2002 года.

Таблица 4

Шаблон рождественской елки порядка 8

				10101010				
			10101000	10101001	10101011			
				10101100				
			10100100	10100101	10101101			
			10100010	10100110	10101110			
		10100000	10100001	10100011	10100111	10101111		
				10110010				
			10110000	10110001	10110011			
				10110100				
			10010100	10010101	10110101			
			10010010	10010110	10110110			
		10010000	10010001	10010011	10010111	10110111		
				10111000				
			10011000	10011001	10111001			
			10001010	10011010	10111010			
		10001000	10001001	10001011	10011011	10111011		
			10001100	10011100	10111100			
		10000100	10000101	10001101	10011101	10111101		
		10000010	10000110	10001110	10011110	10111110		
	10000000	10000001	10000011	10000111	10001111	10011111	10111111	
				11001010				
			11001000	11001001	11001011			
				11001100				
			11000100	11000101	11001101			
			11000010	11000110	11001110			
		11000000	11000001	11000011	11000111	11001111		
				11010010				
			11010000	11010001	11010011			
				11010100				
			01010100	01010101	11010101			
			01010010	01010110	11010110			
		01010000	01010001	01010011	01010111	11010111		
				11011000				
			01011000	01011001	11011001			
			01001010	01011010	11011010			
		01001000	01001001	01001011	01011011	11011011		
			01001100	01011100	11011100			
		01000100	01000101	01001101	01011101	11011101		
		01000010	01000110	01001110	01011110	11011110		
	01000000	01000001	01000011	01000111	01001111	01011111	11011111	
				11100010				
			11100000	11100001	11100011			
				11100100				
			01100100	01100101	11100101			
			01100010	01100110	11100110			
		01100000	01100001	01100011	01100111	11100111		
				11101000				
			01101000	01101001	11101001			
			00101010	01101010	11101010			
		00101000	00101001	00101011	01101011	11101011		
			00101100	01101100	11101100			
		00100100	00100101	00101101	01101101	11101101		
		00100010	00100110	00101110	01101110	11101110		
	00100000	00100001	00100011	00100111	00101111	01101111	11101111	
				11110000				
			01110000	01110001	11110001			
			00110010	01110010	11110010			
		00110000	00110001	00110011	01110011	11110011		
			00110100	01110100	11110100			
		00010100	00010101	00110101	01110101	11110101		
		00010010	00010110	00110110	01110110	11110110		
	00010000	00010001	00010011	00010111	00110111	01110111	11110111	
			00111000	01111000	11111000			
		00011000	00011001	00111001	01111001	11111001		
		00001010	00011010	00111010	01111010	11111010		
	00001000	00001001	00001011	00011011	00111011	01111011	11111011	
		00001100	00011100	00111100	01111100	11111100		
	00000100	00000101	00001101	00011101	00111101	01111101	11111101	
	00000010	00000110	00001110	00011110	00111110	01111110	11111110	
00000000	00000001	00000011	00000111	00001111	00011111	00111111	01111111	11111111

iii) в пределах каждой строки каждая битовая строка получается из предыдущей путем единственного изменения нуля на единицу.

Если рассматривать битовые строки как представляющие подмножества множества $\{1, \ldots, n\}$, где единичные биты указывают члены множества, то свойство (iii) гласит, что каждая строка представляет *цепочку*, в которой каждое подмножество покрывается следующим за ним. В символьном виде с применением обозначений из раздела 7.1.3 можно записать: каждая строка $\sigma_1\ \sigma_2\ \ldots\ \sigma_s$ обладает тем свойством, что $\sigma_j \subseteq \sigma_{j+1}$ и $\nu(\sigma_{j+1}) = \nu(\sigma_j) + 1$ при $1 \le j < s$.

Свойства (i) и (ii) говорят о том, что если пронумеровать столбцы от 0 до n, то в столбце k будет находиться ровно $\binom{n}{k}$ элементов. Это наблюдение вместе с тем фактом, что каждая строка отцентрирована по столбцам, доказывает, что общее количество строк, как и утверждалось, равно $\max_{0\le k\le n}\binom{n}{k} = \binom{n}{\lfloor n/2\rfloor}$. Обозначим эту величину как M_n.

Множество битовых строк C называется *клаттером* (*clutter*),* или антицепочкой подмножеств, если его битовые строки несравнимы в том смысле, что $\sigma \not\subseteq \tau$, когда σ и τ являются различными элементами C. Знаменитая теорема Эмануэля Спернера (Emanuel Sperner) [*Math. Zeitschrift* **27** (1928), 544–548] утверждает, что ни один клаттер на множестве $\{1, \ldots, n\}$ не может иметь более чем M_n элементов; шаблон рождественской елки предоставляет простое доказательство, поскольку ни один клаттер не может содержать более одного элемента из каждой строки.

На самом деле шаблон рождественской елки можно использовать для того, чтобы доказать существенно большее. Сначала заметим, что имеется ровно $\binom{n}{k} - \binom{n}{k-1}$ строк длиной $n + 1 - 2k$ при $0 \le k \le n/2$, поскольку в столбце k ровно $\binom{n}{k}$ элементов. Например, в табл. 4 имеется одна строка длиной 9, а именно нижняя; в этой таблице также $\binom{8}{1} - \binom{8}{0} = 7$ строк длиной 7, $\binom{8}{2} - \binom{8}{1} = 20$ строк длиной 5, $\binom{8}{3} - \binom{8}{2} = 28$ строк длиной 3 и $\binom{8}{4} - \binom{8}{3} = 14$ строк длиной 1. Кроме того, эти числа $\binom{n}{k} - \binom{n}{k-1}$ появляются в треугольнике Каталана (22), поскольку они равны $C_{k(n-k)}$ в соответствии с формулой (23).

Дальнейшее изучение показывает, что эта связь с числами Каталана — не простое совпадение; ключом к более глубокому пониманию шаблона рождественской елки являются вложенные скобки, потому что теория скобок говорит нам, где в массиве располагается произвольная битовая строка. Предположим, что вместо единицы и нуля мы используем символы (и) соответственно. Любая строка скобок, вложенных или нет, может быть записана единственным способом в виде

$$\alpha_0\,)\,\ldots\,\alpha_{p-1}\,)\,\alpha_p\,(\alpha_{p+1}\,\ldots\,(\alpha_q \tag{37}$$

для некоторых p и q $(0 \le p \le q)$, где подстроки $\alpha_0, \ldots, \alpha_q$ — корректно вложенные (возможно, пустые). Ровно p правых скобок и $q - p$ левых скобок "свободны" в том смысле, что не имеют соответствующей пары. Например, у строки

$$)\,\underbrace{(\,(\,)\,)\,(\,)}\,)\,\underbrace{(\,)}\,)\,)\,)\,(\,(\,(\,(\,(\,\underbrace{(\,)}\,\underbrace{(\,(\,)\,(\,)}\,\underbrace{(\,(\,(\,)}\,) \tag{38}$$

$p = 5$, $q = 12$, $\alpha_0 = \epsilon$, $\alpha_1 = (()\,)\,()$, $\alpha_2 = ()$, $\alpha_3 = \epsilon$, $\ldots$, $\alpha_{12} = (())$. В общем случае строка (37) является частью цепочки длиной $q + 1$,

$$\alpha_0\,)\,\ldots\,\alpha_{q-1}\,)\,\alpha_q,\quad \alpha_0\,)\,\ldots\,\alpha_{q-2}\,)\,\alpha_{q-1}\,(\alpha_q,\quad \ldots,\quad \alpha_0\,(\alpha_1\,\ldots\,(\alpha_q, \tag{39}$$

* Дословно — "беспорядком". — *Примеч. пер.*

которая начинается с q свободных правых скобок) и заменяет их по одной свободными левыми скобками (. Каждая строка шаблона рождественской елки получается точно таким же способом, но с использованием единицы и нуля вместо (и); если цепочка $\sigma_1 \ \ldots \ \sigma_s$ соответствует строкам вложенных скобок $\alpha_0, \ldots, \alpha_{s-1}$, то следующие за ней цепочки в (36) соответствуют строкам $\alpha_0, \ldots, \alpha_{s-3}, \alpha_{s-2}(\alpha_{s-1})$ и $\alpha_0, \ldots, \alpha_{s-3}, \alpha_{s-2}, \alpha_{s-1}, \epsilon$ соответственно. [См. Curtis Greene and Daniel J. Kleitman, *J. Combinatorial Theory* **A20** (1976), 80–88.]

Обратите также внимание, что крайние справа элементы в каждой строке шаблона, такие, как 10101010, 10101011, 10101100, 10101101, ..., 11111110, 11111111 в случае $n = 8$, находятся в лексикографическом порядке. Таким образом, например, четырнадцать строк длиной 1 в табл. 4 в точности соответствуют четырнадцати строкам вложенных скобок в табл. 1. Это наблюдение позволяет легко генерировать строки табл. 4 последовательно, снизу вверх, с помощью метода, аналогичного алгоритму P; см. упр. 77.

Пусть $f(x_1, \ldots, x_n)$ — произвольная монотонная булева функция от n переменных. Если $\sigma = a_1 \ldots a_n$ — произвольная битовая строка длиной n, для удобства можно записывать $f(\sigma) = f(a_1, \ldots, a_n)$. Любая строка $\sigma_1 \ \ldots \ \sigma_s$ шаблона рождественской елки образует цепочку, так что мы имеем

$$0 \ \le \ f(\sigma_1) \ \le \ \cdots \ \le \ f(\sigma_s) \ \le \ 1. \tag{40}$$

Другими словами, имеется индекс t, такой, что $f(\sigma_j) = 0$ при $j < t$ и $f(\sigma_j) = 1$ при $j \ge t$; нам будут известны значения $f(\sigma)$ для всех 2^n битовых строк σ, если мы узнаем индексы t для каждой строки шаблона.

Джордж Хансель (Georges Hansel) [*Comptes Rendus Acad. Sci.* (A) **262** (Paris, 1966), 1088–1090] заметил, что шаблон рождественской елки обладает еще одним важным свойством: если σ_{j-1}, σ_j и σ_{j+1} представляют собой три последовательных элемента любой строки, битовая строка

$$\sigma'_j \ = \ \sigma_{j-1} \oplus \sigma_j \oplus \sigma_{j+1} \tag{41}$$

находится в *предыдущей* строке. В самом деле, σ'_j находится в том же столбце, что и σ_j, и удовлетворяет условию

$$\sigma_{j-1} \ \subseteq \ \sigma'_j \ \subseteq \ \sigma_{j+1}; \tag{42}$$

это значение называется относительным дополнением σ_j в интервале $(\sigma_{j-1} \,..\, \sigma_{j+1})$. Наблюдение Ханселя легко доказать по индукции с помощью рекурсивного правила (36), определяющего шаблон рождественской елки. Он воспользовался им для того, чтобы показать, что можно вывести значения $f(\sigma)$ для всех σ, вычисляя значения функции в относительно небольшом количестве мест; если известно значение $f(\sigma'_j)$, то в силу (42) известно либо значение $f(\sigma_{j-1})$, либо значение $f(\sigma_{j+1})$.

Алгоритм H (*Исследование монотонной булевой функции*). Пусть $f(x_1, \ldots, x_n)$ — булева функция, неубывающая по каждой из булевых переменных (больше о ней ничего не известно). Обозначим для данной битовой строки σ длиной n через $r(\sigma)$ номер строки, в которой σ находится в шаблоне рождественской елки; очевидно, что $1 \le r(\sigma) \le M_n$. Для $1 \le m \le M_n$ обозначим через $s(m)$ количество битовых строк в строке m; пусть также $\chi(m, k)$ означает битовую строку в столбце k этой

строки при $(n+1-s(m))/2 \le k \le (n-1+s(m))/2$. Данный алгоритм определяет последовательность пороговых значений $t(1)$, $t(2)$, $\ldots$, $t(M_n)$, таких, что

$$f(\sigma) = 1 \quad \Longleftrightarrow \quad \nu(\sigma) \ge t\bigl(r(\sigma)\bigr), \tag{43}$$

путем вычисления f не более чем в двух точках на строку.

H1. [Цикл по m.] Выполнить шаги H2–H4 для $m = 1, \ldots, M_n$; после этого завершить работу алгоритма.

H2. [Начало строки m.] Установить $a \leftarrow (n+1-s(m))/2$ и $z \leftarrow (n-1+s(m))/2$.

H3. [Бинарный поиск.] Если $z \le a+1$, перейти к шагу H4. В противном случае установить $k \leftarrow \lfloor (a+z)/2 \rfloor$ и

$$\sigma \;\leftarrow\; \chi(m,k-1) \oplus \chi(m,k) \oplus \chi(m,k+1). \tag{44}$$

Если $k \ge t\bigl(r(\sigma)\bigr)$, установить $z \leftarrow k$; в противном случае установить $a \leftarrow k$. Повторить шаг H3.

H4. [Вычисление.] Если $f(\chi(m,a)) = 1$, установить $t(m) \leftarrow a$; в противном случае, если $a = z$, установить $t(m) \leftarrow a+1$; в противном случае установить $t(m) \leftarrow z + 1 - f(\chi(m,z))$. ▮

Алгоритм Ханселя *оптимален* в том смысле, что вычисляет f в наименьшем возможном количестве точек в наихудшем случае. Если f представляет собой пороговую функцию

$$f(\sigma) \;=\; \bigl[\nu(\sigma) > n/2\bigr], \tag{45}$$

любой корректный алгоритм, который исследует f на первых m строках шаблона рождественской елки, должен вычислять $f(\sigma)$ в столбце $\lfloor n/2 \rfloor$ каждой строки и в столбце $\lfloor n/2 \rfloor + 1$ каждой строки, размер которой больше 1. В противном случае невозможно отличить f от функции, которая не совпадает с ней только в неисследованных точках. [См. В. К. Коробков, *Проблемы кибернетики* **13** (1965), 5–28, теорема 5.]

Ориентированные деревья и леса. Обратимся теперь к другому виду деревьев, в которых важны отношения "родитель–потомок", но не порядок дочерних узлов в каждом семействе. *Ориентированный лес* из n узлов можно определить с помощью последовательности указателей $p_1 \ldots p_n$, где p_j — родительский по отношению к узлу j узел (если j — корень, $p_j = 0$); ориентированный граф с вершинами $\{0, 1, \ldots, n\}$ и дугами $\{j \to p_j \mid 1 \le j \le n\}$ не имеет ориентированных циклов. *Ориентированное дерево* представляет собой ориентированный лес с единственным корнем (см. раздел 2.3.4.2). Каждый n-узловой ориентированный лес эквивалентен $(n+1)$-узловому ориентированному дереву, поскольку корень этого дерева можно рассматривать как родительский узел для всех корневых узлов леса. В разделе 2.3.4.4 мы видели, что имеется A_n ориентированных деревьев с n узлами; вот первые несколько значений A_n.

$$\begin{array}{rcccccccccccccccc} n &=& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ A_n &=& 1 & 1 & 2 & 4 & 9 & 20 & 48 & 115 & 286 & 719 & 1842 & 4766 & 12486 & 32973 & 87811 \end{array} \tag{46}$$

Асимптотически $A_n = c\alpha^n n^{-3/2} + O(\alpha^n n^{-5/2})$, где $\alpha \approx 2.9558$ и $c \approx 0.4399$. Так, например, только 9 из 14 лесов в табл. 1 различны, если игнорировать порядок по горизонтали и рассматривать только вертикальную ориентацию.

Каждый ориентированный лес соответствует единственному упорядоченному лесу, если соответствующим образом отсортировать члены каждого семейства с использованием упорядочения деревьев, предложенного Г. Я. Скоинсом (H. I. Scoins) [*Machine Intelligence* **3** (1968), 43–60]: вспомним из (11), что упорядоченные леса могут характеризоваться кодами их уровней $c_1 \ldots c_n$, где c_j — уровень, на котором находится j-й в прямом порядке обхода узел. Упорядоченный лес называется *каноническим*, если последовательность кодов уровней поддеревьев в каждом семействе находится в невозрастающем лексикографическом порядке. Например, каноническими лесами в табл. 1 являются те леса, коды уровней $c_1c_2c_3c_4$ которых равны 0000, 0100, 0101, 0110, 0111, 0120, 0121, 0122 и 0123. Последовательность 0112 канонической не является, поскольку поддеревья корня имеют соответственно коды уровней 1 и 12; строка 1 лексикографически меньше строки 12. По индукции можно легко убедиться, что *канонические коды уровней лексикографически наибольшие среди всех способов переупорядочения поддеревьев данного ориентированного леса.*

Т. Бейер (T. Beyer) и С. М. Хедетниеми (S. M. Hedetniemi) [*SICOMP* **9** (1980), 706–712] заметили, что существует удивительно простой способ генерации ориентированных лесов, если посещать их в *уменьшающемся* лексикографическом порядке канонических кодов уровней. Предположим, что $c_1 \ldots c_n$ — канонический код, в котором $c_k > 0$ и $c_{k+1} = \cdots = c_n = 0$. Следующая наименьшая последовательность получается путем уменьшения c_k и увеличения $c_{k+1} \ldots c_n$ до наибольших уровней, согласующихся с условием каноничности; эти уровни достаточно легко вычислить. Если $j = p_k$ — узел, родительский по отношению к узлу k, имеем $c_j = c_k - 1 < c_l$ для $j < l \le k$, а следовательно, уровни $c_j \ldots c_k$ представляют поддерево, корнем которого является узел j. Для получения наибольшей последовательности уровней, меньших $c_1 \ldots c_n$, мы заменяем $c_k \ldots c_n$ первыми $n + 1 - k$ элементами бесконечной последовательности $(c_j \ldots c_{k-1})^\infty = c_j \ldots c_{k-1}c_j \ldots c_{k-1}c_j \ldots$. (Результатом будет удаление k из его текущей позиции крайнего справа дочернего по отношению к узлу j узла и добавление новых поддеревьев, являющихся “братьями” j, путем клонирования j и его наследников настолько часто, насколько это возможно. Этот процесс клонирования можно завершить посреди последовательности $c_j \ldots c_{k-1}$, но это не приведет к сложностям, поскольку каждый префикс канонической последовательности уровней является каноническим.) Например, для получения последующего элемента любой последовательности канонических кодов, заканчивающейся 23443433000000000, мы заменяем окончание 3000000000 на 2344343234.

Алгоритм O (*Ориентированные леса*). Этот алгоритм генерирует все ориентированные леса из n узлов путем посещения всех канонических n-узловых лесов в уменьшающемся лексикографическом порядке их кодов уровней $c_1 \ldots c_n$. Однако коды уровней при этом явно не вычисляются; каждый канонический лес представляется непосредственно последовательностью указателей на родительские узлы $p_1 \ldots p_n$ в порядке прямого обхода узлов. Для генерации всех ориентированных деревьев из $n + 1$ узлов можно представить, что корнем является узел 0. Алгоритм устанавливает $p_0 \leftarrow -1$.

O1. [Инициализация.] Установить $p_k \leftarrow k - 1$ для $0 \le k \le n$. (В частности, этот шаг делает ненулевым p_0 для использования при проверке условия завершения алгоритма; см. шаг O4.)

O2. [Посещение.] Посетить лес, представленный указателями на родительские узлы $p_1 \ldots p_n$.

O3. [Простой случай?] Если $p_n > 0$, установить $p_n \leftarrow p_{p_n}$ и вернуться к шагу O2.

O4. [Поиск j и k.] Найти наибольшее $k < n$, такое, что $p_k \neq 0$. Завершить работу алгоритма, если $k = 0$; в противном случае установить $j \leftarrow p_k$ и $d \leftarrow k - j$.

O5. [Клонирование.] Если $p_{k-d} = p_j$, установить $p_k \leftarrow p_j$; в противном случае установить $p_k \leftarrow p_{k-d} + d$. Вернуться к шагу O2, если $k = n$; в противном случае установить $k \leftarrow k + 1$ и повторить данный шаг. ▮

Как и в других рассмотренных алгоритмах, циклы на шагах O4 и O5 обычно достаточно коротки (см. упр. 88). В упр. 90 доказывается, что небольших изменений данного алгоритма достаточно для генерации всех размещений ребер, формирующих *свободные* деревья.

Остовные деревья. Теперь рассмотрим минимальные подграфы, которые "охватывают" данный граф. Если G — связный граф с n вершинами, его *остовными деревьями* (*spanning trees*) являются подмножества из $n-1$ ребер, не содержащие циклов; или, что эквивалентно, они представляет собой множества ребер, которые образуют свободное дерево, соединяющее все вершины. Остовные деревья играют важную роль во многих приложениях, особенно при изучении сетей, так что задача генерации всех остовных деревьев рассматривалась многими авторами. Фактически систематические способы их перечисления были разработаны в начале XX века Вильгельмом Фойсснером (Wilhelm Feussner) [*Annalen der Physik* (4) **9** (1902), 1304–1329] задолго до того, как стали задумываться о способах генерации других видов деревьев.

В дальнейшем рассмотрении мы допускаем, что граф может содержать любое количество ребер между двумя вершинами; однако петли (ребра, выходящие из вершины и возвращающиеся в нее же) запрещены, поскольку петля не может быть частью дерева. Основная идея Фойсснера очень проста и хорошо подходит для вычислений: если e — произвольное ребро графа G, то остовное дерево либо содержит его, либо нет. Предположим, что ребро e соединяет вершину u с вершиной v, и пусть оно является частью остовного дерева; тогда остальные $n-2$ ребер этого дерева покрывают граф $G \,/\, e$, который мы получим, рассматривая вершины u и v как идентичные. Другими словами, остовные деревья, которые содержат e, по сути, те же, что и остовные деревья сжатого графа $G \,/\, e$, который получается в результате сжатия ребра e в точку. С другой стороны, остовные деревья, которые *не* содержат e, являются остовными деревьями приведенного графа $G \setminus e$, полученного в результате удаления ребра e. Таким образом, множество $S(G)$ всех остовных деревьев графа G удовлетворяет соотношению

$$S(G) \;=\; e\,S(G \,/\, e) \;\cup\; S(G \setminus e). \tag{47}$$

В своей диссертации в Университете Виктории (1997) Малкольм Д. Смит (Malcolm J. Smith) представил красивый способ обеспечения рекурсии (47) путем поиска всех остовных деревьев в порядке "кода Грея двери-вертушки": каждое дерево в его схеме получается из предшествующего путем простого удаления одного ребра и подстановки другого. Найти такой порядок деревьев нетрудно, проблема в том, чтобы сделать это эффективно.

Основная идея алгоритма Смита состоит в генерации $S(G)$ таким образом, чтобы первое остовное дерево включало данное *близкое дерево* (*near tree*), состоящее из $n-2$ ребер и не содержащее циклов. При $n = 2$ эта задача тривиальна; мы просто перечисляем все ребра. При $n > 2$ и данном близком дереве $\{e_1, \ldots, e_{n-2}\}$ мы действуем следующим образом. Считаем, что граф G связный; в противном случае остовных деревьев не существует. Образуем G / e_1 и добавляем e_1 к каждому из его остовных деревьев, начиная с того, которое содержит $\{e_2, \ldots, e_{n-2}\}$; заметим, что $\{e_2, \ldots, e_{n-2}\}$ — близкое дерево для G / e_1, так что эта рекурсия имеет смысл. Если последним найденным таким путем остовным деревом для G/e_1 является $f_1 \ldots f_{n-2}$, завершим работу перечислением всех остовных деревьев для $G \setminus e_1$, начиная с того, которое содержит близкое дерево $\{f_1, \ldots, f_{n-2}\}$.

Предположим, например, что G представляет собой граф

$G =$

(48)

с четырьмя вершинами и пятью ребрами $\{p, q, r, s, t\}$. Начиная с близкого дерева $\{p, q\}$, процедура Смита сначала образует сжатый граф

$G / p =$

(49)

и перечисляет его остовные деревья, начиная с того, которое содержит $\{q\}$. Это может быть список qs, qt, ts, tr, rs; таким образом, деревья pqs, pqt, pts, ptr и prs покрывают G. Остается только перечислить остовные деревья графа

$G \setminus p =$

, (50)

начиная с того, которое содержит $\{r, s\}$; это деревья rsq, rqt, qts.

Детальная реализация алгоритма Смита весьма поучительна. Как обычно, мы представляем граф, используя для представления каждого ребра u — v две дуги — $u \longrightarrow v$ и $v \longrightarrow u$ — и поддерживая список "узлов дуг" для представления дуг, покидающих каждую вершину. Нам потребуется уменьшать и увеличивать количество ребер графа, так что эти списки лучше делать дважды связанными. Если a указывает на узел дуги, представляющей $u \longrightarrow v$, то

$a \oplus 1$ указывает "напарника" a, который представляет дугу $v \longrightarrow u$;
t_a является "верхушкой" a, т. е. v (следовательно, $t_{a\oplus 1} = u$);
i_a представляет собой необязательное имя, идентифицирующее данное ребро (эквивалентно $i_{a\oplus 1}$);
n_a указывает на следующий элемент в списке дуг u;
p_a указывает на предыдущий элемент в списке дуг u;
l_a представляет собой связь, используемую для восстановления дуг, как поясняется ниже.

Вершины представлены целыми числами $\{1, \ldots, n\}$; номер дуги $v - 1$ представляет собой головной узел дважды связанного списка дуг для вершины v. Головной узел

a распознается по тому факту, что его верхушка t_a равна нулю. Обозначим степень вершины v как d_v. Так, например, граф (48) может быть представлен с помощью $(d_1, d_2, d_3, d_4) = (2, 3, 3, 2)$ и следующих четырнадцати узлов дуг:

$$\begin{array}{rcccccccccccccc}
a = & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
t_a = & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 3 & 2 & 3 & 2 & 4 & 3 & 4 \\
i_a = & & & & & p & p & q & q & r & r & s & s & t & t \\
n_a = & 5 & 4 & 6 & 10 & 9 & 7 & 8 & 0 & 13 & 11 & 12 & 1 & 3 & 2 \\
p_a = & 7 & 11 & 13 & 12 & 1 & 0 & 2 & 5 & 6 & 4 & 3 & 9 & 10 & 8
\end{array}$$

Управлять неявной рекурсией алгоритма Смита удобно с помощью массива указателей на дуги $a_1 \ldots a_{n-1}$. На уровне l процесса ду́ги $a_1 \ldots a_{l-1}$ означают ребра, которые были включены в текущее остовное дерево; a_l игнорируется, а дуги $a_{l+1} \ldots a_{n-1}$ обозначают ребра близкого дерева сжатого графа $(\ldots(G/a_1)\ldots)/a_{l-1}$, которое должно быть частью остовного дерева, посещаемого следующим.

Существует и другой массив указателей на дуги $s_1 \ldots s_{n-2}$, представляющий стеки дуг, временно удаленных из текущего графа. Верхним элементом стека для уровня l является s_l, и каждая дуга a связывается со следующей за ней, l_a (которая на дне стека равна нулю).

Ребро, удаление которого разъединяет связный граф, называется *мостом*. Одним из ключевых моментов данного алгоритма является то, что мы хотим сохранять текущий граф связным; следовательно, мы не можем устанавливать $G \leftarrow G \setminus e$, если e представляет собой мост.

Алгоритм S (*Все остовные деревья*). Этот алгоритм посещает все остовные деревья заданного связного графа, представленного структурами данных, рассмотренными выше.

Для удаления и восстановления элементов в дважды связанных списках здесь используется метод "танцующих связей", который будет всесторонне рассмотрен в разделе 7.2.2.1. Обозначение "delete(a)" в описании шагов алгоритма представляет собой сокращенную запись пары операций

$$n_{p_a} \leftarrow n_a, \quad p_{n_a} \leftarrow p_a; \tag{51}$$

аналогично "undelete(a)" означает

$$p_{n_a} \leftarrow a, \quad n_{p_a} \leftarrow a. \tag{52}$$

S1. [Инициализация.] Установить $a_1 \ldots a_{n-1}$ равным остовному дереву графа (см. упр. 94.) Установить также $x \leftarrow 0$, $l \leftarrow 1$ и $s_1 \leftarrow 0$. Если $n = 2$, установить $v \leftarrow 1$, $e \leftarrow n_0$ и перейти к шагу S5.

S2. [Вход на уровень l.] Установить $e \leftarrow a_{l+1}$, $u \leftarrow t_e$ и $v \leftarrow t_{e \oplus 1}$. Если $d_u > d_v$, выполнить обмен $v \leftrightarrow u$ и установить $e \leftarrow e \oplus 1$.

S3. [Сжатие e.] (Сейчас u будет сделано идентичным v путем вставки списка смежности u в список v. Следует также удалить все бывшие ребра между u и v, включая само e, поскольку иначе такие ребра станут петлями. Удаленные ребра связываются вместе, так что мы сможем восстановить их на шаге S7.) Установить $k \leftarrow d_u + d_v$, $f \leftarrow n_{u-1}$ и $g \leftarrow 0$. Пока $t_f \neq 0$, выполнять следующее: если $t_f = v$, delete(f), delete($f \oplus 1$), и установить $k \leftarrow k - 2$, $l_f \leftarrow g$, $g \leftarrow f$;

в противном случае установить $t_{f\oplus 1} \leftarrow v$. Затем установить $f \leftarrow n_f$ и повторять эти операции до тех пор, пока не будет выполнено условие $t_f = 0$. Наконец установить $l_e \leftarrow g$, $d_v \leftarrow k$, $g \leftarrow v - 1$, $n_{p_f} \leftarrow n_g$, $p_{n_g} \leftarrow p_f$, $p_{n_f} \leftarrow g$, $n_g \leftarrow n_f$ и $a_l \leftarrow e$.

S4. [Продвижение l.] Установить $l \leftarrow l + 1$. Если $l < n - 1$, установить $s_l \leftarrow 0$ и вернуться к шагу S2. В противном случае установить $e \leftarrow n_{v-1}$.

S5. [Посещение.] (Текущий граф в настоящий момент имеет только две вершины, одна из которых — v.) Установить $a_{n-1} \leftarrow e$ и посетить остовное дерево $a_1 \ldots a_{n-1}$. (Если $x = 0$, это первое посещаемое остовное дерево; в противном случае оно отличается от своего предшественника удалением x и вставкой e.) Установить $x \leftarrow e$ и $e \leftarrow n_e$. Повторить шаг S5, если $t_e \neq 0$.

S6. [Уменьшение l.] Установить $l \leftarrow l - 1$. Завершить работу алгоритма, если $l = 0$; в противном случае установить $e \leftarrow a_l$, $u \leftarrow t_e$ и $v \leftarrow t_{e\oplus 1}$.

S7. [Восстановление e.] Установить $f \leftarrow u - 1$, $g \leftarrow v - 1$, $n_g \leftarrow n_{p_f}$, $p_{n_g} \leftarrow g$, $n_{p_f} \leftarrow f$, $p_{n_f} \leftarrow f$ и $f \leftarrow p_f$. Пока $t_f \neq 0$, устанавливать $t_{f\oplus 1} \leftarrow u$ и $f \leftarrow p_f$. Затем установить $f \leftarrow l_e$, $k \leftarrow d_v$; пока $f \neq 0$, выполнять следующее: установить $k \leftarrow k + 2$, undelete$(f \oplus 1)$, undelete(f) и установить $f \leftarrow l_f$. Наконец установить $d_v \leftarrow k - d_u$.

S8. [Проверка моста.] Если e — мост, перейти к шагу S9. (См. один из способов выполнения данной проверки в упр. 95.) В противном случае установить $x \leftarrow e$, $l_e \leftarrow s_l$, $s_l \leftarrow e$; delete(e) и delete$(e \oplus 1)$. Установить $d_u \leftarrow d_u - 1$, $d_v \leftarrow d_v - 1$ и перейти к шагу S2.

S9. [Отмена удалений на уровне l.] Установить $e \leftarrow s_l$. Пока $e \neq 0$, установить $u \leftarrow t_e$, $v \leftarrow t_{e\oplus 1}$, $d_u \leftarrow d_u + 1$, $d_v \leftarrow d_v + 1$, undelete$(e \oplus 1)$, undelete(e) и $e \leftarrow l_e$. Вернуться к шагу S6. ▮

Вы можете выполнить шаги алгоритма вручную на небольшом графе, таком, как (48). Обратите внимание на тонкий момент, возникающий на шагах S3 и S7, когда список смежности u становится пустым. Заметим также, что возможно некоторое сокращение алгоритма ценой его усложнения; такие улучшения будут рассмотрены позже в данном разделе.

***Последовательно-параллельные графы.** Задача поиска всех остовных деревьев становится особенно простой, когда данный граф имеет последовательное и/или параллельное разложение. *Последовательно-параллельный граф между s и t* представляет собой граф G с двумя выделенными вершинами, s и t, ребра которого могут быть рекурсивно построены следующим образом: G либо состоит из единственного ребра s — t, либо представляет собой *последовательное сверхребро*, состоящее из $k \geq 2$ последовательно-параллельных подграфов G_j между s_j и t_j, последовательно соединенных так, что $s = s_1$ и $t_j = s_{j+1}$ для $1 \leq j < k$ и $t_k = t$, либо G представляет собой *параллельное сверхребро*, состоящее из $k \geq 2$ последовательно-параллельных подграфов G_j между s_j и t_j, соединенных параллельно. Такое разложение для данных s и t единственное, если потребовать, чтобы подграфы G_j последовательных сверхребер сами не являлись последовательными сверхребрами, а подграфы G_j параллельных сверхребер сами не являлись параллельными сверхребрами.

Любой последовательно-параллельный граф можно удобно представить в виде дерева, в котором нет узлов степени 1. Листья такого дерева представляют ребра, а внутренние узлы — сверхребра, причем последовательные и параллельные сверхребра чередуются от уровня к уровню. Например, дерево

(53)

соответствует последовательно-параллельным графам и подграфам

$A =$

, $B =$

, $C =$

, $D =$

, (54)

если верхний узел A рассматривать как параллельный. В (54) именованы ребра, но не вершины, поскольку именно ребра играют главную роль в остовных деревьях.

Будем считать, что *близкое дерево* последовательно-параллельного графа между s и t представляет собой множество из $n-2$ ребер без циклов, которые не соединяют s с t. Остовные деревья и близкие деревья последовательно-параллельного графа легко описать рекурсивно следующим образом. (1) Остовное дерево последовательного сверхребра соответствует остовным деревьям всех его главных подграфов G_j; близкое дерево соответствует остовным деревьям всех подграфов, кроме одного, у которого соответствие выполняется для близкого дерева. (2) Близкое дерево параллельного сверхребра соответствует близким деревьям всех его главных подграфов G_j; остовное дерево соответствует близким деревьям всех подграфов, кроме одного, у которого соответствие выполняется для остовного дерева.

Правила (1) и (2) наводят на мысль об использовании следующих структур данных для перечисления остовных деревьев и/или близких деревьев последовательно-параллельных графов. Пусть p указывает на узел в дереве наподобие (53). Тогда определим

$t_p = 1$ для последовательных сверхребер, 0 в противном случае (“тип” p);
$v_p = 1$ в случае остовного дерева для p, 0 в случае близкого дерева;
$l_p =$ указатель на крайний слева потомок p или 0, если p — лист;
$r_p =$ указатель на крайний справа “брат” p (по циклу);
$d_p =$ указатель на выбранный потомок p или 0, если p — лист.

Если q указывает на крайний справа дочерний по отношению к p узел, то его “правый брат” r_q равен l_p. А если q указывает на *любой* непосредственный потомок p, то правила (1) и (2) гласят, что

$$v_q = \begin{cases} v_p, & \text{если } q = d_p; \\ t_p, & \text{если } q \neq d_p. \end{cases} \tag{55}$$

(Например, если p — внутренний узел, представляющий последовательное сверхребро, то для всех, кроме одного, потомков p должно выполняться $v_q = 1$; един-

ственным исключением является выбранный потомок d_p. Таким образом, у нас должно быть остовное дерево для всех последовательно соединенных подграфов, образующих p, за исключением одного выбранного подграфа, и в этом случае мы имеем для p близкое дерево.)

Для заданных выбранных указателей d_p на дочерние узлы и заданного для v_p значения 0 или 1 в корне дерева формула (55) говорит о том, каким образом происходит распространение этих значений вниз по дереву вплоть до всех листьев. Например, если установить $v_A \leftarrow 1$ в дереве (53), а выбранным потомком каждого внутреннего узла считать крайний слева (так что $d_A = a$, $d_B = b$, $d_C = c$ и $d_D = f$), то можно последовательно найти значения

$$v_a = 1,\ v_B = 0,\ v_b = 0,\ v_C = 1,\ v_c = 1,\ v_d = 0,\ v_e = 1,\ v_D = 0,\ v_f = 0,\ v_g = 1. \tag{56}$$

Лист q присутствует в остовном дереве тогда и только тогда, когда $v_q = 1$; следовательно, (56) определяет остовное дерево $aceg$ последовательно-параллельного графа A из (54).

Будем для удобства говорить, что *конфигурациями* p являются его остовные деревья, если $v_p = 1$, и его близкие деревья, если $v_p = 0$. Мы бы хотели сгенерировать все конфигурации корня. Внутренний узел p называется простым, если $v_p = t_p$; т. е. последовательный узел прост, если его конфигурациями являются остовные деревья, а параллельный узел прост, если его конфигурациями являются близкие деревья. Если p прост, его конфигурации представляют собой декартово произведение конфигураций его дочерних узлов, а именно все независимо изменяющиеся k-кортежи дочерних конфигураций; выбранный потомок d_p в этом простом случае несуществен. Но если p не является простым, то его конфигурации представляют собой объединение таких декартовых k-кортежей, взятых по всем возможным выборам d_p.

Как назло, простые узлы относительно редки: простым может быть максимум один из дочерних узлов узла, не являющегося простым (а именно выбранный дочерний узел), и все дочерние узлы простого узла не являются простыми, если только это не листья.

Даже при этом представление последовательно-параллельного графа в виде дерева делает рекурсивную генерацию всех остовных и/или близких деревьев достаточно простой и эффективной. Операции алгоритма S — сжатие и восстановление, удаление и его отмена, проверка моста — не требуются при работе с последовательно-параллельными графами. Кроме того, в упр. 99 показано, что имеется красивый способ получить остовные или близкие деревья в порядке кода Грея двери-вертушки с использованием фокусных указателей, как в некоторых встречавшихся ранее алгоритмах.

***Усовершенствование алгоритма S.** Хотя алгоритм S обеспечивает простой и достаточно эффективный способ посещения всех остовных деревьев обобщенного граф, его автор Малкольм Смит (Malcolm Smith) нашел, что свойства последовательно-параллельных графов позволяют его улучшить. Например, если граф имеет два или более ребер, проходящих между одними и теми же вершинами u и v, можно скомбинировать их в сверхребро; остовные деревья исходного графа в таком случае могут быть получены из этого более простого графа. А если граф имеет вершину v степени 2, так что с ней связаны только ребра u — v и v — w, можно удалить

вершину v и заменить указанные ребра одним сверхребром между u и w. Кроме того, можно удалить любую вершину степени 1 вместе с ее ребром, просто включая его в каждое остовное дерево.

Применив описанные в предыдущем абзаце приведения к данному графу G, мы получим приведенный граф $\hat{G}$, не имеющий параллельных ребер и вершин степеней 1 и 2, и множество $m \geq 0$ последовательно-параллельных графов S_1, $\ldots$, S_m, представляющих ребра (или сверхребра), которые должны быть включены во все остовные деревья G. Каждое оставшееся ребро u — v графа $\hat{G}$ фактически соответствует последовательно-параллельному графу S_{uv} между вершинами u и v. *Остовные деревья графа G могут быть получены как объединение, взятое по всем остовным деревьям T графа $\hat{G}$, декартова произведения остовных деревьев графов S_1, $\ldots$, S_m и остовных деревьев всех графов S_{uv} для ребер u — v в T вместе с близкими деревьями всех графов S_{uv} для ребер u — v, которые входят в $\hat{G}$, но не в T.* А все остовные деревья T графа $\hat{G}$ можно получить с использованием стратегии из алгоритма S.

Фактически при таком расширении алгоритма S его операции по замене текущего графа G на $G \,/\, e$ или $G \setminus e$ обычно влекут за собой дальнейшие приведения, если появляются новые параллельные ребра или если степень некоторых вершин графа падает ниже 3. Таким образом, это приводит к тому, что "останавливающее состояние" неявной рекурсии алгоритма S, а именно ситуация, когда остаются только две вершины (шаг S5), в действительности никогда не достигается: приведенный граф $\hat{G}$ либо состоит из одной вершины без ребер, либо имеет как минимум четыре вершины и шесть ребер.

Полученный в результате алгоритм обладает требуемым свойством двери-вертушки, которым обладает исходный алгоритм S, и достаточно красив (несмотря на то, что он примерно в четыре раза длиннее исходного); см. упр. 100. Смит доказал, что он обладает наилучшим возможным асимптотическим временем работы: если G имеет n вершин, m ребер и N остовных деревьев, алгоритм посещает все их за $O(m + n + N)$ шагов.

Производительность алгоритма S и его улучшенной версии — алгоритма S′ — лучше всего можно оценить, рассмотрев количество обращений к памяти, выполняемых этими алгоритмами при генерации всех остовных деревьев типичных графов, как показано в табл. 5. Нижняя строка таблицы соответствует графу *plane_miles*$(16, 0, 0, 1, 0, 0, 0)$ из Stanford GraphBase, который служит "естественным" противоядием для чисто математических примеров в предыдущих строках таблицы. Случайный мультиграф в предпоследней строке, также взятый из Stanford GraphBase, может быть описан более точно его официальным именем *random_graph*$(16, 37, 1, 0, 0, 0, 0, 0, 0, 0)$. Хотя тор 4×4 изоморфен четырехмерному кубу (см. упр. 7.2.1.1–17), эти изоморфные графы дают немного разное время работы из-за того, что их вершины и ребра алгоритм обходит в разном порядке.

В общем случае алгоритм S не так уж плох для небольших примеров, за исключением случаев сильно разреженных графов; но алгоритм S′ начинает превосходить его при наличии большого количества остовных деревьев. Когда алгоритм S′ входит в полную силу, он тратит на каждое новое дерево примерно 18–19 обращений к памяти.

Таблица 5

Время работы в обращениях к памяти, необходимое для генерации всех остовных деревьев

	m	n	N	Алгоритм S	Алгоритм S′	μ на одно дерево	
Путь P_{10}	9	10	1	794μ	473μ	794.0	473.0
Путь P_{100}	99	100	1	$9\,974\mu$	$5\,063\mu$	9974.0	5063.0
Цикл C_{10}	10	10	10	$3\,480\mu$	998μ	348.0	99.8
Цикл C_{100}	100	100	100	$355\,605\mu$	$10\,538\mu$	3556.1	105.4
Полный граф K_4	6	4	16	$1\,213\mu$	$1\,336\mu$	75.8	83.5
Полный граф K_{10}	45	10	100 000 000	$3\,759.58\mathrm{M}\mu$	$1\,860.95\mathrm{M}\mu$	37.6	18.6
Полный биграф $K_{5,5}$	25	10	390 625	$23.43\mathrm{M}\mu$	$8.88\mathrm{M}\mu$	60.0	22.7
4×4-решетка $P_4\square P_4$	24	16	100 352	$12.01\mathrm{M}\mu$	$1.87\mathrm{M}\mu$	119.7	18.7
5×5-решетка $P_5\square P_5$	40	25	557 568 000	$54.68\mathrm{G}\mu$	$10.20\mathrm{G}\mu$	98.1	18.3
4×4-цилиндр $P_4\square C_4$	28	16	2 558 976	$230.96\mathrm{M}\mu$	$49.09\mathrm{M}\mu$	90.3	19.2
5×5-цилиндр $P_5\square C_5$	45	25	38 720 000 000	$3\,165.31\mathrm{G}\mu$	$711.69\mathrm{G}\mu$	81.7	18.4
4×4-тор $C_4\square C_4$	32	16	42 467 328	$3\,168.15\mathrm{M}\mu$	$823.08\mathrm{M}\mu$	74.6	19.4
Четырехмерный куб $P_2\square P_2\square P_2\square P_2$	32	16	42 467 328	$3\,172.19\mathrm{M}\mu$	$823.38\mathrm{M}\mu$	74.7	19.4
Случайный мультиграф	37	16	59 933 756	$3\,818.19\mathrm{M}\mu$	$995.91\mathrm{M}\mu$	63.7	16.6
16 городов	37	16	179 678 881	$11\,772.11\mathrm{M}\mu$	$3\,267.43\mathrm{M}\mu$	65.5	18.2

Из табл. 5 также видно, что математически определенные графы часто имеют на удивление “круглое” количество остовных деревьев. Например, Д. М. Цветкович (D. M. Cvetković) [*Srpska Akademija Nauka, Matematicheski Institut* **11** (Belgrade: 1971), 135–141] открыл, помимо прочего, что n-мерный куб имеет ровно

$$2^{2^n-n-1}\,1^{\binom{n}{1}}\,2^{\binom{n}{2}}\ldots n^{\binom{n}{n}} \tag{57}$$

остовных деревьев. В упр. 104–109 рассматриваются некоторые причины этого.

Обобщенный квазикод Грея. Завершим данный раздел обсуждением еще одного вопроса, совершенно отличного от предыдущих, но, тем не менее, связанного с деревьями. Рассмотрим гибридные варианты двух стандартных способов обхода непустого леса.

Прямообратный порядок обхода (prepostorder)	*Обратнопрямой порядок обхода (postpreorder)*
Посетить корень первого дерева	Обойти поддеревья первого дерева в прямообратном порядке
Обойти поддеревья первого дерева в обратнопрямом порядке	Посетить корень первого дерева
Обойти остальные деревья в прямообратном порядке	Обойти остальные деревья в обратнопрямом порядке

В первом случае каждое дерево леса обходится в прямообратном порядке, причем первым посещается корень дерева; однако поддеревья этих корней обходятся в обратнопрямом порядке, причем корень посещается последним. Второй вариант

аналогичен, но в нем “прямой” и “обратный” переставлены. В общем случае прямообратный порядок посещает корни первыми на каждом четном уровне леса, а на нечетном — последними. Например, лес (2) превращается в

(58)

при нумерации его узлов в прямообратном порядке.

Прямообратный и обратнопрямой порядки не просто курьез — они приносят реальную пользу. Это связано с тем, что соседние узлы при любом из этих порядков всегда находятся в лесу близко один к другому. Например, узлы k и $k+1$ являются смежными в (58) при $k = 1, 4, 6, 8, 10, 13$; они разделены только одним узлом при $k = 3, 12, 14$ и разнесены на три шага при $k = 2, 5, 7, 9, 11$ (если представить невидимый сверхродительский узел на вершине леса). Минутное размышление позволяет по индукции доказать, что между соседями как в прямообратном, так и в обратнопрямом порядке может располагаться не более двух узлов, поскольку обратнопрямой порядок всегда начинается с корня первого дерева или его крайнего слева дочернего узла, а прямообратный порядок всегда заканчивается корнем последнего дерева или его правым дочерним узлом.

Предположим, что мы хотим сгенерировать все комбинаторные шаблоны некоторого вида и посетить их в порядке наподобие кода Грея, так, чтобы последовательные шаблоны всегда были “близки” один к другому. Можно сформировать, по крайней мере концептуально, граф всех возможных шаблонов p с ребрами p — q для всех пар шаблонов, близких один к другому. В следующей теореме, открытой Миланом Секаниной (Milan Sekanina) [*Spisy Přírodovědecké Fakulty University v Brně*, No. 412 (1960), 137–140], доказывается, что всегда возможен хороший код Грея, обеспечивающий получение одного шаблона из другого с помощью последовательности коротких шагов.

Теорема S. *Вершины любого связного графа могут быть перечислены в циклическом порядке* $(v_0, v_1, \ldots, v_{n-1})$ *так, что расстояние между* v_k *и* $v_{(k+1) \bmod n}$ *не превышает* 3 *для* $0 \le k < n$.

Доказательство. Найдите остовное дерево графа и обойдите его в прямообратном порядке. ▮

Ученые в области теории графов традиционно говорят, что k-й степенью графа G является граф G^k, вершины которого те же, что и у графа G, а ребро u — v присутствует в графе G^k тогда и только тогда, когда в G существует путь длиной не более k от u до v. Это позволяет сформулировать теорему S более кратко при $n > 2$: *куб связного графа является гамильтонианом.*

Прямообратный обход полезен также в случае, когда мы хотим посетить узлы дерева с помощью ограниченного количества шагов без циклов.

Алгоритм Q (*Прямообратный преемник в трижды связанном лесу*). Если P указывает на узел в лесу, представленном связями PARENT, CHILD и SIB, соответствующими родителю каждого узла, его крайнему слева дочернему узлу и правому брату, то данный алгоритм вычисляет узел Q, следующий за узлом P в прямообратном порядке. Предполагается, что известен уровень L, на котором в лесу находится узел P; значение L обновляется так, что указывает уровень узла Q. Если P — последний узел в прямообратном порядке, то алгоритм устанавливает Q $\leftarrow \Lambda$ и L $\leftarrow -1$.

Q1. [Прямой или обратный?] Если L четно, перейти к шагу Q4.

Q2. [Продолжение обратнопрямое.] Установить Q $\leftarrow$ SIB(P). Перейти к шагу Q6, если Q $\neq \Lambda$.

Q3. [Перемещение вверх.] Установить P $\leftarrow$ PARENT(P) и L $\leftarrow$ L $- 1$. Перейти к шагу Q7.

Q4. [Продолжение прямообратное.] Если CHILD(P) $= \Lambda$, перейти к шагу Q7.

Q5. [Перемещение вниз.] Установить Q $\leftarrow$ CHILD(P) и L $\leftarrow$ L $+ 1$.

Q6. [Перемещение вниз по возможности.] Если CHILD(Q) $\neq \Lambda$, установить Q $\leftarrow$ CHILD(Q) и L $\leftarrow$ L $+ 1$. Завершить работу алгоритма.

Q7. [Перемещение вправо или вверх.] Если SIB(P) $\neq \Lambda$, установить Q $\leftarrow$ SIB(P); в противном случае установить Q $\leftarrow$ PARENT(P) и L $\leftarrow$ L $- 1$. Завершить работу алгоритма. ▮

Заметим, что, как и в алгоритме 2.4C, связь PARENT(P) проверяется, только если SIB(P) $= \Lambda$. Полный обход представляет собой путешествие червяка вокруг леса наподобие (3): червяк "видит" узлы на четных уровнях при проходе слева, а на нечетных — справа.

Упражнения

1. [*15*] Каким образом можно реконструировать строку скобок (1) при проползании червяка по бинарному дереву (4)?

2. [*20*] (Ш. Закс (S. Zaks), 1980.) Модифицируйте алгоритм P так, чтобы он выдавал сочетания $z_1 z_2 \ldots z_n$ из (8) вместо строк скобок $a_1 a_2 \ldots a_{2n}$.

▶ **3.** [*23*] Докажите, что (11) преобразует $z_1 z_2 \ldots z_n$ в таблицу инверсий $c_1 c_2 \ldots c_n$.

4. [*20*] Истинно или ложно следующее утверждение? Если строки $a_1 \ldots a_{2n}$ сгенерированы в лексикографическом порядке, то в том же порядке находятся строки $d_1 \ldots d_n$, $z_1 \ldots z_n$, $p_1 \ldots p_n$ и $c_1 \ldots c_n$.

5. [*15*] Какие таблицы $d_1 \ldots d_n$, $z_1 \ldots z_n$, $p_1 \ldots p_n$ и $c_1 \ldots c_n$ соответствуют строке вложенных скобок (1)?

▶ **6.** [*20*] Какое *паросочетание* (см. последний столбец табл. 1) отвечает строке (1)?

7. [*16*] (а) В каком состоянии находится строка $a_1 a_2 \ldots a_{2n}$ по окончании работы алгоритма P? (б) Что содержится в массивах $l_1 l_2 \ldots l_n$ и $r_1 r_2 \ldots r_n$ по окончании работы алгоритма B?

8. [*15*] Как выглядят таблицы $l_1 \ldots l_n$, $r_1 \ldots r_n$, $e_1 \ldots e_n$ и $s_1 \ldots s_n$, соответствующие примеру леса (2)?

9. [*M20*] Покажите, что таблицы $c_1 \ldots c_n$ и $s_1 \ldots s_n$ связаны соотношением

$$c_k = [s_1 \geq k-1] + [s_2 \geq k-2] + \cdots + [s_{k-1} \geq 1].$$

10. [*M20*] (*Обход червяка.*) Для заданной строки вложенных скобок $a_1 a_2 \ldots a_{2n}$ обозначим через w_j превышение количества левых скобок над правыми в подстроке $a_1 a_2 \ldots a_j$, $0 \leq j \leq 2n$. Докажите, что $w_0 + w_1 + \cdots + w_{2n} = 2(c_1 + \cdots + c_n) + n$.

11. [*11*] Если F — некоторый лес, *сопряженный* лес F^R получается путем зеркального отражения слева направо. Например, для 14 лесов из табл. 1

сопряженными являются соответственно

(солексные леса из табл. 2). Если F соответствует некоторой строке вложенных скобок $a_1 a_2 \ldots a_{2n}$, то какой строке соответствует F^R?

12. [*15*] Если F — некоторый лес, то *транспонированный* лес F^T получается путем обмена в каждом бинарном дереве леса F левых и правых связей. Например, для 14 лесов из табл. 1 транспонированными являются соответственно

Что дает транспонирование леса (2)?

13. [*20*] Продолжая выполнять упр. 11 и 12, ответьте, как соотносятся прямой и обратный порядки обхода помеченного леса F и прямой и обратный порядки обхода (а) F^R? (б) F^T?

▶ **14.** [*21*] Найдите все помеченные леса F, такие, что $F^{RT} = F^{TR}$.

15. [*20*] Предположим, что B — бинарное дерево, полученное из леса F путем связывания каждого узла с его левым братом и крайним справа дочерним узлом, как в упр. 2.3.2–5 и последнем столбце табл. 2. Пусть F' — лес, естественным образом соответствующий B посредством связей с левым братом и крайним справа дочерним узлом. Докажите, что $F' = F^{RT}$ (с использованием обозначений из упр. 11 и 12).

16. [*20*] Пусть F и G — леса и пусть FG означает лес, полученный путем размещения деревьев F слева от деревьев G. Определим также $F|G = (G^T F^T)^T$. Приведите интуитивное пояснение оператора $|$ и докажите, что он ассоциативен.

17. [*M46*] Охарактеризуйте все *непомеченные* леса F, такие, что $F^{RT} = F^{TR}$ (см. упр. 14).

18. [*30*] Два леса называются *родственными* (*cognate*), если один может быть получен из другого с помощью некоторого количества операций сопряжения и/или транспонирования. Примеры в упр. 11 и 12 показывают, что каждый лес из четырех узлов принадлежит одному из трех классов родственности:

Изучите множество всех лесов из 15 узлов. Сколько классов эквивалентности родственных лесов они образуют? Какой класс наибольший? Какой наименьший? Чему равен размер класса, содержащего (2)?

19. [*28*] Пусть F_1, F_2, …, F_N — последовательность непомеченных лесов, соответствующих вложенным скобкам, сгенерированным алгоритмом P, и пусть G_1, G_2, …, G_N — последовательность непомеченных лесов, соответствующих бинарным деревьям, сгенерированным алгоритмом B. Докажите, что $G_k = F_k^{RTR}$ (с использованием обозначений из

упр. 11 и 12). (Лес F^{RTR} называется *дуальным* к лесу F и в ряде упражнений далее обозначается как F^D.)

20. [*25*] Вспомним из раздела 2.3, что *степенью* узла в дереве называется количество его дочерних узлов и что расширенное бинарное дерево характеризуется тем свойством, что каждый его узел имеет степень либо 0, либо 2. В расширенном бинарном дереве (4) последовательность степеней узлов в прямом порядке обхода представляет собой 2200222002220220002002202200000; эта строка нулей и двоек идентична последовательности скобок в (1), за исключением того, что каждая скобка '(' заменена двойкой, каждая ')' — нулем, и добавлен дополнительный нуль.

а) Докажите, что последовательность неотрицательных целых чисел $b_1b_2\ldots b_N$ представляет собой последовательность степеней в прямом порядке обхода леса тогда и только тогда, когда она удовлетворяет следующему свойству для $1 \le k \le N$:

$$b_1 + b_2 + \cdots + b_k + f \;>\; k \qquad \text{тогда и только тогда, когда} \qquad k < N.$$

Здесь $f = N - b_1 - b_2 - \cdots - b_N$ — количество деревьев в лесу.

б) Вспомним из упр. 2.3.4.5–6, что *расширенное тернарное дерево* характеризуется тем свойством, что каждый его узел имеет степень либо 0, либо 3; расширенное тернарное дерево с n внутренними узлами имеет $2n+1$ внешних узлов, следовательно, всего $N = 3n+1$ узлов. Разработайте алгоритм генерации всех тернарных деревьев с n внутренними узлами путем генерации соответствующих последовательностей $b_1b_2\ldots b_N$ в лексикографическом порядке.

▶ **21.** [*26*] (Ш. Закс (S. Zaks) и Д. Ричардс (D. Richards), 1979.) Продолжая выполнять упр. 20, поясните, как сгенерировать последовательности степеней в прямом порядке обхода для всех лесов с $N = n_0 + \cdots + n_t$ узлами, среди которых степень j имеют ровно n_j узлов. *Пример*: при $n_0 = 4$, $n_1 = n_2 = n_3 = 1$ и $t = 3$ корректными последовательностями $b_1b_2b_3b_4b_5b_6b_7$ являются

1203000, 1230000, 1300200, 1302000, 1320000, 2013000, 2030010, 2030100, 2031000, 2103000, 2130000, 2300010, 2300100, 2301000, 2310000, 3001200, 3002010, 3002100, 3010200, 3012000, 3020010, 3020100, 3021000, 3100200, 3102000, 3120000, 3200010, 3200100, 3201000, 3210000.

▶ **22.** [*30*] (Д. Корш (J. Korsh), 2004.) В качестве альтернативы алгоритму B покажите, что бинарные деревья могут также быть сгенерированы непосредственно в связном виде, если генерировать их в *солексном* порядке чисел $d_1\ldots d_{n-1}$, определенных в (9). (Реальные значения $d_1\ldots d_{n-1}$ не должны вычисляться явно; должна выполняться работа со связями $l_1\ldots l_n$ и $r_1\ldots r_n$ таким образом, чтобы получались бинарные деревья, последовательно соответствующие значениям $d_1d_2\ldots d_{n-1} = 000\ldots0$, $100\ldots0$, $010\ldots0$, $110\ldots0$, $020\ldots0$, $001\ldots0$, $\ldots$, $000\ldots(n{-}1)$.)

▶ **23.** [*25*] (а) Какова последняя строка, посещаемая алгоритмом N? (б) Каково последнее бинарное дерево (или лес), посещаемое алгоритмом L? *Указание*: см. упр. 40.

24. [*22*] Какие последовательности $l_0l_1\ldots l_{15}$, $r_1\ldots r_{15}$, $k_1\ldots k_{15}$, $q_1\ldots q_{15}$ и $u_1\ldots u_{15}$ соответствуют бинарному дереву (4) и лесу (2) при использовании обозначений из табл. 3?

▶ **25.** [*30*] (*Обрезка и прививка.*) Представляя бинарные деревья способом, использующимся в алгоритме B, разработайте алгоритм, который посещает все таблицы связей $l_1\ldots l_n$ и $r_1\ldots r_n$ таким образом, что между визитами только одна связь меняется с j на 0 и одна — с 0 на j для некоторого индекса j. (Другими словами, на каждом шаге некоторое поддерево j удаляется из бинарного дерева и помещается в другое место, сохраняя при этом прямой порядок обхода.)

26. [*M31*] (*Решетка Кревераса.*) Пусть F и F' — n-узловые леса, узлы которых пронумерованы от 1 до n в прямом порядке обхода. Мы записываем $F \ltimes F'$ ("F соединяется с F'"), если из того, что j и k являются братьями в F', вытекает, что они братья и в F, $1 \le j < k \le n$. На рис. 60 показано такое частичное упорядочение для случая $n = 4$; каждый лес закодирован последовательностью $c_1 \ldots c_n$ из (10) и (11), которая определяет глубину каждого узла. (При таком кодировании j и k являются братьями тогда и только тогда, когда $c_j = c_k \le c_{j+1}, \ldots, c_{k-1}$.)

Рис. 60. Решетка Кревераса четвертого порядка. Каждый лес представлен последовательностью глубин узлов $c_1c_2c_3c_4$ в прямом порядке обхода (см. упр. 26–28.)

а) Пусть Π — разбиение $\{1, \ldots, n\}$. Покажите, что существует лес F с узлами, помеченными $(1, \ldots, n)$ в прямом порядке обхода, такой, что

$$j \equiv k \ (\text{по модулю } \Pi) \iff j \text{ брат } k \text{ в } F$$

тогда и только тогда, когда Π удовлетворяет условию *непересекаемости*

$$\text{из } i<j<k<l \text{ и из } i\equiv k \text{ и } j\equiv l \ (\text{по модулю } \Pi) \quad \text{вытекает} \quad i\equiv j\equiv k\equiv l \ (\text{по модулю } \Pi).$$

б) Поясните, как для двух заданных n-узловых лесов F и F' вычислить их наименьшую верхнюю границу $F \veebar F'$, которая представляет собой элемент, такой, что $F \ltimes G$ и $F' \ltimes G$ тогда и только тогда, когда $F \veebar F' \ltimes G$.

в) Когда F' покрывает F по отношению к $\ltimes$? (См. упр. 7.2.1.4–55.)

г) Покажите, что если F' покрывает F, то он меньше F ровно на один лист.

д) Сколько лесов покрывают F, когда узел k имеет e_k дочерних узлов ($1 \le k \le n$)?

е) Какой лес является дуальным по отношению к лесу (2) при использовании определения дуальности из упр. 19?

ж) Докажите, что $F \ltimes F'$ тогда и только тогда, когда $F'^D \ltimes F^D$. (Из-за этого свойства дуальные элементы располагаются симметрично относительно центра рис. 60.)

з) Объясните, как для заданных n-узловых лесов F и F' вычислить их наибольшую нижнюю границу $F \barwedge F'$; т. е. $G \ltimes F$ и $G \ltimes F'$ тогда и только тогда, когда $G \ltimes F \barwedge F'$.

и) Удовлетворяет ли эта решетка полумодулярному закону, аналогичному закону из упр. 7.2.1.5–12, (е)?

▶ **27.** [*M33*] (*Решетка Тамари.*) Продолжая выполнять упр. 26, будем записывать $F \dashv F'$, если j-й узел в прямом порядке обхода для всех j имеет как минимум столько же потомков в F', как и в F. Другими словами, если F и F' характеризуются своими последовательностями $s_1 \ldots s_n$ и $s'_1 \ldots s'_n$ (см. табл. 2), то $F \dashv F'$ тогда и только тогда, когда $s_j \le s'_j$ для $1 \le j \le n$ (рис. 61).

а) Покажите, что координаты $\min(s_1, s'_1)\min(s_2, s'_2)\ldots\min(s_n, s'_n)$ определяют лес, который является наибольшей нижней границей F и F' (обозначим ее как $F \perp F'$).

Рис. 61. Решетка Тамари четвертого порядка. Каждый лес представлен последовательностью в прямом порядке обхода (а) глубин узлов и (б) количеством потомков (см. упр. 26–28.)

Указание: докажите, что $s_1 \ldots s_n$ соответствует лесу тогда и только тогда, когда из $0 \le k \le s_j$ вытекает $s_{j+k} + k \le s_j$ для $0 \le j \le n$, если положить $s_0 = n$.

б) Когда при таком частичном упорядочении F' покрывает F?

в) Докажите, что $F \dashv F'$ тогда и только тогда, когда $F'^D \dashv F^D$. (Сравните с упр. 26, (ж).)

г) Поясните, как вычислить наименьшую верхнюю границу $F \top F'$ для заданных F и F'.

д) Докажите, что из $F \ltimes F'$ в решетке Кревераса вытекает $F \dashv F'$ в решетке Тамари.

е) Истинно или ложно следующее утверждение? $F \barwedge F' \dashv F \perp F'$.

ж) Истинно или ложно следующее утверждение? $F \veebar F' \ltimes F \top F'$.

з) Каковы длиннейший и кратчайший пути от верха решетки Тамари до ее низа, такие, что на этих путях каждый лес покрывает следующий за ним? (Такие пути называются *максимальными цепочками* решетки; сравните с упр. 7.2.1.4–55, (з).)

28. [*M26*] (*Решетка Стенли.*) Продолжая выполнять упр. 26 и 27, определим еще одно частичное упорядочение на n-узловых лесах, говоря, что $F \subseteq F'$, если координаты глубины $c_1 \ldots c_n$ и $c'_1 \ldots c'_n$ удовлетворяют соотношению $c_j \le c'_j$ для $1 \le j \le n$ (рис. 62).

Рис. 62. Решетка Стенли четвертого порядка. Каждый лес представлен последовательностью глубин узлов в прямом порядке обхода (см. упр. 26–28).

а) Докажите, что это частичное упорядочение представляет собой решетку, пояснив, каким образом можно вычислить наибольшую нижнюю границу $F \cap F'$ и наименьшую верхнюю границу $F \cup F'$ любых двух заданных лесов.

б) Покажите, что решетка Стенли удовлетворяет законам дистрибутивности

$$F \cap (G \cup H) = (F \cap G) \cup (F \cap H), \qquad F \cup (G \cap H) = (F \cup G) \cap (F \cup H).$$

в) Когда в этой решетке F' покрывает F?

г) Истинно или ложно следующее утверждение? $F \subseteq G$ тогда и только тогда, когда $F^R \subseteq G^R$.

д) Докажите, что $F \subseteq F'$ в решетке Стенли, когда $F \dashv F'$ в решетке Тамари.

29. [*HM31*] Покрывающий граф для решетки Тамари иногда называют ассоциаэдром (associahedron) из-за его связи с законом ассоциативности (14), доказанным в упр. 27, (б). Ассоциаэдр четвертого порядка, изображенный на рис. 61, имеет три квадратные грани и шесть пятиугольных. (Сравните с рис. 43 из упр. 7.2.1.2–60, на котором показан "пермутаэдр" четвертого порядка, известное Архимедово тело.) Почему фигура на рис. 61 не входит в классический список однородных многогранников?

30. [*M26*] *Следом* леса называется битовая строка $f_1 \ldots f_n$, определяемая следующим образом:

$$f_j = [\text{узел } j \text{ в прямом порядке обхода не является листом}].$$

а) Если F имеет след $f_1 \ldots f_n$, какой след имеет F^D? (См. упр. 27.)

б) Сколько лесов имеют след 101011011111100001010100010110 00?

в) Докажите, что $f_j = [d_j = 0]$, $1 \le j < n$, с использованием обозначений из (6).

г) Два элемента решетки называются *комплементарными*, если их наибольшая нижняя граница представляет собой нижний элемент, а наименьшая верхняя граница — верхний элемент. Покажите, что F и F' комплементарны в решетке Тамари тогда и только тогда, когда их следы комплементарны в том смысле, что $f'_1 \ldots f'_{n-1} = \bar{f}_1 \ldots \bar{f}_{n-1}$.

▶ **31.** [*M28*] Бинарное дерево с n внутренними узлами называется *вырожденным*, если его высота равна n.

а) Сколько среди n-узловых бинарных деревьев являются вырожденными?

б) В табл. 1–3 мы видели, что бинарные деревья и леса могут быть закодированы различными n-кортежами чисел. Поясните для каждой из кодировок $c_1 \ldots c_n$, $d_1 \ldots d_n$, $e_1 \ldots e_n$, $k_1 \ldots k_n$, $p_1 \ldots p_n$, $s_1 \ldots s_n$, $u_1 \ldots u_n$ и $z_1 \ldots z_n$, как можно, бегло взглянув на кодировку, сказать, вырождено ли соответствующее бинарное дерево.

в) Истинно или ложно следующее утверждение? Если F вырождено, то вырождено и F^D.

г) Докажите, что если F и F' вырождены, то вырождены и $F \barwedge F' = F \perp F'$, и $F \barvee F' = F \top F'$.

▶ **32.** [*M30*] Докажите, что если $F \dashv F'$, существует лес F'', такой, что для всех G

$$F' \perp G = F \quad \text{тогда и только тогда, когда} \quad F \dashv G \dashv F''.$$

Следовательно, в решетке Тамари выполняются *полудистрибутивные законы*:

$$\text{из} \quad F \perp G = F \perp H \quad \text{вытекает} \quad F \perp (G \top H) = F \perp G;$$
$$\text{из} \quad F \top G = F \top H \quad \text{вытекает} \quad F \top (G \perp H) = F \top G.$$

▶ **33.** [*M27*] (*Представление деревьев с помощью перестановок.*) Пусть σ представляет собой цикл $(1\ 2\ \ldots\ n)$.

а) Докажите, что для заданного бинарного дерева, узлы которого пронумерованы от 1 до n в симметричном порядке обхода, существует единственная перестановка λ множества $\{1, \ldots, n\}$, такая, что для $1 \le k \le n$

$$\texttt{LLINK}[k] = \begin{cases} k\lambda, & \text{если } k\lambda < k; \\ 0 & \text{в противном случае;} \end{cases} \qquad \texttt{RLINK}[k] = \begin{cases} k\sigma\lambda, & \text{если } k\sigma\lambda > k; \\ 0 & \text{в противном случае.} \end{cases}$$

Таким образом, λ аккуратно упаковывает $2n$ полей связей в единый n-элементный массив.

б) Покажите, что эту перестановку λ наиболее легко можно описать в циклическом виде, когда бинарное дерево является представлением "левый брат/правый потомок" леса F. Как выглядит циклический вид $\lambda(F)$, если представляет собой лес (2)?

в) Найдите простое соотношение между $\lambda(F)$ и дуальной перестановкой $\lambda(F^D)$.

г) Докажите, что в упр. 26 F' покрывает F тогда и только тогда, когда $\lambda(F') = (j\,k)\,\lambda(F)$, где j и k являются братьями в F.

д) Следовательно, количество максимальных цепочек в решетке Кревераса порядка n равно количеству способов разложения n-цикла в виде произведения $n-1$ транспозиций. Вычислите это значение. *Указание*: см. формулу 1.2.6–(16).

34. [*M25*] (Р. П. Стенли (R. P. Stanley).) Покажите, что количество максимальных цепочек в решетке Стенли порядка n равно $(n(n-1)/2)!/(1^{n-1}3^{n-2}\ldots(2n-5)^2(2n-3)^1)$.

35. [*HM37*] (Д. Б. Тайлер (D. B. Tyler) и Д. Р. Хиккерсон (D. R. Hickerson).) Объясните, почему все знаменатели в асимптотической формуле (16) являются степенями 2.

▶ **36.** [*M25*] Проанализируйте алгоритм генерации тернарных деревьев, приведенный в упр. 20, (б). *Указание*: в соответствии с упр. 2.3.4.4–11 всего имеется $(2n+1)^{-1}\binom{3n}{n}$ тернарных деревьев с n внутренними узлами.

▶ **37.** [*M40*] Проанализируйте алгоритм Закса–Ричардса для генерации всех деревьев с данным распределением $n_0, n_1, n_2, \ldots, n_t$ степеней (упр. 21). *Указание*: см. упр. 2.3.4.4–32.

38. [*M22*] Чему равно общее количество обращений к памяти, выполняемых алгоритмом L, выраженное как функция от n?

39. [*22*] Докажите формулу (23), показав, что элементы A_{pq} в (5) соответствуют диаграмме Юнга с двумя строками.

40. [*M22*] (а) Докажите, что C_{pq} нечетно тогда и только тогда, когда $p \,\&\, (q+1) = 0$, т. е. когда бинарные представления p и $q+1$ не имеют общих битов. (б) Следовательно, C_n нечетно тогда и только тогда, когда $n+1$ является степенью 2.

41. [*M21*] Покажите, что баллотировочные числа имеют простую производящую функцию $\sum C_{pq} w^p z^q$.

▶ **42.** [*M22*] Сколько непомеченных лесов с n узлами являются (а) самосопряженными? (б) самотранспонированными? (в) самодуальными? (См. упр. 11, 12, 19 и 26.)

43. [*M21*] Выразите C_{pq} через числа Каталана $\langle C_0, C_1, C_2, \ldots\rangle$. Целью является получение простой формулы для малых величин $q-p$. (Например, $C_{(q-2)q} = C_q - C_{q-1}$.)

▶ **44.** [*M27*] Докажите, что алгоритм B делает $8\frac{2}{3} + O(n^{-1})$ обращений к памяти на одно посещенное бинарное дерево.

45. [*M26*] Проанализируйте обращения к памяти, выполняемые алгоритмом из упр. 22. Сравните полученное значение со значением для алгоритма B.

46. [*M30*] (*Обобщенные числа Каталана.*) Обобщим соотношения (21), определив

$$C_{pq}(x) = C_{p(q-1)}(x) + x^{q-p}C_{(p-1)q}(x), \quad \text{если } 0 \le p \le q \ne 0; \qquad C_{00}(x) = 1;$$

и $C_{pq}(x) = 0$, если $p < 0$ или $p > q$; таким образом, $C_{pq} = C_{pq}(1)$. Пусть также $C_n(x) = C_{nn}(x)$, так что

$$\langle C_0(x), C_1(x), \ldots\rangle = \langle 1, 1, 1+x, 1+2x+x^2+x^3, 1+3x+3x^2+3x^3+2x^4+x^5+x^6, \ldots\rangle.$$

а) Покажите, что $[x^k]\,C_{pq}(x)$ — количество путей от $\textcircled{pq}$ до $\textcircled{00}$ в графе (28), которые имеют площадь k, где "площадь" пути представляет собой количество прямоугольных ячеек над ним. (Таким образом, L-образный путь имеет максимально возможную площадь, равную $p(q-p) + \binom{p}{2}$.)

б) Докажите, что $C_n(x) = \sum_F x^{c_1+\cdots+c_n} = \sum_F x^{\text{длина внутреннего пути}(F)}$, где сумма берется по всем n-узловым лесам F.

в) Покажите, что если $C(x,z) = \sum_{n=0}^{\infty} C_n(x) z^n$, то $C(x,z) = 1 + zC(x,z)C(x,xz)$.

г) Кроме того, $C(x,z)C(x,xz)\ldots C(x,x^r z) = \sum_{p=0}^{\infty} C_{p(p+r)}(x) z^p$.

47. [*M27*] Продолжая выполнять предыдущее упражнение, обобщите тождество (27).

48. [*M28*] (Ф. Раски (F. Ruskey) и А. Проскуровски (A. Proskurowski).) Вычислите $C_{pq}(x)$ при $x = -1$ и используйте полученный результат, чтобы показать, что "идеальный" код Грея для вложенных скобок при нечетном $n \geq 5$ невозможен.

49. [*17*] Какова миллионная в лексикографическом порядке строка из 15 вложенных пар скобок?

50. [*20*] Разработайте алгоритм, обратный алгоритму U: для данной строки вложенных скобок $a_1 \ldots a_{2n}$ определите ее ранг $N-1$ в лексикографическом порядке. Чему равен ранг (1)?

51. [*M22*] Пусть $\bar{z}_1\bar{z}_2\ldots\bar{z}_n$ является дополнением $z_1z_2\ldots z_n$ до $2n$; другими словами, $\bar{z}_j = 2n - z_j$, где z_j определено в формуле (8). Покажите, что если $\bar{z}_1\bar{z}_2\ldots\bar{z}_n$ представляет собой $(N+1)$-е n-сочетание $\{0, 1, \ldots, 2n-1\}$, сгенерированное алгоритмом 7.2.1.3L, то $z_1z_2\ldots z_n$ является $(N - \kappa_n N + 1)$-м n-сочетанием $\{1, 2, \ldots, 2n\}$, сгенерированным алгоритмом из упр. 2. (Здесь κ_n означает n-ю функцию Крускала, определенную в 7.2.1.3–(60).)

52. [*M23*] Найдите среднее и дисперсию величины d_n из табл. 1 при случайном выборе строки вложенных скобок $a_1 \ldots a_{2n}$.

53. [*M28*] Пусть X — расстояние от корня расширенного бинарного дерева до крайнего слева внешнего узла. (а) Каково математическое ожидание значения X, если все бинарные деревья с n узлами равновероятны? (б) Каково математическое ожидание значения X в случайном *бинарном дереве поиска*, построенном с использованием алгоритма 6.2.2T на основе случайной перестановки $K_1 \ldots K_n$? (в) Каково математическое ожидание значения X в случайном *вырожденном* (в смысле упр. 31) бинарном дереве? (г) Каково математическое ожидание значения 2^X во всех трех случаях?

54. [*HM29*] Чему равны математическое ожидание и дисперсия $c_1 + \cdots + c_n$ (см. упр. 46)?

55. [*HM33*] Вычислите $C'_{pq}(1)$ — общую площадь всех путей в упр. 46, (а).

56. [*M23*] (Ренцо Спруньоли (Renzo Sprugnoli), 1990.) Докажите формулу суммирования

$$\sum_{k=0}^{m-1} C_k C_{n-1-k} = \frac{1}{2}C_n + \frac{2m-n}{2n(n+1)}\binom{2m}{m}\binom{2n-2m}{n-m} \qquad \text{для } 0 \leq m \leq n.$$

57. [*M28*] Выразите суммы $S_p(a,b) = \sum_{k\geq 0}\binom{2a}{a-k}\binom{2b}{b-k}k^p$ в аналитическом виде для $p = 0$, 1, 2, 3 и используйте полученные формулы для доказательства (30).

58. [*HM34*] Обозначим через t_{lmn} количество n-узловых бинарных деревьев, в которых внешний узел m (при нумерации внешних узлов в симметричном порядке обхода от 0 до n) находится на уровне l. Пусть также $t_{mn} = \sum_{l=1}^{n} l t_{lmn}$, так что t_{mn}/C_n — средний уровень внешнего узла m; и пусть $t(w,z)$ — сверхпроизводящая функция

$$\sum_{m,n} t_{mn} w^m z^n = (1+w)z + (3+4w+3w^2)z^2 + (9+13w+13w^2+9w^3)z^3 + \cdots.$$

Докажите, что $t(w,z) = \big(C(z) - wC(wz)\big)/(1-w) - 1 + zC(z)t(w,z) + wzC(wz)t(w,z)$, и выведите простую формулу для чисел t_{mn}.

59. [*HM29*] Аналогично пусть T_{lmn} — количество всех n-узловых бинарных деревьев, в которых *внутренний* узел m находится на уровне l. Найдите простую формулу для $T_{mn} = \sum_{l=1}^{n} l T_{lmn}$.

▶ **60.** [*M26*] (*Сбалансированные строки.*) Строка вложенных скобок α является *атомарной*, если она имеет вид (α'), где α' — строка вложенных скобок; каждая строка вложенных скобок может быть единственным способом представлена как произведение атомов $\alpha_1 \ldots \alpha_r$. Строка с равным количеством левых и правых скобок называется *сбалансированной*; каждая сбалансированная строка может быть единственным способом представлена как $\beta_1 \ldots \beta_r$, где каждое β_j является либо атомом, либо *соатомом* (обращением атома). *Дефектом* сбалансированной строки является половина длины ее соатомов. Например, сбалансированная строка

$$(\,(\,)\,)\,(\,(\,(\,)\,)\,)\,)\,(\,(\,)\,(\,(\,(\,)\,)\,(\,(\,(\,)\,)\,(\,(\,)$$

имеет разложение $\beta_1\beta_2\beta_3\beta_4\beta_5\beta_6\beta_7\beta_8 = \alpha_1\alpha_2^R\alpha_3\alpha_4^R\alpha_5^R\alpha_6\alpha_7^R\alpha_8$ с четырьмя атомами и четырьмя соатомами; ее дефект равен $|\alpha_2\alpha_4\alpha_5\alpha_7|/2 = 9$.

а) Докажите, что дефект сбалансированной строки равен количеству индексов k, для которых k-я правая скобка *предшествует* k-й левой скобке.

б) Если строка $\beta_1 \ldots \beta_r$ сбалансирована, ее можно отобразить на строку вложенных скобок простым обращением ее соатомов. Однако описанное далее отображение более интересно, поскольку производит несмещенные (равномерно распределенные) строки вложенных скобок из несмещенных сбалансированных строк. Пусть имеется s соатомов $\beta_{i_1} = \alpha_{i_1}^R, \ldots, \beta_{i_s} = \alpha_{i_s}^R$. Заменим каждый соатом на '('; затем добавим строку $)\alpha'_{i_s} \ldots)\alpha'_{i_1}$, где $\alpha_j = (\alpha'_j)$. Например, приведенная выше строка отображается на строку $\alpha_1(\alpha_3((\alpha_6(\alpha_8)\alpha'_7)\alpha'_5)\alpha'_4)\alpha'_2$, которая представляет собой строку (1) из начала данного раздела.

Разработайте алгоритм, применяющий это отображение к заданной сбалансированной строке $b_1 \ldots b_{2n}$.

в) Разработайте также алгоритм для обратного отображения: для данной строки вложенных скобок $\alpha = a_1 \ldots a_{2n}$ и целого числа $0 \le l \le n$ он должен находить сбалансированную строку $\beta = b_1 \ldots b_{2n}$ с дефектом l, для которой $\beta \mapsto \alpha$. Какая сбалансированная строка с дефектом 11 отображается на строку (1)?

▶ **61.** [*M26*] (*Лемма Рани (Raney) о цикле.*) Пусть $b_1 b_2 \ldots b_N$ — строка неотрицательных целых чисел, такая, что $f = N - b_1 - b_2 - \cdots - b_N > 0$.

а) Докажите, что свойству последовательности степеней леса в прямом порядке обхода из упр. 20 удовлетворяют ровно f циклических сдвигов $b_{j+1} \ldots b_N b_1 \ldots b_j$, $1 \le j \le N$.

б) Разработайте эффективный алгоритм для определения всех таких j для заданной строки $b_1 b_2 \ldots b_N$.

в) Поясните, как сгенерировать случайный лес с $N = n_0 + \cdots + n_t$ узлами, в котором степень j имеет n_j узлов. (Например, в качестве частного случая этой общей процедуры при $N = tn + 1$, $n_0 = (t-1)n + 1$, $n_1 = \cdots = n_{t-1} = 0$ и $n_t = n$ мы получаем случайные n-узловые t-арные деревья.)

62. [*22*] Бинарное дерево может быть также представлено битовыми строками $(l_1 \ldots l_n, r_1 \ldots r_n)$, где l_j и r_j указывают, пусты ли левое и правое поддеревья узла j в прямом порядке обхода (см. теорему 2.3.1A). Докажите, что если $l_1 \ldots l_n$ и $r_1 \ldots r_n$ — произвольные битовые строки, такие, что $l_1 + \cdots + l_n + r_1 + \cdots + r_n = n - 1$, то только один циклический сдвиг $(l_{j+1} \ldots l_n l_1 \ldots l_j, r_{j+1} \ldots r_n r_1 \ldots r_j)$ дает корректное представление бинарного дерева, и поясните, как его найти.

63. [*16*] Если первые две итерации алгоритма Реми дают

, то какие декорированные бинарные деревья возможны после очередной итерации?

64. [*20*] Какая последовательность значений X в алгоритме R соответствует декорированным деревьям из (34) и каковы конечные значения $L_0 L_1 \ldots L_{12}$?

65. [*38*] Обобщите алгоритм Реми (алгоритм R) для t-арных деревьев.

66. [*21*] *Деревом Шрёдера* называется бинарное дерево, в котором каждая ненулевая правая связь окрашена либо в белый, либо в черный цвет. Вот количества S_n деревьев Шрёдера для небольших n.

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12
$S_n =$	1	1	3	11	45	197	903	4 279	20 793	103 049	518 859	2 646 723	13 648 869

Например, $S_3 = 11$, как видно из приведенных вариантов:

(Белые связи изображены как "пустотелые"; на рисунке показаны также внешние узлы.)

а) Найдите простое соответствие между деревьями Шрёдера с n внутренними узлами и обычными деревьями с $n+1$ листьями и без узлов степени 1.

б) Разработайте код Грея для деревьев Шрёдера.

67. [*M22*] Какова производящая функция $S(z) = \sum_n S_n z^n$ для чисел Шрёдера?

68. [*10*] Что собой представляет шаблон рождественской елки нулевого порядка?

69. [*20*] Содержатся ли в табл. 4 шаблоны рождественской елки порядков 6 и 7, возможно, в замаскированном виде?

▶ **70.** [*20*] Найдите простое правило, которое определяет для каждой битовой строки σ другую битовую строку σ', которая называется *напарником* (*mate*) первой строки и обладает следующими свойствами: (i) $\sigma'' = \sigma$; (ii) $|\sigma'| = |\sigma|$; (iii) либо $\sigma \subseteq \sigma'$, либо $\sigma' \subseteq \sigma$; (iv) $\nu(\sigma) + \nu(\sigma') = |\sigma|$.

71. [*M21*] Пусть M_{tn} — размер наибольшего возможного множества S, состоящего из n-битовых строк, обладающих тем свойством, что если σ и τ являются членами S, такими, что $\sigma \subseteq \tau$, то $\nu(\tau) < \nu(\sigma) + t$. (Таким образом, например, $M_{1n} = M_n$ согласно теореме Спернера.) Найдите формулу для M_{tn}.

▶ **72.** [*M28*] Если начать с единственной строки $\sigma_1\ \sigma_2\ \ldots\ \sigma_s$ длиной s и применить к ней правило роста (36) n раз подряд, то сколько строк мы получим?

73. [*15*] Каковы первый и последний элементы строки шаблона рождественской елки порядка 30, содержащей битовую строку 011001001000011111101101011100?

74. [*M26*] Продолжая выполнять предыдущее упражнение, ответьте, сколько строк предшествует указанной в нем строке?

▶ **75.** [*HM23*] Пусть $(r_1^{(n)}, r_2^{(n)}, \ldots, r_{n-1}^{(n)})$ — номера строк, в которых шаблон рождественской елки порядка n содержит $n-1$ элементов; например, из табл. 4 видно, что $(r_1^{(8)}, \ldots, r_7^{(8)}) = (20, 40, 54, 62, 66, 68, 69)$. Найдите формулы для $r_{j+1}^{(n)} - r_j^{(n)}$ и для $\lim_{n\to\infty} r_j^{(n)}/M_n$.

76. [*HM46*] Рассмотрите предельную форму шаблонов рождественской елки при $n \to \infty$. Имеет ли она, например, фрактальную размерность при выборе некоторого подходящего масштабирования?

77. [*21*] Разработайте алгоритм для генерации последовательности крайних справа элементов $a_1 \ldots a_n$ строк шаблона рождественской елки для заданного n. *Указание*: эти битовые строки характеризуются тем свойством, что $a_1 + \cdots + a_k \geq k/2$ для $0 \leq k \leq n$.

78. [*20*] Истинно или ложно следующее утверждение? Если $\sigma_1\ \ldots\ \sigma_s$ — строка шаблона рождественской елки, то ею же является и строка $\bar\sigma_s^R\ \ldots\ \bar\sigma_1^R$ (обращенная последовательность обращенных дополнений).

79. [*M26*] Количество перестановок $p_1 \ldots p_n$, у которых имеется ровно один "спуск", на котором $p_k > p_{k+1}$, равно согласно 5.1.3–(12) числу Эйлера $\left\langle {n \atop 1} \right\rangle = 2^n - n - 1$. Количество элементов в шаблоне рождественской елки, расположенных выше нижней строки, такое же.

а) Найдите комбинаторное пояснение этого совпадения, приведя взаимно однозначное соответствие между перестановками с одним спуском и неотсортированными битовыми строками.

б) Покажите, что две неотсортированные битовые строки принадлежат одной и той же строке шаблона рождественской елки тогда и только тогда, когда они соответствуют перестановкам, которые определяют одну и ту же диаграмму P при соответствии Робинсона–Шенстеда (теорема 5.1.4A).

80. [*30*] Будем говорить, что две битовые строки *совместимы*, если одну из них можно получить из другой путем преобразования подстрок $010 \leftrightarrow 100$ и $101 \leftrightarrow 110$. Например, строки

$$\begin{array}{c} 011100 \leftrightarrow 011010 \leftrightarrow 010110 \leftrightarrow 010101 \leftrightarrow 011001 \\ \qquad\qquad\qquad\quad \updownarrow \qquad\qquad \updownarrow \\ 100110 \leftrightarrow 100101 \leftrightarrow 101001 \leftrightarrow 110001 \end{array}$$

взаимно совместимы, но ни одна другая строка не совместима ни с одной из приведенных.

Докажите, что строки взаимно совместимы тогда и только тогда, когда они принадлежат одному и тому же столбцу шаблона рождественской елки и строкам одинаковой длины в этом шаблоне.

81. [*M30*] *Биклаттером* (*biclutter*) порядка (n, n') называется семейство S пар битовых строк (σ, σ'), где $|\sigma| = n$ и $|\sigma'| = n'$, обладающих тем свойством, что для различных членов (σ, σ') и (τ, τ') семейства S условия $\sigma \subseteq \tau$ и $\sigma' \subseteq \tau'$ выполняются, только если $\sigma \neq \tau$ и $\sigma' \neq \tau'$.

Воспользуйтесь шаблонами рождественской елки для того, чтобы доказать, что S содержит не более $M_{n+n'}$ пар строк.

▶ **82.** [*M26*] Пусть $E(f)$ — количество вычислений функции f алгоритмом H.

а) Покажите, что $M_n \leq E(f) \leq M_{n+1}$, причем равенство достигается, когда f — константа.

б) Какая из всех функций f, таких, что $E(f) = M_n$, минимизирует $\sum_\sigma f(\sigma)$?

в) Какая из всех функций f, таких, что $E(f) = M_{n+1}$, максимизирует $\sum_\sigma f(\sigma)$?

83. [*M20*] (Д. Хансель (G. Hansel).) Покажите, что существует не более 3^{M_n} монотонных булевых функций $f(x_1, \ldots, x_n)$ от n булевых переменных.

▶ **84.** [*HM27*] (Д. Кляйтман (D. Kleitman).) Пусть A — матрица действительных чисел размером $m \times n$, каждый столбец v которой имеет длину $\|v\| \geq 1$, и пусть b — m-мерный вектор-столбец. Докажите, что не более M_n векторов-столбцов $x = (a_1, \ldots, a_n)^T$ с компонентами $a_j = 0$ или 1 удовлетворяют условию $\|Ax - b\| < \frac{1}{2}$. *Указание*: воспользуйтесь конструкцией, аналогичной шаблону рождественской елки.

85. [*HM35*] (Филипп Голль (Philippe Golle).) Пусть V — произвольное векторное пространство, содержащееся во множестве всех действительных n-мерных векторов, но не содержащее ни одного из единичных векторов $(1, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, $\ldots$, $(0, \ldots, 0, 1)$. Докажите, что V содержит не более M_n векторов, все компоненты которых равны 0 или 1; кроме того, верхняя граница M_n достижима.

86. [*15*] Если (2) рассматривать как *ориентированный лес*, а не как упорядоченный, то какой канонический лес ему соответствует? Укажите этот лес как с использованием кодов уровней $c_1 \ldots c_{15}$, так и с помощью указателей на родительские узлы $p_1 \ldots p_{15}$.

87. [*M20*] Пусть F — упорядоченный лес, в котором k-й в прямом порядке обхода узел находится на уровне c_k, а его родительский узел — p_k, причем если узел представляет собой корень, то $p_k = 0$.

а) Сколько лесов удовлетворяет условию $c_k = p_k$ для $1 \leq k \leq n$?

б) Предположим, что F и F' имеют коды уровней $c_1 \ldots c_n$ и $c'_1 \ldots c'_n$ соответственно и родительские связи $p_1 \ldots p_n$ и $p'_1 \ldots p'_n$. Докажите, что лексикографически $c_1 \ldots c_n \le c'_1 \ldots c'_n$ тогда и только тогда, когда $p_1 \ldots p_n \le p'_1 \ldots p'_n$.

88. [*M20*] Проанализируйте алгоритм O: как часто выполняется шаг O4? Чему равно общее количество изменений p_k на шаге O5?

89. [*M46*] Как часто на шаге O5 выполняется присваивание $p_k \leftarrow p_j$?

▶ **90.** [*M27*] Если $p_1 \ldots p_n$ — каноническая последовательность указателей на родительские узлы для ориентированного леса, то граф с вершинами $\{0, 1, \ldots, n\}$ и ребрами $\{k — p_k \mid 1 \le k \le n\}$ является *свободным деревом*, т. е. связным графом без циклов (см. теорему 2.3.4.1A). И обратно: каждое свободное дерево соответствует таким способом как минимум одному ориентированному лесу. Однако указатели на родительские узлы 011 и 000 дают одно и то же свободное дерево ; аналогично 012 и 010 дают .

Цель данного упражнения — наложить на последовательности $p_1 \ldots p_n$ ограничения с тем, чтобы каждое свободное дерево получалось только один раз. В 2.3.4.4–(9) было доказано, что количество структурно различных свободных деревьев с $n + 1$ вершинами имеет весьма простую производящую функцию. С этой целью было показано, что свободное дерево всегда имеет как минимум один *центроид*.

а) Покажите, что канонический n-узловой лес соответствует свободному дереву с одним центроидом тогда и только тогда, когда ни одно дерево в лесу не имеет больше $\lfloor n/2 \rfloor$ узлов.

б) Модифицируйте алгоритм O для генерации всех последовательностей $p_1 \ldots p_n$, удовлетворяющих пункту (а).

в) Поясните, как найти все $p_1 \ldots p_n$ для свободных деревьев с *двумя* центроидами.

91. [*M37*] (Ньенхьюз (Nijenhuis) и Вильф (Wilf).) Покажите, что случайное ориентированное дерево может быть сгенерировано с помощью процедуры, аналогичной алгоритму случайного разбиения из упр. 7.2.1.4–47.

92. [*15*] Являются ли первое и последнее остовные деревья, посещенные алгоритмом S, смежными в том смысле, что они имеют $n - 2$ общих ребер?

93. [*20*] Восстанавливает ли алгоритм S первоначальное состояние графа по завершении работы?

94. [*22*] Алгоритм S требует "прокрутить стартер", т. е. найти начальное остовное дерево на шаге S1. Поясните, как это сделать.

95. [*26*] Завершите алгоритм S, реализовав проверку моста на шаге S8.

▶ **96.** [*28*] Проанализируйте приближенное время работы алгоритма S, когда передаваемый ему граф представляет собой (а) путь P_n длиной $n - 1$; (б) цикл C_n длиной n.

97. [*15*] Является ли граф (48) последовательно-параллельным?

98. [*16*] Какой последовательно-параллельный граф соответствует графу (53), если узел A — *последовательный*?

▶ **99.** [*30*] Рассмотрим последовательно-параллельный граф, представленный деревом, как в (53), со значениями узлов, удовлетворяющими условию (55). Эти значения определяют остовное или близкое дерево в соответствии с тем, равно ли значение v_p корня p единице или нулю. Покажите, что описанный далее метод генерирует все прочие конфигурации корня.

i) Вначале все непростые узлы являются активными, а все остальные — пассивными.

ii) Выбираем крайний справа активный узел p (в прямом порядке обхода); если все узлы пассивны, завершаем работу.

iii) Изменяем $d_p \leftarrow r_{d_p}$, обновляем все значения в дереве и посещаем новую конфигурацию.

iv) Активизируем все непростые узлы справа от p.

v) Если d_p прошло по всем дочерним узлам p, поскольку p — последний ставший активным узел, делаем узел p пассивным. Возвращаемся к шагу (ii).

Поясните также, как эффективно выполнить описанные шаги. *Указание*: для реализации шага (v) введите указатель z_p; делайте узел p пассивным, когда d_p становится равным z_p, и в этот момент сбрасывайте также значение z_p, делая его равным предыдущему значению d_p. Для реализации шагов (ii) и (iv) воспользуйтесь фокусными указателями f_p, аналогичными используемым в алгоритмах 7.2.1.1L и 7.2.1.1K.

100. [*40*] Реализуйте "алгоритм S′ двери-вертушки" для генерации всех остовных деревьев путем комбинирования алгоритма S и идей из упр. 99.

101. [*46*] Имеется ли простой способ перечисления в порядке двери-вертушки всех n^{n-2} остовных деревьев полного графа K_n? (Порядок, получаемый с использованием алгоритма S, достаточно сложен.)

102. [*46*] *Ориентированным остовным деревом* ориентированного графа D с n вершинами, известным также как "остовная древовидность" (spanning arborescence), является ориентированное поддерево графа D, содержащее $n-1$ дуг. Теорема о матрице, соответствующей дереву (упр. 2.3.4.2–19), гласит, что ориентированные поддеревья, имеющие данный корень, могут быть легко подсчитаны путем вычисления детерминанта размером $(n-1) \times (n-1)$.

Можно ли перечислить эти поддеревья в порядке двери-вертушки, при котором на каждом шаге происходит удаление одной дуги и замещение ее другой?

▶ **103.** [*HM39*] (*Барханы.*) Рассмотрим произвольный ориентированный граф D с вершинами V_0, V_1, …, V_n и дугами e_{ij} от V_i к V_j, где $e_{ii} = 0$. Будем считать, что D имеет как минимум одно ориентированное остовное дерево с корнем в V_0; это предположение означает, что если мы соответствующим образом перенумеруем вершины, то будут выполняться соотношения $e_{i0} + \cdots + e_{i(i-1)} > 0$ для $1 \le i \le n$. Пусть $d_i = e_{i0} + \cdots + e_{in}$ — общая исходящая степень вершины V_i. Поместим x_i песчинок на вершину V_i для $0 \le i \le n$ и сыграем в следующую игру. Если $x_i \ge d_i$ для любого $i \ge 1$, уменьшим x_i на d_i и установим $x_j \leftarrow x_j + e_{ij}$ для всех $j \ne i$. (Другими словами, если это возможно, перешлем по одной песчинке из V_i по всем исходящим дугам, за исключением вершины $i = 0$.) Назовем эту операцию рассыпанием вершины V_i, а последовательность рассыпаний — лавиной. Вершина V_0 особая; вместо того чтобы рассыпáть песчинки, она их собирает (и в результате они, по сути, покидают систему). Продолжаем до тех пор, пока не будет выполнено условие $x_i < d_i$ для $1 \le i \le n$. Такое состояние $x = (x_1, \ldots, x_n)$ назовем *устойчивым*.

а) Докажите, что каждая лавина завершается устойчивым состоянием после конечного числа рассыпаний. Более того, конечное состояние зависит только от начального состояния, но не от порядка выполнения рассыпаний.

б) Пусть $\sigma(x)$ — устойчивое состояние, являющееся результатом начального состояния x. Устойчивое состояние называется *рекуррентным*, если это $\sigma(x)$ для некоторого x с $x_i \ge d_i$ для $1 \le i \le n$. (Рекуррентные состояния соответствуют "барханам", которые образовались за длинный промежуток времени после многократного случайного введения новых песчинок.) Найдите рекуррентные состояния в частном случае $n = 4$ и дуг D

$$V_1 \to V_0,\ V_1 \to V_2,\ V_2 \to V_0,\ V_2 \to V_1,\ V_3 \to V_0,\ V_3 \to V_4,\ V_4 \to V_0,\ V_4 \to V_3.$$

в) Пусть $d = (d_1, \ldots, d_n)$. Докажите, что x является рекуррентным тогда и только тогда, когда $x = \sigma(x + t)$, где t — вектор $d - \sigma(d)$.

г) Пусть a_i — вектор $(-e_{i1}, \ldots, -e_{i(i-1)}, d_i, -e_{i(i+1)}, \ldots, -e_{in})$ для $1 \le i \le n$; таким образом, рассыпание V_i соответствует изменению вектора состояния $x = (x_1, \ldots, x_n)$, который становится равным $x - a_i$. Будем говорить, что состояния x и x' *конгруэнтны* (записывается как $x \equiv x'$), если $x - x' = m_1 a_1 + \cdots + m_n a_n$ для некоторых целых чисел $m_1, \ldots, m_n$. Докажите, что существует ровно столько же классов эквивалентности конгруэнтных состояний, сколько и ориентированных остовных деревьев в D с корнем V_0. *Указание*: см. теорему о матрице, соответствующей дереву (упр. 2.3.4.2–19).

д) Докажите, что $x = x'$, если $x \equiv x'$ и если и x, и x' рекуррентны.

е) Докажите, что каждый класс конгруэнтности содержит единственное рекуррентное состояние.

ж) Пусть D — *сбалансированный* граф, в том смысле, что входящая степень каждой вершины равна ее исходящей степени. Докажите, что в таком случае x является рекуррентным состоянием тогда и только тогда, когда $x = \sigma(x+a)$, где $a = (e_{01}, \ldots, e_{0n})$.

з) Проиллюстрируйте эту концепцию для случая, когда D представляет собой "колесо" с n спицами, когда всего имеется $3n$ дуг, $V_j \to V_0$ и $V_j \leftrightarrow V_{j+1}$ для $1 \le j \le n$, где вершина V_{n+1} рассматривается как идентичная V_1. Найдите взаимно однозначное соответствие между ориентированными остовными деревьями этого ориентированного графа и рекуррентными состояниями его барханов.

и) Аналогично проанализируйте рекуррентные барханы, когда D является *полным* графом из $n + 1$ вершин, а именно когда $e_{ij} = [i \ne j]$ для $0 \le i, j \le n$. *Указание*: см. упр. 6.4–31.

▶ **104.** [*HM21*] Пусть G — граф с n вершинами $\{V_1, \ldots, V_n\}$ и с ребрами e_{ij} между V_i и V_j. Обозначим через $C(G)$ матрицу с элементами $c_{ij} = -e_{ij} + \delta_{ij} d_i$, где $d_i = e_{i1} + \cdots + e_{in}$ — степень вершины V_i. Будем говорить, что *сторонами* (*aspects*) G являются собственные значения $C(G)$, т. е. корни $\alpha_0, \ldots, \alpha_{n-1}$ уравнения $\det(\alpha I - C(G)) = 0$. Поскольку $C(G)$ — симметричная матрица, ее собственные значения являются действительными числами, и мы можем считать, что $\alpha_0 \le \alpha_1 \le \cdots \le \alpha_{n-1}$.

а) Докажите, что $\alpha_0 = 0$.

б) Докажите, что G имеет ровно $c(G) = \alpha_1 \ldots \alpha_{n-1}/n$ остовных деревьев.

в) Чему равны стороны полного графа K_n?

105. [*HM38*] Продолжая выполнять упр. 104, мы хотим доказать, что часто есть простой путь определения сторон графа G, когда G построен из других графов, стороны которых известны. Предположим, что G' имеет стороны $\alpha'_0, \ldots, \alpha'_{n'-1}$, а G'' имеет стороны $\alpha''_0, \ldots, \alpha''_{n''-1}$. Какими будут стороны графа G в следующих случаях?

а) $G = \overline{G'}$ представляет собой дополнение G'. (Считаем, что в этом случае $e'_{ij} \le [i \ne j]$.)

б) $G = G' \oplus G''$ представляет собой прямую сумму (соединение) G' и G''.

в) $G = G' — G''$ представляет собой объединение G' и G''.

г) $G = G' \square G''$ представляет собой декартово произведение G' и G''.

д) $G = L(G')$ представляет собой линейный граф графа G', где G' является однородным графом степени d' (т. е. все вершины G' имеют ровно по d' и в графе нет петель).

е) $G = G' \otimes G''$ представляет собой прямое произведение (конъюнкцию) G' и G'', причем G' — однородный граф степени d', а G'' — однородный граф степени d''.

ж) $G = G' \boxtimes G''$ представляет собой сильное произведение однородных графов G' и G''.

з) $G = G' \triangle G''$ представляет собой нечетное произведение однородных графов G' и G''.

и) $G = G' \circ G''$ представляет собой лексикографическое произведение однородных графов G' и G''.

▶ **106.** [*HM37*] Найдите общее количество остовных деревьев (а) в решетке $P_m \square P_n$ размером $m \times n$; (б) в цилиндре $P_m \square C_n$ размером $m \times n$; (в) в торе $C_m \square C_n$ размером $m \times n$. Почему эти числа имеют тенденцию состоять только из малых простых сомножителей? *Указание*: покажите, что стороны P_n и C_n можно выразить через числа $\sigma_{kn} = 4\sin^2 \frac{k\pi}{2n}$.

107. [*M24*] Определите стороны всех связных графов, содержащих $n \le 5$ вершин и не имеющих ни петель, ни параллельных ребер.

108. [*HM40*] Распространите результаты упр. 104–106 на ориентированные графы.

109. [*M46*] Найдите комбинаторное пояснение того факта, что (57) представляет собой количество остовных деревьев в n-мерном кубе.

▶ **110.** [*M27*] Докажите, что если G — произвольный связный мультиграф без петель, то он имеет

$$c(G) > \sqrt{(d_1 - 1)\ldots(d_n - 1)}$$

остовных деревьев, где d_j — степень вершины j.

111. [*05*] Перечислите узлы дерева (58) в обратнопрямом порядке обхода.

112. [*15*] Если узел p в лесу предшествует узлу q в прямообратном порядке и следует за ним в порядке обратнопрямом, то что можно сказать об узлах p и q?

▶ **113.** [*20*] Как прямообратный и обратнопрямой порядки обхода леса F соотносятся с прямообратным и обратнопрямым порядками обхода сопряженного леса F^R (см. упр. 13)?

114. [*15*] Если вы хотите обойти весь лес в прямообратном порядке с применением алгоритма Q, то с чего вы должны начать процесс?

115. [*20*] Проанализируйте алгоритм Q: как часто выполняется каждый из шагов алгоритма в процессе полного обхода леса?

▶ **116.** [*28*] Если узлы леса F помечены от 1 до n в прямообратном порядке обхода, то будем называть узел k ($1 \le k \le n$) *везучим*, если он смежен с узлом $k+1$ в F, *невезучим*, если он удален на три шага, и *обычным* в противном случае. При таком определении узел $n+1$ представляет собой воображаемый суперкорень, рассматриваемый как родительский для каждого корня.

а) Докажите, что везучие узлы встречаются только на четных уровнях, а невезучие — только на нечетных.

б) Покажите, что количество везучих узлов ровно на один больше количества невезучих узлов, кроме случая $n = 0$.

117. [*21*] Продолжая выполнять упр. 116, ответьте, сколько n-узловых лесов не содержат невезучих узлов?

118. [*M28*] Сколько везучих узлов имеется (а) в полном t-арном дереве с $(t^k - 1)/(t - 1)$ внутренними узлами? (б) в дереве Фибоначчи порядка k с $F_{k+1} - 1$ внутренними узлами? (См. 2.3.4.5–(6) и рис. 8 в разделе 6.2.1.)

119. [*21*] *Скрученное биномиальное дерево* $\tilde{T}_n$ порядка n рекурсивно определяется правилами

$\tilde{T}_0 = \circ\,,$ $\qquad \tilde{T}_n =$

для $n > 0$.

(Сравните с 7.2.1.3–(21); мы обращаем порядок дочерних узлов на чередующихся уровнях.) Покажите, что прямообратный обход $\tilde{T}_n$ имеет простую связь с бинарным кодом Грея.

120. [*22*] Истинно или ложно следующее утверждение? Квадрат графа гамильтонов, если граф связный и не имеет мостов.

121. [*M34*] (Ф. Нойман (F. Neuman), 1964.) *Производной* графа G называется граф $G^{(\prime)}$, полученный путем удаления всех вершин степени 1 и ребер, которые с ними соединены. Докажите, что если T — свободное дерево, то его квадрат T^2 содержит гамильтонов путь тогда и только тогда, когда его производная не имеет вершин степени, большей 4, и выполняются два следующих условия.

i) Все вершины степени 3 или 4 в $T^{(\prime)}$ лежат на единственном пути.

ii) Между любыми двумя вершинами степени 4 в $T^{(\prime)}$ есть как минимум одна вершина, степень которой в T равна 2.

▶ **122.** [*31*] (*Головоломка Дьюдени (Dudeney) о представлении сотни.*) Имеется много интересных способов представить число 100 путем вставки арифметических операторов (и, возможно, скобок) в последовательность 123456789, например:

$$100 = 1 + 2 \times 3 + 4 \times 5 - 6 + 7 + 8 \times 9 = (1 + 2 - 3 - 4) \times (5 - 6 - 7 - 8 - 9)$$
$$= \big((1/((2 + 3)/4 - 5 + 6)) \times 7 + 8\big) \times 9\,.$$

а) Сколько всего имеется таких представлений числа 100? Чтобы сделать этот вопрос точным, с учетом закона ассоциативности и прочих алгебраических свойств, будем считать, что выражения записываются в каноническом виде в соответствии с приведенным далее синтаксисом.

$\langle\,\text{expression}\,\rangle \to \langle\,\text{number}\,\rangle \mid \langle\,\text{sum}\,\rangle \mid \langle\,\text{product}\,\rangle \mid \langle\,\text{quotient}\,\rangle$
$\langle\,\text{sum}\,\rangle \to \langle\,\text{term}\,\rangle + \langle\,\text{term}\,\rangle \mid \langle\,\text{term}\,\rangle - \langle\,\text{term}\,\rangle \mid \langle\,\text{sum}\,\rangle + \langle\,\text{term}\,\rangle \mid \langle\,\text{sum}\,\rangle - \langle\,\text{term}\,\rangle$
$\langle\,\text{term}\,\rangle \to \langle\,\text{number}\,\rangle \mid \langle\,\text{product}\,\rangle \mid \langle\,\text{quotient}\,\rangle$
$\langle\,\text{product}\,\rangle \to \langle\,\text{factor}\,\rangle \times \langle\,\text{factor}\,\rangle \mid \langle\,\text{product}\,\rangle \times \langle\,\text{factor}\,\rangle \mid (\langle\,\text{quotient}\,\rangle) \times \langle\,\text{factor}\,\rangle$
$\langle\,\text{quotient}\,\rangle \to \langle\,\text{factor}\,\rangle/\langle\,\text{factor}\,\rangle \mid \langle\,\text{product}\,\rangle/\langle\,\text{factor}\,\rangle \mid (\langle\,\text{quotient}\,\rangle)/\langle\,\text{factor}\,\rangle$
$\langle\,\text{factor}\,\rangle \to \langle\,\text{number}\,\rangle \mid (\langle\,\text{sum}\rangle)$
$\langle\,\text{number}\,\rangle \to \langle\,\text{digit}\,\rangle$

Используемые цифры — от 1 до 9 в указанном порядке.

б) Расширим задачу (а), допустив применение чисел, состоящих из нескольких цифр.

$$\langle\,\text{number}\,\rangle \to \langle\,\text{digit}\,\rangle \mid \langle\,\text{number}\rangle\langle\,\text{digit}\,\rangle$$

Например, $100 = \big(1/(2 - 3 + 4)\big) \times 567 - 89$. Какое из таких представлений наиболее короткое и какое наиболее длинное?

в) Расширим задачу (б), разрешив применение десятичных точек.

$$\langle\,\text{number}\,\rangle \to \langle\,\text{digit string}\,\rangle \mid .\langle\,\text{digit string}\rangle$$
$$\langle\,\text{digit string}\,\rangle \to \langle\,\text{digit}\,\rangle \mid \langle\,\text{digit string}\rangle\langle\,\text{digit}\,\rangle$$

Например, $100 = (.1 - 2 - 34 \times .5)/(.6 - .789)$.

123. [*21*] Продолжая выполнять предыдущее упражнение, ответьте, каковы наименьшие положительные целые числа, которые *не могут* быть представлены с использованием соглашений (а)–(в)?

▶ **124.** [*40*] Эксперимент с методами черчения расширенных бинарных деревьев, вызванный простыми природными моделями. Например, можно назначить каждому узлу x значение $v(x)$, именуемое *числом Хортона–Страхлера*, следующим образом. Каждый внешний узел (лист) имеет $v(x) = 0$; внутренний узел с дочерними узлами (l, r) имеет значение $v(x) = \max(v(l), v(r)) + [v(l) = v(r)]$. Ребро от внутреннего узла x до его родителя можно изобразить как прямоугольник высотой $h(v(x))$ и шириной $w(v(x))$, а прямоугольники ребер к дочерним узлам (l, r) могут быть повернуты на углы $\theta(v(l(x)), v(r(x)))$ и $-\theta(v(r(x)), v(l(x)))$ для некоторых функций h, w и θ. На рис. 63 показаны типичные результаты при выборе $w(k) = 3 + k$, $h(k) = 18k$, $\theta(k, k) = 30^\circ$, $\theta(j, k) = ((k + 1)/j) \times 20^\circ$ для $0 \le k < j$

Рис. 63. “Органические” бинарные деревья.

и $\theta(j,k) = ((k-j)/k) \times 30^\circ$ для $0 \le j < k$; корни находятся внизу. На рис. 63, (а) показано бинарное дерево (4); на рис. 63, (б) — случайное 100-узловое дерево, сгенерированное алгоритмом R; на рис. 63, (в) — дерево Фибоначчи порядка 11 со 143 узлами; а на рис. 63, (г) — случайное 100-узловое бинарное дерево поиска. (Очевидно, что деревья (б)–(г) относятся к различным видам.)

[Эта тема] связана почти со всеми полезными знаниями, которые только может воспринять человеческий разум.

— ЯКОБ БЕРНУЛЛИ (JAMES BERNOULLI), *Ars Conjectandi* (*Искусство догадки*) (1713)

7.2.1.7. Исторические и иные сведения. Первые работы по генерации комбинаторных шаблонов появились одновременно с появлением цивилизацией. Это очень интересная история, и, как мы увидим, она охватывает многие части мира и многие области человеческой деятельности, включая поэзию, музыку и религию. Здесь будут рассмотрены лишь некоторые основные вопросы, но такой краткий экскурс в историю не может не стимулировать интерес читателя и не подвигнуть его на более глубокое изучение данной темы.

Списки всех кортежей из n элементов прослеживаются тысячи лет назад в древних Китае, Индии и Греции. Наиболее примечательный источник (в силу того, что в современном переводе эта книга стала бестселлером) — это китайская *I Ching*, или *Yijing*, что означает "Книга изменений". В этой книге, одной из пяти классических книг конфуцианства, содержится ровно $2^6 = 64$ главы; каждая глава символизирует гексаграмму, образованную из шести линий, каждая из которых имеет вид либо **– –** ("инь"), либо **—** ("ян"). Например, гексаграмма 1 представляет собой чистый ян, ䷀; гексаграмма 2 — чистый инь, ䷁; а гексаграмма 64 — чередующиеся инь и ян, причем верхним элементом является ян: ䷿. Вот полный список гексаграмм.

(1)

Такое размещение 64 возможных вариантов называется упорядочением Вэнь-вана, так как авторство книги *I Ching* традиционно приписывается Вэнь-вану (ок. 1100 до н. э.), легендарному основателю династии Чжоу. К сожалению, древние тексты трудно надежно датировать, так что современные историки не видят достаточных оснований считать, что такой список гексаграмм был составлен ранее III века до н. э.

Обратите внимание, что в перечне (1) гексаграммы располагаются парами: за гексаграммой с нечетным номером идет гексаграмма, представляющая собой зеркальное отражение относительно горизонтальной оси, кроме случаев, когда гексаграмма симметрична: в этих случаях гексаграмма спарена со своим дополнением ($1 = \overline{2}$, $27 = \overline{28}$, $29 = \overline{30}$, $61 = \overline{62}$). Гексаграммы, состоящие из двух триграмм, представляющих четыре базовых элемента — воздух (☰), землю (☷), огонь (☲) и воду (☵), — также размещаются специальным образом. Остальные гексаграммы располагаются, по сути, случайным образом. Определенные шаблоны, имеющиеся в расположении пар, — не более чем совпадения, подобные совпадениям в записи числа π (см. 3.3–(1)).

Инь и ян представляют собой взаимодополняющие стороны элементарных сил природы, всегда находящиеся в противоборстве и переходящие друг в друга. В определенном смысле *I Ching* представляет собой тезаурус, в котором гексаграммы

служат указателями к накопленным знаниям о фундаментальных концепциях наподобие следующих: давать (䷀), брать (䷊), скромность (䷎), радость (䷤), товарищество (䷌), извлечение (䷩), мир (䷊), война (䷅), организация (䷬), коррупция (䷑), незрелость (䷿), изящество (䷕) и т. д. Можно выбрать пару гексаграмм случайным образом, получив вторую из первой, например, независимо заменяя каждый инь на ян и наоборот с вероятностью 1/4; такой метод дает 4096 способов размышлений об экзистенциалистских загадках, например: не может ли соответствующий марковский процесс приоткрыть тайну смысла жизни?

Строгий логический способ упорядочения гексаграмм был создан Шао Юном (Shao Yung) около 1060 года н. э. Его лексикографическое упорядочение от ䷁ к ䷗, к ䷘, к ䷙, к ䷚, к $\cdots$, к ䷾, к ䷿ (читать каждую гексаграмму нужно снизу вверх) существенно более понятное, чем порядок Вэнь-вана. Когда Г. В. Лейбниц (G. W. Leibniz) изучал эту последовательность гексаграмм в 1702 году, он пришел к ложному выводу о том, что китайские математики были знакомы с бинарной арифметикой. [См. Frank Swetz, *Mathematics Magazine* **76** (2003), 276–291. Дополнительную информацию об *I Ching* можно найти, например, в Joseph Needham, *Science and Civilisation in China* **2** (Cambridge University Press, 1956), 304–345; R. J. Lynn, *The Classic of Changes* (New York: Columbia University Press, 1994).]

Другой древнекитайский философ, Ян Сюн (Yang Hsiung), предложил систему, основанную на 81 тернарной тетраграмме вместо 64 бинарных гексаграмм. Его *Канон высшего таинства* (*Canon of Supreme Mystery*), написанный около 2 года до н. э., был не так давно переведен на английский язык Майклом Найланом (Michael Nylan) (Albany, New York: 1993). Ян описывает структуру полного иерархического тернарного дерева, в которой имеется три края, по три провинции в каждом крае, по три департамента в каждой провинции, по три семьи в каждом департаменте, и по девять коротких стихов (именуемых “оценками” (appraisals)) на каждую семью, следовательно, всего 729 стихов, т. е. почти в точности два стиха на каждый день года. Его тетраграммы располагаются в строгом лексикографическом порядке при чтении сверху вниз: 𝌀, 𝌁, 𝌂, 𝌃, 𝌄, 𝌅, 𝌆, $\ldots$, 𝍖. В действительности, как поясняется в книге Найлана (с. 28), Ян разработал простой способ вычисления ранга каждой тетраграммы, как если бы он пользовался троичной системой счисления. Таким образом, нас не должно удивлять упорядочение бинарных гексаграмм Шао Юном, хотя он и жил более чем на тысячу лет раньше.

Индийское стихосложение. Бинарные кортежи из n элементов изучались в совершенно ином контексте мудрецами Древней Индии, посвятившими себя священным ведическим песнопениям. Слоги в санскрите либо короткие (ı), либо длинные (ऽ), а изучение шаблонов слогов называется просодией (prosody). Современные авторы вместо символов ı и ऽ используют символы $\smile$ и —. Типичная ведическая строфа состоит из четырех строк с n слогами в строке для некоторого $n \geq 8$. Таким образом, нужен способ классификации всех 2^n возможных шаблонов. В классической работе Пингалы (Piṅgala) *Чандасутра* (*Chandaḥśāstra*), написанной до 400 года н. э. (вероятно, существенно ранее — точно неизвестно), описана процедура поиска индекса k любого заданного шаблона, состоящего из $\smile$ и —, а также поиск шаблона для заданного значения k. Другими словами, Пингала пояснил, как найти *ранг* данного шаблона и как найти шаблон по заданному рангу. Таким образом, он зашел в своих исследованиях дальше Ян Сюня (Yang Hsiung), который рассматривал

только задачу поиска ранга, но не поиска шаблона по заданному рангу. Методы Пингалы также связаны с возведением в степень, как мы видели ранее в связи с алгоритмом 4.6.3A.

Следующий важный шаг был сделан просодистом Кедараой (Kedāra) в его книге *Vṛttaratnākara*, которая, как считают, была написана в VIII веке. Кедара пошагово описал процедуру перечисления всех кортежей из n элементов от $— — — \ldots —$ к $\smile — — \ldots —$, к $— \smile — \ldots —$, к $\smile \smile — \ldots —$, к $— — \smile \ldots —$, к $\smile — \smile \ldots —$, к $\cdots$, к $\smile \smile \smile \ldots \smile$, т. е., по сути, алгоритм 7.2.1.1M для основания системы счисления, равного 2. Его метод можно считать первым явным алгоритмом для генерации комбинаторной последовательности. [См. B. van Nooten, *J. Indian Philos.* **21** (1993), 31–50.]

Стихотворные измерения можно также рассматривать как ритмы, с одним тактом для каждого $\smile$ и двумя для каждого $—$. Хотя n-тактовый шаблон может включать от n до $2n$ тактов, музыкальные ритмы для маршей и танцев обычно основываются на фиксированном количестве тактов. Следовательно, естественно рассматривать множество всех последовательностей $\smile$ и $—$, содержащих фиксированное количество тактов, равное m. Такие последовательности называются последовательностями кода Морзе длиной m, и из упр. 4.5.3–32 известно, что их ровно F_{m+1}. Например, 21 последовательность для $m = 7$ имеет вид

$$\begin{array}{l}
\smile — — —,\ — \smile — —,\ \smile \smile \smile — —,\ — — \smile —,\ \smile \smile — \smile —,\\
\smile — \smile \smile —,\ — \smile \smile \smile —,\ \smile \smile \smile \smile \smile —,\ — — — \smile,\\
\smile \smile — — \smile,\ \smile — \smile — \smile,\ — \smile \smile — \smile,\ \smile \smile \smile \smile — \smile,\\
\smile — — \smile \smile,\ — \smile — \smile \smile,\ \smile \smile \smile — \smile \smile,\ — — \smile \smile \smile,\\
\smile \smile — \smile \smile \smile,\ \smile — \smile \smile \smile \smile,\ — \smile \smile \smile \smile \smile,\ \smile \smile \smile \smile \smile \smile \smile.
\end{array} \tag{2}$$

Таким путем индийские просодисты пришли к открытию последовательности Фибоначчи, как уже отмечалось в разделе 1.2.8.

Кроме того, анонимный автор *Prākṛta Paiṅgala* (ок. 1320 г.) разработал элегантный алгоритм для определения ранга ритма и восстановления ритма по рангу для m-тактовых ритмов. Чтобы найти k-й шаблон, начинаем с записи m символов $\smile$, затем выражаем разность $d = F_{m+1} - k$ в виде суммы чисел Фибоначчи $F_{j_1} + \cdots + F_{j_t}$; здесь F_{j_1} — наибольшее число Фибоначчи $\le d$, F_{j_2} — наибольшее число Фибоначчи $\le d - F_{j_1}$, и так до тех пор, пока не будет получен нулевой остаток. Тогда такты $j-1$ и j заменяются с $\smile \smile$ на $—$ для $j = j_1, \ldots, j_t$. Например, для получения пятого элемента последовательности (2) вычисляем $21 - 5 = 16 = 13 + 3 = F_7 + F_4$; в результате ответом является $\smile \smile — \smile —$.

Несколькими годами позже Нараяна Пандита (Nārāyaṇa Paṇḍita) рассматривал более общую задачу поиска всех композиций числа m, части которых $\le q$, где q — *любое* заданное положительное целое число. Как следствие он открыл последовательности Фибоначчи q-го порядка 5.4.2–(4), которые пригодились через 600 лет после этого в многофазной сортировке; он также разработал соответствующие алгоритмы определения ранга элемента и элемента с заданным рангом. [См. Parmanand Singh, *Historia Mathematica* **12** (1985), 229–244, и упр. 16.]

Пингала дает специальные имена всем трехсложным измерениям:

$$\begin{array}{llll} ——— & = \text{म (m)}, & ——\smile & = \text{त (t)}, \\ \smile—— & = \text{य (y)}, & \smile—\smile & = \text{ज (j)}, \\ —\smile— & = \text{र (r)}, & —\smile\smile & = \text{भ (bh)}, \\ \smile\smile— & = \text{स (s)}, & \smile\smile\smile & = \text{न (n)}, \end{array} \tag{3}$$

и с тех пор все изучающие санскрит должны заучивать их наизусть. Давным-давно кто-то разработал способ запоминания этих кодов, придумав несуществующее слово *yamātārājabhānasalagām* (यमाताराजभानसलगाम्). Смысл в том, что десять слогов этого слова можно записать как

$$\begin{array}{cccccccccc} \text{ya} & \text{mā} & \text{tā} & \text{rā} & \text{ja} & \text{bhā} & \text{na} & \text{sa} & \text{la} & \text{gām,} \\ \smile & — & — & — & \smile & — & \smile & \smile & \smile & — \end{array} \tag{4}$$

после чего каждый трехсложный шаблон начинается с его имени. Происхождение слова yamā…lagām неизвестно, но Сабхаш Как (Subhash Kak) [*Indian J. History of Science* **35** (2000), 123–127] проследил его до книги C. P. Brown, *Sanskrit Prosody* (1869), с. 28; таким образом, его можно считать наиболее ранним известным “циклом де Брейна” для кодирования бинарных n-кортежей.

Тем временем в Европе. Классическая греческая поэзия также основана на группах коротких и/или длинных слогов, называемых метрической стопой, аналогично тактам в музыке. Каждый базовый тип стопы получил греческое имя; например, два коротких слога ‘$\smile\smile$’ получили имя *пиррихий*, а два длинных слога ‘——’ получили имя *спондей*, поскольку эти ритмы использовались соответственно в песнях войны (πυρρίχη) и мира (σπονδαί). Греческие названия метрических стоп ассимилировались латынью и в конечном счете современными языками, в том числе английским.

$\smile$ арсис (arsis)
— тезис (thesis)

$\smile\smile$ пиррихий (pyrrhic)
$\smile—$ ямб (iambus)
$—\smile$ хорей (трохей) (trochee)
—— спондей (spondee)

$\smile\smile\smile$ трибрахий (tribrach)
$\smile\smile—$ анапест (anapest)
$\smile—\smile$ амфибрахий (amphibrach)
$\smile——$ бакхий (bacchius)
$—\smile\smile$ дактиль (dactyl)
$—\smile—$ амфимакр (amphimacer)
$——\smile$ антибакхий (palimbacchius)
——— молосс (molossus)

$\smile\smile\smile\smile$ гиперпиррихий (прокелевсаматик) (proceleusmatic)
$\smile\smile\smile—$ четвертый пеон (fourth p?on)
$\smile\smile—\smile$ третий пеон (third p?on)
$\smile\smile——$ ионик меньший (нисходящий) (minor ionic)
$\smile—\smile\smile$ второй пеон (second p?on)
$\smile—\smile—$ диямб (diiambus)
$\smile——\smile$ антиспаст (antispast)
$\smile———$ первый эпитрит (first epitrite)
$—\smile\smile\smile$ первый пеон (first p?on)
$—\smile\smile—$ хориямб (choriambus)
$—\smile—\smile$ дихорей (дитрохей) (ditrochee)
$—\smile——$ второй эпитрит (second epitrite)
$——\smile\smile$ ионик больший (восходящий) (major ionic)
$——\smile—$ третий эпитрит (third epitrite)
$———\smile$ четвертый эпитрит (fourth epitrite)
———— диспондей (dispondee)

(5)

Зачастую используются альтернативные имена, например “хорей” (choree) вместо “трохей ” (trochee) или “кретик” (cretic) вместо “амфимакр” (amphimacer). Более того, во времена, когда Диомед (Diomedes) писал свою латинскую грамматику (ок. 375 г. н. э.), каждой из 32 *пятисложных* стоп было присвоено как минимум одно имя. Диомед также указал на отношения между дополняющими шаблонами; он привел пример трибрахия и молосса как *противоположностей*, подобно амфибрахию и амфимакру. Однако он также рассматривал как противоположности дактиль и анапест, а также бакхий и антибакхий, хотя сам термин *palimbacchius* означает “обратный бакхий”. У греков не было стандартного порядка перечисления стоп, а их названия никак не связаны с бинарными числами. [См. H. Keil, *Grammatici Latini* **1** (1857), 474–482; W. von Christ, *Metrik der Griechen und Römer* (1879), 78–79.]

Дошедшие до нас фрагменты работы Аристоксена (Aristoxenus) под названием *Элементы ритмов* (ок. 325 г. до н. э.) показывают, что та же терминология применялась и в музыке. Эти традиции пережили эпоху Возрождения; например, в книге Athanasius Kircher, *Musurgia Universalis* **2** (Rome: 1650) на с. 32 можно встретить следующий фрагмент.

Кирхер описал все трех- и четырехнотные ритмы, соответствующие (5).

Ранние списки перестановок. Мы уже проследили историю формулы для подсчета перестановок в разделе 5.1.2; однако нетривиальные *списки* перестановок не публиковались еще сотни лет после того, как была открыта формула $n!$. Первая из таких известных ныне таблиц была составлена итальянским врачом Шаббетаи Донноло (Shabbetai Donnolo) в его комментариях к каббалистической книге *Sefer Yetzirah*, написанной в 946 году. В табл. 1 приведен его список для $n = 5$ в том виде, в котором он был опубликован в Варшаве в 1884 году. (Буквы иврита в этой таблице написаны с использованием средневекового стиля, традиционно применявшегося в комментариях; обратите внимание, что буква מ заменяется буквой ם на левом конце слова.) Донноло перечислил все 120 перестановок шестибуквенного слова סכתבמ, начинающиеся с буквы ‘ס’; затем он отметил, что следующие 120 перестановок можно получить при использовании другой буквы в качестве первой, т. е. всего 720 перестановок. Его список включает группирование по шесть перестановок, но оно выполнено случайным образом, что и привело к ошибке (см. упр. 4). Хотя Донноло и знал общее количество перестановок и перестановок, начинающихся с данной буквы, алгоритма для их генерации у него не было.

Полный список всех 720 перестановок множества $\{a, b, c, d, e, f\}$ имеется на с. 668–671 книги Jeremias Drexel, *Orbis Phaëthon* (Munich: 1629); в Кёльнском издании 1631 года этот список находится на с. 526–531. Автор приводит его в качестве доказательства того, что хозяин может размещать шестерых гостей за завтраком и обедом разными способами в течение года — всего 360 дней; пять дней в году он отвел для поста на Страстной неделе. Вскоре после этого Марен Мерсенн (Marin Mersenne) привел все 720 перестановок шести нот {до, ре, ми, фа, соль, ля} на

Таблица 1

Средневековый список перестановок

דבריס	דבירס	דבימר	דבמיר	דבמרי	דברמי	דרבים	דרבמי	דרימב	דריבס	דרמבי	דרמיב
דיברס	דירבס	דירמב	דיבמר	דימרב	דימבר	דמביר	דמברי	דמריב	דמרבי	דמיבר	דמירב
בדימר	בדירס	בדרמי	בדרים	בדמרי	בדמיר	בירמד	בירדס	בימדר	בימרד	בידמר	בידרס
במדרי	במדיר	במריד	במרדי	במדיר	במירד	בדמיר	בדמרי	בדרמי	בדרים	בדימר	בדירס
רימבד	רימבס	רידמב	רידבס	ריבמד	ריבדס	רדבמי	רדמבי	רדימב	רדיבס	רדמיב	רדבים
רבמדי	רבדמי	רבדים	רבירס	רבימד	רבמיד	רמיבד	רמיבס	רמדיב	רמבדי	רמבדי	רמדבי
ימדרב	ימרבד	ימבדר	ימבבר	ימרבד	ימבדר	ירבמד	ירבדס	ירדבס	ירדמב	ידמבר	ידרמב
יבדרס	יבדמר	יבדרס	יברמד	יבמדר	יבמרד	ירדמב	ירדבס	ירמבד	ירבמד	ירבדס	ירדמב
מדבריי	מדריי	מדיבר	מדירב	מדריב	מדבריי	מבירד	מבדיר	מבריד	מבדרי	מבדרי	מבדריי
מרדבי	מרבדי	מרדיב	מרבדי	מרבדי	מרדיב	מיבבר	מיבדר	מיבדר	מיבבר	מיבדר	מיבבר

с. 111–115 своей книги *Traitez de la Voix et des Chants* (2-й том *Harmonie Universelle*, 1636); далее, на с. 117–128, он привел те же данные в нотной записи.

Таблица Дрекселя организована лексикографически по столбцам; таблицы Мерсенна упорядочены лексикографически в соответствии с отношением до < ре < ми < фа < соль < ля и начинаются с “до, ре, ми, фа, соль, ля”, а заканчиваются “ля, соль, фа, ми, ре, до”. Мерсенн также подготовил огромную рукопись, в которой перечислил все 40 320 перестановок *восьми* нот на 672 страницах, после чего привел алгоритмы для поиска ранга перестановки и перестановки по заданному рангу [Bibliothèque nationale de France, Fonds Francais, no. 24256].

Из раздела 7.2.1.2 мы знаем, что важные идеи алгоритма 7.2.1.2P были разработаны в Англии несколькими годами позднее.

Методы перечисления всех перестановок мультимножества с *повторяющимися* элементами часто неверно понимались ранними авторами. Например, когда Бхаскара (Bhāskara) представил перестановки множества $\{4, 5, 5, 5, 8\}$ в разделе 271 своей книги *Līlāvatī* (ок. 1150 г.), он привел их в следующем порядке.

(6)

Мерсенн использовал более практичный, но не совсем систематический порядок при перечислении шестидесяти анаграмм слова IESVS (“Иисус” в латинской транскрипции) на с. 131 своей книги. Когда Анастасиус Кирхер решил проиллюстрировать 30 перестановок пятинотной мелодии на с. 10–11 книги *Musurgia Universalis* **2** (1650), такое отсутствие системы привело его к неприятностям (см. упр. 5).

(7)

Джон Уоллис (John Wallis) справился со своей задачей успешнее. На с. 117 своей книги *Discourse of Combinations* (1685) он корректно перечислил 60 анаграмм слова "messes" в лексикографическом порядке, если считать $\mathrm{m} < \mathrm{e} < \mathrm{s}$; а на с. 126 он рекомендовал использовать алфавитный порядок, "при котором мы можем быть уверены, что ничего не пропустим".

Позже мы увидим, что индийские ученые Сарнгадева (Śārṅgadeva) и Нараяна (Nārāyaṇa) разработали теорию генерации перестановок в XIII и XIV веках, но эти работы опередили свое время и остались незамеченными.

Список Секи. Такаказу Секи (Takakazu Seki) (1642–1708) был харизматичным учителем и ученым, революционизировавшим изучение математики в Японии XVII века. При изучении исключения переменных из систем однородных уравнений он пришел к выражениям типа $a_1b_2 - a_2b_1$ и $a_1b_2c_3 - a_1b_3c_2 + a_2b_3c_1 - a_2b_1c_3 + a_3b_1c_2 - a_3b_2c_1$, которые сегодня известны как *определители* (*детерминанты*). В 1683 году он опубликовал брошюру с этим открытием, в которую включил остроумную схему для перечисления всех перестановок таким образом, чтобы одна половина из них были "живыми" (четными), а другая половина — "мертвыми" (нечетными). Начиная со случая $n = 2$, когда '12' — живая перестановка, а '21' — мертвая, он сформулировал следующие правила для $n > 2$.

1) Возьми каждую живую перестановку для $n-1$, увеличь все ее элементы на 1 и вставь 1 впереди. Это правило дает $(n-1)!/2$ "базовых перестановок" множества $\{1, \ldots, n\}$.
2) Из каждой базовой перестановки образуй $2n$ других перестановок путем поворотов и отражений:

$$a_1a_2\ldots a_{n-1}a_n,\ a_2\ldots a_{n-1}a_na_1,\ \ldots,\ a_na_1a_2\ldots a_{n-1}; \tag{8}$$
$$a_na_{n-1}\ldots a_2a_1,\ a_1a_na_{n-1}\ldots a_2,\ \ldots,\ a_{n-1}\ldots a_2a_1a_n. \tag{9}$$

Если n нечетно, перестановки в первой строке живые, а во второй — мертвые; если n четно, перестановки в каждой строке чередуются — живая, мертвая, ..., живая, мертвая.

Например, при $n = 3$ единственной базовой перестановкой является 123. Таким образом, перестановки 123, 231, 312 — живые, в то время как перестановки 321, 132, 213 — мертвые, и мы успешно генерируем шесть членов определителя 3×3. Базовыми перестановками при $n = 4$ являются 1234, 1342, 1423; скажем, из 1342 мы получаем множество из восьми элементов, а именно

$$+1342 - 3421 + 4213 - 2134 + 2431 - 1243 + 3124 - 4312, \tag{10}$$

с чередующимися живыми (+) и мертвыми (−) перестановками. Определитель 4×4, таким образом, включает члены $a_1b_3c_4d_2 - a_3b_4c_2d_1 + \cdots - a_4b_3c_1d_2$, а также шестнадцать других.

Правило Секи для генерации перестановок достаточно привлекательно, но, к сожалению, у него имеется серьезная проблема: оно не работает при $n > 4$. Похоже, однако, что эта ошибка оставалась незамеченной столетиями. [См. Y. Mikami, *The Development of Mathematics in China and Japan* (1913), 191–199; *Takakazu Seki's Collected Works* (Osaka: 1974), 18–20, 八五一-一四一; и упр. 7–8.]

Списки сочетаний. Наиболее ранний исчерпывающий список *сочетаний*, выдержавший разрушительное действие времени, находится в главе 63 последней книги Сусруты (Suśruta), известном медицинском трактате на санскрите, написанном до 600 года (вероятнее всего, намного ранее). Заметив, что лекарства могут быть сладкими, кислыми, солеными, острыми, горькими и вяжущими, Сусрута старательно перечислил все $(15, 20, 15, 6, 1, 6)$ случаи, когда одновременно встречаются два, три, четыре, пять, шесть и одно из этих качеств.

Бхаскара (Bhāskara) повторил этот пример в разделах 110–114 своей книги *Līlāvatī* и заметил, что те же самые рассуждения применимы и к шестисложным стихотворным шаблонам с заданным количеством длинных слогов. Однако он просто упомянул их количества, $(6, 15, 20, 15, 6, 1)$, без перечисления самих сочетаний. В разделах 274 и 275 он заметил, что числа $(n(n-1)\ldots(n-k+1))/(k(k-1)\ldots(1))$ указывают количество *композиций*, т. е. упорядоченных разбиений, однако и это замечание сделано без приведения полного списка.

Рассмотрим этот вопрос вкратце, избегая многословия,
поскольку наука о вычислениях — безбрежный океан.

— БХАСКАРА (BHĀSKARA) (ок. 1150 г.)

Отдельный, но интересный список сочетаний приведен в знаменитой работе по алгебре *Al-Bāhir fi'l-ḥisāb* (*Блистательная книга о вычислениях*), написанной аль-Самавалем (al-Samaw'al) из Багдада в 1144 году, когда ему было всего лишь 19 лет. В заключительной части работы он представил список из $\binom{10}{6} = 210$ линейных уравнений с 10 неизвестными.

Арабский оригинал аль-Самаваля			Эквивалентная современная запись	
٦٥	٦٥٤٣٢١	ا	(1) $x_1+x_2+x_3+x_4+x_5+x_6=65$	
٧٠	٧٥٤٣٢١	ب	(2) $x_1+x_2+x_3+x_4+x_5+x_7=70$	
٧٥	٨٥٤٣٢١	ج	(3) $x_1+x_2+x_3+x_4+x_5+x_8=75$	(11)
	⋮		⋮	
٩١	١٠٩٨٧٦٤	رط	(209) $x_4+x_6+x_7+x_8+x_9+x_{10}=91$	
١٠٠	١٠٩٨٧٦٥	ري	(210) $x_5+x_6+x_7+x_8+x_9+x_{10}=100$	

Каждое сочетание шести из десяти элементов дает одно из уравнений. Целью аль-Самаваля, несомненно, была демонстрация переопределенной системы уравнений, которая, тем не менее, имеет единственное решение; для приведенной системы это $(x_1, x_2, \ldots, x_{10}) = (1, 4, 9, 16, 25, 10, 15, 20, 25, 5)$. [Salah Ahmad and Roshdi Rashed, *Al-Bāhir en Algèbre d'As-Samaw'al* (Damascus: 1972), 77–82, ٢٤٨–٢٣١.]

Игральные кости. Определенный проблеск в области элементарной комбинаторики был и в средневековой Европе, особенно в связи с вопросом о всех возможных сочетаниях выпадения трех костей. Разумеется, всего имеется $\binom{8}{3} = 56$ способов выбрать три элемента из шести при разрешенных повторениях. В то время азартные игры были официально запрещены, но это не помешало упомянутым 56 способам стать общеизвестными. Около 965 года епископ Вибольд (Wibold) из Камбраи на севере Франции разработал игру под названием Ludus Clericalis, дабы ею могли насладиться смиренные пастыри, оставаясь при этом достаточно набожными. Его

идея состояла в том, чтобы связать с каждым набором очков одну из 56 добродетелей в соответствии с приведенной таблицей.

⚀⚀⚀	Любовь	⚀⚅⚅	Великодушие	⚁⚁⚅	Молитва
⚀⚀⚁	Вера	⚀⚄⚄	Мудрость	⚁⚁⚄	Привязанность
⚀⚀⚂	Надежда	⚀⚀⚀	Сострадание	⚁⚄⚄	Суд Божий
⚀⚀⚃	Справедливость	⚀⚀⚁	Радость	⚁⚄⚅	Бдительность
⚀⚀⚅	Благоразумие	⚀⚀⚄	Воздержанность	⚁⚄⚄	Покорность
⚀⚀⚄	Умеренность	⚀⚀⚅	Удовлетворенность	⚁⚅⚅	Невинность
⚀⚁⚀	Храбрость	⚀⚀⚄	Безмятежность	⚁⚅⚄	Раскаяние
⚀⚁⚁	Дружелюбие	⚀⚁⚁	Мастерство	⚁⚄⚄	Исповедь
⚀⚁⚃	Целомудрие	⚀⚁⚃	Простота	⚃⚃⚃	Зрелость
⚀⚁⚅	Милосердие	⚀⚁⚅	Гостеприимство	⚃⚃⚅	Забота
⚀⚁⚄	Послушание	⚀⚁⚄	Бережливость	⚃⚃⚄	Постоянство
⚀⚂⚁	Страх Божий	⚀⚃⚃	Терпение	⚃⚅⚅	Ум
⚀⚂⚃	Предусмотрительность	⚀⚃⚅	Усердие	⚃⚅⚄	Томление
⚀⚂⚅	Осторожность	⚀⚃⚄	Бедность	⚃⚄⚄	Плач
⚀⚂⚄	Настойчивость	⚀⚅⚅	Мягкость	⚅⚅⚅	Веселье
⚀⚃⚃	Доброжелательность	⚀⚅⚄	Девственность	⚅⚅⚄	Сочувствие
⚀⚃⚅	Скромность	⚀⚄⚄	Уважение	⚅⚄⚄	Самообладание
⚀⚃⚄	Смирение	⚁⚁⚁	Благочестие	⚄⚄⚄	Повиновение
⚀⚅⚅	Доброта	⚁⚁⚃	Снисхождение		

Игроки бросают кости, и первый, у кого выпало сочетание очков, соответствующее некоторой добродетели, получает ее. После того как все сочетания окажутся выпавшими, победителем становится наиболее добродетельный из игроков. Вибольд заметил, что любовь — наивысшая добродетель из всех. Он разработал сложную систему подсчета очков, по которой две добродетели могли комбинироваться, если сумма очков на всех шести костях добродетелей равнялась 21; например, так можно скомбинировать "любовь + повиновение" или "целомудрие + ум", и такие комбинации ценились выше любых отдельных добродетелей. Вибольд рассматривал и более сложные варианты игры, в которых вместо точек на костях использовались гласные буквы.

При первом описании (примерно 150 лет спустя) игры Балдериком (Baldéric) в его *Chronicon Cameracense* таблица Вибольда была представлена в лексикографическом порядке, как это сделано выше. [*Patrologia Latina* **134** (Paris: 1884), 1007–1016.] Однако в другом средневековом манускрипте возможные результаты бросания костей представлены в совсем ином порядке.

⚄⚄⚄	⚅⚅⚅	⚅⚅⚀	⚁⚁⚁	⚀⚀⚅	⚀⚀⚀	⚄⚄⚅
⚄⚄⚀	⚅⚅⚀	⚅⚅⚅	⚁⚁⚀	⚀⚅⚅	⚀⚅⚅	⚄⚄⚁
⚅⚅⚅	⚅⚅⚅	⚅⚅⚄	⚁⚁⚅	⚅⚅⚅	⚅⚅⚅	⚄⚄⚀
⚁⚁⚄	⚅⚅⚁	⚅⚅⚁	⚁⚁⚁	⚁⚁⚁	⚁⚁⚄	⚄⚄⚅
⚀⚀⚀	⚅⚅⚅	⚅⚅⚀	⚀⚀⚄	⚁⚁⚁	⚁⚁⚁	⚄⚄⚁
⚀⚀⚀	⚅⚅⚅	⚅⚅⚅	⚀⚀⚁	⚁⚁⚅	⚁⚁⚁	⚄⚄⚁
⚄⚄⚁	⚅⚅⚁	⚅⚅⚄	⚁⚁⚄	⚁⚁⚁	⚁⚁⚁	⚄⚄⚁
⚁⚁⚁	⚅⚅⚁	⚅⚅⚁	⚁⚁⚁	⚁⚁⚁	⚁⚁⚁	⚄⚄⚁

(12)

В этом случае автор знал, как следует поступить с повторяющимися значениями, но использовал очень сложный, разработанный для данной конкретной задачи способ

обработки ситуаций, когда очки на всех трех костях различны. [См. D. R. Bellhouse, *International Statistical Review* **68** (2000), 123–136.]

В начале 1400-х годов было написано забавное стихотворение для вечеринок, "Chaunce of the Dyse", приписываемое Джону Лидгейту (John Lydgate). Его начальные строфы приглашают каждого бросить три игральные кости; в остальных строфах, индексированных в обратном лексикографическом порядке от ⚅⚅⚅ до ⚅⚅⚁, до $\cdots$, до ⚀⚀⚀, приводится 56 веселых описаний характера бросающего. [Полный текст опубликован Э. П. Хаммонд (E. P. Hammond) в *Englische Studien* **59** (1925), 1–16; перевод на современный английский язык был бы очень кстати.]

I pray to god that euery wight may caste
Vpon three dyse ryght as is in hys herte
Whether he be rechelesse or stedfaste
So moote he lawghen outher elles smerte
He that is gilty his lyfe to converte
They that in trouthe haue suffred many a throwe
Moote ther chaunce fal as they moote be knowe.

— *The Chaunce of the Dyse* (ок. 1410 г.)

Раймунд Луллий. Значительный вклад в комбинаторные концепции внесен энергичным донкихотствующим каталонским поэтом, писателем-романистом, энциклопедистом, преподавателем, мистиком и миссионером по имени Раймунд Луллий (Ramon Llull) (ок. 1235–1315 гг.). Подход Луллия к знаниям состоял, по сути, в определении базовых принципов с последующим рассмотрением всех возможных их сочетаний.

Например, одна из глав его *Ars Compendiosa Inveniendi Veritatem* (ок. 1274 г.) начинается с перечисления шестнадцати свойств, присущих Богу: доброта, величие, бессмертие, могущество, мудрость, любовь, добродетель, истина, слава, совершенство, справедливость, великодушие, милосердие, скромность, владычество и терпение. Затем Луллий пишет $\binom{16}{2} = 120$ кратких эссе (приблизительно по 80 слов каждое), рассматривающих доброту Бога в связи с величием, доброту в связи с бессмертием и т. д. В другой главе он рассматривает семь добродетелей (веру, надежду, милосердие, справедливость, благоразумие, силу духа и уверенность) и семь пороков (чревоугодие, похоть, жадность, леность, гордыню, зависть и гнев), написав $\binom{14}{2} = 91$ подраздел, в каждом из которых попарно рассматриваются добродетель и порок. Другие главы так же систематически разделены на $\binom{8}{2} = 28$, $\binom{15}{2} = 105$, $\binom{4}{2} = 6$ и $\binom{16}{2} = 120$ подразделов. (Интересно, что было бы, если бы Луллий был знаком со списком Вибольда из 56 добродетелей? Написал бы он комментарии к каждой из $\binom{56}{2} = 1540$ пар?)

Луллий иллюстрировал свою методологию, изображая круговые диаграммы наподобие приведенных на рис. 64. Фигура слева, *Deus*, именует шестнадцать свойств Бога — те же, которые перечислены выше, за исключением того, что любовь (*amor*) здесь названа волей (*voluntas*), а последними четырьмя являются простота, авторитет, милосердие и владычество в указанном порядке. Каждому свойству назначена буква, и диаграмма иллюстрирует их взаимоотношения в виде полного графа K_{16} на множестве вершин (B, C, D, E, F, G, H, I, K, L, M, N, O, P, Q, R). Фигура справа, *vir-*

Рис. 64. Иллюстрация из манускрипта, подаренного Раймундом Луллием венецианскому дожу в 1280 году. [Из его *Ars Demonstrativa*, Biblioteca Marciana, VI 200, folio 3^{v}.]

tutes et vitia, показывает семь добродетелей (b, c, d, e, f, g, h), чередующихся с семью пороками (i, k, l, m, n, o, p); в оригинале рукописи добродетели изображены синими чернилами, а пороки — красными. Обратите внимание, что в этом случае на иллюстрации содержатся два независимых полных графа K_7, по одному для каждого цвета. (Луллий больше не пытается сравнивать каждую отдельную добродетель с каждым отдельным пороком, поскольку и так понятно, что каждая добродетель лучше любого порока.)

Тот же подход использован Луллием и тогда, когда он писал о медицине: вместо сопоставления теологических концепций в его книге *Liber Principiorum Medicin?* (ок. 1275 г.) рассматриваются сочетания симптомов и лечения. Луллий также писал книги о философии, логике, юриспруденции, астрологии, зоологии, геометрии, риторике и рыцарстве — всего более двухсот работ. Следует отметить, однако, высокую повторяемость его книг. Современные методы сжатия данных, вероятно, привели бы к тому, что труды Луллия после сжатия были бы существенно меньшего размера, чем, скажем, труды Аристотеля.

В конечном счете Луллий решил упростить свою систему, работая в основном с группами из девяти элементов. Взгляните, например, на рис. 65, на котором в окружности, изображенной слева вверху, показаны только первые девять из всех свойств, присущих Богу (B, C, D, E, F, G, H, I, K). В ступенчатой диаграмме, расположенной рядом с окружностью, приведены $\binom{9}{2} = 36$ пар свойств (BC, BD, ..., IK). Добавив к семи две добродетели (терпение и сострадание) и к семи два порока (ложь и непостоянство), Луллий рассматривает пары добродетелей и пороков с помощью той же диаграммы. Он предложил также использовать эту диаграмму для проведения выборов при девяти кандидатах [см. I. McLean and J. London, *Studia Lulliana* **32** (1992), 21–37].

Рис. 65. Иллюстрации из манускрипта Луллия, подаренного королеве Франции (ок. 1325 г.) [Badische Landesbibliothek Karlsruhe, Codex St. Peter perg. 92, folios 28^{v} and 39^{v}.]

Вписанные в окружность треугольники (см. рис. 65, слева внизу) иллюстрируют еще один ключевой аспект подхода Луллия. Треугольник (B, C, D) означает (различность, совместимость, противоположность), треугольник (E, F, G) означает (начало, средина, конец), а треугольник (H, I, K) — (больше, равно, меньше). Эти три вложенных графа K_3 представляют три вида трехзначной логики. Луллий ранее экспериментировал с другими такими тройками, в особенности с тройкой (истинно, неизвестно, ложно). Представление о том, как он использовал эти треугольники, можно получить, познакомившись с анализом Луллием четырех основных элементов (земля, воздух, огонь, вода). Все четыре элемента различны. Земля совместима с огнем, который совместим с воздухом, который совместим с водой, которая совместима с землей. Земля является противоположностью воздуху, а огонь — воде. Этим завершается анализ с использованием треугольника (B, C, D). Переходя к треугольнику (E, F, G), Луллий замечает, что разные процессы в природе начинаются при доминировании одного элемента над другим; затем выполняется переход, или среднее состояние, после чего достигается конечное состояние, например воздух становится горячим. По поводу треугольника (H, I, K) Луллий говорит, что в общем случае огонь > воздух > вода > земля по отношению к их “сферам”, “скоростям” и “благородству”; тем не менее мы также имеем, например, воздух > огонь по отношению к поддержанию жизни, а при совместной работе воздух и огонь равны.

На рис. 65 справа приведена вертикальная таблица, смысл которой станет понятен из упр. 11. Луллий также разработал подвижные концентрические колеса, помеченные буквами (B, C, D, E, F, G, H, I, K) и другими именами, что позволило ему работать со многими разнообразными типами элементов одновременно. Работая таким способом, верный последователь Луллия мог быть уверен, что рассматривает все возможные случаи. [Луллий мог видеть подобные колеса в ближайшей еврейской общине; см. M. Idel, *J. Warburg and Courtauld Institutes* **51** (1988), 170–174, а также вклейки 16 и 17.]

Несколькими столетиями позже Анастсиус Кирхер (Athanasius Kircher) опубликовал развитие системы Луллия как часть большого тома, озаглавленного *Ars Magna Sciendi sive Combinatoria* (Amsterdam: 1669), в котором на с. 173 имелось пять подвижных колес. Кирхер также расширил множество используемых Луллием полных графов K_n, приведя иллюстрации полных *двудольных* графов $K_{m,n}$; например, рис. 66 взят со с. 171 книги Кирхера; на с. 170 той же книги показан граф $K_{18,18}$.

Рис. 66. Граф $K_{9,9}$ опубликованный Кирхером в 1669 году.

Это искусство исследовать и изобретать.
Когда идеи комбинируются всеми возможными способами,
новые комбинации открывают новые пути для размышлений
и приводят к открытию новых истин и аргументов.

— МАРТИН ГАРДНЕР (MARTIN GARDNER),
Логические машины и диаграммы (*Logic Machines and Diagrams*) (1958)

Наиболее далеко идущим современным развитием методов в духе Луллия, вероятно, является работа Иосифа Шиллингера (Joseph Schillinger) *The Schillinger System of Musical Composition* (New York: Carl Fischer, 1946), замечательный двухтомник, рассматривающий теории ритма, мелодии, гармонии, контрапунктов, композиции, оркестровки и прочего с точки зрения комбинаторики. Например, на с. 56 Шиллингер перечисляет 24 перестановки $\{a, b, c, d\}$ в порядке кода Грея (алгоритм 7.2.1.2P); затем на с. 57 он применяет их не к мелодии, а к ритму, к длительности нот. На с. 364 он приводит симметричный цикл

$$(2, 0, 3, 4, 2, 5, 6, 4, 0, 1, 6, 2, 3, 1, 4, 5, 3, 6, 0, 5, 1), \tag{13}$$

универсальный цикл 2-сочетаний для семи объектов $\{0, 1, 2, 3, 4, 5, 6\}$; другими словами, (13) представляет собой эйлеров путь в K_7: вся $\binom{7}{2} = 21$ пара цифр встре-

чаются ровно по одному разу. Такие шаблоны — настоящая золотая жила композитора, но мы должны быть благодарны за то, что лучшие ученики Шиллингера (наподобие Джорджа Гершвина (George Gershwin)) все же не ограничились строго математическим смыслом эстетики.

Тако, ван Шутен и Изкуэрдо. В 1650-х годах были опубликованы еще три книги, связанные с рассматриваемой нами темой. Андрэ Тако (André Tacquet) написал общедоступную книгу *Arithmetic? Theoria et Praxis* (Louvain: 1656), которая перепечатывалась и исправлялась в течение следующих 50 лет. В конце книги, на с. 376 и 377, Тако привел процедуру для перечисления сочетаний по два, по три и т. д.

Книга Франса ван Шутена (Frans van Schooten) *Exercitationes Mathematicæ* (Leiden: 1657) была более серьезной и академичной. На с. 373 в ней перечислены все сочетания в виде привлекательной схемы

$$\begin{array}{c} a \\ \hline b.\ ab \\ \hline c.\ ac.\ bc.\ abc \\ \hline d.\ ad.\ bd.\ abd.\ cd.\ acd.\ bcd.\ abcd \end{array}, \tag{14}$$

и следующие несколько страниц посвящены расширению данного шаблона при введении букв e, f, g, h, i, k “и так до бесконечности”. На с. 376 автор замечает, что в выражении (14) можно заменить (a, b, c, d) на $(2, 3, 5, 7)$ и получить все делители числа 210, превышающие единицу:

$$\begin{array}{c} 2 \\ \hline 3\ \ 6 \\ \hline 5\ \ 10\ \ 15\ \ 30 \\ \hline 7\ \ 14\ \ 21\ \ 42\ \ 35\ \ 70\ \ 105\ \ 210 \end{array}. \tag{15}$$

На следующей странице первоначальная идея распространяется на

$$\begin{array}{c} a \\ \hline a.\ aa \\ \hline b.\ ab.\ aab \\ \hline c.\ ac.\ aac.\ bc.\ abc.\ aabc \end{array}, \tag{16}$$

таким образом позволяя иметь две буквы a. Тем не менее в действительности автор не понял это расширение; его следующий пример

$$\begin{array}{c} a \\ \hline a.\ aa \\ \hline a.\ aaa \\ \hline b.\ ab.\ aab.\ aaab \\ \hline b.\ bb.\ abb.\ aabb.\ aaabb \end{array} \tag{17}$$

выполнен неверно, указывая границы знаний того времени (см. упр. 13.)

На с. 411 ван Шутен замечает, что веса $(a, b, c, d) = (1, 2, 4, 8)$ в (14) при использовании сложения дают

$$\begin{array}{c} 1 \\ \hline 2\ \ 3 \\ \hline 4\ \ 5\ \ 6\ \ 7 \\ \hline 8\ \ 9\ \ 10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15 \end{array}. \tag{18}$$

Однако он не увидел здесь связи с двоичными числами.

Двухтомник Себастьяна Изкуэрдо (Sebastián Izquierdo) *Pharus Scientiarum* (Lyon: 1659) — "Маяк науки" — включает в себя хорошо организованное обсуждение комбинаторики под заголовком "Disputatio 29, *De Combinatione*". Автор приводит детальное обсуждение четырех ключевых частей "двенадцатизадачия" Стенли, а именно n-кортежей, n-размещений, n-мультисочетаний и n-сочетаний из m объектов, которые находятся в первых двух строках и двух столбцах табл. 7.2.1.4–1.

В разделах 81–84 *De Combinatione* автор перечисляет (в лексикографическом порядке) все сочетания m букв по n для $2 \le n \le 5$ и $n \le m \le 9$; он также протабулировал их для $m = 10$ и 20 при $n = 2$ и 3. Но при перечислении $m^{\underline{n}}$ *размещений* m элементов по n он использовал более сложный порядок (см. упр. 14).

Изкуэрдо был первым, кто открыл формулу $\binom{m+n-1}{n}$ для сочетаний m элементов по n с неограниченными повторениями; это правило встречается в §48–51 его книги. Но в §105, при попытке перечислить все такие сочетания при $n = 3$, он не знает, что существует простой способ сделать это. Фактически его перечисление 56 случаев для $m = 6$ больше напоминает старый запутанный порядок (12).

Сочетания с повторениями не были хорошо поняты до тех пор, пока в 1713 году не вышла книга Якоба Бернулли (James Bernoulli) *Ars Conjectandi* ("Искусство догадки"). В части 2, главе 5, Бернулли просто перечислил все возможности в лексикографическом порядке и показал, что формула $\binom{m+n-1}{n}$ является простым следствием применения индукции. [Никколо Тарталья (Niccolò Tartaglia), кстати, близко подошел к получению этой формулы в своей книге *General trattato di numeri, et misure* **2** (Venice: 1556), 17^{r} и 69^{v}; то же справедливо и в отношении магрибского математика Ибн Мунима (Ibn Mun'im) и его книги *Fiqh al-Ḥisāb*, написанной в XIII веке.]

Нулевой случай. Перед тем как завершить наше рассмотрение ранних работ по сочетаниям, следует вспомнить небольшой, но важный шаг, сделанный Джоном Уоллисом (John Wallis) на с. 110 книги *Discourse of Combinations* (1685), где он, в частности, рассматривает сочетание 0 объектов из m элементов: "очевидно, что, когда мы *выбираем Ничто*, то мы тем самым *оставляем Все*, а это можно сделать только одним способом, сколько бы объектов у нас ни было". Кроме того, на с. 113, он утверждает, что $\binom{0}{0} = 1$: "(в этом случае выбрать все или оставить все означает одно и то же)".

Однако, приводя таблицу факториалов $n!$ для $n \le 24$, он не зашел настолько далеко, чтобы указать, что $0! = 1$ или что имеется ровно одна перестановка пустого множества.

Работа Нараяны. Замечательная монография *Gaṇita Kaumudī* ("Наслаждение лотосом вычислений"), написанная Нараяной Пандита (Nārāyaṇa Paṇḍita) в 1356 году, не так давно стала известной за пределами Индии благодаря переводу на английский язык Парманандом Сингхом (Parmanand Singh) [*Gaṇita Bhāratī* **20** (1998), 25–82; **21** (1999), 10–73; **22** (2000), 19–85; **23** (2001), 18–82; **24** (2002), 35–98]; см. также диссертацию Таканори Кусуба (Takanori Kusuba) из Brown University (1993). Глава 13 книги Нараяны, озаглавленная *Aṅka Pāśa* ("Соединение чисел"), посвящена комбинаторной генерации. Хотя 97 "сутр" этой главы довольно загадочны, в них содержится исчерпывающая теория, опередившая мировую науку в этой области на несколько столетий.

Например, в сутрах 49–55a Нараяна описывает генерацию перестановок и приводит алгоритм для перечисления всех перестановок в уменьшающемся солексном порядке, приводя при этом способ вычисления ранга данной перестановки и получения перестановки по ее рангу. Эти алгоритмы появились более чем на сто лет раньше в известной работе *Saṅgītaratnākara* ("Добыча жемчуга музыки"), принадлежащей перу Сарнгадевы (Śārṅgadeva), в §1.4.60–71. Таким образом, Сарнгадева, по сути, открыл факториальное представление положительных целых чисел. В сутрах 57–60 Нараяна развил алгоритмы Сарнгадевы так, чтобы было можно легко выполнять перестановки мультимножеств общего вида; например, он перечисляет перестановки мультимножества $\{1,1,2,4\}$ в уменьшающемся солексном порядке:

$$1124, 1214, 2114, 1142, 1412, 4112, 1241, 2141, 1421, 4121, 2411, 4211.$$

В сутрах 88–92 Нараяна занимается систематической генерацией сочетаний. Помимо иллюстрации сочетаний элементов множества $\{1,\ldots,8\}$ по три, а именно

$$(678,\ 578,\ 478,\ \ldots,\ 134,\ 124,\ 123),$$

он также рассматривает представление этих сочетаний в виде битовых строк в обратном порядке (*возрастающий* солексный порядок), тем самым развивая метод Бхаттотпалы (Bhaṭṭotpala), разработанный в X веке:

$$(11100000,\ 11010000,\ 10110000,\ \ldots,\ 00010011,\ 00001011,\ 00000111).$$

Он почти — но не до конца — открыл теорему 7.2.1.3L.

Таким образом, можно вполне законно рассматривать Нараяну Пандита как основателя науки о комбинаторной генерации — несмотря на то что, как и в случае множества других пионерских работ, опередивших свое время, его работы не были хорошо известны даже в его собственной стране.

Перестановочная поэзия. Обратимся теперь к одному занимательному вопросу, привлекавшему внимание ряда видных математиков XVII века; это прольет свет на состояние комбинаторных знаний в Европе того времени. Иезуитский священник Бернард Баухуис (Bernard Bauhuis) сочинил однострочную хвалу Деве Марии в виде латинского гекзаметра:

$$\text{Tot tibi sunt dotes, Virgo, quot sidera c?lo.} \tag{19}$$

["У тебя столько добродетелей, Дево, сколько звезд на небе"; см. его *Epigrammatum Libri V* (Cologne: 1615), 49.] Его стих побудил Эрикуса Путеануса (Erycius Puteanus), профессора университета в Лувене (University of Louvain), написать книгу *Pietatis Thaumata* (Antwerp: 1617), в которой он представил 1022 перестановки слов Баухуиса. Например, у Путеануса имеются следующие варианты.

107	Tot dotes tibi, quot c?lo sunt sidera, Virgo.	
270	Dotes tot, c?lo sunt sidera quot, tibi Virgo.	
329	Dotes, c?lo sunt quot sidera, Virgo tibi tot.	
384	Sidera quot c?lo, tot sunt Virgo tibi dotes.	(20)
725	Quot c?lo sunt sidera, tot Virgo tibi dotes.	
949	Sunt dotes Virgo, quot sidera, tot tibi c?lo.	
1022	Sunt c?lo tot Virgo tibi, quot sidera, dotes.	

Он остановился на 1022 перестановках, поскольку 1022 — это количество видимых звезд в знаменитом каталоге Птолемея (Ptolemy).

Идея такой перестановки слов была хорошо известна в то время; эту игру слов в своей книге *Poetices Libri Septem* (Lyon: 1561), Book 2, Chapter 30, Юлиус Скалигер (Julius Scaliger) назвал "стихами-хамелеонами" (Proteus verses). Латинский язык приспособлен для перестановок наподобие (20), поскольку окончания латинских слов зачастую выражают грамматическое значение существительного, делая относительный порядок слов существенно менее важным для смысла предложения, чем, например, в английском языке. Путеанус указал, однако, что он специально избегал неподходящих перестановок, таких как, например

$$\text{Sidera tot c?lo, Virgo, quot sunt tibi dotes,} \tag{21}$$

поскольку они указывают не нижнюю, а *верхнюю* границу количества достоинств Девы Марии. [См. с. 12 и 103 книги Путеануса.]

Конечно, всего имеется $8! = 40\,320$ способов перестановки слов в строке (19). Но цель состоит не в том, чтобы получить их все — большинство из них "нечитаемо". Каждый же из 1022 стихов Путеануса соответствует строгим правилам классического "гекзаметра", которым следовали греческие и латинские поэты со времен Гомера и Вергилия.

i) Каждое слово состоит из длинных (—) или коротких (⏑) слогов.
ii) Слоги каждой строки относятся к одному из 32 шаблонов:

$$\left\{\begin{matrix}—⏑⏑\\——\end{matrix}\right\}\left\{\begin{matrix}—⏑⏑\\——\end{matrix}\right\}\left\{\begin{matrix}—⏑⏑\\——\end{matrix}\right\}\left\{\begin{matrix}—⏑⏑\\——\end{matrix}\right\}—⏑⏑\left\{\begin{matrix}—⏑\\——\end{matrix}\right\}. \tag{22}$$

Другими словами, имеется шесть метрических стоп, причем каждая из первых четырех может быть либо дактилем, либо спондеем в терминологии (5); пятая стопа обязана быть дактилем, а последняя — хореем или спондеем.

Правила для определения длинных и коротких слогов в латинской поэзии в общем случае достаточно запутанны, но восемь слов Баухуиса соответствуют следующим шаблонам:

$$\text{tot} = —,\ \text{tibi} = \left\{\begin{matrix}⏑⏑\\⏑—\end{matrix}\right\},\ \text{sunt} = —,\ \text{dotes} = ——,$$
$$\text{Virgo} = \left\{\begin{matrix}—⏑\\——\end{matrix}\right\},\ \text{quot} = —,\ \text{sidera} = —⏑⏑,\ \text{c?lo} = ——. \tag{23}$$

Обратите внимание, что у поэтов есть по два варианта при использовании слов 'tibi' и 'Virgo'. Таким образом, например, строка (19) укладывается в шаблон гекзаметра

$$\overset{—}{\text{Tot}}\ \overset{⏑}{\text{ti-}}\ \overset{⏑}{\text{bi}}\quad \overset{—}{\text{sunt}}\ \overset{—}{\text{do-}}\quad \overset{—}{\text{tes,}}\ \overset{—}{\text{Vir-}}\quad \overset{—}{\text{go,}}\ \overset{—}{\text{quot}}\quad \overset{—}{\text{si-}}\ \overset{⏑}{\text{de-}}\ \overset{⏑}{\text{ra}}\quad \overset{—}{\text{c?-}}\ \overset{—}{\text{lo.}} \tag{24}$$

(Дактиль, спондей, спондей, спондей, дактиль, спондей; "там-тата там-там там-там там-там там-тата там-там". Запятые представляют небольшие паузы при чтении стиха, именуемые "цезурами" (c?suras); здесь они не рассматриваются, хотя Путеанус аккуратно вставил их в каждую из 1022 перестановок.)

Возникает естественный вопрос: если переставить слова Баухуиса случайным образом, то каковы шансы, что получится гекзаметр? Другими словами, сколько

перестановок подчиняются правилам (i) и (ii) с учетом шаблонов из (23)? Г. В. Лейбниц (G. W. Leibniz) рассматривал этот вопрос среди прочих в своей работе *Dissertatio de Arte Combinatoria* (1666), которая была опубликована, когда он претендовал на место в Университете Лейпцига. В то время Лейбницу было всего 19 лет, и по большей части он был самоучкой, так что его понимание комбинаторики было весьма ограниченным; например, он считал, что имеется 600 перестановок множества {до, до, ре, ми, фа, соль} и 480 — множества {до, до, ре, ре, ми, фа}, и даже утверждал, что (22) представляет 76 вариантов, а не 32. [См. §5 и 8 в его задаче 6.]

Однако Лейбниц понимал, что следовало бы разработать общие методы для подсчета всех "подходящих" перестановок в ситуациях, когда многие подстановки таковыми не являются. Он рассмотрел несколько примеров стихов-хамелеонов, корректно проведя подсчеты для более простых и наделав массу ошибок там, где слова оказывались сложнее. Хотя Лейбниц и упомянул работу Путеануса, он не пытался подсчитать количество гекзаметров среди перестановок строки (19).

Существенно более успешный подход был предложен несколько лет спустя Жаном Престе (Jean Prestet) в его *Élémens des Mathématiques* (Paris: 1675), 342–438. Из рассмотрения вопроса Престе вытекало наличие ровно 2196 перестановок слов хвалы Баухуиса, являющихся гекзаметрами. Однако вскоре он обнаружил, что упустил некоторое количество случаев, в частности случаи 270, 384 и 725 из (20). Поэтому он полностью переписал данный материал при переиздании книги *Nouveaux Élémens des Mathématiques* в 1689 году. На с. 127–133 новой книги Престе показывает, что правильное количество перестановок-гекзаметров равно 3276, что почти на 50% превышает указанное им ранее число.

Тем временем Джон Уоллис (John Wallis) рассмотрел данную задачу в работе *Discourse of Combinations* (London: 1685), 118–119, опубликованной в качестве дополнения к его книге *Treatise of Algebra*. Объяснив, почему он считает, что корректное количество равно 3096, Уоллис допустил предположение, что он мог упустить некоторые варианты и/или сосчитать другие более чем по одному разу; "однако в настоящее время я не заметил за собой ни того, ни другого".

Анонимный критик работы Уоллиса указал, что верное количество перестановок составляет 2580, но не дал этому никакого доказательства [*Acta Eruditorum* **5** (1686), 289]. Этим критиком почти наверняка был Лейбниц, хотя никакого ключа к пониманию числа 2580 среди его многочисленных неопубликованных заметок найдено не было.

Наконец на сцене появляется Якоб Бернулли (James Bernoulli), который в инаугурационной лекции при вступлении в должность декана факультета философии базельского университета (University of Basel) в 1692 году упомянул данную задачу и сообщил, что аккуратный анализ дал корректный ответ, равный 3312(!) перестановкам. Доказательство Бернулли было опубликовано посмертно в первом издании *Ars Conjectandi* (1713), 79–81. [Бернулли не собирался публиковать данные страницы в этой ныне знаменитой книге; корректор, нашедший их среди заметок, решил включить их "для удовлетворения любопытства". См. *Die Werke von Jakob Bernoulli* **3** (Basel: Birkhäuser, 1975), 78, 98–106, 108, 154–155.]

Так кто же был прав? Сколько же гекзаметров среди перестановок — 2196, 3276, 3096, 2580 или 3312? Этот вопрос был заново рассмотрен В. А. Уитвортом (W. A. Whitworth) и В. Э. Хартли (W. E. Hartley) в *The Mathematical Gazette* **2**

(1902), 227–228, где каждый из них представил элегантные доказательства и сделал выводы, что ни одно из представленных чисел не является корректным ответом. Их общим решением было 2880, и это был первый случай, когда два математика независимо получили одно и то же решение данной задачи.

Но из упр. 21 и 22 вы узнаете истину: прав был только Бернулли, все остальные ошибались. Кроме того, изучение трехстраничного вывода Бернулли указывает, что он был успешен в основном потому, что последний строго придерживался метода, который сейчас называется *методом с возвратом* (*backtrack method*). Мы тщательно изучим этот метод в разделе 7.2.2, из которого также узнаем, что рассматриваемая здесь задача легко решается как частный случай *задачи точного покрытия* (*exact cover problem*).

Даже мудрейшие и осторожнейшие люди часто страдают от того, что логики называют неполным перечислением возможностей.

— ЯКОБ БЕРНУЛЛИ (JAMES BERNOULLI) (1692)

Разбиения множеств. Похоже, впервые разбиения множеств изучались в Японии, где в 1500-е годы среди высших классов стала популярной игра *генджи-ко* (*genji-ko*). Ведущий должен скрытно выбрать пять пакетов фимиама, причем некоторые из них могут быть одинаковыми, и воскурить их по одному. Игроки должны попытаться определить, какие запахи были одинаковы, а какие различны — словом, угадать, какое из $\varpi_5 = 52$ разбиений множества $\{1, 2, 3, 4, 5\}$ выбрано ведущим.

Рис. 67. Диаграммы, использованные для представления разбиений множества в XVI веке в Японии. [С копии из коллекции Тамаки Яно (Tamaki Yano) из Университета Сайтамы.]

Вскоре стало привычным изображать 52 возможных варианта с помощью диаграмм, подобных приведенным на рис. 67. Например, верхняя диаграмма при чтении справа налево означает, что одинаковы два первых запаха и три последних, т. е. это разбиение $12|345$. Две другие диаграммы представляют соответственно разбиения $124|35$ и $1|24|35$. Чтобы упростить запоминание, каждый из 52 шаблонов получил свое название благодаря знаменитой книге XI века г-жи Мурасаки (Murasaki) *Рассказ о генджи*, в соответствии с приведенной ниже последовательностью [*Encyclopedia Japonicæ* (Tokyo: Sanseido, 1910), 1299].

(25)

(Как и во множестве других примеров, варианты здесь перечислены без какого бы то ни было логичного порядка.)

Привлекательная природа шаблонов генджи-ко привела к тому, что многие семьи использовали их в качестве геральдических символов. Например, далее приведены стилизованные варианты (25), найденные в стандартных каталогах узоров кимоно начала XX века.

[См. Fumie Adachi, *Japanese Design Motifs* (New York: Dover, 1972), 150–153.]

В начале XVIII века Такаказу Секи (Takakazu Seki) со своими учениками приступил к изучению количества разбиений множества ϖ_n для произвольного n, вдохновленный известным результатом $\varpi_5 = 52$. Ёшисуке Мацунага (Yoshisuke Matsunaga) нашел формулы для количества разбиений множества, когда имеется k_j подмножеств размером n_j для $1 \le j \le t$ и $k_1 n_1 + \cdots + k_t n_t = n$ (см. ответ к упр. 1.2.5–21). Он также открыл базовое рекуррентное соотношение 7.2.1.5–(14), а именно

$$\varpi_{n+1} \;=\; \binom{n}{0}\varpi_n + \binom{n}{1}\varpi_{n-1} + \binom{n}{2}\varpi_{n-2} + \cdots + \binom{n}{n}\varpi_0, \tag{26}$$

с помощью которого легко вычислить значения ϖ_n.

Открытия Мацунаги оставались неопубликованными до выхода в 1769 году книги Ёриюки Аримы (Yoriyuki Arima) *Shūki Sanpō*. Задача 56 из этой книги состоит в решении уравнения "$\varpi_n = 678\,570$" относительно n; детальное решение Аримы (с надлежащими благодарностями Мацунаке) дает ответ $n = 11$.

Вскоре после этого Масанобу Сака (Masanobu Saka) изучал количество $\left\{{n \atop k}\right\}$ способов разбиения множества из n элементов на k подмножеств в своей работе *Sanpō-Gakkai* (1782). Он открыл рекуррентную формулу

$$\left\{{n+1 \atop k}\right\} \;=\; k\left\{{n \atop k}\right\} + \left\{{n \atop k-1}\right\} \tag{27}$$

и протабулировал результаты для $n \le 11$. Джеймс Стирлинг (James Stirling) в своей работе *Methodus Differentialis* (1730) открыл числа $\left\{{n \atop k}\right\}$ в чисто алгебраическом контексте; таким образом, Сака был первым, кто осознал их комбинаторную важность.

Интересный алгоритм для перечисления всех разбиений множества был впоследствии открыт Тошиаки Хондой (Toshiaki Honda) (см. упр. 24). Более детальную информацию о генджи-ко и ее связи с историей математики можно найти в статьях Тамаки Яно (Tamaki Yano) *Sugaku Seminar* **34**, 11 (Nov. 1995), 58–61; **34**, 12 (Dec. 1995), 56–60.

Разбиения множеств оставались практически неизвестными в Европе до гораздо более позднего времени, за исключением трех не связанных между собой случаев. Во-первых, в 1589 году Георг и/или Ричард Паттенхам (George and/or Richard Puttenham) опубликовал книгу *The Arte of English Poesie*, на с. 70–72 которой

содержатся диаграммы, похожие на диаграммы генджи-ко. Например, приведенные ниже семь диаграмм

(28)

использованы для иллюстрации возможных схем рифм в пятистрочных стихах, "где некоторые из них грубее и неприятнее для уха, чем иные." Но этот визуально привлекательный список был неполным (см. упр. 25).

Во-вторых, неопубликованная рукопись Г. В. Лейбница (G. W. Leibniz) конца XVII века свидетельствует, что он пытался вычислить количество способов разбиения множества $\{1,\ldots,n\}$ на три или четыре подмножества, но практически безуспешно. Он подсчитал $\left\{{n \atop 2}\right\}$ очень громоздким методом, который не позволил ему заметить, что $\left\{{n \atop 2}\right\} = 2^{n-1}-1$. Лейбниц пытался вычислить $\left\{{n \atop 3}\right\}$ и $\left\{{n \atop 4}\right\}$ только для $n \le 5$ и сделал несколько числовых ошибок, приведших его к неверному ответу. [См. E. Knobloch, *Studia Leibnitiana Supplementa* **11** (1973), 229–233; **16** (1976), 316–321.]

Третий европейский случай разбиения множества носит совершенно иной характер. Джон Уоллис (John Wallis) посвятил главу 3 своей книги *Discourse of Combinations* (1685) вопросу о "кратных частях", истинных делителях чисел, в частности он изучал множество всех способов разложения данного числа на множители. Этот вопрос эквивалентен задаче о разбиениях "мультимножества"; например, разложение p^3q^2r, по сути, то же, что и разбиения мультимножества $\{p,p,p,q,q,r\}$, где p, q и r — простые числа. Уоллис разработал превосходный алгоритм для перечисления всех разложений данного числа n, по сути, предвосхитив алгоритм 7.2.1.5M (см. упр. 28). Однако он не исследовал важные частные случаи, когда n представляет собой степень простого числа (эквивалентно разбиениям целого числа) или когда n не содержит квадратов (эквивалентно разбиениям множества). Таким образом, хотя Уоллис и смог решить более общую задачу, ее сложность парадоксальным образом увела его с пути открытия разбиений чисел, чисел Белла, количества подмножеств Стирлинга и разработки простых алгоритмов генерации разбиений целых чисел или разбиений множеств.

Разбиения целых чисел. Разбиения целых чисел выходили на сцену комбинаторики еще медленнее. Выше мы видели, что епископ Вибольд (Wibold) (ок. 965 г.) знал о разбиениях целых чисел на три части ≤ 6. О том же знал и Галилей (Galileo), написавший о них небольшую работу (ок. 1627 г.) и изучавший частоты выпадения очков при бросании трех игральных костей. ["Sopra le scoperte de i dadi", в своей работе *Opere*, Volume 8, 591–594; Галилей перечислил разбиения в уменьшающемся лексикографическом порядке.] Томас Харриот (Thomas Harriot) в неопубликованной работе несколькими годами ранее рассматривал бросания до шести игральных костей [см. J. Stedall, *Historia Math.* **34** (2007), 398].

Мерсенн (Mersenne) перечислил разбиения 9 на любое количество частей на с. 130 своей книги *Traitez de la Voix et des Chants* (1636). Для каждого разбиения $9 = a_1 + \cdots + a_k$ он также вычислил полиномиальный коэффициент $9!/(a_1!\ldots a_k!)$; как мы видели ранее, его интересовало количество различных мелодий, например он знал, что имеется $9!/(3!\,3!\,3!) = 1680$ мелодий из девяти нот $\{a,a,a,b,b,b,c,c,c\}$.

Однако он не упомянул случаи $8+1$ и $3+2+1+1+1+1$, вероятно, потому, что не перечислял возможности каким-либо систематическим путем.

Лейбниц (Leibniz) рассматривал разбиения на две части в задаче 3 в работе *Dissertatio de Arte Combinatoria* (1666), а его неопубликованные заметки указывают, что впоследствии он потратил немало времени, пытаясь перечислить разбиения, состоящие из трех и более слагаемых. Он называл их (разумеется, на латыни) "discerptions" или реже "divulsions", иногда разделами ("sections"), рассеянием ("dispersions") или даже разбиениями ("partitions"). Они интересовали его, в первую очередь, из-за связи с мономиальными симметричными функциями $\sum x_{i_1}^{a_1} x_{i_2}^{a_2} \ldots$. Однако многие попытки Лейбница всегда приводили к неудаче, за исключением случая трех слагаемых, когда он почти (но не совсем) открыл формулу для $\left|{n \atop 3}\right|$ (см. упр. 7.2.1.4–31). Например, он аккуратно подсчитал только 21 разбиение числа 8, пропустив случай $2+2+2+1+1$, а для $p(9)$ получил значение 26, потеряв $3+2+2+2$, $3+2+2+1+1$, $2+2+2+1+1+1$ и $2+2+1+1+1+1+1$, несмотря на то что пытался перечислять разбиения систематически в убывающем лексикографическом порядке. [См. E. Knobloch, *Studia Leibnitiana Supplementa* **11** (1973), 91–258; **16** (1976), 255–337; *Historia Mathematica* **1** (1974), 409–430.]

Абрахам де Муавр (Abraham de Moivre) был первым, реально достигшим успеха при изучении разбиений в своей статье "Метод возведения бесконечного многочлена в любую заданную степень или извлечение из него заданного корня" [*Philosophical Transactions* **19** (1697), 619–625 и рис. 5]. Он доказал, что коэффициент при z^{m+n} в $(az+bz^2+cz^3+\cdots)^m$ имеет по одному члену для каждого разбиения n; например, коэффициент при z^{m+6} равен

$$\begin{aligned}&\tbinom{m}{6}a^{m-6}b^6+5\tbinom{m}{5}a^{m-5}b^4c+4\tbinom{m}{4}a^{m-4}b^3d+6\tbinom{m}{4}a^{m-4}b^2c^2\\&\quad+3\tbinom{m}{3}a^{m-3}b^2e+6\tbinom{m}{3}a^{m-3}bcd+2\tbinom{m}{2}a^{m-2}bf+\tbinom{m}{3}a^{m-3}c^3\\&\qquad+2\tbinom{m}{2}a^{m-2}ce+\tbinom{m}{2}a^{m-2}d^2+\tbinom{m}{1}a^{m-1}g.\end{aligned}\tag{29}$$

Если мы положим $a=1$, то член со степенями $b^ic^jd^ke^l\ldots$ соответствует разбиению с i единицами, j двойками, k тройками, l четверками и т. д. Так, например, при $n=6$ он, по сути, получил разбиения в порядке

$$111111, 11112, 1113, 1122, 114, 123, 15, 222, 24, 33, 6. \tag{30}$$

Де Муавр пояснил, как перечислить разбиения рекурсивно (хотя и другим языком, связанным с его собственными обозначениями): для $k=1, 2, \ldots, n$ начинаем с k и добавляем (ранее перечисленные) разбиения $n-k$, наименьшая часть которых $\geq k$.

[Мое решение] упорядочено перед публикацией не столько для красоты, сколько в процессе некоторых размышлений, не достойных внимания любителей истины.

— АБРАХАМ ДЕ МУАВР (ABRAHAM DE MOIVRE) (1717)

П. Р. де Монмор (P. R. de Montmort) протабулировал все разбиения чисел ≤ 9 на количество частей ≤ 6 в своем труде *Essay d'Analyse sur les Jeux de Hazard* (1708) в связи с задачей об игральных костях. Его разбиения перечислены в ином, чем в (30), порядке, например

$$111111, 21111, 2211, 222, 3111, 321, 33, 411, 42, 51, 6. \tag{31}$$

Вероятно, он не был знаком с более ранней работой де Муавра.

До сих пор практически никто из рассмотренных нами авторов не описывал использовавшуюся им процедуру генерации комбинаторных шаблонов. Мы можем только догадываться об их методах (или их отсутствии), изучая опубликованные ими списки. Более того, в редких случаях, таких как статья де Муавра, в которой *имеется* явно описанный метод, автор полагает, что существуют списки всех шаблонов для случаев 1, 2, ..., $n-1$, перед тем как приступить к составлению списка для n. За исключением Кедары (Kedāra) и Нараяны (Nārāyaṇa), ни один из авторов не привел метода генерации шаблонов "на лету", когда очередной шаблон получается из предшествующего непосредственно, без просмотра вспомогательных таблиц. Естественно, что современные программисты предпочитают более прямые методы, требующие меньшего количества памяти.

Р. И. Бошкович (R. J. Boscovich) опубликовал первый прямой алгоритм для генерации разбиений в *Giornale de' Letterati* (Rome, 1747), с. 393–404, вместе с двумя таблицами (с. 404). Его метод, который для $n = 6$ дает вывод

$$111111, 11112, 1122, 222, 1113, 123, 33, 114, 24, 15, 6, \tag{32}$$

генерирует разбиения в точности в обратном порядке по отношению к порядку, получаемому при работе алгоритма 7.2.1.4P. Метод Бошковича, по сути, описан в разделе 7.2.1.4, с тем отличием, что обратный порядок приводит к несколько более простому и быстрому алгоритму, чем порядок, выбранный Бошковичем.

Продолжение своей работы он опубликовал в *Giornale de' Letterati* (Rome, 1748), с. 12–27 и 84–99, развив свой алгоритм в двух направлениях. Во-первых, он рассмотрел генерацию только тех разбиений, части которых принадлежат заданному множеству S, с тем чтобы возводить в m-ю степень символьные полиномы с разреженными коэффициентами. (Он утверждал, что наибольший общий делитель всех элементов S должен быть равен 1; в действительности, однако, его метод может не работать, если $1 \notin S$.) Во-вторых, он разработал алгоритм для генерации разбиений n на m частей для заданных m и n. И опять ему не повезло: для решения этой задачи впоследствии был разработан более удачный алгоритм 7.2.1.4H, что лишило Бошковича шансов на славу.

Ода Гинденбургу. Изобретателем алгоритма 7.2.1.4H был Карл Фридрих Гинденбург (Carl Friedrich Hindenburg), который "переоткрыл" алгоритм Нараяны 7.2.1.2L, представляющий собой способ генерации перестановок мультимножества. К сожалению, этот маленький успех привел его к вере в то, что он совершил революционный прорыв в математике (хотя он и снизошел до упоминания других исследователей, в частности де Муавра, Эйлера и Ламберта, которые близко подошли к аналогичным открытиям).

Гинденбург был в числе основателей первых математических журналов Германии (издававшихся в 1786–1789 и 1794–1800 годах) и автором ряда длинных статей в них. Он неоднократно занимал должность декана в Университете Лейпцига, а в 1792 году был его ректором. Если бы он имел немного больше способностей в области математики по сравнению с тем, чем был наделен на самом деле, германская математика процветала бы в Лейпциге, а не в Берлине и Геттингене.

Однако первая же его математическая работа, *Beschreibung einer ganz neuen Art, nach einem bekannten Gesetze fortgehende Zahlen durch Abzählen oder Abmessen bequem und sicher zu finden* (Leipzig: 1776), достаточно ярко показала, чего следует

ожидать: его “ganz neue” (совершенно новая) идея заключалась в придании комбинаторного смысла цифрам десятичной записи чисел. В это невозможно поверить, но Гинденбург завершил свою монографию большими таблицами — таблицей чисел от 0000 до 9999, после которой следовали таблицы четных и нечетных чисел по отдельности (!).

Гинденбург публиковал письма людей, хваливших его работу, и приглашал их писать статьи в журналы. В 1796 году он редактировал *Sammlung combinatorisch-analytischer Abhandlungen*, где в подзаголовке было указано, что теорема о полиномах де Муавра является “наиболее важной во всем математическом анализе”. Около десятка людей объединили свои усилия и организовали то, что впоследствии стало известно как “школа комбинаторики Гинденбурга”, и опубликовали тысячи страниц, заполненных эзотерическими символизмами, поражавшими воображение множества людей, не имеющих отношения к математике.

С точки зрения информатики работа этой школы не может считаться совершенно тривиальной. Например, лучший студент Гинденбурга Г. А. Роте (H. A. Rothe) заметил, что имеется простой способ перейти от последовательности кода Морзе к следующей за ней в лексикографическом порядке или к предшествующей. Другой ученик, И. К. Буркхардт (J. K. Burckhardt), заметил, что последовательности кодов Морзе длиной n могут быть легко сгенерированы, если сначала рассмотреть коды без тире, затем — с одним тире, далее — с двумя и т. д. Их целью была не табуляция стихотворных форм с n тактами, как это делалось в Индии, а перечисление членов континуантов $K(x_1, x_2, \ldots, x_n)$, формула 4.5.3–(4). [См. *Archiv der reinen und angewandten Mathematik* **1** (1794), 154–195.] Кроме того, на с. 53 упоминавшегося выше *Sammlung* Г. С. Клюгель (G. S. Klügel) привел способ перечисления всех перестановок, который впоследствии получил название алгоритма Орда-Смита; см. формулы (23)–(26) в разделе 7.2.1.2.

Гинденбург верил, что его методы заслуживают достойного места в учебных курсах алгебры, геометрии и вычислений. Однако и он, и его ученики ограничивались в своих работах составлением списков комбинаторных объектов. Закопавшись в своих формулах и формализме, они редко открывали что-то действительно интересное с точки зрения математики. Ойген Нетто (Eugen Netto) замечательно охарактеризовал их работу в *Geschichte der Mathematik* **4** (1908), 201–219, М. Кантора (M. Cantor): “Какое-то время они контролировали германский рынок, но большинство из откопанного ими тут же засыпало песком забвения”.

Грустным итогом их исследований стало то, что в результате комбинаторика в целом стала восприниматься как лженаука. Геста Миттаг-Леффлер (Gösta Mittag-Leffler), собравший великолепную библиотеку математической литературы примерно через сто лет после смерти Гинденбурга, решил поместить все такие работы на отдельной полке, помеченной “Декаденс”. Эта категория есть в шведском Институте Миттаг-Леффлера и сегодня, несмотря на то что данный институт привлекает со всего мира математиков, работающих в области комбинаторики, труды которых можно назвать как угодно, но никак не упадническими.

Во всем есть хорошая сторона, и мы можем отметить как минимум одну хорошую книгу, появившуюся в результате всей этой деятельности. Это *Die combinatorische Analysis* (Vienna: 1826) Андреаса фон Эттингсхаузена (Andreas von Ettingshausen), заслуживающая внимания как первая книга, в которой методы генерации

комбинаторных объектов рассматриваются ясно и последовательно. Эттингсхаузен рассмотрел общие принципы лексикографической генерации в §8 и применил их для построения метода перечисления всех перестановок (§11), сочетаний (§30) и разбиений (§41–44).

А где же деревья? Итак, мы уже видели, что на протяжении всей истории человечества неоднократно составлялись списки кортежей, перестановок, сочетаний и разбиений. Таким образом, мы рассмотрели эволюцию тем из разделов 7.2.1.1–7.2.1.5, но наш рассказ будет неполным, если мы не проследим происхождение генерации деревьев — тему раздела 7.2.1.6.

Однако исторические сведения по этой теме до прихода компьютеров практически отсутствуют, если не считать нескольких статей, опубликованных в XIX веке Артуром Кейли (Arthur Cayley). Главная работа Кейли по деревьям была опубликована в 1875 году и перепечатана в его *Collected Mathematical Papers*, Volume 4, на с. 427–460; она содержала большую иллюстрацию со всеми свободными деревьями не более чем с девятью непомеченными вершинами. Ранее в этой статье он также привел девять *ориентированных* деревьев с пятью вершинами. Методы, использовавшиеся Кейли для получения этих списков, были весьма сложными, совершенно отличными от алгоритма 7.2.1.6O и упр. 7.2.1.6–90. Все свободные деревья не более чем с десятью вершинами были перечислены много лет спустя Ф. Харари (F. Harary) и Д. Принсем (G. Prins) [*Acta Math.* **101** (1958), 158–162], которые дошли до $n = 12$ для случаев свободных деревьев, не имеющих узлов степени 2 и не обладающих симметрией.

Деревья, наиболее любимые кибернетиками — бинарные деревья, или эквивалентные упорядоченные леса, или вложенные скобки, — как ни странно, в литературе отсутствуют. В разделе 2.3.4.5 мы видели, что многие математики в XVIII–XIX веках изучали количество бинарных деревьев; мы также знаем, что числа Каталана C_n подсчитывают десятки разных комбинаторных объектов. Однако до 1950 года, похоже, никто не публиковал реальный *список* из $C_4 = 14$ объектов порядка 4 *любого* вида, и еще меньше авторов публиковали список из $C_5 = 42$ объектов порядка 5. (За косвенным исключением: 42 диаграммы генджи-ко из (25), в которых нет пересекающихся линий, эквивалентны пятиузловым бинарным деревьям и лесам. Однако этот факт не изучался до XX века.)

Имеется несколько отдельных примеров, когда в былые времена авторы составляли списки из $C_3 = 5$ объектов, связанных с числами Каталана. Здесь также первым был Кейли (Cayley), который привел бинарные деревья с тремя внутренними узлами и четырьмя листьями в *Philosophical Magazine* **18** (1859), 374–378.

(33)

(В той же статье проиллюстрированы другие разновидности деревьев, эквивалентные так называемым слабым упорядочениям (weak orderings).) Затем в 1901 году Э. Нетто (E. Netto) перечислил пять способов расстановки скобок в выражении '$a + b + c + d$':

$$(a+b)+(c+d),\ [(a+b)+c]+d,\ [a+(b+c)]+d,\ a+[(b+c)+d],\ a+[b+(c+d)]. \tag{34}$$

[*Lehrbuch der Combinatorik*, §122.] Пять перестановок множества $\{+1, +1, +1, -1, -1, -1\}$, частичные суммы которых неотрицательны, перечислены Палом Эрдешем (Paul Erdös) и Ирвингом Каплански (Irving Kaplansky) [*Scripta Math.* **12** (1946), 73–75]:

$$1{+}1{+}1{-}1{-}1{-}1,\quad 1{+}1{-}1{+}1{-}1{-}1,\quad 1{+}1{-}1{-}1{+}1{-}1,\quad 1{-}1{+}1{+}1{-}1{-}1,\quad 1{-}1{+}1{-}1{+}1{-}1. \tag{35}$$

Несмотря на то что в приведенных примерах используется только пять объектов, мы видим, что упорядочение в (33) и (34) выполнено "как получится"; и только упорядочение (35), отвечающее алгоритму 7.2.1.6P, систематическое и лексикографическое.

Следует также вкратце отметить работу Вальтера фон Дика (Walther von Dyck), поскольку во многих современных статьях о строках вложенных скобок используется термин "Слова Дика" ("Dyck words"). Дик был педагогом, помимо прочего известным как один из основателей Немецкого музея в Мюнхене (Deutsches Museum in Munich). Он написал две пионерские статьи по теории свободных групп [*Math. Annalen* **20** (1882), 1–44; **22** (1883), 70–108]. Так называемые слова Дика имеют очень слабое отношение к его реальным исследованиям. Он изучал слова на алфавите $\{x_1, x_1^{-1}, \ldots, x_k, x_k^{-1}\}$, которые приводятся к пустой строке после многократного удаления пар соседних букв вида $x_i x_i^{-1}$ или $x_i^{-1} x_i$; связь со скобками и деревьями возникает, только если ограничиться первым случаем, $x_i x_i^{-1}$.

Итак, можно заключить, что, несмотря на огромный рост интереса к бинарным деревьям и их "родственникам" после 1950 года, такие деревья — единственный предмет нашего рассмотрения, исторические корни которого крайне неглубоки.

После 1950 года. Разумеется, выход на сцену электронных вычислительных машин резко изменил ситуацию. Первой ориентированной на применение компьютеров публикацией о методах комбинаторной генерации стала статья C. B. Tompkins, "Machine attacks on problems whose variables are permutations" [*Proc. Symp. Applied Math.* **6** (1956), 202–205]. За ней не замедлили последовать тысячи других.

Ряд статей Д. Г. Лемера (D. H. Lehmer), в частности его "Teaching combinatorial tricks to a computer" ("Обучение компьютера комбинаторным трюкам") в *Proc. Symp. Applied Math.* **10** (1960), 179–193, оказали исключительно большое влияние на исследования того времени. [См. также *Proc. 1957 Canadian Math. Congress* (1959), 160–173; *Proc. IBM Scientific Computing Symposium on Combinatorial Problems* (1964), 23–30, и главу 1 в *Applied Combinatorial Mathematics*, ed. by E. F. Beckenbach (Wiley, 1964), 5–31.] Лемер оказался важным связующим звеном с предыдущими поколениями. Так, например, записи в Станфордской библиотеке свидетельствуют о том, что в январе 1932 года он изучал *Lehrbuch der Combinatorik* Э. Нетто (E. Netto).

Основные публикации, посвященные рассматривавшимся нами конкретным алгоритмам, упоминались в предыдущих разделах, так что повторять их здесь нет необходимости. Однако следует упомянуть три книги, которые заслуживают особого внимания, поскольку именно в них были заложены общие принципы.

- Mark B. Wells, *Elements of Combinatorial Computing* (Pergamon Press, 1971), в особенности глава 5.

- Albert Nijenhuis and Herbert S. Wilf, *Combinatorial Algorithms* (Academic Press, 1975). Второе издание книги вышло в 1978 году и содержало дополнительный материал; впоследствии Вильф написал *Combinatorial Algorithms: An Update* (Philadelphia: SIAM, 1989).
- Edward M. Reingold, Jurg Nievergelt, and Narsingh Deo, *Combinatorial Algorithms: Theory and Practice* (Prentice–Hall, 1977), в особенности материал главы 5.

Роберт Седжвик (Robert Sedgewick) опубликовал первый серьезный обзор по методам генерации перестановок в *Computing Surveys* **9** (1977), 137–164, 314. Второй вехой стал обзор Карлы Сэведж (Carla Savage), посвященный кодам Грея [*SIAM Review* **39** (1997), 605–629].

Ранее мы отмечали, что алгоритмы для генерации объектов, подсчитываемых с помощью чисел Каталана, не были созданы до тех пор, пока разработчики программного обеспечения не ощутили в них потребность. Первые опубликованные алгоритмы не цитировались в разделе 7.2.1.6, поскольку были вытеснены более эффективными методами, однако здесь самое подходящее место для беглого их упоминания. Во-первых, Г. Я. Скоинс (H. I. Scoins) привел два рекурсивных алгоритма для генерации упорядоченных деревьев в той же статье, которую мы уже цитировали, говоря о генерации *ориентированных* деревьев [*Machine Intelligence* **3** (1968), 43–60]. Его алгоритмы работают с представлением бинарных деревьев в виде битовых строк, что, по сути, эквивалентно польской префиксной записи или вложенным скобкам. Затем Марк Уэллс (Mark Wells) в разделе 5.5.4 своей упоминавшейся выше книги генерировал бинарные деревья, представляя их в виде непересекающихся разбиений множеств. Наконец Гари Кнотт (Gary Knott) [*CACM* **20** (1977), 113–115] разработал рекурсивные алгоритмы для определения ранга бинарного дерева и получения бинарного дерева по заданному рангу, представляя деревья с помощью перестановок $q_1 \ldots q_n$ из табл. 7.2.1.6–3.

Алгоритмы для генерации всех остовных деревьев заданного графа были опубликованы рядом авторов еще в 1950-е годы; толчком к этому стало изучение электрических сетей. Вот некоторые из ранних работ в этом направлении: N. Nakagawa, *IRE Trans.* **CT-5** (1958), 122–127; W. Mayeda, *IRE Trans.* **CT-6** (1959), 136–137, 394; H. Watanabe, *IRE Trans.* **CT-7** (1960), 296–302; S. Hakimi, *J. Franklin Institute* **272** (1961), 347–359.

В главах 2 и 3 книги Donald L. Kreher and Douglas R. Stinson, *Combinatorial Algorithms: Generation, Enumeration, and Search* (CRC Press, 1999), можно найти полезную информацию по всей рассматриваемой теме.

Фрэнк Раски (Frank Ruskey) подготовил к изданию книгу *Combinatorial Generation*, которая будет содержать всестороннее рассмотрение комбинаторных вопросов и исчерпывающую библиографию по данной теме. Черновики некоторых глав из этой книги можно найти в Интернете.

Упражнения

Во многих из предлагаемых упражнений от современного читателя требуется найти и/или исправить ошибки в литературе давно минувших дней. Цель этих упражнений отнюдь не в том, чтобы, злорадно ухмыляясь, показать, насколько мы в XXI веке умнее, а в том, чтобы понять, что даже пионеры в некоторой области могут оступаться. Полное понимание

того, что множество идей не так просты, как может показаться на первый взгляд сегодняшнему программисту и математику, придет к вам лишь тогда, когда вы обнаружите, как много сил вынуждены были тратить ученые мирового уровня, чтобы разобраться в этих концепциях, когда они были новинкой.

1. [*15*] Существует ли понятие "вычислений" (computing) в *I Ching*?

▶ **2.** [*M30*] (*Генетический код.*) Молекулы ДНК представляют собой строки "нуклеотидов" из 4-буквенного алфавита $\{\mathtt{T},\mathtt{C},\mathtt{A},\mathtt{G}\}$, а большинство молекул белков представляют собой строки "аминокислот" из 20-буквенного алфавита $\{\mathrm{A},\mathrm{C},\mathrm{D},\mathrm{E},\mathrm{F},\mathrm{G},\mathrm{H},\mathrm{I},\mathrm{K},\mathrm{L},\mathrm{M},\mathrm{N},\mathrm{P},\mathrm{Q},\mathrm{R},\mathrm{S},\mathrm{T},\mathrm{V},\mathrm{W},\mathrm{Y}\}$. Три последовательных нуклеотида xyz образуют базовый блок — "кодон", а цепочка $x_1y_1z_1x_2y_2z_2\ldots$ ДНК определяет белок $f(x_1,y_1,z_1)f(x_2,y_2,z_2)\ldots$, где $f(x,y,z)$ — элемент в строке z и столбце y матрицы x в массиве

$$\begin{pmatrix} \mathrm{F} & \mathrm{S} & \mathrm{Y} & \mathrm{C} \\ \mathrm{F} & \mathrm{S} & \mathrm{Y} & \mathrm{C} \\ \mathrm{L} & \mathrm{S} & - & - \\ \mathrm{L} & \mathrm{S} & - & \mathrm{W} \end{pmatrix} \quad \begin{pmatrix} \mathrm{L} & \mathrm{P} & \mathrm{H} & \mathrm{R} \\ \mathrm{L} & \mathrm{P} & \mathrm{H} & \mathrm{R} \\ \mathrm{L} & \mathrm{P} & \mathrm{Q} & \mathrm{R} \\ \mathrm{L} & \mathrm{P} & \mathrm{Q} & \mathrm{R} \end{pmatrix} \quad \begin{pmatrix} \mathrm{I} & \mathrm{T} & \mathrm{N} & \mathrm{S} \\ \mathrm{I} & \mathrm{T} & \mathrm{N} & \mathrm{S} \\ \mathrm{I} & \mathrm{T} & \mathrm{K} & \mathrm{R} \\ \mathrm{M} & \mathrm{T} & \mathrm{K} & \mathrm{R} \end{pmatrix} \quad \begin{pmatrix} \mathrm{V} & \mathrm{A} & \mathrm{D} & \mathrm{G} \\ \mathrm{V} & \mathrm{A} & \mathrm{D} & \mathrm{G} \\ \mathrm{V} & \mathrm{A} & \mathrm{E} & \mathrm{G} \\ \mathrm{V} & \mathrm{A} & \mathrm{E} & \mathrm{G} \end{pmatrix}.$$

(Здесь $(\mathtt{T},\mathtt{C},\mathtt{A},\mathtt{G}) = (1,2,3,4)$; например, $f(\mathtt{CAT})$ представляет собой элемент в строке 1 и столбце 3 матрицы 2, а именно H.) Кодирование выполняется до тех пор, пока кодон не приведет к останавливающему элементу '$-$'.

а) Покажите, что существует простой способ сопоставить каждый кодон гексаграмме из *I Ching*, причем 21 возможный выход $\{\mathrm{A},\mathrm{C},\mathrm{D},\ldots,\mathrm{W},\mathrm{Y},-\}$ соответствует 21 *последовательной* гексаграмме в упорядочении Вень-вана (1).

б) Является ли это открытие сенсационным?

3. [*20*] Какой вид имеет миллионная строфа из 30 тактов в солексном порядке, аналогичном порядку (2)? Каков ранг ◡◡◡—◡——◡◡◡◡——◡◡◡◡◡◡◡◡—◡—?

4. [*19*] Проанализируйте недочеты списка перестановок Донноло (Donnolo) в табл. 1.

5. [*16*] Где ошибка в списке перестановок пяти нот Кирхера (Kircher) в (7)?

6. [*25*] Мерсенн (Mersenne) на с. 108–110 своей книги *Traitez de la Voix et des Chants* (1636) опубликовал таблицу первых 64 факториалов. Его значение для 64! приближенно равно $\approx 2.2\times10^{89}$, в то время как оно должно быть равно $\approx 1.3\times10^{89}$. Найдите копию упомянутой книги и выясните, где он ошибся.

7. [*20*] Какие перестановки множества $\{1,2,3,4,5\}$ являются "живыми" и "мертвыми" в соответствии с правилами Секи (Seki) (8) и (9)?

▶ **8.** [*M27*] Придумайте исправление, которое нужно внести в правило (9), чтобы процедура Секи стала корректной.

9. [*15*] Выведите из (11) арабский способ записи арабских цифр $(0,1,\ldots,9)$.

▶ **10.** [*HM27*] Каково математическое ожидание количества бросков костей в игре Ludus Clericalis для получения всех возможных добродетелей?

11. [*21*] Расшифруйте вертикальную таблицу Луллия на рис. 65 справа. Какие 20 комбинаторных объектов она представляет? *Указание*: не позвольте опечаткам ввести вас в заблуждение.

12. [*M20*] Свяжите универсальный цикл Шиллингера (Schillinger) (13) с универсальным циклом Пуансо (Poinsot) из упр. 7.2.1.3–106.

13. [*21*] Что ван Шутен (van Schooten) должен был написать вместо (17)? Приведите также соответствующую таблицу для сочетаний мультимножества $\{a,a,a,b,b,c\}$.

▶ **14.** [*20*] Завершите последовательность из §95 *De Combinatione* Изкуэрдо (Izquierdo):

ABC ABD ABE ACD ACE ACB ADE ADB ADC AEB ....

15. [*15*] Если все n-сочетания с повторениями $x_1 \ldots x_n$ множества $\{1, \ldots, m\}$ перечислить в лексикографическом порядке, причем $x_1 \le \cdots \le x_n$, то сколько из них будет начинаться с числа j?

16. [*20*] (Нараяна Пандита (Nārāyaṇa Paṇḍita), 1356.) Разработайте алгоритм для генерации всех композиций числа n на части $\le q$, т. е. всех упорядоченных разбиений $n = a_1 + \cdots + a_t$, где $1 \le a_j \le q$ для $1 \le j \le t$ и произвольного значения t. Проиллюстрируйте ваш метод для $n = 7$ и $q = 3$.

17. [*HM27*] Проанализируйте алгоритм из упр. 16.

18. [*10*] Шуточный вопрос. Лейбниц (Leibniz) опубликовал свою *Dissertatio de Arte Combinatoria* в 1666 году. Чем интересен этот год с точки зрения перестановок?

19. [*17*] В какой из строф Путеануса (Puteanus) (20) 'tibi' трактуется как ⏑—, а не как ⏑⏑?

20. [*M25*] К визиту трех титулованных особ в Дрезден в 1617 году поэт опубликовал 1617 перестановок гекзаметра

Dant tria jam Dresdæ, ceu sol dat, lumina lucem

("Трое даны ныне Дрездену, как Солнце дает луч за лучом"). [Gregor Kleppis, *Proteus Poeticus* (Leipzig: 1617).] Сколько перестановок этих слов являются гекзаметрами? *Указание*: указанная строфа в первой и пятой стопах имеет дактили, в остальных — спондеи.

21. [*HM30*] Пусть $f(p, q, r; s, t)$ — количество способов получить (o^p, o^q, o^r) путем конкатенации строк $\{s \cdot o, t \cdot oo\}$ при $p + q + r = s + 2t$. Например, $f(2, 3, 2; 3, 2) = 5$, поскольку этими пятью способами являются

$$(o|o, o|oo, oo), \quad (o|o, oo|o, oo), \quad (oo, o|o|o, oo), \quad (oo, o|oo, o|o), \quad (oo, oo|o, o|o).$$

а) Покажите, что $f(p, q, r; s, t) = [u^p v^q w^r z^s]\, 1/((1 - zu - u^2)(1 - zv - v^2)(1 - zw - w^2))$.
б) Воспользуйтесь функцией f для подсчета гекзаметров среди перестановок (19) при дополнительном условии, что пятая стопа не начинается посреди слова.
в) Подсчитайте остальные случаи.

▶ **22.** [*M40*] Познакомьтесь с решениями задачи о количестве гекзаметров среди перестановок (19), опубликованными Престе (Prestet), Уоллисом (Wallis), Уитвортом (Whitworth) и Хартли (Hartley). Какие ошибки они допустили?

23. [*20*] Какой порядок 52 диаграмм генджи-ко соответствует алгоритму 7.2.1.5H?

▶ **24.** [*23*] В начале 1800-х годов Тошиаки Хонда (Toshiaki Honda) предложил рекурсивное правило для генерации всех разбиений $\{1, \ldots, n\}$. При $n = 4$ его алгоритм давал такую последовательность разбиений.

Можете ли вы сказать, какой будет последовательность при $n = 5$? *Указание*: см. (26).

25. [*15*] Автора изданной в XVI веке книги *The Arte of English Poesie* интересовали только "полные" в смысле упр. 7.2.1.5–35 схемы рифм; другими словами, каждая строка должна рифмоваться как минимум с одной другой строкой. Кроме того, схемы должны быть "неприводимыми" в смысле упр. 7.2.1.2–100: разбиение наподобие 12|345 можно разделить на двустрочный стих, за которым следует трехстрочный. Наконец схема не должна тривиально состоять из строк, которые рифмуются каждая с каждой. Является ли при этих условиях (28) полным списком пятистрочных схем рифм?

▶ **26.** [*HM25*] Сколько схем рифм, состоящих из n строк, удовлетворяют ограничениям упр. 25?

▶ **27.** [*HM31*] Разбиение множества $14|25|36$ может быть представлено диаграммой генджи-ко наподобие ; однако каждая такая диаграмма для данного разбиения должна иметь как минимум три места пересечения линий, а такие пересечения иногда рассматриваются как нежелательные. Сколько разбиений множества $\{1,\ldots,n\}$ имеют диаграммы генджи-ко не более чем с одним пересечением?

▶ **28.** [*25*] Пусть a, b и c — простые числа. Джон Уоллис (John Wallis) перечислил все возможные разложения a^3b^2c следующим образом: $cbbaaa$, $cbbaa\cdot a$, $cbaaa\cdot b$, $bbaaa\cdot c$, $cbba\cdot aa$, $cbba\cdot a\cdot a$, $cbaa\cdot ba$, $cbaa\cdot b\cdot a$, $bbaa\cdot ca$, $bbaa\cdot c\cdot a$, $caaa\cdot bb$, $caaa\cdot b\cdot b$, $baaa\cdot cb$, $baaa\cdot c\cdot b$, $cbb\cdot aaa$, $cbb\cdot aa\cdot a$, $cbb\cdot a\cdot a\cdot a$, $cba\cdot baa$, $cba\cdot ba\cdot a$, $cba\cdot aa\cdot b$, $cba\cdot b\cdot a\cdot a$, $bba\cdot caa$, $bba\cdot ca\cdot a$, $bba\cdot aa\cdot c$, $bba\cdot c\cdot a\cdot a$, $caa\cdot bb\cdot a$, $caa\cdot ba\cdot b$, $caa\cdot b\cdot b\cdot a$, $baa\cdot cb\cdot a$, $baa\cdot ca\cdot b$, $baa\cdot ba\cdot c$, $baa\cdot c\cdot b\cdot a$, $aaa\cdot cb\cdot b$, $aaa\cdot bb\cdot c$, $aaa\cdot c\cdot b\cdot b$, $cb\cdot ba\cdot aa$, $cb\cdot ba\cdot a\cdot a$, $cb\cdot aa\cdot b\cdot a$, $cb\cdot b\cdot a\cdot a\cdot a$, $bb\cdot ca\cdot aa$, $bb\cdot ca\cdot a\cdot a$, $bb\cdot aa\cdot c\cdot a$, $bb\cdot c\cdot a\cdot a\cdot a$, $ca\cdot ba\cdot ba$, $ca\cdot ba\cdot b\cdot a$, $ca\cdot aa\cdot b\cdot b$, $ca\cdot b\cdot b\cdot a\cdot a$, $ba\cdot ba\cdot c\cdot a$, $ba\cdot aa\cdot c\cdot b$, $ba\cdot c\cdot b\cdot a\cdot a$, $aa\cdot c\cdot b\cdot b\cdot a$, $c\cdot b\cdot b\cdot a\cdot a\cdot a$. Какой алгоритм он использовал для генерации разбиения в указанном порядке?

▶ **29.** [*24*] В каком порядке Уоллис сгенерировал бы все разложения числа $abcde = 5\cdot 7\cdot 11\cdot 13\cdot 17$? Ваш ответ должен быть выражен в форме последовательности диаграмм генджи-ко.

30. [*M20*] Чему равен коэффициент при $a_1^{i_1}a_2^{i_2}\ldots z^{m+n}$ в $(a_0z+a_1z^2+a_2z^3+\cdots)^m$? (См. формулу (29).)

31. [*20*] Сравните упорядочения разбиений де Муавра (de Moivre) и де Монмора (de Montmort) (30) и (31) с алгоритмом 7.2.1.4P.

32. [*21*] (Р. И. Бошкович (R. J. Boscovich), 1748.) Перечислите все разбиения 20, все части которых представляют собой 1, 7 или 10. Разработайте алгоритм для перечисления всех таких разбиений заданного целого числа $n > 0$.

ОТВЕТЫ К УПРАЖНЕНИЯМ

Не отвечай глупому по глупости его,
чтоб и тебе не сделаться подобным ему.

— Притчи 26:4

ПРИМЕЧАНИЯ К УПРАЖНЕНИЯМ

1. Средняя задача для читателя с математическими наклонностями.

2. Автор вознаградит вас, если вы первым сообщите об ошибке в условии упражнения или в его ответе.

3. См. H. Poincaré, *Rendiconti Circ. Mat. Palermo* **18** (1904), 45–110; R. H. Bing, *Annals of Math.* (2) **68** (1958), 17–37; G. Perelman, `arXiv:math.DG/0211159`, `0303109`, `0307245`.

РАЗДЕЛ 7

1. Следуя указанию, мы хотим, чтобы непосредственно за вторым '$4m-4$' следовало первое '$2m-1$'. Требуемое размещение можно вывести из первых четырех примеров, приведенных в шестнадцатеричной записи: `231213`, `46171435623725`, `86a31b1368597a425b2479`, `ca8e531f1358ac7db9e6427f2469bd`. [R. O. Davies, *Math. Gazette* **43** (1959), 253–255.]

2. Такое расположение существует тогда и только тогда, когда $n \bmod 4 = 0$ или 1. Это условие является необходимым, поскольку должно быть четное число нечетных элементов. Оно также достаточно, поскольку мы можем поместить '00' перед ответом из предыдущего упражнения.

Примечания. Этот вопрос впервые был задан Маршаллом Холлом (Marshall Hall) в 1951 году и решен в следующем году Ф. Т. Лихи-мл. (F. T. Leahy, Jr.) в неопубликованной работе [Armed Forces Security Agency report 343 (28 January 1952)]. Эта задача была независимо поставлена и решена Т. Сколемом (T. Skolem) и Т. Бангом (T. Bang), *Math. Scandinavica* **5** (1957), 57–58. Полное решение для других диапазонов чисел дано Д. Э. Симпсоном (J. E. Simpson), *Discrete Math.* **44** (1983), 97–104.

3. Да. Например, цикл (0072362435714165) "выпрямить" нельзя.

4. k-е появление b находится в позиции $\lfloor k\phi \rfloor$ слева, а k-е появление a — в позиции $\lfloor k\phi^2 \rfloor$. Ясно, что $\lfloor k\phi^2 \rfloor - \lfloor k\phi \rfloor = k$, поскольку $\phi^2 = \phi + 1$. (Целые числа $\lfloor k\phi \rfloor$ образуют "спектр" ϕ; см. упр. 3.13 из *CMath*.)

5. Указанному условию удовлетворяют $2n-k-1$ из $\binom{2n}{2}$ равновероятных пар позиций. Если бы эти вероятности были независимы (но это не так), значение $2L_n$ было бы равно

$$\binom{2n}{2,2,\ldots,2} \prod_{k=1}^{n} \bigl((2n-1-k)/\tbinom{2n}{2}\bigr) = \frac{(2n)!^2 n(n-1)}{n!(2n)^{n+1}(2n-1)^{n+1}}$$
$$= \exp\Bigl(n \ln \frac{4n}{e^3} + \ln \sqrt{\frac{\pi e n}{2}} + O(n^{-1})\Bigr).$$

6. (а) При раскрытии произведений мы получим полином из $(2n-2)!/(n-2)!$ членов, каждый степени $4n$. Для каждой пары Лэнгфорда имеется член $x_1^2 \ldots x_{2n}^2$; все прочие члены имеют как минимум одну переменную степени 1. Суммирование по $x_1, \ldots, x_{2n} \in \{-1, +1\}$, следовательно, сокращает все плохие члены, но дает 2^{2n} для хороших. Дополнительный множитель 2 возникает из-за того, что имеется $2L_n$ размещений Лэнгфорда (включая отражения).

(б) Пусть $f_k = \sum_{j=1}^{2n-k-1} x_j x_{j+k+1}$ представляет собой главную часть k-го множителя. Мы можем пройти по всем 4^n случаям $x_1, \ldots, x_{2n} \in \{-1, +1\}$ в порядке кода Грея (алгоритм 7.2.1.1L), всякий раз меняя знак только у одного x_j. Изменение x_j вызывает не более двух корректировок в каждой f_k; так что стоимость каждого шага кода Грея составляет $O(n)$.

Нам нет необходимости вычислять сумму точно; достаточно работать по модулю 2^N, где 2^N удобно превышает $2^{2n+1}L_n$. Что еще лучше, когда $n = 24$, вычисления можно было бы делать по модулю $2^{60}-1$, или по модулю $2^{30}-1$ и $2^{30}+1$, поскольку $2^{49} \perp 2^{60}-1$. Можно также сэкономить $\lceil n/2 \rceil$ бит точности, используя тот факт, что $f_k \equiv k+1$ (по модулю 2).

(в) Третье равенство в действительности корректно, только когда $n \bmod 4 = 0$ или 3; но это интересные n. Сумма может быть вычислена за n этапов, где этап p для $p < n$ включает случаи, где $x_{n-1} = x_{n+2}$, $x_{n-2} = x_{n+3}$, $\ldots$, $x_{n-p+1} = x_{n+p}$, $x_{n-p} = x_n = x_{n+1} = +1$ и $x_{n+p+1} = -1$; он имеет внешний цикл, который выбирает $(x_{n-p+1}, \ldots, x_{n-1})$ всеми 2^{p-1} способами, и внутренний цикл, который выбирает $(x_1, \ldots, x_{n-p-1}, x_{n+p+2}, \ldots, x_{2n})$ всеми $2^{2n-2p-2}$ способами. (Внутренний цикл использует бинарный код Грея, предпочтительно с "порядком органных труб" для назначения приоритетов индексам так, что x_1 и x_{2n} варьируют наиболее быстро. Внешний цикл не требуется делать особенно эффективным.) Этап n охватывает 2^{n-1} палиндромных случаев с $x_j = x_{2n+1-j}$ для $1 \le j < n$ и $x_n = x_{n+1} = +1$. Если s_p обозначает сумму на этапе p, то $s_1 + \cdots + s_{n-1} + \frac{1}{2}s_n = 2^{2n-2}L_n$.

Существенная доля членов оказывается равной нулю. Например, когда $n = 16$, нули появляются примерно в 76% случаев (в 408 838 754 случаях из $2^{29} + 2^{14}$). Этот факт можно использовать для того, чтобы избежать умножения во внутреннем цикле. (Нулем может быть только f_1, f_3, $\ldots$.)

7. Пусть d_k — количество неполных пар после прочтения k чисел; таким образом, $d_0 = d_{2n} = 0$ и $d_k = d_{k-1} \pm 1$ для $1 \le k \le 2n$. Наибольшая такая последовательность, в которой d_k никогда не превышает 6, представляет собой $(d_0, d_1, \ldots, d_{2n}) = (0, 1, 2, 3, 4, 5, 6, 5, 6, \ldots, 5, 6, 5, 4, 3, 2, 1, 0)$, для которой $\sum_{k=1}^{2n} d_k = 11n - 30$. Но в любой последовательности Лэнгфорда $\sum_{k=1}^{2n} d_k = \sum_{k=1}^{n}(k+1) = \binom{n+1}{2} + n$. Следовательно, $\binom{n+1}{2} + n \le 11n - 30$ и $n \le 15$. (На самом деле ширина 6 невозможна и при $n = 15$. Значения наибольшей и наименьшей ширины в общем случае неизвестны.)

8. Решений при $n = 4$ и $n = 7$ нет. При $n = 8$ имеется четыре таких решения:

(Эта задача приводит к интересной механической головоломке, использующей элементы шириной $k+1$ и высотой $\lceil k/2 \rceil$ для разных k. В своей заметке [*Math. Gazette* **42** (1958), 228] Ч. Дадли Лэнгфорд (C. Dudley Langford) проиллюстрировал аналогичные элементы и привел планарное решение для $n = 12$. Этот вопрос можно привести к задаче точного покрытия, в которой непростые столбцы представляют места, в которых не могут пересекаться два элемента; см. упр. 7.2.2.1–00. Жан Бретт (Jean Brette) разработал похожую головоломку на основе варианта задачи, предложенного Сколемом и использующего ширину вместо планарности; он передал ее копию Дэвиду Сингмастеру (David Singmaster) в 1992 году.)

9. Лишь тремя способами: 181915267285296475384639743, 191218246279458634753968357, 191618257269258476354938743 (и обратными к ним). [Впервые найдены в 1969 году Г. Бароном (G. Baron); см. *Combinatorial Theory and Its Applications* (Budapest: 1970), 81–92. Метод "танцующих связей" из раздела 7.2.2.1 решает этот вопрос с помощью обхода дерева, имеющего всего лишь 360 узлов для данной задачи точного покрытия со 132 строками.]

10. Например, положите A = 12, K = 8, Q = 4, J = 0, ♠ = 4, ♡ = 3, ♢ = 2, ♣ = 1 и просуммируйте.

[В связи с этим следует упомянуть, что ортогональные латинские квадраты, эквивалентные рис. 1, были неявно представлены уже в средневековых исламских талисманах, проиллюстрированных Ибн аль-Хаджем (Ibn al-Hajj) в его *Kitab Shumus al-Anwar* (Cairo: 1322), который привел также пример 5×5. См. E. Doutté, *Magie et Religion dans l'Afrique du Nord* (Algiers: 1909), 193–194, 214, 247; W. Ahrens, *Der Islam* **7** (1917), 228–238. См. также статью по истории латинских квадратов, подготовленную Ларсом Д. Андерсеном (Lars D. Andersen).]

11. $\begin{pmatrix} d\gamma\aleph & a\delta\beth & b\beta\gimel & c\alpha\daleth \\ c\beta\beth & b\alpha\aleph & a\gamma\daleth & d\delta\gimel \\ a\alpha\gimel & d\beta\daleth & c\delta\aleph & b\gamma\beth \\ b\delta\daleth & c\gamma\gimel & d\alpha\beth & a\beta\aleph \end{pmatrix}$. [Жозе Сове (Joseph Sauveur) представил самый ранний из известных примеров таких квадратов в *Mémoires de l'Académie Royale des Sciences* (Paris, 1710), 92–138, §83.]

12. Если n нечетно, можно положить $M_{ij} = (i - j) \bmod n$. Но если n четно, то секущих не существует: если $\{(t_0+0) \bmod n, \ldots, (t_{n-1}+n-1) \bmod n\}$ является секущей, то мы имеем $\sum_{k=0}^{n-1} t_k \equiv \sum_{k=0}^{n-1}(t_k + k)$ (по модулю n), следовательно, $\sum_{k=0}^{n-1} k = \frac{1}{2}n(n-1)$ кратно n.

13. Заменим каждый элемент l на $\lfloor l/5 \rfloor$, чтобы получить матрицу, состоящую из нулей и единиц. Дадим имена четвертям следующим образом: $\left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$; тогда и A, и D содержат по k единиц, в то время как и B, и C содержат по k нулей. Предположим, что исходная матрица имеет десять непересекающихся секущих. Если $k \le 2$, то не более четырех из них проходят через 1 в A или D и не более четырех из них проходят через 0 в B или C. Таким образом, как минимум две из них проходят только через нули в A и D и только через единицы в B и C. Но такие секущие имеют четное число нулей (не пять), поскольку пересекают A и D одинаково часто.

Аналогично латинский квадрат порядка $4m+2$ с ортогональным напарником должен иметь больше m нарушителей в каждой из подматриц $(2m+1) \times (2m+1)$ при всех переименованиях элементов. [H. B. Mann, *Bull. Amer. Math. Soc.* (2) **50** (1944), 249–257.]

14. В случаях (б) и (г) таких квадратов нет. В случаях (а), (в) и (д) их имеется соответственно 2, 6 и 12 265 168(!), лексикографически первыми и последними из которых являются

(а)	(а)	(в)	(в)	(д)	(д)
0456987213	0691534782	0362498571	0986271435	0214365897	0987645321
1305629847	1308257964	1408327695	1354068792	1025973468	1795402638
2043798165	2169340578	2673519408	2741853960	2690587143	2506913874
3289176504	3250879416	3521970846	3572690814	3857694201	3154067289
4518263790	4587902631	4890253167	4630789251	4168730925	4231850967
5167432089 ,	5412763890 ,	5736841920 ,	5218947306 ,	5473829016 ,	5348276190 .
6894015372	6945081327	6259784013	6095324178	6942158730	6820394715
7920341658	7836425109	7915602384	7869512043	7309216584	7069128543
8731504926	8723196045	8147036259	8407136529	8531402679	8412739056
9672850431	9074618253	9084165732	9123405687	9786041352	9673581402

Примечания. Квадраты (а), (б), (в) и (г) получены из десятичных цифр π, e, γ и ϕ путем отбрасывания цифр, не согласующихся с полным латинским квадратом. Хотя они не являются истинно случайными, вероятно, в общем случае это типичные латинские квадраты 10×10, примерно половина из которых имеет ортогональные им квадраты.

Паркер (Parker) построил квадрат (д) так, чтобы получить необычно большое количество секущих; в данном случае их 5504. (Эйлер (Euler) изучал подобный пример порядка 6, так что он просто "прозевал" открытие пары 10×10.)

15. Паркер (Parker) был потрясен, обнаружив, что среди квадратов, ортогональных квадрату 14, (д), не оказалось квадратов, ортогональных друг другу. Позже вместе с Д. У. Брауном (J. W. Brown) и А. С. Хедаятом (A. S. Hedayat) [*J. Combinatorics, Inf. and System Sci.* **18** (1993), 113–115] он нашел два квадрата размером 10×10, которые имели четыре непересекающиеся общие секущие (но не десять). [См. также B. Ganter, R. Mathon, and A. Rosa, *Congressus Numerantium* **20** (1978), 383–398; **22** (1979), 181–204.] Продолжая идею Л. Вайзнера (L. Weisner) [*Canadian Math. Bull.* **6** (1963), 61–63], автор данной книги случайно заметил некоторые квадраты, более близкие ко взаимно ортогональному трио: квадрат ниже ортогонален себе же транспонированному; и он имеет пять диагонально симметричных секущих, в ячейках $(0, p_0), \ldots, (9, p_9)$ для $p_0 \ldots p_9 = 0132674598$, 2301457689, 3210896745, 4897065312 и 6528410937, которые *почти* не пересекаются: они покрывают 49 ячеек:

$$L \quad = \quad \begin{pmatrix} 0234567891 \\ 3192708546 \\ 6528139407 \\ 8753241960 \\ 1689473025 \\ 4970852613 \\ 5047986132 \\ 9416320758 \\ 7361095284 \\ 2805614379 \end{pmatrix} \quad \perp \quad \begin{pmatrix} 0368145972 \\ 2157690438 \\ 3925874160 \\ 4283907615 \\ 5712489306 \\ 6034758291 \\ 7891326054 \\ 8549061723 \\ 9406213587 \\ 1670532849 \end{pmatrix} \quad = \quad L^T.$$

Обширные вычисления, проведенные Б. Д. Мак-Кеем (B. D. McKay), Э. Мейнертом (A. Meynert) и В. Мирвольд (W. Myrvold) [*J. Comb. Designs* **15** (2007), 98–119], доказали, что не существует латинских квадратов размером 10×10 с нетривиальной симметрией, которые имели бы два ортогональных квадрата, ортогональных также друг другу. Известно, что три взаимно ортогональных латинских квадрата существуют для всех порядков $n > 10$ [см. S. M. P. Wang and R. M. Wilson, *Congressus Numerantium* **21** (1978), 688; D. T. Todorov, *Ars Combinatoria* **20** (1985), 45–47].

16. См. R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory* (Cambridge University Press, 1991), §8.2.

17. (а) Пусть имеется $3n$ столбцов r_j, c_j, v_j для $0 \le j < n$ и n^2 строк; строка (i, j) имеет 1 в столбцах r_i, c_j и v_l, где $l = L_{ij}$, для $0 \le i, j < n$.

(б) Пусть имеется $4n^2$ столбцов r_{ij}, c_{ij}, x_{ij}, y_{ij} для $0 \le i, j < n$ и $n^3 - n^2 + n$ строк; строка (i, j, k) имеет 1 в столбцах r_{ik}, c_{jk}, x_{ij} и y_{lk}, где $l = L_{ij}$, для $0 \le i, j, k < n$ и ($i = k$ или $j > 0$).

18. Определим для заданного ортогонального массива A со строками A_i для $1 \le i \le m$ латинский квадрат $L_i = (L_{ijk})$ для $1 \le i \le m-2$, полагая $L_{ijk} = A_{iq}$ когда $A_{(m-1)q} = j$ и $A_{mq} = k$, для $0 \le j, k < n$. (Значение q единственным образом определяется значениями j и k.) Перестановка столбцов массива не изменяет соответствующие латинские квадраты.

Это построение может быть обращено, производя ортогональные массивы порядка n из взаимно ортогональных латинских квадратов порядка n. В упр. 11, например, можно положить $a = \alpha = \aleph = 0$, $b = \beta = \beth = 1$, $c = \gamma = \gimel = 2$ и $d = \delta = \daleth = 3$, получая

$$A = \begin{pmatrix} 3012210303211230 \\ 2310102301323201 \\ 0123103223013210 \\ 0000111122223333 \\ 0123012301230123 \end{pmatrix}.$$

(Концепция ортогонального массива математически "чище", чем концепция ортогональных латинских квадратов, поскольку она лучше учитывает имеющиеся симметрии. Заметим, например, что матрица L размером $n \times n$ с элементами из $\{1, 2, \ldots, n\}$ является латинским квадратом тогда и только тогда, когда он ортогонален двум нелатинским квадратам специального вида, а именно

$$L \perp \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 2 & 2 & \ldots & 2 \\ \vdots & \vdots & \ddots & \vdots \\ n & n & \ldots & n \end{pmatrix} \quad \text{и} \quad L \perp \begin{pmatrix} 1 & 2 & \ldots & n \\ 1 & 2 & \ldots & n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 2 & \ldots & n \end{pmatrix}.$$

Следовательно, латинские квадраты, греко-латинские квадраты, иврито-греко-латинские квадраты и т. д. эквивалентны ортогональным массивам глубиной 3, 4, 5, ... Более того, рассматриваемые здесь ортогональные массивы представляют собой просто частный случай $t = 2$ и $\lambda = 1$ более общей концепции n-арных массивов $m \times \lambda n^t$ с "мощностью t" и "индексом λ", введенной К. Р. Рао (C. R. Rao) в *Proc. Edinburgh Math. Soc.* **8** (1949), 119–125; см. книгу *Orthogonal Arrays* А. С. Хедаята (A. S. Hedayat), Н. Д. А. Слоана (N. J. A. Sloane) и Д. Стафкена (J. Stufken) (Springer, 1999).)

19. Можно переупорядочить столбцы таким образом, что первая строка примет вид $0^n 1^n \ldots (n{-}1)^n$. Тогда можно перенумеровать элементы других строк так, чтобы они начинались с $01 \ldots (n{-}1)$. Элементы в каждом оставшемся столбце должны быть различны во всех строках, кроме первой.

Для достижения верхней границы при $n = p$ проиндексируем каждый столбец двумя числами, x и y, где $0 \le x, y < p$, и поместим числа y, x, $(x + y) \bmod p$, $(x + 2y) \bmod p$, $\ldots$, $(x + (p{-}1)y) \bmod p$ в этот столбец. Например, когда $p = 5$, мы получим следующий ортогональный массив, эквивалентный четырем взаимно ортогональным латинским квадратам:

$$\begin{pmatrix} 0&0&0&0&0&1&1&1&1&1&2&2&2&2&2&3&3&3&3&3&4&4&4&4&4 \\ 0&1&2&3&4&0&1&2&3&4&0&1&2&3&4&0&1&2&3&4&0&1&2&3&4 \\ 0&1&2&3&4&1&2&3&4&0&2&3&4&0&1&3&4&0&1&2&4&0&1&2&3 \\ 0&1&2&3&4&2&3&4&0&1&4&0&1&2&3&1&2&3&4&0&3&4&0&1&2 \\ 0&1&2&3&4&3&4&0&1&2&1&2&3&4&0&4&0&1&2&3&2&3&4&0&1 \\ 0&1&2&3&4&4&0&1&2&3&3&4&0&1&2&2&3&4&0&1&1&2&3&4&0 \end{pmatrix}.$$

[По сути, такая же идея работает и тогда, когда n является степенью простого числа, с использованием конечного поля $\mathrm{GF}(p^e)$; см. E. H. Moore, *American Journal of Mathematics* **18** (1896), 264–303, §15(l). Эти массивы эквивалентны конечным *проективным плоскостям*; см. Marshall Hall, Jr., *Combinatorial Theory* (Blaisdell, 1967), Chapters 12 and 13.]

20. Пусть $\omega = e^{2\pi i/n}$, и предположим, что $a_1 \ldots a_{n^2}$ и $b_1 \ldots b_{n^2}$ являются векторами из разных строк. Тогда $a_1 b_1 + \cdots + a_{n^2} b_{n^2} = \sum_{0 \le j,k < n} \omega^{j+k} = 0$, поскольку $\sum_{k=0}^{n-1} \omega^k = 0$.

21. (а) Чтобы показать, что отношение "эквивалентность или параллельность" является отношением эквивалентности, нам нужно убедиться в законе транзитивности: если $L \parallel M$, $M \parallel N$ и $L \ne N$, то должно выполняться $L \parallel N$. В противном случае согласно (ii) должна иметься такая точка p, что $L \cap N = \{p\}$); и p будет лежать на двух разных прямых, параллельных M, что приводит к противоречию с (iii).

(б) Пусть $\{L_1, \ldots, L_n\}$ является классом параллельных прямых, и пусть M — прямая из другого класса. Тогда каждая L_j пересекается с M в единственной точке p_j; и так встречаются все точки M, поскольку каждая точка геометрии лежит ровно на одной прямой каждого класса согласно (iii). Таким образом, M содержит ровно n точек.

(в) Мы уже заметили, что каждая точка принадлежит m прямым, если имеется m классов. Если прямые L, M и N принадлежат трем разным классам, то M и N имеют

то же количество точек, что и количество точек прямых из класса L. Так что существует общий размер прямых n, и в действительности общее количество точек равно n^2. (Конечно, n может быть бесконечно.)

22. Определим для данного ортогонального массива A порядка n и глубиной m геометрическую сеть с n^2 точками и m классами параллельных прямых, рассматривая столбцы A как точки; прямая j из класса k представляет собой множество столбцов, в которых символ j находится в строке k массива A.

Так получаются все конечные геометрические сети с $m \geq 3$ классами. Но геометрическая сеть только с одним классом представляет собой тривиальное разбиение точек на непересекающиеся подмножества. Геометрическая сеть с $m = 2$ классами имеет nn' точек (x, x'), где всего имеется n прямых '$x =$ constant' в одном классе и n' прямых '$x' =$ constant' в другом. [Более подробную информацию можно найти в R. H. Bruck, *Canadian J. Math.* **3** (1951), 94–107; *Pacific J. Math.* **13** (1963), 421–457.]

23. (а) Если $d(x, y) \leq t$ и $d(x', y) \leq t$ и $x \neq x'$, то $d(x, x') \leq 2t$. Таким образом, код с расстоянием $> 2t$ между кодовыми словами позволяет исправлять до t ошибок — по крайней мере в принципе, хотя необходимые вычисления могут оказаться весьма сложными. И наоборот, если $d(x, x') \leq 2t$ и $x \neq x'$, существует элемент y, такой, что $d(x, y) \leq t$ и $d(x', y) \leq t$; следовательно, мы не можем однозначно восстановить x при получении y.

(б, в) Пусть $m = r + 2$, и заметим, что множество b^2 b-арных m-кортежей имеет расстояние Хэмминга $\geq m - 1$ между всеми парами элементов тогда и только тогда, когда оно образует столбцы b-арного ортогонального массива глубиной m. [См. S. W. Golomb и E. C. Posner, *IEEE Trans.* **IT-10** (1964), 196–208. В литературе по теории кодов код $C(b, n, r)$ с расстоянием d часто обозначают как $(n + r, b^n, d)_b$. Таким образом, b-арный ортогональный массив глубиной m, по сути, является кодом $(m, b^2, m - 1)_b$.]

24. (а) Допустим, что $x_j \neq x'_j$ для $1 \leq j \leq l$ и $x_j = x'_j$ для $l < j \leq N$. Мы имеем $x = x'$ при $l = 0$. В противном случае рассмотрим биты четности, соответствующие m прямым, проходящим через точку 1. Не более $l-1$ из этих битов соответствуют прямым, проходящим через точки $\{2, \ldots, l\}$. Следовательно, x' имеет как минимум $m-(l-1)$ изменений четности и $d(x, x') \geq l + (m - (l-1)) = m + 1$.

(б) Пусть $l_{p1}, \ldots, l_{pm}$ — номера индексов прямых, проходящих через точку p. После получения сообщения $y_1 \ldots y_{N+R}$ вычислим x_p для $1 \leq p \leq N$, взяв значение большинства из $m + 1$ "битов-свидетелей" $\{y_{p0}, \ldots, y_{pm}\}$, где $y_{p0} = y_p$ и

$$y_{pk} = \bigl(y_{N+l_{pk}} + \textstyle\sum\{y_j \mid j \neq p \text{ и точка } j \text{ лежит на прямой } l_{pk}\}\bigr) \bmod 2, \quad \text{для } 1 \leq k \leq m.$$

Этот метод работает, поскольку каждый полученный бит y_j влияет не более чем на один бит-свидетель.

Например, в 25-точечной геометрии из упр. 19 предположим, что бит четности $x_{26+5i+j}$ каждого кодового слова соответствует прямой j строки i, для $0 \leq i \leq 5$ и $0 \leq j < 5$; таким образом, $x_{26} = x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5$, $x_{27} = x_6 \oplus x_7 \oplus x_8 \oplus x_9 \oplus x_{10}$, $\ldots$, $x_{55} = x_5 \oplus x_6 \oplus x_{12} \oplus x_{18} \oplus x_{24}$. Для данного сообщения $y_1 \ldots y_{55}$ мы декодируем бит x_1, скажем, путем вычисления значения большинства из семи битов y_1, $y_{26} \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_5$, $y_{31} \oplus y_6 \oplus y_{11} \oplus y_{16} \oplus y_{21}$, $y_{36} \oplus y_{10} \oplus y_{14} \oplus y_{18} \oplus y_{22}$, $y_{41} \oplus y_9 \oplus y_{12} \oplus y_{20} \oplus y_{23}$, $y_{46} \oplus y_8 \oplus y_{15} \oplus y_{17} \oplus y_{24}$, $y_{51} \oplus y_7 \oplus y_{13} \oplus y_{19} \oplus y_{25}$. [В разделе 7.1.2 поясняется, как эффективно вычислять функции мажоризации. Заметим, что мы можем исключить последние 10 бит, если мы хотим корректировать до двух ошибок, и последние 20, если достаточно коррекции одной ошибки. См. M. Y. Hsiao, D. C. Bossen and R. T. Chien, *IBM J. Research and Development* **14** (1970), 390–394.]

25. Рассматривая анаграммы из {l, e, a, s, t} (см. упр. 5–21), мы приходим к квадрату

```
stela
telas
elast
laste
astel
```

и циклическим сдвигам его строк. Здесь telas — испанская ткань; elast — префикс, означающий гибкость; а laste — императивный чосеровский глагол. (Конечно, практически все произносимые комбинации пяти букв были кем-то где-то когда-то использованы для тех или иных целей, например для заклинаний...)

26. "every night, young video buffs catch rerun fever forty years after those great shows first aired." ("Каждую ночь молодых любителей видео постоянно лихорадит — через сорок лет после первого выхода в эфир этих великих шоу".) [Robert Leighton, *GAMES* **16**, 6 (December 1992), 34, 47.]

27. $(0, 4, 163, 1756, 3834)$ для $k = (1, 2, 3, 4, 5)$; для любителей покера — mamma и esses дают "фул-хаус".

28. Да, всего 38 пар. "Наиболее распространенным" решением являются слова needs (ранг 180) и offer (ранг 384). Имеется только три случая отличия всех компонентов на +1: (adder beefs, sheer tiffs, sneer toffs). Другими незабываемыми примерами являются ghost hints и strut rusts. Во всех случаях одно из слов пары заканчивается буквой s, за исключением четырех пар наподобие robed spade. [См. Leonard J. Gordon, *Word Ways* **23** (1990), 59–61.]

29. Всего имеется 18 палиндромов, от level (ранг 184) до dewed (ранг 5688). Вот некоторые из 34 зеркальных пар: 'devil lived', 'knits stink', 'smart trams', 'faced decaf'.

30. Среди 105 таких слов в SGB наиболее распространенными являются first, below, floor, begin, cells, empty и hills; abbey и pssst являются лексикографически первым и последним словами. (Если вам не нравится pssst, то предпоследнее слово — mossy.) В *обратном* алфавитном порядке находятся буквы только у 37 слов, от mecca до zoned.

31. Слово посредине представляет собой арифметическое среднее двух других, так что крайние слова должны быть конгруэнтны по модулю 2; это наблюдение уменьшает количество просмотров словаря примерно в 32 раза. В WORDS(5757) имеется 119 таких троек, но в WORDS(2000) их только две: marry, photo, solve; risky, tempo, vague. [*Word Ways* **25** (1992), 13–15.]

32. Похоже, единственным разумным примером является peopleless.

33. about, bacon, faced, under, chief, fight, right, which, ouija, jokes, ankle, films, hymns, known, crops, pique, quart, first, first, study, mauve, vowel, waxes, proxy, crazy, pizza. (Идея заключается в том, чтобы найти наиболее распространенное слово, в котором за x следует $(x+1) \bmod 26$, для $x =$ a (0), $x =$ b (1), ..., $x =$ z (25). Мы также минимизируем промежуточное расстояние, предпочитая, таким образом, bacon более распространенному слову black. В одном случае, где такого слова не существует, crazy показалось самым разумным. См. *OMNI* **16**, 8 (May 1994), 94.)

34. Первые два слова (и общее количество слов) в каждой категории следующие: pssst и pffft (2), schwa и schmo (2), threw и throw (36), three и spree (5), which и think (709), there и these (234), their и great (291), whooo и wheee (3), words и first (628), large и since (376), water и never (1313), value и radio (84), would и could (460), house и voice (101), quiet и queen (25), queue только (1), ahhhh и ankhs (4), angle и extra (20), other и after (227), agree и issue (20), along и using (124), above и alone (92), about и again (58), adieu и aquae (2), earth и eight (16), eagle и ounce (8), outer и eaten (42), eerie

и `audio` (4), (0), `ouija` и `aioli` (2), (0), (0); `years` и `every` — наиболее распространенные из 868 опущенных слов. [Для заполнения трех пропусков в Интернете нашлись `ooops`, `ooooh` и `ooooo`. См. P. M. Cohen, *Word Ways* **10** (1977), 221–223.]

35. Рассмотрим коллекцию `WORDS`(n) для $n = 1, 2, \ldots, 5757$. Проиллюстрированный луч с корнем `s` впервые становится возможным, когда n достигает 978 (ранг `stalk`). Следующая корневая буква, поддерживающая такой луч, — `c`, которая получает достаточное ветвление среди потомков при $n = 2503$ (ранг `craze`). Следующие прорывы обнаруживаются при $n = 2730$ (`bulks`), 3999 (`ducky`), 4230 (`panty`), 4459 (`minis`), 4709 (`whooo`), 4782 (`lardy`), 4824 (`herem`), 4840 (`firma`), 4924 (`ridgy`), 5343 (`taxol`).

(Прорыв осуществляется, когда луч верхнего уровня получает число Хортона–Страхлера 4; см. упр. 7.2.1.6–124. Забавные множества слов, вызывающие мысли о новом виде поэзии, возникают также при ветвлении справа налево вместо ветвления слева направо: `black`, `slack`, `crack`, `track`, `click`, `slick`, `brick`, `trick`, `blank`, `plank`, `crank`, `drank`, `blink`, `clink`, `brink`, `drink`. В действительности ветвление справа налево дает полный *тернарный* луч с 81 листом: `males`, `sales`, `tales`, `files`, `miles`, `piles`, `holes`, ..., `tests`, `costs`, `hosts`, `posts`.)

36. При обозначении элементов куба как a_{ijk} для $1 \le i, j, k \le 5$, получаем условия симметрии $a_{ijk} = a_{ikj} = a_{jik} = a_{jki} = a_{kij} = a_{kji}$. В общем случае куб размером $n \times n \times n$ содержит $3n^2$ слов, получаемых путем фиксации двух координат с позволением третьей меняться от 1 до n; но условие симметрии означает, что нам нужно только $\binom{n+1}{2}$ слов. Следовательно, когда $n = 5$, количество необходимых слов снижается с 75 до 15. [Джефф Грант (Jeff Grant) смог найти 75 подходящих слов в *Oxford English Dictionary*; см. *Word Ways* **11** (1978), 156–157.]

Замена (`stove`, `event`) на (`store`, `erect`) или (`stole`, `elect`) дает еще два куба.

37. Наиболее плотная часть графа, которую можно назвать "активной зоной" ("bare core"), содержит вершины `bares` и `cores`, каждая из которых имеет степень 25.

38. `tears` $\to$ `raise` $\to$ `aisle` $\to$ `smile`; вторым словом может быть также `reals`. [Переход от `tears` к `smile`, показанный в (11), был первым пятибуквенным примером Льюиса Кэрролла (Lewis Carroll). Ему было бы интересно узнать, что "ориентированное правило" затрудняет переход от `smile` к `tears`, поскольку в этом направлении требуется *четыре* шага.]

39. Всегда остовный и никогда — порожденный.

40. (а) 2^e, (б) 2^n, по одному для каждого подмножества E или V.

41. (а) $n = 1$ и $n = 2$; P_0 не определен. (б) $n = 0$ и $n = 3$.

42. G имеет $65/2$ ребер (а следовательно, не существует).

43. Да: первые три графа изоморфны рис. 2, (д). [Крайняя слева диаграмма в действительности представляет собой наиболее раннее из известных изображений графа Петерсена в печати: см. A. B. Kempe, *Philosophical Transactions* **177** (1886), 1–70, в особенности рис. 13 в §59.] Крайний справа граф, определенно, отличается; он планарный, гамильтонов и имеет обхват 3.

44. Любой автоморфизм должен переводить угловую точку в угловую точку, поскольку три различных пути длиной 2 можно найти только между определенными парами неугловых точек. Следовательно, граф имеет только восемь симметрий C_4.

45. Все ребра этого графа соединяют вершины в одной и той же строке или в соседних. Следовательно, мы можем поочередно использовать цвета 0 и 2 в четных строках, а цвета

1 и 3 — аналогично в нечетных. Соседи NV образуют цикл длиной 5, следовательно, четыре цвета необходимы.*

46. (а) Каждая вершина имеет степень ≥ 2, и ее соседи имеют точно определенный циклический порядок, соответствующий входящим ребрам. Если u — v и u — w, где v и w являются циклически последовательными соседями u, то мы должны иметь v — w. Таким образом, все точки в окрестности любой вершины u принадлежат единственной треугольной области.

(б) Формула выполняется при $n = 3$. Если $n > 3$, стянем любое ребро в точку; такое преобразование удаляет одну вершину и три ребра. (Если мы стягиваем u — v, предположим, что оно является частью треугольников x — u — v — x и y — u — v — y. Тогда мы теряем вершину v и ребра $\{x$ — v, u — v, y — $v\}$; все прочие ребра вида w — v становятся ребрами w — u.)

47. Планарная диаграмма делит плоскость на области с 4 или 6 вершинами на границе каждой из них (поскольку $K_{3,3}$ не имеет нечетных циклов). Если имеется f_4 и f_6 областей каждого вида, должно выполняться соотношение $4f_4 + 6f_6 = 18$, поскольку всего имеется 9 ребер; следовательно, $(f_4, f_6) = (3, 1)$ или $(0, 3)$. Мы можем также триангулировать граф, добавляя к нему еще $f_4 + 3f_6$ ребер; но тогда в нем будет как минимум 15 ребер, что противоречит упр. 46.

[Тот факт, что граф $K_{3,3}$ не планарный, возвращает нас к головоломке о подключении трех домов к источникам воды, газа и электричества без пересечений. Происхождение этой головоломки неизвестно; Г. Э. Дьюдени (H. E. Dudeney) называет ее в *Strand* **46** (1913), 110 "древней."]

48. Если u, v, w являются вершинами и u — v, то мы должны иметь $d(w, u) \not\equiv d(w, v)$ (по модулю 2); в противном случае кратчайшие пути от w до u и от w до v должны давать нечетный цикл. После того как w окрашена в цвет 0, процедура назначает цвет $d(w, v) \bmod 2$ каждой новой неокрашенной вершине v, смежной с окрашенной вершиной u; а каждая вершина v с $d(w, v) < \infty$ окрашивается до того, как выбирается новая вершина w.

49. Их только три: K_4, $K_{3,3}$ и (который представляет собой $\overline{C}_6$).

50. Граф должен быть связным, поскольку количество 3-раскрашиваний при наличии r компонентов делится на 3^r. Он должен также содержаться в полном двудольном графе $K_{m,n}$, который может быть раскрашен в три цвета $3(2^m + 2^n - 2)$ способами. Удаление ребер из $K_{m,n}$ не уменьшает количество раскрашиваний; следовательно, $2^m + 2^n - 2 \leq 8$, и мы получаем $\{m, n\} = \{1, 1\}$, $\{1, 2\}$, $\{1, 3\}$ или $\{2, 2\}$. Таким образом, единственно возможными решениями являются "лапа" $K_{1,3}$ и путь P_4.

51. Цикл длиной 4 p_1 — L_1 — p_2 — L_2 — p_1 соответствовал бы двум различным прямым $\{L_1, L_2\}$ с двумя общими точками $\{p_1, p_2\}$, что противоречит (ii). Так что обхват как минимум равен 6.

Если имеется только один класс параллельных прямых, обхват равен ∞; если имеется два класса с $n \leq n'$ членами, обхват равен 8, или ∞ при $n = 1$. (См. ответ к упр. 22.) В противном случае можно найти цикл длиной 6, делая треугольник из трех прямых, выбранных из разных классов.

52. Если диаметр обозначить через d, а обхват через g, то $d \geq \lfloor g/2 \rfloor$, если только g не равен ∞.

53. `happy` (которое связано с `tears` и `sweat`, но не с `world`).

---

* Те же рассуждения (а значит, и способ раскрашивания) применимы и для приведенного там же графа областей Украины (17, у), включая цикл длиной 5, образуемый соседями `Тернополя`, или длиной 7 из соседей `Днепропетровска`. Для раскрашивания графа республик СССР (17с) также требуются четыре краски — из-за цикла длиной 3, образуемого соседями `Литвы`. — *Примеч. пер.*

54. (а) Это один высокосвязный компонент. (Кстати, этот граф представляет собой *реберный граф* двудольного графа, в котором одна часть соответствует первым буквам {A, C, D, F, G, ..., W}, а вторая — последним буквам {A, C, D, E, H, ..., Z}.)

(б) Вершина WY изолирована. Прочие вершины с нулевой входящей степенью, а именно FL, GA, PA, UT, WA, WI и WV, образуют сильные компоненты сами по себе; все они предшествуют гигантскому сильному компоненту, за которым следуют оставшиеся одновершинные сильные компоненты с нулевой исходящей степенью: AZ, DE, KY, ME, NE, NH, NJ, NY, OH, TX.

(в) Теперь сильный компонент {GU} предшествует {UT}; NH, OH, PA, WA, WI и WV присоединяются к гигантскому компоненту; {FM} предшествует ему; {AE} и {WY} следуют за ним.

55. $\binom{N}{2} - \binom{n_1}{2} - \cdots - \binom{n_k}{2}$, где $N = n_1 + \cdots + n_k$.

56. Истинно. Заметим, что J_n прост, но не соответствует ни одному мультиграфу.

57. Ложно — в связном ориентированном графе $u \longrightarrow w \longleftarrow v$. (Но u и v находятся в одном и том же *сильно связанном* компоненте тогда и только тогда, когда $d(u,v) < \infty$ и $d(v,u) < \infty$; см. раздел 2.3.4.2.)

58. Каждый компонент представляет собой цикл, порядок которого не менее (а) 3 и (б) 1.

59. (а) Для индукции по n можно воспользоваться простой сортировкой вставкой. Предположим, что $v_1 \longrightarrow \cdots \longrightarrow v_{n-1}$. Тогда либо $v_n \longrightarrow v_1$, либо $v_{n-1} \longrightarrow v_n$, либо $v_{k-1} \longrightarrow v_n \longrightarrow v_k$, где k — наименьшее такое, что $v_n \longrightarrow v_k$. [L. Rédei, *Acta litterarum ac scientiarum* **7** (Szeged, 1934), 39–43.]

(б) 15: 01234, 02341, 02413 и их циклические сдвиги. [Количество таких ориентированных путей всегда нечетно; см. T. Szele, *Matematikai és Fizikai Lapok* **50** (1943), 223–256.]

(в) Да. (По индукции: если существует только одно место для вставки v_n, как в п. (а), то турнир транзитивен.)

60. Пусть $A = \{x \mid u \longrightarrow x\}$, $B = \{x \mid x \longrightarrow v\}$, $C = \{x \mid v \longrightarrow x\}$. Если $v \notin A$ и $A \cap B = \emptyset$, то мы имеем $|A| + |B| = |A \cup B| \le n-2$, поскольку $u \notin A \cup B$ и $v \notin A \cup B$. Но $|B| + |C| = n-1$; следовательно, $|A| < |C|$. [H. G. Landau, *Bull. Math. Biophysics* **15** (1953), 148.]

61. $1 \longrightarrow 1$, $1 \longrightarrow 2$, $2 \longrightarrow 2$; тогда $A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ и $A^k = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}$ для всех целых чисел k.

62. (а) Предположим, что вершинами являются $\{1, \ldots, n\}$. Каждый из $n!$ членов $a_{1p_1} \ldots a_{np_n}$ в разложении перманента равен количеству остовных перестановочных ориентированных графов, имеющих дуги $j \longrightarrow p_j$. (б) Аналогичные рассуждения показывают, что $\det A$ равен количеству четных остовных перестановочных ориентированных графов минус количество нечетных остовных перестановочных ориентированных графов. [См. F. Harary, *SIAM Review* **4** (1962), 202–210, где перестановочные ориентированные графы названы "линейными подграфами".]

63. Пусть v — любая вершина. Если $g = 2t+1$, как минимум $d(d-1)^{k-1}$ вершин x удовлетворяет условию $d(v,x) = k$ для $1 \le k \le t$. Если $g = 2t+2$ и v' — любой сосед v, имеется также как минимум $(d-1)^t$ вершин x, для которых $d(v,x) = t+1$ и $d(v',x) = t$.

64. Для достижения нижней границы в упр. 63 *каждая* вершина v должна иметь степень d и все d соседей v должны быть смежны с остальными $d-1$ вершинами. В действительности этот граф представляет собой $K_{d,d}$.

65. (а) Согласно ответу к упр. 63, граф G должен быть регулярным со степенью d и должен быть ровно один путь длиной ≤ 2 между любыми двумя различными вершинами.

(б) Можно взять $\lambda_1 = d$ с $x_1 = (1 \ldots 1)^T$. Все другие собственные векторы удовлетворяют условию $Jx_j = (0 \ldots 0)^T$; следовательно, $\lambda_j^2 + \lambda_j = d-1$ для $1 < j \le N$.

(в) Если $\lambda_2 = \cdots = \lambda_m = (-1 + \sqrt{4d-3})/2$ и $\lambda_{m+1} = \cdots = \lambda_N = (-1 - \sqrt{4d-3})/2$, должно выполняться $m-1 = N-m$. При этом значении мы находим $\lambda_1 + \cdots + \lambda_N = d - d^2/2$.

(г) Если $4d-3=s^2$ и m такие, как в п. (в), собственные значения суммируются до

$$\frac{s^2+3}{4}+(m-1)\frac{s-1}{2}-\Bigl(\frac{(s^2+3)^2}{16}+1-m\Bigr)\frac{s+1}{2},$$

что равно 15/32 плюс кратное s. Следовательно, s должно быть делителем 15.

[Эти результаты были получены А. Д. Хоффманом (A. J. Hoffman) и Р. Р. Синглтоном (R. R. Singleton), *IBM J. Research and Development* **4** (1960), 497–504, которые также доказали, что граф для $d=7$ единственный.]

66. Обозначим 50 вершин как $[a,b]$ и (a,b) для $0\le a,b<5$, и определим три типа ребер, используя арифметику по модулю 5:

$$[a,b]\text{ — }[a+1,b];\qquad (a,b)\text{ — }(a+2,b);\qquad (a,b)\text{ — }[a+bc,c]\quad \text{для } 0\le a,b,c<5.$$

[См. W. G. Brown, *Canadian J. Math.* **19** (1967), 644–648; *J. London Math. Soc.* **42** (1967), 514–520. Без ребер первых двух типов граф имеет обхват 6 и соответствует геометрической сети из упр. 51, с использованием ортогонального массива из ответа к упр. 19.]

67. Некоторые возможности были рассмотрены Майклом Ашбахером (Michael Aschbacher) в *Journal of Algebra* **19** (1971), 538–540.

68. Если G имеет s автоморфизмов, то он имеет $n!/s$ матриц смежности, поскольку имеется s перестановочных матриц P, таких, что $P^{-}AP=A$.

69. Сначала установите IDEG(v) $\leftarrow 0$ для всех вершин v. Затем выполните (31) для всех v, а также установите $u\leftarrow$ TIP(a) и IDEG(u) $\leftarrow$ IDEG(u) $+1$ во второй строке этого мини-алгоритма.

Чтобы сделать что-то "для всех v", используя SGB-формат, сначала установите $v\leftarrow$ VERTICES(g); затем, пока $v<$ VERTICES(g) $+$ N(g), выполняйте операцию и устанавливайте $v\leftarrow v+1$.

70. Шаг B1 выполняется однократно (но требует $O(n)$ единиц времени). Шаги (B2, B3, ..., B8) выполняются соответственно $(n+1,n,n,m+n,m,m,n)$ раз, каждый со стоимостью $O(1)$.

71. Возможны различные варианты. Здесь мы используем 32-битовые указатели, все относительно символьного адреса Pool, который лежит в Data_Segment. Приведенное далее объявление предоставляет один способ установления соглашений для работы с базовыми структурами данных SGB.

```
VSIZE IS 32 ;ASIZE IS 20            Размеры узла.
ARCS IS 0 ;COLOR IS 8 ;LINK IS 12   Смещения полей вершины.
TIP IS 0 ;NEXT IS 4                 Смещения полей дуги.

arcs GREG Pool+ARCS ;color GREG Pool+COLOR ;link GREG Pool+LINK
tip GREG Pool+TIP ;next GREG Pool+NEXT
u GREG ;v GREG ;w GREG ;s GREG ;a GREG ;mone GREG -1

AlgB  BZ     n,Success            Выход, если граф нулевой.
      MUL    $0,n,VSIZE           B1. Инициализация.
      ADDU   v,v0,$0              v ← v0 + n.
      SET    w,v0                 w ← v0.
1H    STT    mone,color,w         COLOR(w) ← −1.
      ADDU   w,w,VSIZE            w ← w + 1.
      CMP    $0,w,v
      PBNZ   $0,1B                Повторять, пока не достигнем w = v.
```

```
0H      SUBU   w,w,VSIZE    w ← w − 1.
3H      LDT    $0,color,w   B3. Окраска w при необходимости.
        PBNN   $0,2F        Переход к B2, если COLOR(w) ≥ 0.
        STCO   0,link,w     COLOR(w) ← 0, LINK(w) ← Λ.
        SET    s,w          s ← w.
4H      SET    u,s          B4. Стек ⇒ u. Установить u ← s.
        LDTU   s,link,s     s ← LINK(s).
        LDT    $1,color,u
        NEG    $1,1,$1      $1 ← 1 − COLOR(u).
        LDTU   a,arcs,u     a ← ARCS(u).
5H      BZ     a,8F         B5. Завершено с u? Переход к B8, если a = Λ.
5H      LDTU   v,tip,a      v ← TIP(a).
6H      LDT    $0,color,v   B6. Обработка v.
        CMP    $2,$0,$1     (Здесь программа немного хитрее.)
        PBZ    $2,7F        Переход к B7, если COLOR(v) = 1 − COLOR(u).
        BNN    $0,Failure   Неудача, если COLOR(v) = COLOR(u).
        STT    $1,color,v   COLOR(v) ← 1 − COLOR(u).
        STTU   s,link,v     LINK(v) ← s.
        SET    s,v          s ← v.
7H      LDTU   a,next,a     B7. Цикл по a. Установить a ← NEXT(a).
        PBNZ   a,5B         К B5, если a ≠ Λ.
8H      PBNZ   s,4B         B8. Стек не пустой? Переход к B4, если s ≠ Λ.
2H      CMP    $0,w,v0      B2. Выполнено?
        PBNZ   $0,0B        Если w ≠ v0, уменьшить w и перейти к шагу B3.
Success LOC    @            (Успешное завершение.)  ▮
```

72. (а) Ясно, что это условие остается инвариантом при вхождении вершин в стек и выходе из него.

(б) Вершина v была окрашена, но еще не исследована, поскольку соседи каждой исследованной вершины имеют правильный цвет.

(в) Непосредственно перед установкой $s \leftarrow v$ на шаге B6 установите PARENT$(v) \leftarrow u$, где PARENT — новое сервисное поле. А непосредственно перед неудачным завершением алгоритма на этом шаге выполните следующее: "многократно выводить NAME(u) и устанавливать $u \leftarrow$ PARENT(u), пока не выполнится условие $u =$ PARENT(v); затем вывести NAME(u) и NAME(v)".

73. K_{10}. (A *random_graph*$(10, 100, 0, 1, 1, 0, 0, 0, 0, 0)$ представляет собой J_{10}.)

74. badness имеет исходящую степень 22; других вершин с исходящей степенью > 20 нет.

75. Параметры $(n_1, n_2, n_3, n_4, p, w, o)$ представляют собой соответственно (а) $(n, 0, 0, 0, -1, 0, 0)$; (б) $(n, 0, 0, 0, 1, 0, 0)$; (в) $(n, 0, 0, 0, 1, 1, 0)$; (г) $(n, 0, 0, 0, -1, 0, 1)$; (д) $(n, 0, 0, 0, 1, 0, 1)$; (е) $(n, 0, 0, 0, 1, 1, 1)$; (ж) $(m, n, 0, 0, 1, 0, 0)$; (з) $(m, n, 0, 0, 1, 2, 0)$; (и) $(m, n, 0, 0, 1, 3, 0)$; (к) $(m, n, 0, 0, -1, 0, 0)$; (л) $(m, n, 0, 0, 1, 3, 1)$; (м) $(n, 0, 0, 0, 2, 0, 0)$; (н) $(2, -n, 0, 0, 1, 0, 0)$.

76. Да, например из C_1 и C_2 в ответе к упр. 75, (в). (Но петли невозможны при $p < 0$, поскольку дуги $x \longrightarrow y = x + k\delta$ генерируются для $k = 1, 2, \ldots,$ пока y не окажется за пределами диапазона или пока не выполнится условие $y = x$.)

77. Предположим, что x и y представляют собой вершины, такие, что $d(x, y) > 2$. Таким образом, $x \not\longrightarrow y$; и если v представляет собой любую другую вершину, то мы должны иметь либо $v \not\longrightarrow x$, либо $v \not\longrightarrow y$. Эти факты дают путь длиной не более 3 в $\overline{G}$ между любыми двумя вершинами u и v.

78. (а) Количество ребер, $\binom{n}{2}/2$, должно быть целым. Наименьшими примерами являются K_0, K_1, P_4, C_5 и ⩛.

(б) Если q — любое нечетное число, то мы имеем u — v тогда и только тогда, когда $\varphi^q(u) \not\perp \varphi^q(v)$. Следовательно, φ^q не может иметь ни двух фиксированных точек, ни содержать цикл длиной 2.

(в) Такая перестановка V определяет также перестановку $\widehat{\varphi}$ ребер K_n, отображающую $\{u,v\} \mapsto \widehat{\varphi}(\{u,v\}) = \{\varphi(u), \varphi(v)\}$, и легко увидеть, что все циклы длиной $\widehat{\varphi}$ четны. Если $\widehat{\varphi}$ имеет t циклов, мы получим 2^t самодополнительных графов, окрашивая ребра каждого цикла чередующимися цветами.

(г) В этом случае φ имеет единственную фиксированную точку v, и $G' = G \backslash v$ является самодополнительным. Предположим, что φ имеет r циклов в дополнение к (v); тогда $\widehat{\varphi}$ имеет r циклов, включая ребра, касающиеся вершины v, и имеется 2^r способов расширить G' до графа G.

[*Ссылки*: H. Sachs, *Publicationes Mathematicæ* **9** (Debrecen, 1962), 270–288; G. Ringel, *Archiv der Mathematik* **14** (1963), 354–358.]

79. Решение 1, принадлежащее X. Саксу (H. Sachs), с $\varphi = (1\,2\,\ldots\,4k)$: пусть u — v, когда $u > v > 0$ и $u + v \bmod 4 \le 1$; а также 0 — v, когда $v \bmod 2 = 0$.

Решение 2, с $\varphi = (a_1\,b_1\,c_1\,d_1)\ldots(a_k\,b_k\,c_k\,d_k)$, где $a_j = 4j-3$, $b_j = 4j-2$, $c_j = 4j-1$ и $d_j = 4j$: пусть 0 — b_j — a_j — c_j — d_j — 0 для $1 \le j \le k$ и a_i — a_j — b_i — d_j — c_i — c_j — d_i — b_j — a_i для $1 \le i < j \le k$.

80. (Решение Г. Рингеля (G. Ringel).) Пусть φ та же, что и в ответе к упр. 79, решение 2. Пусть E_0 состоит из $3k$ ребер b_j — a_j — c_j — d_j для $1 \le j \le k$; E_1 представляет собой $8\binom{k}{2}$ ребер между $\{a_i, b_i, c_i, d_i\}$ и $\{b_j, d_j\}$ для $1 \le i < j \le k$; а E_2 — $8\binom{k}{2}$ ребер между $\{a_i, b_i, c_i, d_i\}$ и $\{a_j, c_j\}$ для $1 \le i < j \le k$. В случае (а) $E_0 \cup E_1$ дает диаметр 2; $E_0 \cup E_2$ дает диаметр 3. Случай (б) аналогичен, но мы добавляем $2k$ ребер b_j — 0 — d_j к E_1 и a_j — 0 — c_j к E_2.

81. $\vec{C_3}$, $\vec{K_3}$, $D =$, и $D^T =$. (Ориентированный граф D^T, *обратный* к ориентированному графу D, получается путем обращения направления его дуг. Имеется 16 неизоморфных простых ориентированных графов порядка 3 без циклов, 10 из которых самообратны, включая $\vec{C_3}$ и $\vec{K_3}$.)

82. (а) Истинно по определению. (б) Истинно: если каждая вершина имеет d соседей, каждое ребро u — v имеет $d-1$ соседей u — w и $d-1$ соседей w — v. (в) Истинно: $\{a_i, b_j\}$ имеет $m+n-2$ соседей для $0 \le i < m$ и $0 \le j < n$. (г) Ложно: $L(K_{1,1,2})$ имеет 5 вершин и 8 ребер. (д) Истинно. (е) Истинно: единственными несмежными ребрами являются $\{0,1\} \not\perp \{2,3\}$, $\{0,2\} \not\perp \{1,3\}$, $\{0,3\} \not\perp \{1,2\}$. (ж) Истинно для всех $n > 0$. (з) Ложно, за исключением случая, когда G не имеет изолированных вершин.

83. Это граф Петерсена. [A. Kowalewski, *Sitzungsberichte der Akademie der Wissenschaften in Wien*, Mathematisch-Nat. Klasse, Abteilung IIa, **126** (1917), 67–90.]

84. Да: положим $\varphi(\{a_u, b_v\}) = \{a_{(u+v) \bmod 3}, b_{(u-v) \bmod 3}\}$ для $0 \le u, v < 3$.

85. Пусть степенями вершин являются $\{d_1, \ldots, d_n\}$. Тогда G имеет $\frac{1}{2}(d_1 + \cdots + d_n)$ ребер, а $L(G)$ — $\frac{1}{2}(d_1(d_1-1) + \cdots + d_n(d_n-1))$. Таким образом, и G, и $L(G)$ имеют ровно по n ребер тогда и только тогда, когда $(d_1-2)^2 + \cdots + (d_n-2)^2 = 0$. Следовательно, упр. 58 дает искомый ответ. [См. V. V. Menon, *Canadian Math. Bull.* **8** (1965), 7–15.]

86. Если $G =$, то $\overline{G} =$ $= L(G)$.

87. (а) Да, легко. [В действительности Р. Л. Брукс (R. L. Brooks) доказал, что *каждый* связный граф с максимальной степенью вершин $d > 2$ является d-раскрашиваемым, за исключением полного графа K_{d+1}; см. *Proc. Cambridge Phil. Soc.* **37** (1941), 194–197.]

(б) Нет. По сути, имеется только один способ 3-раскрашивания ребер внешнего цикла длиной 5 на рис. 2, (д); но он приводит к конфликту с внутренним циклом длиной 5. [Петерсен доказал это в 1898 году.]

88. Один цикл, который не использует центральную вершину, и $(n-1)(n-2)$ циклов, которые это делают (а именно, по одному для каждой упорядоченной пары различных вершин на "ободе" колеса). При этом подсчете мы не учитываем C_0.

89. Обе части равны $\begin{pmatrix} A & O & O \\ O & B & O \\ O & O & C \end{pmatrix}$, $\begin{pmatrix} A & J & J \\ J & B & J \\ J & J & C \end{pmatrix}$, $\begin{pmatrix} A & J & J \\ O & B & J \\ O & O & C \end{pmatrix}$, $\begin{pmatrix} A & O & O \\ J & B & O \\ J & J & C \end{pmatrix}$, соответственно.

90. K_4 и $\overline{K_4}$; $K_{1,1,2}$ и $\overline{K_{1,1,2}}$; $K_{2,2} = C_4$ и $\overline{K_{2,2}}$; $K_{1,3}$ и $\overline{K_{1,3}}$; $K_1 \oplus K_{1,2}$ и его дополнение; все графы K_α являются сографами согласно (39). Не является таковым $P_4 = \overline{P_4}$. (Все связанные подграфы сографа имеют диаметр ≤ 2; сографом является W_5, но не W_6.)

91. (а) ; (б) ; (в) ; (г) ; (д) ; (е) ; (ж) . (В общем случае имеем $K_2 \triangle H = (K_2 \square H) \cup (K_2 \otimes \overline{H})$ и $K_2 \circ H = H$ — H. Таким образом, совпадение $K_2 \triangle H = K_2 \square H$ и $K_2 \circ H = K_2 \boxtimes H$ осуществляется тогда и только тогда, когда H является полным графом.)

Мнемотехника: используемые обозначения $G \square H$ и $G \boxtimes H$ отлично соответствуют диаграммам (а) и (в), как отмечал Я. Нешетржил (J. Nesetřil), *Lecture Notes in Comp. Sci.* **118** (1981), 94–102. Его аналогичная рекомендация писать $G \times H$ для (б) не менее привлекательна; но здесь мы ею не пользуемся, так как сотни авторов применяют $G \times H$ для обозначения $G \square H$.

92. (а) ; (б) ; (в) ; (г) ; (д) .

93. $K_m \boxtimes K_n = K_m \circ K_n \cong K_{mn}$.

94. Нет; они являются порожденными подграфами $K_{26} \square K_{26} \square K_{26} \square K_{26} \square K_{26}$.

95. (а) $d_u + d_v$; (б) $d_u d_v$; (в) $d_u d_v + d_u + d_v$; (г) $d_u(n - d_v) + (m - d_u)d_v$; (д) $d_u n + d_v$.

96. (а) $A \square B = A \otimes I + I \otimes B$; (б) $A \boxtimes B = A \square B + A \otimes B$; (в) $A \triangle B = A \otimes J + J \otimes B - 2A \otimes B$; (г) $A \circ B = A \otimes J + I \otimes B$. (Формулы (а), (б) и (г) определяют произведения произвольных ориентированных графов и мультиграфов. Формула (в) справедлива в общем случае для простых ориентированных графов; однако, когда A и B содержат значения > 1, в результирующей матрице могут наблюдаться отрицательные элементы.)

Историческая справка: прямое произведение матриц часто называется произведением Кронекера, поскольку К. Хенсель (K. Hensel) [*Crelle* **105** (1889), 329–344] говорил, что слышал о нем на лекциях Кронекера; однако Кронекер о нем никогда ничего не публиковал. Первое известное появление его в печати — в статье И. Г. Зехфусса (J. G. Zehfuss) [*Zeitschrift für Math. und Physik* **3** (1858), 298–301], который доказал, что $\det(A \otimes B) = (\det A)^n (\det B)^m$ при $m = m'$ и $n = n'$. Основные формулы $(A \otimes B)^T = A^T \otimes B^T$, $(A \otimes B)(A' \otimes B') = AA' \otimes BB'$ и $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ были открыты А. Гурвицем (A. Hurwitz) [*Math. Annalen* **45** (1894), 381–404].

97. Операции над матрицами смежности доказывают, что $(G \oplus G') \square H = (G \square H) \oplus (G' \square H)$; $(G \oplus G') \boxtimes H = (G \boxtimes H) \oplus (G' \boxtimes H)$; $(G \oplus G') \circ H = (G \circ H) \oplus (G' \circ H)$. Поскольку $G \square H \cong H \square G$, $G \otimes H \cong H \otimes G$ и $G \boxtimes H \cong H \boxtimes G$, мы также получаем праводистрибутивные законы $G \square (H \oplus H') \cong (G \square H) \oplus (G \square H')$; $G \otimes (H \oplus H') \cong (G \otimes H) \oplus (G \otimes H')$; $G \boxtimes (H \oplus H') \cong (G \boxtimes H) \oplus (G \boxtimes H')$. Лексикографическое произведение удовлетворяет уравнению $\overline{G \circ H} = \overline{G} \circ \overline{H}$; также $K_m \circ H = H$ — $\cdots$ — H, следовательно, $K_m \circ \overline{K}_n = K_{n,\ldots,n}$. Кроме того $G \circ K_n = G \boxtimes K_n$; $K_m \square K_n = \overline{K_m \otimes K_n} = L(K_{m,n})$.

98. Имеется kl компонентов (в силу закона дистрибутивности из предыдущего упражнения и того факта, что $G \square H$ и $G \boxtimes H$ связны, когда связны G и H).

99. Каждый путь от (u, v) до (u', v') в $G \square H$ должен использовать как минимум $d_G(u, u')$ "G-шагов" и как минимум $d_H(v, v')$ "H-шагов"; и этот минимум достижим. Аналогичные рассуждения показывают, что $d_{G \boxtimes H}((u, v), (u', v')) = \max(d_G(u, u'), d_H(v, v'))$.

100. Если G и H связны и если каждый из них имеет как минимум две вершины, $G \otimes H$ является несвязным тогда и только тогда, когда G и H двудольны. Часть "тогда"

проста; и обратно, если существует нечетный цикл в G, можно добраться от (u,v) до (u',v') следующим образом: сначала перейти к (u'',v'), где u'' — любая подходящая вершина G. Затем пройти четное число шагов в G от u'' до u', при чередовании в H между v' и одним из ее соседей. [P. M. Weichsel, *Proc. Amer. Math. Soc.* **13** (1962), 47–52.]

101. Выберем вершины u и v с максимальной степенью. Тогда $d_u + d_v = d_u d_v$ согласно упр. 95; так что либо $G = H = K_1$, либо $d_u = d_v = 2$. В последнем случае $G = P_m$ или C_m и $H = P_n$ или C_n. Но граф $G \square H$ связный, так что G или H должен быть не двудольным; скажем, это G. Тогда $G \square H$ является не двудольным, так что H также должен быть не двудольным; таким образом, $G = C_m$ и $H = C_n$, где и m, и n нечетны. Кратчайший нечетный цикл в $C_m \square C_n$ имеет длину $\min(m,n)$; в $C_m \otimes C_n$ он имеет длину $\max(m,n)$; следовательно, $m = n$. И наоборот, если $n \geq 3$ нечетно, то мы имеем $C_n \square C_n \cong C_n \otimes C_n$ при изоморфизме, который отображает $(u,v) \mapsto ((u+v) \bmod n, (u-v) \bmod n)$ для $0 \leq u, v < n$. [D. J. Miller, *Canadian J. Math.* **20** (1968), 1511–1521.]

102. $P_m \boxtimes P_n$. (Это планарный граф только тогда, когда $\min(m,n) \leq 2$ или $m = n = 3$.)

103.

1	2	3	4	5	7
2	1	3	4	6	8
3	1	2	5	6	8
4	1	2	5	6	
5	3	4	1	7	
6	2	3	4		
7	5	1			
8	2	3			

1	2	3	4	5	6	7	8	9
2	1	3	4	6	8	9		
3	1	2	5	6	8	9		
4	1	2	5	7				
5	3	4	1	7				
6	2	3	1	7				
7	4	5	6	1				
8	2	3	1	9				
9	8	2	3	1				

104. Ребра должны создаваться в несколько непрямом порядке, чтобы поддерживать форму таблицы. Переменные i и r разграничивают доступные строки в столбце t. Например, вторая часть упр. 103 начинается с $i \leftarrow 1$, $t \leftarrow 8$, $r \leftarrow 1$; затем $9 — 1$, $i \leftarrow 2$, $t \leftarrow 6$, $r \leftarrow 3$; затем $9 — 3$, $9 — 2$, $i \leftarrow 4$, $t \leftarrow 4$, $r \leftarrow 8$; а затем $9 — 8$.

105. Заметим, что $d_k \geq k$ тогда и только тогда, когда $c_k \geq k$. Когда $d_k \geq k$, мы имеем

$$c_1 + \cdots + c_k \;=\; k^2 + \min(k, d_{k+1}) + \min(k, d_{k+2}) + \cdots + \min(k, d_n);$$

следовательно, условие $d_1 + \cdots + d_k \leq c_1 + \cdots + c_k - k$ эквивалентно условию

$$d_1 + \cdots + d_k \leq f(k), \quad \text{где } f(k) = k(k-1) + \min(k, d_{k+1}) + \cdots + \min(k, d_n). \qquad (*)$$

Если $k \geq s$, имеем $f(k+1) - f(k) = 2k - d_{k+1} \geq d_{k+1}$; следовательно, $(*)$ выполняется для $1 \leq k \leq n$ тогда и только тогда, когда оно выполняется для $1 \leq k \leq s$. Условие $(*)$ было открыто П. Эрдешем (P. Erdős) и Т. Галлаи (T. Gallai) [*Matematikai Lapok* **11** (1960), 264–274]. Его необходимость очевидна, если рассмотреть ребра между $\{1, \ldots, k\}$ и $\{k+1, \ldots, n\}$.

Пусть $a_k = d_1 + \cdots + d_k - c_1 - \cdots - c_k + k$, и предположим, что мы достигли $a_k > 0$ на шаге H2 для некоторого $k \leq s$. Пусть A_j, C_j, D_j, N и S представляют собой числа, соответствующие a_j, c_j, d_j, n и s *перед* шагами H3 и H4; таким образом, $N = n + 1$, $D_j = d_j + (0$ или $1)$, и т. д. Мы хотим доказать, что $A_K > 0$ для некоторого $K \leq S$.

Шаги H3 и H4 должны удалять строку N и остающиеся q нижних ячеек в столбце t, для некоторого $t \geq S$ и $q > 0$, вместе с крайними справа ячейками в строках с 1 по p. Если $p > 0$, имеем $C_{t+1} = p$. Пусть $r = D_N = p + q$ и $u = C_t$. Заметим, что $D_j = t$ для $p < j \leq u$ и $C_j = N$ для $1 \leq j \leq r$; также $A_j = a_j$ для $1 \leq j \leq p$.

Если k минимально, имеем $1 \leq a_k \leq d_k - c_k + 1$; следовательно, $c_k \leq d_k$. Если $D_k > t$, то $k \leq p$ и $A_k = a_k$. Если $D_k < t$, отсюда следует, что $A_k = a_k + r - \min(k, r) \geq a_k$, поскольку $k \leq D_k$. Таким образом, можно положить, что $D_k = t$.

Предположим, что $t > S$; следовательно, $u \leq S$. Для $k < j \leq u$ мы имеем $d_j \geq D_j - 1 = t - 1 \geq d_k - 1 \geq c_k - 1 \geq c_j - 1$. Таким образом, $a_u \geq a_k > 0$. Но $A_u = a_u$,

поскольку $r \le u \le S < t$. Следовательно, можно считать, что $t = S$. Предположим, что $k < t$; тогда $c_k = d_k = t$, поскольку $S \le c_k \le d_k \le t$. Но $r = t$ приводит к $c_k = N - 1$ и к противоречию; а $r < t$ приводит к $u = t$, откуда следует, что $A_t > A_{t-1} = a_{t-1} - 1 \ge 0$.

(Глубокий вздох.) Хорошо; мы свели задачу к случаям $k = t = S$. Следовательно, $t = s \le c_t \le d_t \le D_t = t$, и мы имеем $a_t = a_{t-1} + 1$. Поэтому $a_{t-1} = 0$.

В действительности можно показать по индукции по $t - j$, что $a_j = 0$ для $p \le j < t$: если $a_{j+1} = 0$, то $0 \ge a_j = c_{j+1} - t - 1 \ge q - 1 \ge 0$, поскольку $c_{j+1} \ge t + q$ при $p \le j < t - 1$.

Если $p < t - 1$, эти рассуждения доказывают, что $q = 1$ и $c_r = N - 1 = t + 1$. Мы заключаем, что, независимо от p, должно выполняться $q = 1$, $N = t + 2$, $D_j = t + 1$ для $1 \le j \le p$, $D_j = t$ для $p < j \le t + 1$ и $D_N = p + 1$. Алгоритм H в действительности заменяет эту "хорошую" последовательность "плохой"; но $D_1 + \cdots + D_N = 2p + t(t + 1) + 1$ нечетно.

106. Ложно в тривиальных случаях, когда $d \le 1$ и $n \ge d+2$. В противном случае истинно: действительно, первые $n - 1$ ребер, сгенерированные на шаге H4, не содержат циклов, так что они образуют остовное дерево.

107. Перестановка φ из упр. 78 отображает вершину со степенью d в вершину со степенью $n - 1 - d$. А φ^2 представляет собой автоморфизм, который объединяет в пары две вершины с одинаковыми степенями, за исключением возможной фиксированной точки со степенью $(n - 1)/2$.

(И обратно, немного запутанное расширение алгоритма H будет строить самодополнительный граф из каждой графической последовательности, которая удовлетворяет этим условиям, давая $d_{(n-1)/2} = (n - 1)/2$ при нечетном n. См. К. Р. Д. Клафам (C. R. J. Clapham) и Д. И. Кляйтман (D. J. Kleitman), *J. Combinatorial Theory* **B20** (1976), 67–74.)

108. Мы можем считать, что $d_1^+ \ge \cdots \ge d_n^+$; входящие степени d_k^- не должны находиться в некотором определенном порядке. Применим алгоритм H к последовательности $d_1 \ldots d_n = d_1^+ \ldots d_n^+$, но со следующими изменениями. Шаг H2 становится следующим: "[Выполнено?] Завершить работу алгоритма успешно, если $d_1 = n = 0$; завершить работу алгоритма неудачно, если $d_1 > n$." На шаге H3 заменить "$j \leftarrow d_n$" на "$j \leftarrow d_n^-$" и завершить работу алгоритма неудачно, если $j > c_1$. На шаге H4 заменить "установить ... и установить" на "если $j > 0$, установить $m \leftarrow c_t$, создать дугу $m \longrightarrow n$ и установить"; и установить $n \leftarrow n - 1$ непосредственно перед возвратом к шагу H2. Рассуждения, подобные лемме M и следствию H, обосновывают этот подход.

(В упр. 7.2.1.4–57 доказывается, что такие ориентированные графы существуют тогда и только тогда, когда $d_1^- + \cdots + d_n^- = d_1^+ + \cdots + d_n^+$ и $d_1^- \ldots d_n^- = \{d'_1, \ldots, d'_n\}$, где $d'_1 \ge \cdots \ge d'_n$ и $d'_1 \ldots d'_n$ мажоризируются сопряженным разбиением $c_1 \ldots c_n = (d_1^+ \ldots d_n^+)^T$. Вариант, когда петли $v \longrightarrow v$ запрещены, оказывается более сложным; см. Д. Р. Фалкерсон (D. R. Fulkerson), *Pacific J. Math.* **10** (1960), 831–836.)

109. Это то же самое, что и упр. 108, если положить $d_k^+ = d_k[k \le m]$ и $d_k^- = d_k[k > m]$.

110. Имеется p вершин степени $d = d_1$ и q вершин степени $d - 1$, где $p + q = n$.

Случай 1, $d = 2k + 1$. Строим u — v, когда $(u - v) \bmod n \in \{2, 3, \ldots, k + 1, n - k - 1, \ldots, n - 3, n - 2\}$; добавим также $p/2$ ребер 1 — 2, 3 — 4, ..., $(p-1)$ — p.

Случай 2, $d = 2k > 0$. Строим u — v, когда $(u - v) \bmod n \in \{2, 3, \ldots, k, n - k, \ldots, n - 3, n - 2\}$; добавим также ребра 1 — 2, ..., $(q-1)$ — q и путь или цикл ($q = 0$? n: q) — $(q+1)$ — $\cdots$ — $(n-1)$ — n. [Д. Л. Ванг (D. L. Wang) и Д. И. Кляйтман (D. J. Kleitman) в *Networks* **3** (1973), 225–239, доказали, что такие графы являются высокосвязными.]

111. Допустим, что $N = n + n'$ и $V' = \{n+1, \ldots, N\}$. Мы хотим построить $e_k = d - d_k$ ребер между k и V' и дополнительные ребра в V', так что каждая вершина V' имеет степень d. Пусть $s = e_1 + \cdots + e_n$. Это возможно, только если (i) $n' \ge \max(e_1, \ldots, e_n)$; (ii) $n'd \ge s$; (iii) $n'd \le s + n'(n' - 1)$; и (iv) $(n + n')d$ четно.

Такие ребра существуют, когда n' удовлетворяет (i)–(iv). Первые s подходящих ребер между V и V' могут быть созданы путем циклического выбора конечных точек $(n{+}1, n{+}2, \ldots, n{+}n', n{+}1, \ldots)$ в связи с (i). Этот процесс назначает каждой вершине из V' либо $\lfloor s/n' \rfloor$, либо $\lceil s/n' \rceil$ ребер; согласно (ii) мы имеем $\lceil s/n' \rceil \le d$ и $d - \lfloor s/n' \rfloor \le n' - 1$ в соответствии с (iii). Следовательно, дополнительные ребра, необходимые внутри V', могут быть построены с применением упр. 110 и (iv).

Выбор $n' = n$ всегда работает. И наоборот, если $G = K_n(V) \setminus \{1 \text{—} 2\}$, условие (iii) требует $n' \ge n$ при $n \ge 4$. [P. Erdös and P. Kelly, *AMM* **70** (1963), 1074–1075.]

112. Единственным наилучшим треугольником в *miles* является

`Saint Louis, MO` $\overset{748}{\text{—}}$ `Toronto, ON` $\overset{746}{\text{—}}$ `Winston-Salem, NC` $\overset{748}{\text{—}}$ `Saint Louis, MO`.

113. Согласно закону Мерфи (Murphy), в ней n строк и m столбцов; так что это матрица размером $n \times m$, а не $m \times n$.

114. Петля в мультиграфе представляет собой ребро $\{a, a\}$ с повторяющимися вершинами, а мультиграф представляет собой 2-однородный гиперграф. Таким образом, мы должны позволить матрице инциденций обобщенного гиперграфа иметь элементы, превосходящие 1, когда ребро содержит вершину более одного раза. (Педант назвал бы это "мультигиперграфом".) С учетом этих соображений матрица инциденций и двудольный мультиграф, соответствующий (26), представляют собой

$$\begin{pmatrix} 210000 \\ 011100 \\ 001122 \end{pmatrix};$$

115. Элемент в строке e и столбце f матрицы B^TB представляет собой $\sum_v b_{ve} b_{vf}$; так что B^TB равно $2I$ плюс матрица смежности реберного графа $L(G)$. Аналогично BB^T представляет собой D плюс матрица смежности графа G, где D — диагональная матрица с $d_{vv} =$ степень v. (См. упр. 2.3.4.2–18, 19 и 20.)

116. Обобщая (38), $\overline{K_{m,n}^{(r)}} = K_m^{(r)} \oplus K_n^{(r)}$ для всех $r \ge 1$.

117. Неизоморфные мультимножества одновершинных ребер для $m = 4$ и $V = \{0, 1, 2\}$ представляют собой $\{\{0\}, \{0\}, \{0\}, \{0\}\}$, $\{\{0\}, \{0\}, \{0\}, \{1\}\}$, $\{\{0\}, \{0\}, \{1\}, \{1\}\}$ и $\{\{0\}, \{0\}, \{1\}, \{2\}\}$. В общем случае ответом является число разбиений m на не более n частей, а именно $\left|{m+n \atop n}\right|$, с использованием обозначений, поясняемых в разделе 7.2.1.4. (Конечно, причин представлять разбиения как 1-однородные гиперграфы нет никаких — разве что для решения таких странных упражнений.)

118. Пусть d — сумма степеней вершин. Соответствующий двудольный граф представляет собой лес с $m + n$ вершинами, d ребрами и p компонентами. Следовательно, согласно теореме 2.3.4.1A $d = m + n - p$.

119. В этом случае имеется дополнительное ребро, содержащее все семь вершин.

120. Было бы логично говорить, что (гипер)дуги представляют собой произвольные последовательности вершин или последовательности разных вершин. Но большинство авторов, похоже, определяют гипердуги как $A \longrightarrow v$, где A — неупорядоченное множество вершин. Когда наконец будет найдено наилучшее определение, вероятнее всего, окажется, что оно имеет наиболее важные практические приложения.

121. $\chi(H) = |F| - \alpha(I(H)^T)$ представляет собой размер наименьшего покрытия V множествами F.

122. (a) Можно убедиться, что имеется всего семь трехэлементных покрытий, а именно вершин ребра; так что всего имеется семь четырехэлементных независимых множеств, а именно дополнений ребра. Мы не можем выполнить 2-раскрашивание гиперграфа, поскольку один цвет должен использоваться 4 раза, а три другие вершины должны быть

ребром. (Гиперграф (56), по сути, представляет собой проективную плоскость с семью точками и семью прямыми.)

(б) Поскольку мы рассматриваем дуальный граф, будем называть вершины и ребра графа Петерсена "точками" и "прямыми"; тогда вершины и ребра дуального графа являются соответственно прямыми и точками. Окрасим в красный цвет пять прямых, которые соединяют внешнюю точку с внутренней. Остальные десять прямых независимы (они не содержат все три прямые, касающиеся любой точки); так что их можно окрасить в зеленый цвет. Никакое множество из одиннадцати прямых не может быть независимым, поскольку никакие четыре прямые не могут касаться всех десяти точек. (Таким образом, граф, дуальный графу Петерсена, представляет собой двудольный гиперграф, несмотря на тот факт, что он содержит циклы длиной 5.)

123. Оно соответствует латинским квадратам $n \times n$, элементы которого представляют собой цвета вершин.

124. Граф легко раскрасить в четыре цвета. Если бы он был раскрашиваемым в три цвета, то должно быть по четыре вершины каждого цвета, поскольку никакие пять вершин не могут быть независимыми. Тогда два противоположных угла должны быть окрашены в один и тот же цвет, что быстро приводит к противоречию.

125. Граф Чватала является наименьшим таким графом при $g = 4$. Г. Бринкманн (G. Brinkmann) нашел наименьший граф для $g = 5$. Он имеет 21 вершину a_j, b_j, c_j для $0 \le j < 7$ с ребрами a_j — a_{j+2}, a_j — b_j, a_j — b_{j+1}, b_j — c_j, b_j — c_{j+2}, c_j — c_{j+3} и индексами по модулю 7. М. Мерингер (M. Meringer) показал, что при $g > 5$ граф должен иметь как минимум 35 вершин. Б. Грюнбаум (B. Grünbaum) предположил, что значение g может быть произвольно большим; однако никаких других построений пока не известно. [See *AMM* **77** (1970), 1088–1092; *Graph Theory Notes of New York* **32** (1997), 40–41.]

126. Когда m и n четны, и C_m, и C_n являются двудольными, и выполнить 4-раскрашивание легко. В противном случае 4-раскрашивание невозможно. Когда $m = n = 3$, согласно упр. 93, оптимальным является 9-раскрашивание. Когда $m = 3$ и $n = 4$ или 5, независимы не более двух вершин; и легко найти оптимальное 6- или 8-раскрашивание. В противном случае мы получаем 5-раскрашивание, окрашивая вершину (j, k) цветом $(a_j + 2b_k) \bmod 5$, где периодические последовательности $\langle a_j \rangle$ и $\langle b_k \rangle$ (имеющие периоды m и n соответственно) обладают тем свойством, что $a_j - a_{j+1} \equiv \pm 1$ и $b_k - b_{k+1} \equiv \pm 1$ для всех j и k. [K. Vesztergombi, *Acta Cybernetica* **4** (1979), 207–212.]

127. (а) Результат истинен при $n = 1$. В противном случае положим $H = G \setminus v$, где v является произвольной вершиной. Тогда $\overline{H} = \overline{G} \setminus v$ и мы имеем $\chi(H) + \chi(\overline{H}) \le n$ по индукции. Ясно, что $\chi(G) \le \chi(H) + 1$ и $\chi(\overline{G}) \le \chi(\overline{H}) + 1$; так что никаких проблем не возникает, если только во всех трех случаях не достигается равенство. Но этого не может случиться; отсюда вытекает, что $\chi(H) \le d$ и $\chi(\overline{H}) \le n - 1 - d$, где d является степенью v в G. [E. A. Nordhaus and J. W. Gaddum, *AMM* **63** (1956), 175–177.]

Для достижения равенства положим $G = K_a \oplus \overline{K_b}$, где $ab > 0$ и $a + b = n$. Тогда мы имеем $\overline{G} = \overline{K_a}$ — K_b, $\chi(G) = a$ и $\chi(\overline{G}) = b + 1$. [Все графы, для которых выполняется равенство, были найдены Г.-И. Финком (H.-J. Finck), *Wiss. Zeit. der Tech. Hochschule Ilmenau* **12** (1966), 243–246.]

(б) k-раскрашивание G имеет как минимум $\lceil n/k \rceil$ вершин некоторого цвета; эти вершины образуют клику в $\overline{G}$. Следовательно, $\chi(G)\chi(\overline{G}) \ge \chi(G)\lceil n/\chi(G) \rceil \ge n$. Равенство достигается, когда $G = K_n$.

(Из (а) и (б) мы выводим, что $\chi(G) + \chi(\overline{G}) \ge 2\sqrt{n}$ и $\chi(G)\chi(\overline{G}) \le \frac{1}{4}(n+1)^2$.)

128. $\chi(G \,\square\, H) = \max(\chi(G), \chi(H))$. Очевидно, что это количество цветов является необходимым. И если функции $a(u)$ и $b(v)$ раскрашивают G и H цветами $\{0, 1, \ldots, k-1\}$, то мы можем раскрасить $G \,\square\, H$ следующим образом: $c(u, v) = (a(u) + b(v)) \bmod k$.

129. Полная строка или столбец (16 случаев); полная диагональ длиной 4 или более (18 случаев); 5-ячеечный шаблон $\{(x,y),(x-a,y-a),(x-a,y+a),(x+a,y-a),(x+a,y+a)\}$, где $a \in \{1,2,3\}$ $(36+16+4$ случаев); 5-ячеечный шаблон $\{(x,y),(x-a,y),(x+a,y),(x,y-a),(x,y+a)\}$, где $a \in \{1,2,3\}$ $(36+16+4$ случаев); шаблон, содержащий четыре из пяти таких ячеек, при пятой, лежащей за пределами доски $(24+32+24$ случаев); или 4-ячеечный шаблон $\{(x,y),(x+a,y),(x,y+a),(x+a,y+a)\}$, где $a \in \{1,3,5,7\}$ $(49+25+9+1$ случаев). Всего 310 максимальных клик, соответственно $(168,116,4,4,18)$ клик размером $(4,5,6,7,8)$.

130. Если граф G имеет p максимальных клик, а граф H имеет их q, то соединение G — H имеет pq максимальных клик, поскольку клики G — H представляют собой просто объединения клик из G и H. Кроме того, пустой граф $\overline{K_n}$ имеет n максимальных клик (а именно их одноэлементные множества).

Таким образом, полный k-дольный граф с размерами частей $\{n_1,\ldots,n_k\}$, будучи соединением пустых графов этих размеров, имеет $n_1\ldots n_k$ максимальных клик.

131. Считаем, что $n>1$. В полном k-дольном графе число $n_1\ldots n_k$ максимально, когда каждая часть имеет размер 3, за исключением, возможно, одной или двух частей размером 2. (См. упр. 7.2.1.4–68, (а).) Так что мы должны доказать, что $N(n)$ не может быть больше этого значения в *любом* графе.

Пусть $m(v)$ — количество максимальных клик, содержащих вершину v. Если $u \mathrel{-\!\!\!/} v$ и $m(u) \le m(v)$, построим граф G', который подобен графу G, с тем отличием, что u теперь смежна всем соседям v вместо своих прежних соседей. Каждая максимальная клика U в любом из этих графов принадлежит одному из трех классов.

i) $u \in U$; имеется $m(u)$ таковых в G и $m(v)$ в G'.
ii) $v \in U$; имеется $m(v)$ таковых как в G, так и в G'.
iii) $u \notin U$ и $v \notin U$; такие максимальные клики в G являются максимальными и в G'.

Следовательно, G' имеет как минимум столько же максимальных клик, сколько и G. А мы можем получить полный k-дольный граф путем соответствующего повторения описанного процесса.

[Это рассуждение П. Эрдеша (P. Erdős) было представлено Д. У. Муном (J. W. Moon) и Л. Мозером (L. Moser) в *Israel J. Math.* **3** (1965), 23–25.]

132. Сильное произведение клик в G и H представляет собой клику в $G \boxtimes H$, согласно упр. 93; следовательно, $\omega(G\boxtimes H) \ge \omega(G)\omega(H) = \chi(G)\chi(H)$. С другой стороны, раскрашивания $a(u)$ и $b(v)$ графов G и H приводят к раскрашиванию $c(u,v)=(a(u),b(v))$ графа $G\boxtimes H$; следовательно, $\chi(G\boxtimes H)\le\chi(G)\chi(H)$, а $\omega(G\boxtimes H)\le\chi(G\boxtimes H)$.

133. (а) 24; (б) 60; (в) 3; (г) 6; (д) 6; (е) 4; (ж) 5; (з) 4; (и) $K_2 \boxtimes C_{12}$; (к) 18; (л) 12; (м) да, степени 5; (н) нет. [На самом деле в 2009 году Маркус Чимани (Markus Chimani) прибег к методу ветвления и отсечений, чтобы доказать, что этот граф нельзя начертить менее чем с 12 пересечениями.] (о) Да; в действительности он является 4-связным (см. раздел 7.4.1). (п) Да; мы рассматриваем *каждый* граф как ориентированный, с двумя дугами для каждого ребра. (р) Конечно же, нет. (с) Да, и это легко увидеть.

[Музыкальный граф представляет простые модуляции между знаками ключей. Он появляется на странице 73 книги *Graphs* Р. Д. Вильсона (R. J. Wilson) и Д. Д. Уоткинса (J. J. Watkins) (1990).]

134. Путем поворотов и/или обменов внутренних и внешних вершин мы можем найти автоморфизм, который отображает любую вершину в C. Если C фиксировано, можно обменивать внутренние и внешние вершины любого подмножества оставшихся 11 пар и/или выполнять отражение по горизонтали. Следовательно, всего имеется $24\times 2^{11}\times 2 = 98\,304$ автоморфизма.

135. Пусть $\omega = e^{2\pi i/12}$, и определим матрицы $Q=(q_{ij})$, $S=(s_{ij})$, где $q_{ij}=[j=(i+1) \bmod 12]$ и $s_{ij}=\omega^{ij}$ для $0\le i,j<12$. Согласно упр. 96, (б), матрица смежности музыкального графа

$K_2 \boxtimes C_{12}$ представляет собой $A = \left(\begin{smallmatrix}1&1\\1&1\end{smallmatrix}\right) \otimes (I+Q+Q^-) - I$. Пусть T — матрица $\left(\begin{smallmatrix}1&1\\1&-1\end{smallmatrix}\right) \otimes S$; тогда $T^- A T$ представляет собой диагональную матрицу D, первые 12 элементов которой — $1 + 4\cos\frac{j\pi}{6}$ для $0 \le j < 12$, а другие 12 элементов равны -1. Следовательно, $A^{2m} = TD^{2m}T^-$, и отсюда следует, что количество $2m$-шаговых блужданий от C до $(\mathrm{C}, \mathrm{G}, \mathrm{D}, \mathrm{A}, \mathrm{E}, \mathrm{B}, \mathrm{F}^\sharp)$ соответственно равно

$$\begin{aligned}
\mathrm{C}_m &= \tfrac{1}{24}(25^m + 2(13+4\sqrt{3})^m + 3^{2m+1} + 2(13-4\sqrt{3})^m + 16);\\
\mathrm{G}_m &= \tfrac{1}{24}(25^m + \sqrt{3}(13+4\sqrt{3})^m - \sqrt{3}(13-4\sqrt{3})^m - 1);\\
\mathrm{D}_m &= \tfrac{1}{24}(25^m + (13+4\sqrt{3})^m + (13-4\sqrt{3})^m - 3);\\
\mathrm{A}_m &= \tfrac{1}{24}(25^m - 3^{2m+1} + 2);\\
\mathrm{E}_m &= \tfrac{1}{24}(25^m - (13+4\sqrt{3})^m - (13-4\sqrt{3})^m + 1);\\
\mathrm{B}_m &= \tfrac{1}{24}(25^m - \sqrt{3}(13+4\sqrt{3})^m + \sqrt{3}(13-4\sqrt{3})^m - 1);\\
\mathrm{F}^\sharp_m &= \tfrac{1}{24}(25^m - 2(13+4\sqrt{3})^m + 3^{2m+1} - 2(13-4\sqrt{3})^m);
\end{aligned}$$

также $\mathrm{a}_m = \mathrm{C}_m - 1$, $\mathrm{d}_m = \mathrm{F}_m = \mathrm{e}_m = \mathrm{G}_m$ и т. д. В частности, $(\mathrm{C}_6, \mathrm{G}_6, \mathrm{D}_6, \mathrm{A}_6, \mathrm{E}_6, \mathrm{B}_6, \mathrm{F}^\sharp_6) = (15\,462\,617,\ 14\,689\,116,\ 12\,784\,356,\ 10\,106\,096,\ 7\,560\,696,\ 5\,655\,936,\ 5\,015\,296)$, Так что искомая вероятность равна $15462617/5^{12} \approx 6.33\%$. При $m \to \infty$ все вероятности равны $\frac{1}{24} + O(0.8^m)$.

136. Нет. Только два графа Кейли порядка 10 являются кубическими, а именно $K_2 \square C_5$ (вершины которого могут быть записаны как $\{e, \alpha, \alpha^2, \alpha^3, \alpha^4, \beta, \beta\alpha, \beta\alpha^2, \beta\alpha^3, \beta\alpha^4\}$, где $\alpha^5 = \beta^2 = (\alpha\beta)^2 = e$), и граф с вершинами $\{0, 1, \ldots, 9\}$ и дугами $v \to (v \pm 1) \bmod 10$, $v \to (v+5) \bmod 10$. [См. D. A. Holton and J. Sheehan, *The Petersen Graph* (1993), упр. 9.10. Кстати, SGB-графы $raman(p, q, t, 0)$ являются графами Кейли.]

137. Пусть $[x, y]$ означает метку (x, y); нам нужно, чтобы условие $[x, y] = [x+a, y+b] = [x+c, y+d]$ выполнялось для всех x и y. Если A представляет собой матрицу $\left(\begin{smallmatrix}a&b\\c&d\end{smallmatrix}\right)$, операция добавления t раз нижней строки A к верхней строке заменяет матрицу A матрицей $A' = \left(\begin{smallmatrix}1&t\\0&1\end{smallmatrix}\right)A = \left(\begin{smallmatrix}a'&b'\\c'&d'\end{smallmatrix}\right)$, где $a' = a + tc$, $b' = b + td$, $c' = c$, $d' = d$. Новое условие $[x, y] = [x+a', y+b'] = [x+c', y+d']$ эквивалентно старому; а $\gcd(a', b', c', d') = \gcd(a, b, c, d)$. Аналогично можно умножить A слева на $\left(\begin{smallmatrix}1&0\\t&1\end{smallmatrix}\right)$ без реального изменения задачи.

Можно также работать и со столбцами, заменяя A на $A'' = A\left(\begin{smallmatrix}1&t\\0&1\end{smallmatrix}\right) = \left(\begin{smallmatrix}a''&b''\\c''&d''\end{smallmatrix}\right)$, где $a'' = a$, $b'' = ta + b$, $c'' = c$, $d'' = tc + d$. Эта операция изменяет задачу, но только немного: если мы найдем присваивание меток, которое удовлетворяет условию $[\![x, y]\!] = [\![x+a'', y+b'']\!] = [\![x+c'', y+d'']\!]$ для всех x и y, то мы получим $[x, y] = [x+a, y+b] = [x+c, x+d]$, если $[x, y] = [\![x, y + tx]\!]$. Аналогично можно умножить A справа на $\left(\begin{smallmatrix}1&0\\t&1\end{smallmatrix}\right)$; задача останется почти той же самой.

Ряд таких операций над строками и столбцами приводит A к простому виду $UAV = \left(\begin{smallmatrix}1&0\\0&n\end{smallmatrix}\right)$, где U и V — целочисленные матрицы, такие, что $\det U = \det V = 1$. Кроме того, если $V = \left(\begin{smallmatrix}\alpha&\beta\\\gamma&\delta\end{smallmatrix}\right)$, присваивание меток для приведенной задачи, которое удовлетворяет простым условиям $[\![x, y]\!] = [\![x+1, y]\!] = [\![x, y+n]\!]$, будет предоставлять решение и для исходной задачи присваивания меток, если определить $[x, y] = [\![\alpha x + \gamma y, \beta x + \delta y]\!]$.

Наконец, приведенная задача присваивания меток проста: положим $[\![x, y]\!] = y \bmod n$. Таким образом, искомый ответ заключается в установке $p = \beta$, $q = \delta$.

138. Действуя, как и ранее, но с матрицей A размером $k \times k$, операции со строками и столбцами приведут задачу к диагональной матрице UAV. Диагональные элементы $(d_1, \ldots, d_k)$ характеризуются тем условием, что $d_1 \ldots d_j$ является наибольшим общим делителем детерминантов всех $j \times j$ подматриц A. [Это "нормальная форма Смита"; см. H. J. S. Smith, *Philosophical Transactions* **151** (1861), 293–326, §14.] Если присваивание меток $[\![x]\!]$ удовлетворяет приведенной задаче, то исходную задачу решает $[x] = [\![xV]\!]$. Количество элементов в обобщенном торе равно $n = \det A = d_1 \ldots d_k$.

Приведенная задача имеет простое решение, как ранее, если $d_1 = \cdots = d_{k-1} = 1$. Но в общем случае приведенное присваивание меток представляет собой r-мерный одинарный тор с размерностями $(d_{k-r+1}, \ldots, d_k)$, где $d_{k-r+1} > d_{k-r} = 1$. (Здесь $d_0 = 1$; r может быть равно k.)

Для примера из условия задачи мы находим $d_1 = 1$, $d_2 = 2$, $d_3 = 10$, $n = 20$; в действительности

$$UAV = \begin{pmatrix} 1 & -2 & 0 \\ 0 & 1 & -1 \\ -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{pmatrix} \begin{pmatrix} 1 & 5 & 6 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 10 \end{pmatrix}.$$

Каждая точка (x, y, z) теперь получает двумерную метку $(u, v) = ((5x + y) \bmod 2, (6x + y + z) \bmod 10)$. Шестью соседями (u, v) являются $((u \pm 1) \bmod 2, v)$, $((u \pm 1) \bmod 2, (v \pm 1) \bmod 10)$, $(u, (v \pm 1) \bmod 10)$. Это — мультиграф, поскольку первые два соседа идентичны; но это не то же, что мультиграф $C_2 \boxtimes C_{10}$, который имеет степень 8.

[Обобщенные торы, по сути, представляют собой графы Кейли абелевых групп; см. упр. 136. Они были предложены в качестве удобных сетей соединений; в этом случае желательно минимизировать диаметр при заданных k и n. См. C. K. Wong and D. Coppersmith, *JACM* **21** (1974), 392–402; C. M. Fiduccia, R. W. Forcade and J. S. Zito, *SIAM J. Discrete Math.* **11** (1998), 157–167.]

139. (Это упражнение помогает уяснить разницу между помеченными графами G, в которых вершины имеют определенные имена, и непомеченными графами H, такими как представленные на рис. 2.) Если N_H — количество помеченных графов на множестве $\{1, 2, \ldots, h\}$, изоморфных H, и если U — любое h-элементное подмножество V, то вероятность того, что $G \mid U$ изоморфен H, равна $N_H/2^{h(h-1)/2}$. Следовательно, искомый ответ — $\binom{n}{h} N_H/2^{h(h-1)/2}$. Нам нужно только вывести значение N_H, которое соответственно равно: (а) 1; (б) $h!/2$; (в) $(h-1)!/2$; (г) $h!/a$, где H имеет a автоморфизмов.

140. (а) $\#(K_3{:}W_n) = n-1$ и $\#(P_3{:}W_n) = \binom{n-1}{2}$ для $n \geq 5$; также $\#(\overline{K_3}{:}W_8) = 7$.

(б) G пропорционален тогда и только тогда, когда $\#(K_3{:}G) = \#(\overline{K_3}{:}G) = \frac{1}{8}\binom{n}{3}$ и $\#(P_3{:}G) = \#(\overline{P}_3{:}G) = \frac{3}{8}\binom{n}{3}$. Если G имеет e ребер, то $(n-2)e = 3\#(K_3{:}G) + 2\#(P_3{:}G) + \#(\overline{P}_3{:}G)$, поскольку каждая пара вершин появляется в $n-2$ порожденных подграфов. Если G имеет последовательность степеней $d_1 \ldots d_n$, то $d_1 + \cdots + d_n = 2e$, $\binom{d_1}{2} + \cdots + \binom{d_n}{2} = 3\#(K_3{:}G) + \#(P_3{:}G)$ и $d_1(n-1-d_1) + \cdots + d_n(n-1-d_n) = 2\#(P_3{:}G) + 2\#(\overline{P}_3{:}G)$. Следовательно, пропорциональный граф удовлетворяет $(*)$, если только n не равно 2. (Упражнение должно исключать такой случай.)

И обратно, если G удовлетворяет $(*)$ и имеет корректное значение $\#(K_3{:}G)$, то он также имеет корректные значения $\#(P_3{:}G)$, $\#(\overline{P}_3{:}G)$ и $\#(\overline{K_3}{:}G)$.

[*Ссылки*: S. Janson and J. Kratochvil, *Random Structures & Algorithms* **2** (1991), 209–224. В *J. Combinatorial Theory* **B47** (1989), 125–145, Э. Д. Барбур (A. D. Barbour), M. Кароньски (M. Karoński) и A. Руциньски (A. Ruciński) показали, что дисперсия $\#(H{:}G)$ пропорциональна либо n^{2h-2}, либо n^{2h-3}, либо n^{2h-4}, где первый случай осуществляется, когда H не имеет $\frac{1}{2}\binom{h}{2}$ ребер, а третий — когда H является пропорциональным графом.]

141. Условию $(*)$ удовлетворяют только 8 последовательностей степеней $d_1 \ldots d_8$, а именно последовательности 73333333 (1/2), 65433322 (26/64), 64444222 (2/10), 64443331 (8/22), 55543222 (8/20), 55533331 (2/10), 55444321 (26/64) и 44444440 (1/2). Каждая последовательность степеней приведена со статистикой (N_1/N), где N — неизоморфные графы, имеющие данную последовательность, а N_1 — сколько из них пропорциональны. Последние три случая являются дополнениями первых трех. Не существует графов порядка 8, являющихся

одновременно пропорциональными и самодополнительными. Максимально симметричными примерами первых пяти случаев являются W_8,

,

,

и

.

142. Указание получается, как и в ответе к упр. 140; $(n-3)\#(\overline{K_3}{:}G)$ и $(n-3)\#(P_3{:}G)$ могут также быть выражены через учет четырех вершин. Кроме того, граф с e ребрами имеет $\binom{e}{2} = \#(P_3 \subseteq G) + \#(K_2 \oplus K_2 \subseteq G)$, поскольку любые два ребра образуют либо P_3, либо $K_2 \oplus K_2$; в этой формуле $\#(P_3 \subseteq G)$ учитывает подграфы, не обязательно являющиеся порожденными.

Мы имеем $\#(P_3 \subseteq G) = \#(P_3{:}G) + 3\#(K_3{:}G)$, а аналогичная формула выражает $\#(K_2 \oplus K_2 \subseteq G)$ через количества порождений. Таким образом, сверхпропорциональный граф должен быть пропорциональным и удовлетворять условиям $e = \frac{1}{2}\binom{n}{2}$, $\#(P_3 \subseteq G) = \frac{3}{4}\binom{n}{3}$, $\#(K_2 \oplus K_2 \subseteq G) = \frac{3}{4}\binom{n}{4}$. Но эти значения противоречат формуле для $\binom{e}{2}$.

143. Рассмотрим граф, вершины которого являются строками A, а ребра u — v означают, что строки u и v совпадают, за исключением одного столбца, j. Пометим такое ребро как j.

Если граф содержит цикл, удалим любое ребро из цикла и будем повторять этот процесс до тех пор, пока циклов не останется. Заметим, что метка каждого удаленного ребра появляется еще где-то в этом цикле; следовательно, удаления не влияют на множество меток ребер. Но мы останемся менее чем с $m \le n$ ребрами согласно теореме 2.3.4.1A; так что имеется менее чем n различных меток. [См. J. A. (Bondy), *J. Combinatorial Theory* **B12** (1972), 201–202.]

144. Пусть G — граф на вершинах $\{1, \ldots, m\}$, в котором ребро i — j имеется тогда и только тогда, когда $* \ne x_{il} \ne x_{jl} \ne *$ для некоторого l. Этот граф является k-раскрашиваемым тогда и только тогда, когда имеется дополнение с не более чем k различными строками. И обратно, если G представляет собой граф на вершинах $\{1, \ldots, n\}$, с матрицей смежности A, то матрица $X = A + *(J - I - A)$ размером $n \times n$ обладает тем свойством, что i — j тогда и только тогда, когда $* \ne x_{il} \ne x_{jl} \ne *$ для некоторого l. [См. M. Sauerhoff and I. Wegener, *IEEE Trans.* **CAD-15** (1996), 1435–1437.]

145. Установить $c \leftarrow 0$ и повторять следующие операции для $1 \le j \le n$. Если $c = 0$, установить $x \leftarrow a_j$ и $c \leftarrow 1$; в противном случае при $x = a_j$ установить $c \leftarrow c+1$; в противном случае установить $c \leftarrow c - 1$. Тогда x является ответом на поставленный вопрос. Идея заключается в отслеживании возможного мажоритарного элемента x, который встречается c раз в неотброшенных элементах; мы отбрасываем a_j и один x, когда находим $x \ne a_j$. [См. *Automated Reasoning* (Kluwer, 1991), 105–117. Расширения для поиска за $O(n \log k)$ шагов всех элементов, встречающихся более чем n/k раз, рассматриваются Д. Мисрой (J. Misra) и Д. Грисом (D. Gries) в *Science of Computer Programming* **2** (1982), 143–152.]

РАЗДЕЛ 7.1.1

1. (Решение К. Сартена (C. Sartena).) Он описал импликацию $x \Rightarrow y$, где "это" означает соответственно y, x, x, y, y, x. (Возможны и другие решения.)

2. Земной операцией, соответствующей наурикaнской $x \circ y$, является $\overline{\bar{x} \circ \bar{y}}$; ее таблица истинности, таким образом, представляет собой обращение дополнения таблицы истинности для $\circ$. Следовательно, ответами на поставленный вопрос являются соответственно $\top$, $\vee$, $\subset$, $\llcorner$, $\supset$, R, $\equiv$, $\wedge$, $\overline{\wedge}$, $\oplus$, $\bar{\mathsf{R}}$, $\overline{\supset}$, $\bar{\llcorner}$, $\overline{\subset}$, $\overline{\vee}$, $\bot$. (Из любого тождества, включающего

16 операций из табл. 1, вытекает соответствующее дуальное тождество, получающееся подстановкой наориканского эквивалента. Например, каждый из законов де Моргана (11) и (12) дуален другому, как и тождества (3) и (4), связывающие $\equiv$ и $\oplus$. В этом смысле $\equiv$ можно рассматривать как столь же полезную операцию, как и дуальная ей $\oplus$.)

3. (а) $\vee$; (б) $\wedge$; (в) $\bar{\sqsubset}$; (г) $\equiv$. [В действительности многие формулы вычисляются лучше при использовании -1 для истины и $+1$ для лжи, несмотря на то, что такое соглашение выглядит прямо-таки аморально; тогда $x \cdot y$ соответствует $\oplus$. Обратите внимание, что при любом соглашении $\langle xyz \rangle = \operatorname{sign}(x + y + z)$.]

4. [*Trans. Amer. Math. Soc.* **14** (1913), 481–488.] (а) Начнем с таблиц истинности для $\sqsubset$ и $\sqsupset$; затем вычислим таблицы истинности $\alpha \mathbin{\overline{\wedge}} \beta$ побитово из каждой известной пары таблиц истинности α и β, генерируя результаты в порядке длины каждой формулы и записывая кратчайшие формулы, приводящие к каждой новой 4-битовой таблице.

$\bot$: $(x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x)) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x))$
$\wedge$: $(x \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} y)$
$\bar{\supset}$: $(x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} y)) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} y))$
$\sqsubset$: x
$\bar{\subset}$: $(y \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x)) \mathbin{\overline{\wedge}} (y \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x))$
$\sqsupset$: y
$\oplus$: $(y \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x)) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} y))$
$\vee$: $(y \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x)$

$\bar{\vee}$: $(x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x)) \mathbin{\overline{\wedge}} ((y \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x))$
$\equiv$: $(x \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} ((y \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x))$
$\bar{\sqsupset}$: $y \mathbin{\overline{\wedge}} y$
$\subset$: $y \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x)$
$\bar{\sqsubset}$: $x \mathbin{\overline{\wedge}} x$
$\supset$: $x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} y)$
$\overline{\wedge}$: $x \mathbin{\overline{\wedge}} y$
$\top$: $x \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} x)$

(б) В этом случае начнем с четырех таблиц, $\bot$, $\top$, $\sqsubset$, $\sqsupset$, и будем предпочитать формулы с меньшим количеством констант при выборе между формулами одной длины.

$\bot$: 0
$\wedge$: $(x \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} 1$
$\bar{\supset}$: $((y \mathbin{\overline{\wedge}} 1) \mathbin{\overline{\wedge}} x) \mathbin{\overline{\wedge}} 1$
$\sqsubset$: x
$\bar{\subset}$: $(y \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} 1)) \mathbin{\overline{\wedge}} 1$
$\sqsupset$: y
$\oplus$: $(y \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} 1)) \mathbin{\overline{\wedge}} ((y \mathbin{\overline{\wedge}} 1) \mathbin{\overline{\wedge}} x)$
$\vee$: $(y \mathbin{\overline{\wedge}} 1) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} 1)$

$\bar{\vee}$: $1 \mathbin{\overline{\wedge}} ((y \mathbin{\overline{\wedge}} 1) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} 1))$
$\equiv$: $(x \mathbin{\overline{\wedge}} y) \mathbin{\overline{\wedge}} ((y \mathbin{\overline{\wedge}} 1) \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} 1))$
$\bar{\sqsupset}$: $y \mathbin{\overline{\wedge}} 1$
$\subset$: $y \mathbin{\overline{\wedge}} (x \mathbin{\overline{\wedge}} 1)$
$\bar{\sqsubset}$: $x \mathbin{\overline{\wedge}} 1$
$\supset$: $(y \mathbin{\overline{\wedge}} 1) \mathbin{\overline{\wedge}} x$
$\overline{\wedge}$: $x \mathbin{\overline{\wedge}} y$
$\top$: 1

5. (а) $\bot$: $x \mathbin{\overline{\subset}} x$; $\wedge$: $(x \mathbin{\overline{\subset}} y) \mathbin{\overline{\subset}} y$; $\bar{\supset}$: $y \mathbin{\overline{\subset}} x$; $\sqsubset$: x; $\bar{\subset}$: $x \mathbin{\overline{\subset}} y$; $\sqsupset$: y; Прочие 10 операторов не могут быть выражены. (б) При использовании констант можно выразить все 16 операторов.

$\bot$: 0
$\wedge$: $(y \mathbin{\overline{\subset}} 1) \mathbin{\overline{\subset}} x$
$\bar{\supset}$: $y \mathbin{\overline{\subset}} x$
$\sqsubset$: x
$\bar{\subset}$: $x \mathbin{\overline{\subset}} y$
$\sqsupset$: y
$\oplus$: $((y \mathbin{\overline{\subset}} x) \mathbin{\overline{\subset}} ((x \mathbin{\overline{\subset}} y) \mathbin{\overline{\subset}} 1)) \mathbin{\overline{\subset}} 1$
$\vee$: $(y \mathbin{\overline{\subset}} (x \mathbin{\overline{\subset}} 1)) \mathbin{\overline{\subset}} 1$

$\bar{\vee}$: $y \mathbin{\overline{\subset}} (x \mathbin{\overline{\subset}} 1)$
$\equiv$: $(y \mathbin{\overline{\subset}} x) \mathbin{\overline{\subset}} ((x \mathbin{\overline{\subset}} y) \mathbin{\overline{\subset}} 1)$
$\bar{\sqsupset}$: $y \mathbin{\overline{\subset}} 1$
$\subset$: $(x \mathbin{\overline{\subset}} y) \mathbin{\overline{\subset}} 1$
$\bar{\sqsubset}$: $x \mathbin{\overline{\subset}} 1$
$\supset$: $(y \mathbin{\overline{\subset}} x) \mathbin{\overline{\subset}} 1$
$\overline{\wedge}$: $((y \mathbin{\overline{\subset}} 1) \mathbin{\overline{\subset}} x) \mathbin{\overline{\subset}} 1$
$\top$: 1

[B. A. Bernstein, *University of California Publications in Mathematics* **1** (1914), 87–96.]

6. (а) $\bot$, $\wedge$, $\sqsubset$, $\sqsupset$, $\oplus$, $\vee$, $\equiv$, $\top$. (б) $\bot$, $\sqsubset$, $\sqsupset$, $\oplus$, $\equiv$, $\top$. [Заметим, что все эти операторы ассоциативны. Фактически из указанного тождества вытекает закон ассоциативности в общем случае. Сначала мы имеем (i) $(x \circ y) \circ ((z \circ y) \circ w) = ((x \circ z) \circ (z \circ y)) \circ ((z \circ y) \circ w) = (x \circ z) \circ w$ и аналогично (ii) $(x \circ (y \circ z)) \circ (y \circ w) = x \circ (z \circ w)$. Кроме того, (iii) $(x \circ y) \circ (z \circ w) = (x \circ y) \circ ((z \circ y) \circ (y \circ w)) = (x \circ z) \circ (y \circ w)$ согласно (i). Таким образом,

$(x\circ z)\circ w = (x\circ z)\circ((z\circ z)\circ w) = (x\circ(z\circ z))\circ(z\circ w) = x\circ(z\circ w)$ согласно (i), (iii), (ii). Свободная система, генерируемая $\{x_1,\ldots,x_n\}$, имеет ровно $n+2^n n^2$ различных элементов, а именно $\{x_j \mid 1\le j\le n\}$ и $\{x_i\circ x_{j_1}\circ\cdots\circ x_{j_r}\circ x_k \mid r\ge 0$ и $1\le i,k\le n$ и $1\le j_1<\cdots<j_r\le n\}$.]

7. Это эквивалентно тому, что мы хотим, чтобы было справедливо тождество $y\circ(x\circ y)=x$, что выполняется только для $\oplus$ и $\equiv$. [Джевонс (Jevons) обратил внимание на это свойство $\oplus$ в *Pure Logic* §151, но не занимался этим вопросом. Мы изучим обобщенные системы этой природы, называемые "gropes", в разделе 7.2.3.]

8. $(\{\bot,\wedge,\overline{\subset}\},S_0)$, $(\{\top,\vee,\supset\},S_1)$, $(\{\llcorner,\overline{\sqsubset}\},S_0\cap S_1)$, $(\{\oplus,\equiv,\bar{\mathsf{R}}\},S_2)$, $(\{\overline{\supset},\overline{\vee}\},S_0\cap S_2)$, $(\{\subset,\overline{\wedge}\}$, $S_1\cap S_2)$ и $(\mathsf{R},$любая$)$, где $S_0=\{\square\mid 0\,\square\,0=0\}$, $S_1=\{\square\mid 1\,\square\,1=1\}$ и $S_2=\{\square\mid \bar{x}\,\square\,\bar{y}=\overline{x\,\square\,y}\}=\{\llcorner,\mathsf{R},\overline{\sqsubset},\bar{\mathsf{R}}\}$. Таким образом, 92 из 256 пар являются леводистрибутивными. [Эта задача и задачи из упр. 6 впервые были рассмотрены Э. Шрёдером (E. Schröder) в §55 его посмертной публикации *Vorlesungen über die Algebra der Logik* **2**, 2 (1905). Он выразил ответ, по сути, указав, что соответствующие таблицы истинности $(pqrs, wxyz)$ операторов $(\circ,\square)$ должны удовлетворять отношению $((pq\vee rs)\wedge\bar z)\vee((\bar p\bar q\vee\bar r\bar s)\wedge w)\vee((p\bar q\vee r\bar s)\wedge((w\equiv z)\vee(x\equiv y)))=0$.]

9. (а) Ложно; $(x\oplus y)\vee z=(x\vee z)\oplus(y\vee z)\oplus z$. (б) Истинно, поскольку тождество очевидным образом выполняется, когда $z=0$ и когда $z=1$. (в) Истинно; оно равно также $(x\oplus y)\vee(x\oplus z)=1-[x=y=z]$.

10. Первый этап декомпозиции (16) дает нам функции с таблицами истинности $g=10100011$ и $h=10100011\oplus 10010011=00110000$; продолжение этого процесса таким же образом дает $1+y+xz+w+wy+wx+wxz$ (по модулю 2).

11. Указанный член присутствует тогда и только тогда, когда функция $f(x_1,\ldots,x_n)$ истинна нечетное число раз, когда $x_1=x_4=x_5=x_7=x_9=x_{10}=\cdots=0$. (Имеется 2^k таких случаев, когда мы устанавливаем все, кроме k переменных, равными нулю.) Другими словами, мультилинейное представление может быть выражено через обозначения наподобие

$$f(x,y,z)=(f_{000}+f_{00*}z+f_{0*0}y+f_{0**}yz+f_{*00}x+f_{*0*}xz+f_{**0}xy+f_{***}xyz)\bmod 2$$

проиллюстрированное здесь для случая $n=3$, где $f_{**0}=f(1,1,0)\oplus f(1,0,0)\oplus f(0,1,0)\oplus f(0,0,0)$ и т. д.

12. (а) Подставим $1-w$ вместо $\bar w$ и т. д. в (23), получая $1-y-xz+2xyz-w+wy+wx+wxz-2wxyz$. [Некоторые авторы называют это полиномом Жегалкина; но сам И. И. Жегалкин всегда работал по модулю 2. Другие названия, встречающиеся в литературе,— "полином доступности"("availability polynomial"), "полином надежности" ("reliability polynomial"), "характеристический полином" ("characteristic polynomial").]

(б) Соответствующие коэффициенты для произвольной n-арной функции могут достигать абсолютных значений 2^{n-1} (и это по индукции является максимальным значением). Например, целочисленное мультилинейное представление $x_1\oplus\cdots\oplus x_n$ над целыми числами оказывается равным $e_1-2e_2+4e_3-\cdots+(-2)^{n-1}e_n$, где e_k — k-я элементарная симметричная функция от $\{x_1,\ldots,x_n\}$. Формула из ответа к предыдущему упражнению принимает вид

$$f(x,y,z)\;=\;f_{000}+f_{00*}z+f_{0*0}y+f_{0**}yz+f_{*00}x+f_{*0*}xz+f_{**0}xy+f_{***}xyz$$

над целыми числами, где мы теперь имеем $f_{**0}=f(1,1,0)-f(1,0,0)-f(0,1,0)+f(0,0,0)$ и т. д. Это расширение представляет собой замаскированный вид преобразования Адамара 4.6.4–(38).

(в, г) Полином представляет собой сумму своих минитермов наподобие $x_1(1-x_2)(1-x_3)x_4$. Каждый минитерм неотрицателен при $0\le x_1,\ldots,x_n\le 1$, а сумма всех минитермов равна 1.

(д) $\partial f/\partial x_j=h(x)-g(x)$, где $h(x)\ge g(x)$ согласно (г). (См. упр. 21.)

13. В действительности F представляет собой в точности целочисленное мультилинейное представление (см. упр. 12).

14. Пусть $r_j = p_j/(1-p_j)$. Нам нужно, чтобы $f(0,0,0)=0$ и $f(1,1,1)=1 \Leftrightarrow r_1r_2r_3>1$, $f(0,0,1)=0$ и $f(1,1,0)=1 \Leftrightarrow r_1r_2>r_3$, $f(0,1,0)=0$ и $f(1,0,1)=1 \Leftrightarrow r_1r_3>r_2$, $f(0,1,1)=0$ и $f(1,0,0)=1 \Leftrightarrow r_1>r_2r_3$. Так что мы получаем (а) $\langle x_1x_2x_3\rangle$; (б) x_1; (в) $\bar{x}_3$.

15. Упражнение 1.2.6–10 говорит нам, что $\binom{x}{k} \bmod 2 = [x \mathbin{\&} k = k]$. Следовательно, например, $\binom{x}{11} \equiv x_4 \land x_2 \land x_1$ (по модулю 2), когда $x = (x_n \ldots x_1)_2$; и этим путем мы можем получить каждый член мультилинейного представления наподобие (19). Более того, нам не требуется работать по модулю 2, поскольку интерполяционный полином $\binom{x}{11}\binom{15-x}{4}$ в точности представляет $x_4 \land \bar{x}_3 \land x_2 \land x_1$.

16. Да, или даже на $+$, поскольку разные минитермы не могут быть одновременно истинны. (Но мы не можем сделать это в обычной дизъюнктивной нормальной форме наподобие (25). См. упр. 35.)

17. Бинарная операция $\overline{\land}$ не является ассоциативной, так что выражение наподобие $x \mathbin{\overline{\land}} y \mathbin{\overline{\land}} z$ должно рассматриваться как *тернарная* операция. Запись Шустрого корректна, если рассматривать НЕ-И как n-арную операцию, при этом учитывая, что НЕ-И *одной* переменной x представляет собой $\bar{x}$.

18. Если это не так, можно установить $u_1 \leftarrow \cdots \leftarrow u_s \leftarrow 1$ и $v_1 \leftarrow \cdots \leftarrow v_t \leftarrow 0$, делая f одновременно ложной и истинной. (И если мы рассмотрим многократное применение закона дистрибутивности (2) к DNF, пока не получим CNF, то обнаружим, что истинно и обратное: дизъюнкция $v_1 \lor \cdots \lor v_t$ вытекает из f тогда и только тогда, когда она имеет литерал, общий со всеми импликантами f, тогда и только тогда, когда она имеет литерал, общий со всеми простыми импликантами f, тогда и только тогда, когда она имеет литерал, общий с каждой импликантой некоторой DNF f.)

19. Максимальными подкубами, содержащимися в 0010, 0011, 0101, 0110, 1000, 1001, 1010 и 1011, являются 0∗10, 0101, ∗01∗ и 10∗∗; так что ответ — $(w \lor \bar{y} \lor z) \land (w \lor \bar{x} \lor y \lor \bar{z}) \land (x \lor \bar{y}) \land (\bar{w} \lor x)$. (Эта конъюнктивная нормальная форма является также кратчайшей.)

20. Истинно. Соответствующий максимальный подкуб содержится в некоторых максимальных подкубах f' и g', и их пересечение не может быть бóльшим. (Это наблюдение принадлежит Самсону (Samson) и Миллсу (Mills), статья которых цитируется ниже, в ответе к упр. 31.)

21. Согласно закону Буля (20) мы видим, что n-арная функция f монотонна тогда и только тогда, когда ее $(n-1)$-арные проекции g и h монотонны и удовлетворяют условию $g \le h$. Следовательно,

$$f = (g \land \bar{x}_n) \lor (h \land x_n) \;=\; (g \land \bar{x}_n) \lor (g \land x_n) \lor (h \land x_n) \;=\; g \lor (h \land x_n),$$

так что можно обойтись без дополнения. Константы 0 и 1 отсутствуют, кроме тех случаев, когда функция идентична константе. И наоборот, любое выражение, построенное из $\land$ и $\lor$, очевидно, является монотонным.

Примечание по терминологии. Строго говоря, мы должны использовать термин “монотонная неубывающая” вместо просто “монотонная”, если хотим, чтобы наш язык соответствовал языку классической математики. Дело в том, что уменьшающаяся функция действительного переменного также монотонна. (См., например, раздел 3.3.2G.) Но “неубывающая” — немного труднопроизносимое слово, в особенности в английском языке; так что исследователи, активно работающие в области булевых функций, почти единогласно сошлись на том, что в булевом контексте “монотонность” автоматически подразумевает неубывание. Аналогично математический термин “положительная функция” обычно означает функцию, значения которой превышают нуль. Но авторы, пишущие

о "положительных булевых функциях", имеют в виду функции, которые мы называем монотонными. Поскольку монотонная функция является сохраняющей порядок, некоторые авторы используют термин *изотонная*; но это слово уже принято на вооружение физиками, химиками и музыковедами.

Булева функция наподобие $\bar{x} \vee y$, которая становится монотонной, если некоторое подмножество ее переменных будет дополнено, называется *унатной* (*unate*). Очевидно, что теорема Q применима к таким унатным функциям.

22. И g, и $g \oplus h$ должны быть монотонными, и $g(x) \wedge h(x) = 0$.

23. $x \wedge (v \vee y) \wedge (v \vee z) \wedge (w \vee z)$. (Следствие Q применимо также к *конъюнктивным* простым формам монотонных функций. Следовательно, для решения любой задачи такого вида нам нужно применять только закон дистрибутивности (2) до тех пор, пока в $\vee$ не останется ни одного $\wedge$, а затем удалить любой дизъюнкт, содержащий все переменные другого дизъюнкта.)

24. По индукции по k аналогичное дерево с $\vee$ в корне дает функцию с $2^{2^{\lceil k/2 \rceil}-1}$ простыми импликантами длиной $2^{\lfloor k/2 \rfloor}$, в то время как дерево с $\wedge$ дает $4^{2^{\lfloor k/2 \rfloor}-1}$ простых импликант длиной $2^{\lceil k/2 \rceil}$. Например, когда $k = 6$, $4^7 = 2^{14}$ простых импликант для случая $\wedge$ имеют вид

$$x_{(0t_00t_00t_000)_2} \wedge x_{(0t_00t_001t_001)_2} \wedge x_{(0t_01t_01t_010)_2} \wedge x_{(0t_01t_01t_011)_2}$$
$$\wedge\, x_{(1t_10t_10t_100)_2} \wedge x_{(1t_10t_101t_101)_2} \wedge x_{(1t_11t_110t_110)_2} \wedge x_{(1t_11t_111t_111)_2},$$

где t — либо 0, либо 1. [Более детальную информацию о таких деревьях можно найти в D. E. Knuth and R. W. Moore, *Artificial Intelligence* **6** (1975), 293–326; V. Gurvich and L. Khachiyan, *Discrete Mathematics* **169** (1997), 245–248.]

25. Пусть ответ представляет собой a_n. Тогда $a_2 = a_3 = 2$, $a_4 = 3$, а $a_n = a_{n-2} + a_{n-3}$ для $n > 4$, поскольку простые импликанты, когда $n > 4$, представляют собой либо $p_{n-2} \wedge x_{n-1}$, либо $p_{n-3} \wedge x_{n-2} \wedge x_n$ для некоторой простой импликанты p_k в случае k переменных. (Эти простые импликанты соответствуют минимальному вершинному покрытию графа пути P_n. Мы имеем $a_n = (7P_n + 10P_{n+1} + P_{n+2})/23$, где P_n представляют собой числа Перрина из упр. 7.1.4–15.)

26. (а) Пусть $x_j = [j \in J]$. Тогда $f(x) = 0$ и $g(x) = 1$. (Этот факт был темой упр. 18.)

(б) Предположим, например, что $k \in J \in \mathcal{G}$ и $k \notin \bigcup_{I \in \mathcal{F}} I$, и будем считать, что тест (а) пройден. Пусть $x_j = [j \in J$и $j \neq k]$. Тогда $f(x) = 1$; а $g(x) = 0$, поскольку каждое $J' \in \mathcal{G}$ с $J' \neq J$ содержит элемент $\notin J$.

(в) Вновь предположим, что условие (а) было исключено. Если, скажем, $|J| > |\mathcal{F}|$, то пусть $x_j = [j$ является наименьшим элементом $I \cap J$ для некоторого $I \in \mathcal{F}]$. Тогда $f(x) = 1$, $g(x) = 0$. (г) Теперь будем считать, что $\bigcup_{I \in \mathcal{F}} I = \bigcup_{J \in \mathcal{G}} J$. Каждое $I \in \mathcal{F}$ означает $2^{n-|I|}$ векторов, где $f(x) = 0$; аналогично каждое $J \in \mathcal{G}$ означает $2^{n-|J|}$ векторов, где $g(x) = 1$. Если сумма s меньше, чем 2^n, можно вычислить $s = s_0 + s_1$, где s_0 учитывает вклад в s при $x_n = 0$. Если $s_0 < 2^{n-1}$, установим $x_n \leftarrow 0$; в противном случае $s_1 < 2^{n-1}$, так что мы устанавливаем $x_n \leftarrow 1$. Тогда мы установим $n \leftarrow n-1$; в конечном итоге все x_j становятся известны, и $f(x) = 1$, $g(x) = 0$.

27. Пусть $m = \min(\{|I| \mid I \in \mathcal{F}\} \cup \{|J| \mid J \in \mathcal{G}\})$ — длина кратчайшего дизъюнкта или импликанты. Тогда $N \cdot 2^{n-m} \geq \sum_{I \in \mathcal{F}} 2^{n-|I|} + \sum_{J \in \mathcal{G}} 2^{n-|J|} \geq 2^n$; так что мы имеем $m \leq \lg N$. Если, скажем, $|I| = m$, некоторый индекс k должен появиться как минимум в $1/m$ членов $J \in \mathcal{G}$, поскольку каждое J пересекается с I. Это наблюдение доказывает указание.

Теперь положим $A(0) = A(1) = 1$ и $A(v) = 1 + A(v-1) + A(\lfloor \rho v \rfloor)$ для $v > 1$. Тогда $A(|\mathcal{F}||\mathcal{G}|)$ представляет собой верхнюю границу числа рекурсивных вызовов (числа выполнений шага X1). Полагая $B(v) = A(v) + 1$, получим $B(v) = B(v-1) + B(\lfloor \rho v \rfloor)$ для

$v > 1$, следовательно, $B(v) \le B(v-k) + kB(\lfloor \rho v \rfloor)$ для $v > k$. Взяв $k = v - \lfloor \rho v \rfloor$, мы показываем, что $B(v) \le ((1-\rho)v + 2)B(\lfloor \rho v \rfloor)$; следовательно, $B(v) = O(((1-\rho)v+2)^t)$ при $\rho^t v \le 1$, а именно, когда $t \ge \ln v / \ln(1/\rho) = \Theta((\log v)(\log N))$. И обратно, $A(|\mathcal{F}||\mathcal{G}|) \le A(N^2/4) = N^{O(\log N)^2}$.

На практике алгоритм будет работать существенно быстрее, чем гласят только что полученные пессимистические границы. Поскольку простые дизъюнкты функции являются простыми импликантами дуальной к ней функции, эта задача, по сути, та же, что и проверка, что одна дизъюнктивная нормальная форма дуальна другой. Кроме того, если мы начнем с $f(x) = 0$ и будем неоднократно находить минимальные x, такие, что $f(x) = g(\bar{x}) = 0$, мы можем "растить" f, пока не получим функцию, дуальную g.

Представленные здесь идеи разработаны М. Л. Фредманом (M. L. Fredman) и Л. Хачияном (L. Khachiyan), *J. Algorithms* **21** (1996), 618–628, которые также представили усовершенствование, которое сокращает время работы до $N^{O(\log N/\log\log N)}$. Алгоритм с полиномиальным временем работы в настоящее время неизвестен; однако непохоже, чтобы данная задача была NP-полной, поскольку ее можно решить за время, меньшее экспоненциального.

28. Этот результат очевиден для понимания, но обозначения и терминология могут запутать его; так что давайте рассмотрим конкретный пример: если, скажем, $y_1 = y_4 = y_6 = 1$, а прочие y_k равны нулю, то функция g истинна тогда и только тогда, когда простые импликанты p_1, p_4 и p_6 покрывают все места, где f истинна. Таким образом, мы видим, что существует взаимно однозначное соответствие между каждой импликантой g и каждой дизъюнктивной нормальной формой f, которая содержит только простые импликанты p_j. В этом соответствии простые импликанты g соответствуют "неприводимым" дизъюнктивным нормальным формам, в которых не могут быть пропущены никакие p_j.

Многочисленные усовершенствования этого принципа обсуждаются в R. B. Cutler and S. Muroga, *IEEE Transactions* **C-36** (1987), 277–292.

29. **B1.** [Инициализация.] Установить $k \leftarrow k' \leftarrow 0$. (Аналогичные методы рассматривались в упр. 5–19.)

B2. [Поиск нуля.] Увеличить k нуль или более раз, пока не будет достигнуто значение $k = m$ (завершение алгоритма) или не будет выполнено равенство $v_k \mathbin{\&} 2^j = 0$.

B3. [Обеспечение $k' > k$.] Если $k' \le k$, установить $k' \leftarrow k + 1$.

B4. [Продвижение k'.] Увеличить k' нуль или более раз, пока не будет достигнуто значение $k' = m$ (завершение алгоритма) или не будет выполнено неравенство $v_{k'} \ge v_k + 2^j$.

B5. [Пропуск большого несоответствия.] Если $v_k \oplus v_{k'} \ge 2^{j+1}$, установить $k \leftarrow k'$ и вернуться к шагу B2.

B6. [Запись соответствия.] Если $v_{k'} = v_k + 2^j$, вывести (k, k').

B7. [Продвижение k.] Установить $k \leftarrow k + 1$ и вернуться к шагу B2. ▮

(Шаги B3 и B5 необязательны, но рекомендуемы.)

30. Следующий алгоритм поддерживает отсортированные списки переменной длины в стеке S, размер которого никогда не превосходит $2m + n$. Когда верхним элементом стека является $S_t = s$, верхний список представляет собой упорядоченное множество $S_s < S_{s+1} < \cdots < S_{t-1}$. Биты дескриптора поддерживаются в другом стеке T, имеющем тот же размер, что и S (после шага инициализации).

P1. [Инициализация.] Установить $T_k \leftarrow 0$ для $0 \le k < m$. Затем для $0 \le j < n$ применить алгоритм сканирования j-двойников из упр. 29 и установить $T_k \leftarrow T_k + 2^j$, $T_{k'} \leftarrow T_{k'} + 2^j$ для всех найденных пар (k, k'). Затем установить $s \leftarrow t \leftarrow 0$

и повторять следующие операции до тех пор, пока не будет выполнено условие $s = m$: если $T_s = 0$, вывести подкуб $(0, v_s)$ и установить $s \leftarrow s + 1$; в противном случае установить $S_t \leftarrow v_s$, $T_t \leftarrow T_s$, $t \leftarrow t + 1$, $s \leftarrow s + 1$. Наконец установить $A \leftarrow 0$ и $S_t \leftarrow 0$.

P2. [Продвижение A.] (В этой точке стек S содержит $\nu(A) + 1$ списков подкубов. А именно, если $A = 2^{e_1} + \cdots + 2^{e_r}$, где $e_1 > \cdots > e_r \geq 0$, стек содержит b-значения всех подкубов $(a, b) \subseteq V$, a-значения которых равны соответственно 0, 2^{e_1}, $2^{e_1} + 2^{e_2}$, ..., A, за исключением тех подкубов, нулевые дескрипторы которых не появлялись. Все эти списки не пустые, возможно, за исключением последнего. Тепер мы увеличим A до следующего значимого значения.) Установить $j \leftarrow 0$. Если $S_t = t$ (т. е. если верхний список пуст), увеличивать j нуль или более раз до тех пор, пока не будет достигнуто $j \geq n$ или $A \mathbin{\&} 2^j \neq 0$. Затем, пока $j < n$ и $A \mathbin{\&} 2^j \neq 0$, устанавливать $t \leftarrow S_t - 1$, $A \leftarrow A - 2^j$ и $j \leftarrow j + 1$. Завершить работу алгоритма, если $j \geq n$; в противном случае установить $A \leftarrow A + 2^j$.

P3. [Генерация списка A.] Установить $r \leftarrow t$, $s \leftarrow S_t$ и применить алгоритм сканирования j-двойника из упр. 29 к $r - s$ числам $S_s < \cdots < S_{r-1}$. Для всех найденных пар (k, k') установить $x \leftarrow (T_k \mathbin{\&} T_{k'}) - 2^j$; и если $x = 0$, вывести подкуб (A, S_k), в противном случае установить $t \leftarrow t + 1$, $S_t \leftarrow S_k$, $T_t \leftarrow x$. Наконец установить $t \leftarrow t + 1$, $S_t \leftarrow r + 1$ и вернуться к шагу P2. ▌

Этот алгоритм основан на части идей Эженио Морреаля (Eugenio Morreale) [*IEEE Trans.* **EC-16** (1967), 611–620; *Proc. ACM Nat. Conf.* **23** (1968), 355–365]. Время работы алгоритма, по сути, пропорционально mn (для шага P1) плюс общее количество подкубов, содержащихся в V. Если $m \leq 2^n(1 - \epsilon)$ и если V размером m выбрано случайным образом, то, как показано в упр. 34, среднее общее количество подкубов составляет не более $O(\log\log n / \log\log\log n)$, умноженное на среднее количество максимальных подкубов; следовательно, среднее время работы в большинстве случаев будет приблизительно пропорционально среднему количеству генерируемых выводов. С другой стороны, в упр. 32 и 116 показано, что количество выводов может быть огромным.

31. (а) Пусть $c = c_{n-1} \ldots c_0$, $c' = c'_{n-1} \ldots c'_0$, $c'' = c''_{n-1} \ldots c''_0$. Должно иметься некоторое j с $c_j \neq *$ и $c_j \neq c''_j$; в противном случае $c'' \subseteq c$. Аналогично должно существовать некоторое k с $c'_k \neq *$ и $c'_k \neq c''_k$. Если $j \neq k$, должна быть точка $x_{n-1} \ldots x_0 \in c''$, не входящая ни в c, ни в c', поскольку мы могли бы положить $x_j = \bar{c}_j$ и $x_k = \bar{c}'_k$. Следовательно, $j = k$, и значение j определяется однозначно. Кроме того, несложно увидеть, что $c'_j = \bar{c}_j$. А если $i \neq j$, то мы имеем либо $c_i = *$, либо $c_i = c''_i$ и либо $c'_i = *$, либо $c'_i = c''_i$.

(б) Это утверждение является очевидным следствием (а).

(в) Во-первых, мы докажем, что примечание в скобках в описании шага E2 истинно, когда мы переходим к этому шагу. Ясно, что оно истинно при $j = 0$. В противном случае пусть $c \subseteq V$ является j-кубом, и допустим, что $c = c_0 \cup c_1$, где c_0 и c_1 — $(j-1)$-кубы. При предыдущем выполнении шага E2 мы имели $c_0 \subseteq c'_0 \in C$ и $c_1 \subseteq c'_1 \in C$ для некоторых c'_0 и c'_1; следовательно, либо $c \subseteq c'_0 \sqcup c'_1$, либо $c \subseteq c'_0$, либо $c \subseteq c'_1$. В любом случае c теперь содержится в некотором элементе C.

Во-вторых, докажем, что вывод на шаге E3 в точности представляет собой максимальные j-кубы, содержащиеся в V. Пусть $c \subseteq V$ — некоторый k-куб. Если c максимален, то c будет в C, когда мы достигнем шага E3 с $j = k$, и c будет выведен. Если c не является максимальным, он имеет двойника $c' \subseteq V$, который представляет собой k-куб, содержащийся в некотором подкубе $c'' \in C$, когда мы достигаем E3. Поскольку $c \not\subseteq c''$, консенсус $c \sqcup c''$ будет $(j+1)$-кубом C', и c выведен не будет.

Ссылки. Понятие консенсуса было впервые определено Арчи Блейком (Archie Blake) в его диссертации (University of Chicago, 1937); см. *J. Symbolic Logic* **3** (1938), 93, 112–113.

Оно было независимо открыто заново Эдвардом У. Самсоном (Edward W. Samson) и Бартоном Э. Миллсом (Burton E. Mills) [Air Force Cambridge Research Center Tech. Report 54–21 (Cambridge, Mass.: April 1954), 54 pp.] и У. В. Куайном (W. V. Quine) [*AMM* **62** (1955), 627–631]. Эта операция иногда также называется *резольвентой* (*resolvent*), поскольку Д. А. Робинсон (J. A. Robinson) использовал ее в более общем виде (но для дизъюнктов, а не импликант) в качестве основы для своего "правила резолюций" ("resolution principle") для доказательства теорем [*JACM* **12** (1965), 23–41]. Алгоритм E разработан Энн К. Эвинг (Ann C. Ewing), Д. П. Ротом (J. P. Roth) и Э. Г. Вагнером (E. G. Wagner), *AIEE Transactions*, Part 1, **80** (1961), 450–458.

32. (а) Заменим определение $\sqcup$ из упр. 31 следующей ассоциативной и коммутативной операцией над четырьмя символами $A = \{0, 1, *, \bullet\}$ для всех $a \in A$ и $x \in \{0, 1\}$:

$$* \sqcup a = a \sqcup * = a, \qquad \bullet \sqcup a = a \sqcup \bullet = x \sqcup \bar{x} = \bullet \qquad \text{и} \qquad x \sqcup x = x.$$

Пусть также $h(0) = 0$, $h(1) = 1$, $h(*) = *$ и $h(\bullet) = *$. Тогда $c = h(c_1 \sqcup \cdots \sqcup c_m)$, вычисляемое покомпонентно, представляет собой единственный подкуб, который может быть обобщенным консенсусом. [См. P. Tison, *IEEE Transactions* **EC-16** (1967), 446–456.]

(б) Например, пусть $c_j = *^{j-1}1*^{m-j}1^{j-1}0*^{m-j}$. [Последний компонент излишний. Все решения охарактеризованы Р. Х. Слоаном (R. H. Sloan), Б. Серени (B. Szörényi) и Г. Тураном (G. Turán) в *SIAM J. Discrete Math.* **21** (2008), 987–998.]

(в) Согласно (а) каждая простая импликанта однозначно соответствует подмножеству импликант, которые она "пересекает". [A. K. Chandra and G. Markowsky, *Discrete Math.* **24** (1978), 7–11.]

(г) Например, $(y_1 \wedge \bar{x}_1) \vee (y_2 \wedge x_1 \wedge \bar{x}_2) \vee \cdots \vee (y_m \wedge x_1 \wedge \cdots \wedge x_{m-1} \wedge \bar{x}_m)$, как в (б). [J.-M. Laborde, *Discrete Math.* **32** (1980), 209–212.]

33. (а) $\binom{2^n - 2^{n-k}}{m - 2^{n-k}} \Big/ \binom{2^n}{m}$. (б) Мы должны исключить случаи, когда $x_1 \wedge \cdots \wedge x_{j-1} \wedge \bar{x}_j \wedge x_{j+1} \wedge \cdots \wedge x_k$ также является импликантой. Согласно принципу включения-исключения ответ равен

$$\sum_l \binom{k}{l} (-1)^l \binom{2^n - (l+1)2^{n-k}}{m - (l+1)2^{n-k}} \Big/ \binom{2^n}{m};$$

при $k = n$ ответ упрощается до $\binom{2^n - n - 1}{m - 1} \Big/ \binom{2^n}{m}$. (См., например, 1.2.6–(24).)

34. (а) Мы имеем $c(m, n) = \sum c_j(m, n)$, где $c_j(m, n) = 2^{n-j} \binom{n}{j} \binom{2^n - 2^j}{m - 2^j} \Big/ \binom{2^n}{m}$ — среднее количество импликант с $n - j$ литералами (среднее количество подкубов размерности j в терминологии упр. 30). Ясно, что $c_0(m, n) = m$, и

$$c_1(m, n) = \frac{nm(m-1)}{2(2^n - 1)} \le \frac{mn}{2} \left(\frac{m}{2^n}\right) \le \frac{1}{2} m;$$

аналогично $c_j(m, n) \le m/(2^j j! n^{2^j - 1 - j})$. Кроме того, $p(m, n) = \sum_j p_j(m, n)$, где мы имеем

$$\begin{aligned} p_0(m, n) &= 2^n \binom{2^n - n - 1}{m - 1} \Big/ \binom{2^n}{m} = m \frac{(2^n - n - 1)^{\underline{m-1}}}{(2^n - 1)^{\underline{m-1}}} \ge m \frac{(2^n - n - m)^{m-1}}{(2^n - m)^{m-1}} \\ &\ge m \left(1 - \frac{n}{2^n - m}\right)^m \ge m \left(1 - \frac{n}{2^n - 2^n/n}\right)^{2^n/n} = m \exp\left(\frac{2^n}{n} \ln\left(1 - \frac{n^2}{2^n(n-1)}\right)\right). \end{aligned}$$

(б) Заметим, что $t = \lfloor \lg \lg n - \lg \lg(2^n/m) + \lg(4/3) \rfloor \le \lg \lg n + O(1)$ достаточно мало́. Мы многократно используем тот факт, что $\binom{2^n - j \cdot 2^t}{m - j \cdot 2^t} \Big/ \binom{2^n}{m} < \alpha_{mn}^j$, и действительно

$$\binom{2^n - j \cdot 2^t}{m - j \cdot 2^t} \Big/ \binom{2^n}{m} = \alpha_{mn}^j (1 + O(j^2 2^{2t}/m))$$

является исключительно хорошим приближением при не слишком большом j. Чтобы подтвердить указание, заметим, что $\sum_{j<t} c_j(m,n)/c_t(m,n) = O(tc_{t-1}(m,n)/c_t(m,n)) = O(t^2/(n\sqrt{\alpha_{mn}})) = O((\log\log n)^2/n^{1/3})$; а $c_{t+j}(m,n)/c_t(m,n) = O((n/(2t))^j \alpha_{mn}^{2^j-1})$. Следовательно, мы имеем $c(m,n)/c_t(m,n) \approx 1 + \frac{1}{2}(\frac{n-t}{t+1})\alpha_{mn}$, где второй член доминирует, когда α_{mn} находится в верхней части диапазона. Кроме того,

$$\sum_l \binom{n-t}{l}(-1)^l \alpha_{mn}^l \Bigl(1 + O\Bigl(\frac{l^2 2^{2t}}{m}\Bigr)\Bigr) = (1-\alpha_{mn})^{n-t} + O(n^2\alpha_{mn}(1+\alpha_{mn})^n 2^{2t}/m)$$

имеет экспоненциально малый член ошибки, поскольку $(1+\alpha_{mn})^n = O(e^{n^{1/3}}) \ll m$. Следовательно, $p(m,n)/c_t(m,n)$ асимптотически равно $e^{-n\alpha_{mn}} + \frac{1}{2}(\frac{n-t}{t+1})\alpha_{mn}e^{-n\alpha_{mn}^2}$.

(в) Здесь $\alpha_{mn} = 2^{-2^t} \approx n^{-1}\ln(t/\ln t)$; так что $c(m,n)/c_t(m,n) = 1 + O(t^{-1}\log t)$, $p(m,n)/c_t(m,n) = t^{-1}\ln t + \frac{1}{2}t^{-1}\ln t + O(t^{-1}\log\log t)$. Мы заключаем, что в этом случае

$$\frac{c(m,n)}{p(m,n)} = \frac{2}{3}\frac{\lg\lg n}{\lg\lg\lg n}\Bigl(1 + O\Bigl(\frac{\log\log\log\log n}{\log\log\log n}\Bigr)\Bigr).$$

(г) Если $n\alpha_{mn} \le \ln t - \ln\ln t$, мы имеем $p(m,n)/c(m,n) \ge p_t(m,n)/c(m,n) \ge t^{-1}\ln t + O(t^{-1}\log t)^2$. С другой стороны, если $n\alpha_{mn} \ge \ln t - \ln\ln t$, мы имеем $p(m,n)/c(m,n) \ge p_{t+1}(m,n)/c(m,n) \ge \frac{1}{2}t^{-1}\ln t + O(t^{-1}\log\log t)$.

[Средние значения $c(m,n)$ и $p(m,n)$, а также дисперсия $c(m,n)$ были впервые изучены Ф. Милето (F. Mileto) и Ж. Путцолу (G. Putzolu), *IEEE Trans.* **EC-13** (1964), 87–92; *JACM* **12** (1965), 364–375. Детальная асимптотическая информация об импликантах, простых импликантах и неприводимых DNF случайных булевых функций, когда каждое значение $f(x_1,\ldots,x_n)$ независимо равно 1 с вероятностью $p(n)$, была получена Карлом Вебером (Karl Weber), *Elektronische Informationsverarbeitung und Kybernetik* **19** (1983), 365–374, 449–458, 529–534.]

35. (а) Переставляя координаты, мы можем считать, что p-м подкубом является $0^k1^u*^v$, так что $B_p = 0^k1^u0^v$ и $S_p = 1^k0^{u+v}$. Тогда все точки $*^k1^u*^v$остаются покрытыми по индукции по p, поскольку все точки $*^{j-1}1*^{k-j}1^u*^v$ были покрыты при $1 \le j \le k$.

(б) j- и k-й подкубы отличаются в каждой координатной позиции, где $B_j \,\&\, S_k$ не равно нулю. С другой стороны, если $B_j \& S_k$ равно нулю, точка $\bar{S}_k$ подкуба k лежит в предыдущем подкубе согласно (а), поскольку мы имеем $\bar{S}_k \supseteq B_j$.

(в) Из списка 1100, 1$\underline{0}$11, $\underline{0}$011 (с подчеркнутыми битами каждого S_k) мы получаем ортогональную DNF $(x_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2 \wedge x_3 \wedge x_4) \vee (\bar{x}_1 \wedge x_3 \wedge x_4)$.

(г) Имеется восемь решений; например, $(01100, 0\underline{0}110, 00\underline{0}11, 11\underline{0}1\underline{0}, 11\underline{0}0\underline{0})$.

(д) $(001100, 011\underline{0}00, 00\underline{0}110, 11\underline{0}\underline{0}10, 11\underline{0}\underline{0}\underline{0}0, \underline{0}1\underline{0}\underline{0}11, 0\underline{0}\underline{0}\underline{0}11)$ является симметричным решением. Имеется гораздо больше возможных вариантов; например, 42 перестановки битовых кодов $\{110000, 011000, 001100, 000110, 000011, 110010, 011010\}$ являются развертками.

[Концепцию развертки для монотонных булевых функций ввели Майкл О. Болл (Michael O. Ball) и Д. Скотт Прован (J. Scott Provan), *Operations Research* **36** (1988), 703–715, которые рассмотрели много важных ее применений.]

36. Если $j < k$, мы имеем $B_j = \alpha1\beta$ и $B_k = \alpha0\gamma$ для некоторых строк α, β, γ. Образуем последовательность $x_0 = \alpha1\gamma$, $x_1 = x_0'$, $\ldots$, $x_l = x_{l-1}'$, где $x_l = \alpha00^{|\gamma|}$. Мы имеем $f(x_0) = 1$, поскольку $x_0 \supseteq B_k$, но $f(x_l) = 0$, так как $x_l \subseteq B_j'$. Так что строка x_i, где $f(x_i) = 1$ и $f(x_{i+1}) = \cdots = f(x_l) = 0$, находится в B. Она предшествует B_k и доказывает, что $B_j \,\&\, S_k \supseteq 0^{|\alpha|}10^{|\beta|}$.

[Это построение и части упр. 35 взяты из статьи E. Boros, Y. Crama, O. Ekin, P. L. Hammer, T. Ibaraki, and A. Kogan, *SIAM J. Discrete Math.* **13** (2000), 212–226.]

37. Порядок развертки (000011, 0011$\underline{0}$1, 00110$\underline{0}$, 11$\underline{0}$1$\underline{0}$1, 11$\underline{0}$10$\underline{0}$, 110$\underline{00}$1, 110$\underline{00}$$\underline{0}$) обобщается на все n. Имеется также интересное решение, не основанное на развертке, наподобие циклически симметричного (110***, 1110**, **110*, **1110, 0***11, 10**11, 111111).

Для получения нижней границы назначим вес $w_x = -\prod_{j=1}^{n}(x_{2j-1} + x_{2j} - 3x_{2j-1}x_{2j})$ каждой точке x и заметим, что сумма w_x по всем x в любом подкубе равна 0 или ± 1. (Достаточно проверить этот любопытный факт для каждого из девяти возможных подкубов при $n = 1$.) Теперь выберем множество непересекающихся подкубов, которые разбивают множество $F = \{x \mid f(x) = 1\}$; мы имеем

$$\sum_{\text{Выбранное } C} 1 \;\ge\; \sum_{\text{Выбранное } C} \sum_{x\in C} w_x \;=\; \sum_{x\in F} w_x \sum_{\text{Выбранное } C} [x \in C] \;=\; \sum_{x\in F} w_x.$$

Имеется $\binom{n}{k}2^{n-k}$ векторов x с ровно k парами $x_{2j-1}x_{2j} = 1$ и ненулевым весом. Их веса равны $(-1)^{k-1}$, и они лежат в F, за исключением случая $k = 0$. Следовательно, $\sum_{x\in F} w_x = \sum_{k>0}\binom{n}{k}2^{n-k}(-1)^{k-1} = 2^n - (2-1)^n$.

[См. M. O. Ball and G. L. Nemhauser, *Mathematics of Operations Research* **4** (1979), 132–143.]

38. Определенно, нет; DNF выполнима тогда и только тогда, когда имеет хотя бы одну импликанту. Сложной задачей в случае DNF является выяснение, не представляет ли она собой *тавтологию* (т. е. всегда истинна).

39. Свяжем переменные $y_1, \ldots, y_N$ с каждым внутренним узлом в прямом порядке обхода, так что каждый узел дерева соответствует ровно одной переменной F. Для каждого внутреннего узла y с дочерними узлами (l, r) и помеченного бинарным оператором $\circ$ построим четыре 3CNF-дизъюнкта $c_{00} \wedge c_{01} \wedge c_{10} \wedge c_{11}$, где

$$c_{pq} = (y^{\overline{p\circ q}N} \vee l^{pN} \vee r^{qN}),$$

а N означает дополнение (так что $x^{0N} = x$ и $x^{1N} = \bar{x}$). Эти дизъюнкты на самом деле указывают, что $y = l \circ r$; например, если $\circ$ представляет собой $\wedge$, четырьмя дизъюнктами являются $(\bar{y} \vee l \vee r) \wedge (\bar{y} \vee l \vee \bar{r}) \wedge (\bar{y} \vee \bar{l} \vee r) \wedge (y \vee \bar{l} \vee \bar{r})$. Наконец, добавим еще один дизъюнкт, $(y_1 \vee y_1 \vee y_1)$, чтобы обеспечить $F = 1$.

Любое большое число можно сформировать путем простого усложнения троек.
...Возьмем факт из четырех частей: А продал Б товар В по цене Г.
Он состоит из двух фактов: первый — что А совершил
с Б некоторое действие, которое можно назвать Д; и второй —
что действие Д есть продажа В по цене Г.

— ЧАРЛЬЗ С. ПИРС (CHARLES S. PEIRCE),
Отгадка в загадке (*A Guess at the Riddle*) (1887)

40. Следуя указанию, A гласит '$u < v \oplus v < u$', а B гласит '$u < v \wedge v < w \Rightarrow u < w$'. Так что $A \wedge B$ утверждает, что существует линейное упорядочение вершин $u_1 < u_2 < \cdots < u_n$. (Имеется $n!$ способов выполнения $A \wedge B$.) Далее, C гласит, что q_{uvw} эквивалентно $u < v < w$; так что D утверждает, что u и w не являются последовательными в этом упорядочении, когда $u \neq w$. Таким образом, $A \wedge B \wedge C \wedge D$ выполнимо тогда и только тогда, когда существует линейное упорядочение, в котором все несмежные вершины не являются последовательными (т. е. в котором все последовательные вершины являются смежными).

41. Решение 0. '$[m \le n]$' и есть искомая формула, но это решение неспортивное, не в духе нашего упражнения.

Решение 1. Пусть x_{jk} означает, что голубь j занимает гнездо k. Тогда дизъюнктами являются $(x_{j1} \vee \cdots \vee x_{jn})$ для $1 \le j \le m$ и $(\bar{x}_{ik} \vee \bar{x}_{jk})$ для $1 \le i < j \le m$ и $1 \le k \le n$. [См.

S. A. Cook и R. A. Reckhow, *J. Symbolic Logic* **44** (1979), 36–50; A. Haken, *Theoretical Comp. Sci.* **39** (1985), 297–308.]

Решение 2. Допустим, что $n = 2^t$, и пусть голубь j занимает гнездо $(x_{j1} \ldots x_{jt})_2$. Дизъюнкты $((x_{i1} \oplus x_{j1}) \vee \cdots \vee (x_{it} \oplus x_{jt}))$ для $1 \le i < j \le m$ можно разместить в CNF-форме $(y_{ij1} \vee \cdots \vee y_{ijt})$, как в упр. 39, вводя вспомогательные дизъюнкты $(\bar{y}_{ijk} \vee x_{ik} \vee x_{jk}) \wedge (y_{ijk} \vee x_{ik} \vee \bar{x}_{jk}) \wedge (y_{ijk} \vee \bar{x}_{ik} \vee x_{jk}) \wedge (\bar{y}_{ijk} \vee \bar{x}_{ik} \vee \bar{x}_{jk})$. Общий размер этой конъюнктивной нормальной формы — $\Theta(m^2 \log n)$, по сравнению с $\Theta(m^2 n)$ в решении 1. Если n не является степенью 2, $O(m \log n)$ дополнительных дизъюнктов размером $O(\log n)$ уберут неподходящие значения.

42. $(\bar{x} \vee y) \wedge (\bar{z} \vee x) \wedge (\bar{y} \vee \bar{z}) \wedge (z \vee z)$.

43. Вероятно, нет, поскольку каждая 3SAT-задача может быть приведена к этому виду. Например, дизъюнкт $(x_1 \vee x_2 \vee \bar{x}_3)$ можно заменить на $(x_1 \vee \bar{y} \vee \bar{x}_3) \wedge (\bar{y} \vee \bar{x}_2) \wedge (y \vee x_2)$, где y представляет собой новую переменную (по сути, эквивалентную $\bar{x}_2$).

44. Предположим, что из $f(x) = f(y) = 1$ вытекает $f(x \,\&\, y) = 1$, а также что, скажем, $c = x_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4$ является простым дизъюнктом f. Тогда $c' = \bar{x}_1 \vee x_2 \vee \bar{x}_3 \vee \bar{x}_4$ *не* является дизъюнктом; в противном случае $c \wedge c' = x_2 \vee \bar{x}_3 \vee \bar{x}_4$ также было бы дизъюнктом, что противоречит условию простоты. Так что существует вектор y, такой, что $f(y) = 1$ и $y_1 = 1$, $y_2 = 0$, $y_3 = y_4 = 1$. Аналогично существует z, такой, что $f(z) = 1$ и $z_1 = 0$, $z_2 = 1$, $z_3 = z_4 = 1$. Но тогда $f(y \,\&\, z) = 1$, и c не является дизъюнктом. Те же самые рассуждения работают для дизъюнкта c, который имеет иное число литералов, пока как минимум два из этих литералов не являются дополнениями.

45. (а) Функция Хорна $f(x_1, \ldots, x_n)$ неопределена тогда и только тогда, когда она не равна определенной функции Хорна $g(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) \vee (x_1 \wedge \cdots \wedge x_n)$. Так что $f \leftrightarrow g$ представляет собой взаимно однозначное соответствие между неопределенными и определенными функциями Хорна. (б) Если функция f монотонна, ее дополнение $\bar{f}$ либо идентично 1, либо является неопределенной функцией Хорна.

46. Алгоритм C помещает в ядро 88 пар `xy`: когда `x` = `a`, `b`, `c`, `0` или `1`, идущим следом символом `y` может быть любой, кроме `(`. Когда `x` = `(`, `*`, `/`, `+`, `-`, мы можем иметь `y` = `(`, `a`, `b`, `c`, `0`, `1`; а также `y` = `-`, когда `x` = `(`, `+` или `-`. Наконец, корректными парами, начинающимися с `x` = `)`, являются `)+`, `)-`, `)*`, `)/`, `))`.

47. Порядок, в котором алгоритм C вводит вершины в ядро, представляет собой топологическую сортировку, поскольку все предшественники k утверждены до того, как алгоритм устанавливает `TRUTH(`x_k`)` $\leftarrow 1$. Но алгоритм 2.2.3T использует очередь вместо стека, так что создаваемое им упорядочение обычно отличается от упорядочения алгоритма C.

48. Пусть $\perp$ — новая переменная, и заменим каждый неопределенный дизъюнкт Хорна определенным, выполнив для него операцию ИЛИ с этой новой переменной. (Например, '$\bar{w} \vee \bar{y}$' превращается в '$\bar{w} \vee \bar{y} \vee \perp$', т. е. '$w \wedge y \Rightarrow \perp$'; определенные дизъюнкты Хорна остаются неизменными.) Затем применим алгоритм C. Исходные дизъюнкты являются невыполнимыми тогда и только тогда, когда $\perp$ находится в ядре новых дизъюнктов. Следовательно, алгоритм можно завершить, как только он собирается установить `TRUTH(`$\perp$`)` $\leftarrow 1$.

(Студент Шустрый предложил иное решение: применить алгоритм C к функции g, построенной в ответе к упр. 45, (а), поскольку f невыполнима тогда и только тогда, когда *каждая* переменная x_j находится в ядре g. Однако неопределенные дизъюнкты f, такие как $\bar{w} \vee \bar{y}$, превращаются во много различных дизъюнктов $(\bar{w} \vee \bar{y} \vee z) \wedge (\bar{w} \vee \bar{y} \vee x) \wedge (\bar{w} \vee \bar{y} \vee v) \wedge (\bar{w} \vee \bar{y} \vee u) \wedge \cdots$ функции g, по одному для каждой переменной, отсутствующей в исходном дизъюнкте. Так что предложение Шустрого, на первый взгляд звучащее вполне элегантно, может увеличить количество дизъюнктов на множитель $\Omega(n)$.)

49. Мы имеем $f \le g$ тогда и только тогда, когда $f \wedge \bar{g}$ невыполнимо, тогда и только тогда, когда $f \wedge \bar{c}$ невыполнимо для любого дизъюнкта c функции g. Но $\bar{c}$ представляет собой И литералов, так что можно применить результаты упр. 48. [См. в H. Kleine

Büning и T. Lettmann, *Aussagenlogik: Deduktion und Algorithmen* (1994), §5.6, другие результаты, включая эффективный способ проверки, является ли g "переименованием" f, т. е. определения, существуют ли такие константы $(y_1, \ldots, y_n)$, что $f(x_1, \ldots, x_n) = g(x_1 \oplus y_1, \ldots, x_n \oplus y_n)$.]

50. См. Gabriel Istrate, *Random Structures & Algorithms* **20** (2002), 483–506.

51. Если вершина v помечена буквой A, введем $\Rightarrow A^+(v)$ и $\Rightarrow B^-(v)$; если она помечена буквой B, введем $\Rightarrow A^-(v)$ и $\Rightarrow B^+(v)$. В противном случае пусть v имеет k исходящих дуг $v \to u_1, \ldots, v \to u_k$. Введем дизъюнкты $A^-(u_j) \Rightarrow B^+(v)$ и $B^-(u_j) \Rightarrow A^+(v)$ для $1 \le j \le k$. Кроме того, если v не помечено буквой C, введем дизъюнкты $A^+(u_1) \wedge \cdots \wedge A^+(u_k) \Rightarrow B^-(v)$ и $B^+(u_1) \wedge \cdots \wedge B^+(u_k) \Rightarrow A^-(v)$. Все обеспечивающие выигрыш или проигрыш стратегии являются следствиями этих дизъюнктов. Дополнительную информацию на эту тему можно получить из упр. 2.2.3–28 и ответа к нему.

Заметим, что в принципе алгоритм C можно использовать для решения вопроса о выигрышной стратегии белых при игре в шахматы — если, конечно, не обращать внимания на тот досадный факт, что соответствующий ориентированный граф больше, чем вся физическая Вселенная.

52. При наилучшей стратегии результаты таковы (см. упр. 51).

n	(а)	(б)	(в)	(г)
2	Побеждает 0	Побеждает второй игрок	Побеждает 1	Побеждает второй игрок
3	Побеждает 0	Побеждает первый игрок	Побеждает первый игрок	Побеждает первый игрок
4	Побеждает первый игрок	Побеждает первый игрок	Побеждает первый игрок	Побеждает первый игрок
5	Побеждает второй игрок	Ничья	Ничья	1 проигрывает, начиная
6	Побеждает второй игрок	Побеждает второй игрок	1 проигрывает, начиная	1 проигрывает, начиная
7	1 проигрывает, начиная	Побеждает второй игрок	1 проигрывает, начиная	1 проигрывает, начиная
8	Ничья	Ничья	Ничья	1 проигрывает, начиная
9	Ничья	Ничья	Ничья	1 проигрывает, начиная

(Здесь "1 проигрывает, начиная" означает, что игра завершается ничьей, если первым ходит игрок 0, в противном случае 0 может выиграть.)

Комментарии. В (а) игрок 1 имеет небольшую невыгодность ситуации, связанную с тем, что $f(x) = 0$, когда $x_1 \ldots x_n$ представляет собой палиндром. Это небольшое отличие влияет на результат даже тогда, когда $n = 7$. Хотя игрок 1 кажется в лучшем положении при игре нулями в левой части таблицы, оказывается, что его первым ходом при $n = 4$ должен быть $*1**$; противоположный ход $*0**$ ведет к ничьей. Игра (б), по сути, представляет собой гонку — кто сможет убрать последнюю $*$. В игре (в) случайный выбор $x_1 \ldots x_n$ дает $f(x) = 1$ с вероятностью $F_{n+2}/2^n = \Theta((\phi/2)^n)$; в игре (г) эта вероятность стремится к нулю медленнее, как $\Theta(1/\log n)$. Тем не менее шансы игрока 1 выше в (в), чем в (г), когда $n = 2$, 5, 8 и 9; и не хуже в прочих случаях.

53. (а) Следует изменить день 1 или день 2 на день 3. (б, е) Несколько возможных вариантов; например, изменить день 2 на день 3. (в) Этот случай проиллюстрирован на рис. 6. Следует изменить Desert либо Excalibur на Aladdin. (г) Следует изменить Caesars или Excalibur на Aladdin. (д) Следует изменить Bellagio или Desert на Aladdin. Само собой разумеется, Williams, отсутствующий в цикле (42), не несет ответственности за конфликты.

54. Если и x, и $\bar{x}$ находятся в S, то $u \in S \iff \bar{u} \in S$, поскольку из существования путей от x до $\bar{x}$ и от $\bar{x}$ до x, и от x до u и от u до x вытекает существование путей от $\bar{u}$ до $\bar{x}$ и от $\bar{x}$ до $\bar{u}$, а следовательно, от u до $\bar{u}$ и от $\bar{u}$ до u.

55. (а) Необходимыми и достаточными условиями для успешного переименования дизъюнкта, такого как $x_1 \vee \bar{x}_2 \vee x_3 \vee \bar{x}_4$, являются $(y_1 \vee \bar{y}_2) \wedge (y_1 \vee y_3) \wedge (y_1 \vee \bar{y}_4) \wedge (\bar{y}_2 \vee y_3) \wedge (\bar{y}_2 \vee \bar{y}_4) \wedge (y_3 \vee \bar{y}_4)$. Аналогичное множество $\binom{k}{2}$ дизъюнктов длиной 2 из переменных $\{y_1, \ldots, y_n\}$ соответствует каждому дизъюнкту длиной k из переменных $\{x_1, \ldots, x_n\}$. [H. R. Lewis, *JACM* **25** (1978), 134–135.]

(б) Заданный дизъюнкт длиной $k > 3$ от $\{x_1, \ldots, x_n\}$ может быть преобразован вместо упомянутых выше $\binom{k}{2}$ дизъюнктов в $3(k-2)$ дизъюнктов длиной 2 путем введения $k-3$ новых переменных $\{t_2, \ldots, t_{k-2}\}$, что проиллюстрировано далее для дизъюнкта $x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5$:

$$(y_1 \vee y_2) \wedge (y_1 \vee t_2) \wedge (y_2 \vee t_2) \wedge (\bar{t}_2 \vee y_3) \wedge (\bar{t}_2 \vee t_3) \wedge (y_3 \vee t_3) \wedge (\bar{t}_3 \vee y_4) \wedge (\bar{t}_3 \vee y_5) \wedge (y_4 \vee y_5).$$

В общем случае дизъюнкты из $x_1 \vee \cdots \vee x_k$ становятся $(\bar{t}_{j-1} \vee y_j) \wedge (\bar{t}_{j-1} \vee t_j) \wedge (y_j \vee t_j)$ для $1 < j < k$, но t_1 заменяется на $\bar{y}_1$, а t_{k-1} — на y_k; заменим y_j на $\bar{y}_j$, если $\bar{x}_j$ появляется вместо x_j. Будем выполнять эти действия для каждого заданного дизъюнкта, используя различные вспомогательные переменные t_j для разных дизъюнктов; результатом является формула в 2CNF-форме, которая имеет длину $< 3m$ и является выполнимой тогда и только тогда, когда возможно переименование Хорна. Теперь применим теорему K.

[См. B. Aspvall, *J. Algorithms* **1** (1980), 97–103. Одним следствием этого, отмеченным Г. Кляйне Бюнингом (H. Kleine Büning) и Т. Леттманном (T. Lettmann) в *Aussagenlogik: Deduktion und Algorithmen* (1994), теорема 5.2.4, является то, что любая выполнимая формула в 2CNF может быть переименована в дизъюнкты Хорна. Заметим, что две конъюнктивные нормальные формы для одной и той же функции могут приводить к разным исходам; например, $(x \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z}) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee z)$ в действительности является функцией Хорна, но дизъюнкты в этом представлении не могут быть преобразованы в форму Хорна путем операции дополнения.]

56. Здесь $f(x, y, z)$ соответствует показанному ориентированному графу (подобному показанному на рис. 6), и он может быть упрощен до $y \wedge (\bar{x} \vee z)$. Каждая вершина является сильно связным компонентом. Таким образом, формула верна по отношению к кванторам $\exists\exists\exists$, $\exists\exists\forall$, $\forall\exists\exists$ и ложна в прочих случаях $\forall\exists\forall$, (любой)$\forall$(любой). В общем случае восемь возможных вариантов можно разместить в вершинах куба, так что каждое изменение $\exists$ на $\forall$ делает формулу ложной со все большей вероятностью.

57. Образуя ориентированный граф, как в теореме K, мы можем доказать, что квантифицированная формула выполняется тогда и только тогда, когда (i) ни один сильно связный компонент не содержит одновременно x и $\bar{x}$; (ii) не существует пути от одной универсальной переменной (переменной, связанной с квантором всеобщности) x к другой универсальной переменной y или к ее дополнению $\bar{y}$; (iii) ни один сильно связный компонент, содержащий универсальную переменную x, не содержит также экзистенциальную переменную (переменную, связанную с квантором существования) v или ее дополнением $\bar{v}$, когда '$\exists v$' находится слева от '$\forall x$'. Эти три условия с очевидностью необходимы, и легко проверяемы после нахождения сильно связных компонентов.

Чтобы показать их достаточность, сначала заметим, что если S является сильно связным компонентом только с экзистенциальными литералами, условие (i) позволяет нам установить все их равными, как в теореме K. В противном случае S имеет только один универсальный литерал, $u_j = x_j$ или $u_j = \bar{x}_j$; все прочие литералы в S являются экзистенциальными и объявлены справа от x_j, так что мы можем приравнять их к u_j.

И все пути внутрь S в таком случае идут от чисто экзистенциальных сильно связных компонентов, значения которых могут быть установлены равными 0, поскольку дополнения таких сильно связных компонентов не могут также вести в S; если из v и $\bar{v}$ вытекает u_j, то из $\bar{u}_j$ вытекает $\bar{v}$ и v.

[*Information Proc. Letters* **8** (1979), 121–123. Напротив, М. Кром (M. Krom) в *J. Symbolic Logic* **35** (1970), 210–216, доказал, что аналогичная задача в исчислении предикатов первого порядка (где параметризованные предикаты занимают место простых булевых переменных, а квантификация выполняется над параметрами) в действительности в общем случае неразрешима.]

58. Мы можем считать, что каждый дизъюнкт является определенным, вводя '$\bot$', как в упр. 48, и помещая слева '$\forall\bot$'. Назовем универсальные переменные x_0, x_1, ..., x_m (где x_0 представляет собой $\bot$), а экзистенциальные переменные — y_1, ..., y_n. Пусть '$u \prec v$' означает, что переменная u появляется в списке кванторов слева от переменной v. Удалим $\bar{x}_j$ из любого дизъюнкта, литералом которого без надчеркивания является y_k, когда $y_k \prec x_j$. Тогда пусть C_j при $0 \le j \le m$ представляет собой ядро дизъюнктов Хорна при присоединении дополнительных дизъюнктов $(x_0) \wedge \cdots \wedge (x_{j-1}) \wedge (x_{j+1}) \wedge \cdots \wedge (x_m) \wedge \bigwedge\{(y_k) \mid y_k \prec x_j \text{ и } y_k \in C_0\}$. (Другими словами, C_j говорит нам, что́ может быть выведено, когда все x, за исключением x_j, рассматриваются как истинные.) Мы утверждаем, что данная формула истинна тогда и только тогда, когда $x_j \notin C_j$ для $0 \le j \le m$.

Для доказательства этого утверждения сначала заметим, что формула, определенно, ложна, если $x_j \in C_j$ для некоторого j. (Когда $y_k \in C_0$ и $y_k \prec x_j$ и $x_i = 1$ для $i \ne j$, мы должны установить $y_k \leftarrow 1$.) В противном случае, чтобы сделать формулу истинной, мы можем выбрать каждый y_k следующим образом: если $y_k \notin C_0$, установить $y_k \leftarrow 0$; в противном случае установить $y_k \leftarrow \bigwedge\{x_j \mid y_k \notin C_j\}$. Заметим, что y_k зависит от x_j, только когда $x_j \prec y_k$. Каждый дизъюнкт c с ненадчеркнутым литералом x_j теперь истинен: если $x_j = 0$, в c обнаруживается некоторое $\bar{y}_k$, для которого $y_k \notin C_j$, поскольку $x_j \notin C_j$; следовательно, $y_k = 0$. И каждый дизъюнкт c с ненадчеркнутым литералом y_k также является истинным: если $y_k = 0$, то мы либо имеем $y_k \notin C_0$, и в этом случае некоторое $\bar{y}_l$ в c не является элементом C_0, следовательно, $y_l = 0$; либо $y_k \in C_0 \setminus C_j$ для некоторого j, и в этом случае некоторое $x_j = 0$ и либо $\bar{x}_j$ находится в c, либо в c имеется некоторое $\bar{y}_l$, где $y_l \notin C_j$, делая $y_l = 0$.

[Это решение принадлежит Т. Далхаймеру (T. Dahlheimer). См. M. Karpinski, H. Kleine Büning, and P. H. Schmitt, *Lecture Notes in Comp. Sci.* **329** (1988), 129–137; H. Kleine Büning, K. Subramani, and X. Zhao, *Lecture Notes in Comp. Sci.* **2919** (2004), 93–104.]

59. По индукции по n: предположим, что $f(0, x_2, \ldots, x_n)$ приводит к квантифицированным результатам y_1, ..., $y_{2^{n-1}}$, в то время как $f(1, x_2, \ldots, x_n)$ приводит аналогично к z_1, ..., $z_{2^{n-1}}$. Тогда $\exists x_1 f(x_1, x_2, \ldots, x_n)$ приводит к $y_1 \vee z_1$, ..., $y_{2^{n-1}} \vee z_{2^{n-1}}$, а $\forall x_1 f(x_1, x_2, \ldots, x_n)$ приводит к $y_1 \wedge z_1$, ..., $y_{2^{n-1}} \wedge z_{2^{n-1}}$. Теперь воспользуемся тем фактом, что $(y \vee z) + (y \wedge z) = y + z$. [См. *Proc. Mini-Workshop on Quantified Boolean Formulas* **2** (QBF-02) (Cincinnati: May 2002), 1–16.]

60. (а) и (б). (в) Всегда 0. (г) Всегда 1. (д) Равно $\overline{\langle xyz \rangle}$. (е) Равно $\bar{x} \vee \bar{y} \vee \bar{z}$.

61. Истинно. Это очевидно и при $w = 0$, и при $w = 1$.

62. Поскольку $\{x_1, x_2, x_3\} \subseteq \{0, 1\}$, из соображений симметрии можно считать, что x_1 равно x_2. Тогда либо $f(x_1, x_1, x_3, x_4, \ldots, x_n) = f(x_1, x_1, x_1, x_4, \ldots, x_n)$, либо $f(x_1, x_1, x_3, x_4, \ldots, x_n) = f(x_3, x_1, x_3, x_4, \ldots, x_n)$, всего лишь в предположении, что f монотонна по первым трем переменным.

63. $\langle xyz \rangle = \langle xxyyz \rangle$. *Примечание*: Эмиль Пост (Emil Post) доказал, что на самом деле одной подпрограммы вычисления *любой* нетривиальной монотонной самодуальной функции

достаточно для вычисления их всех. (По индукции по n как минимум один подходящий способ вызова такой n-арной подпрограммы даст $\langle xyz\rangle$.)

64. [*FOCS* **3** (1962), 149–157.] (а) Если f монотонна и самодуальна, теорема P гласит, что $f(x) = x_k$ или $f(x) = \langle f_1(x)f_2(x)f_3(x)\rangle$. Следовательно, условие выполняется либо непосредственно, либо по индукции. И наоборот, если условие выполняется, значит, f монотонна (когда x и y отличаются только одним битом) и самодуальна (когда они отличаются во всех битах).

(б) Нам просто нужно показать, что можно определить f в одной новой точке, не вызвав конфликт. Пусть x является лексикографически наименьшей точкой, где $f(x)$ не определена. Если $f(\bar{x})$ определена, установим $f(x) = \overline{f(\bar{x})}$. В противном случае если $f(x') = 1$ для некоторого $x' \subseteq x$, установим $f(x) = 1$; в противном случае установим $f(x) = 0$. Тогда условие все еще будет выполняться.

65. Если семейство $\mathcal{F}$ является максимальным перекрывающимся, то мы имеем (i) $X \in \mathcal{F} \implies \overline{X} \notin \mathcal{F}$, где $\overline{X}$ представляет собой дополнительное множество $\{1, 2, \ldots, n\} \setminus X$; (ii) $X \in \mathcal{F}$ и $X \subseteq Y \implies Y \in \mathcal{F}$, поскольку $\mathcal{F} \cup \{Y\}$ является перекрывающимся; и (iii) $X \notin \mathcal{F} \implies \overline{X} \in \mathcal{F}$, поскольку $\mathcal{F} \cup \{X\}$ должно содержать элемент $Y \subseteq \overline{X}$. И наоборот, можно без труда доказать, что любое семейство $\mathcal{F}$, удовлетворяющее (i) и (ii), является перекрывающимся, и максимальным, если оно удовлетворяет также условию (iii).

Это интересно. Все три утверждения на языке булевых функций очень просты: (i) $f(x) = 1 \implies f(\bar{x}) = 0$; (ii) $x \subseteq y \implies f(x) \le f(y)$; (iii) $f(x) = 0 \implies f(\bar{x}) = 1$.

66. [T. Ibaraki and T. Kameda, *IEEE Transactions on Parallel and Distributed Systems* **4** (1993), 779–794.] Каждое семейство, обладающее тем свойством, что из $Q \subseteq Q'$ вытекает $Q = Q'$, очевидно, соответствует простым импликантам монотонной булевой функции f. Следующее условие, $Q \cap Q' \ne \emptyset$, соответствует отношению $f(\bar{x}) \le \overline{f(x)}$, поскольку $f(\bar{x}) = f(x) = 1$ выполняется тогда и только тогда, когда и x, и $\bar{x}$ делают простые импликанты истинными.

Если комитеты $\mathcal{C}$ и $\mathcal{C}'$ соответствуют таким способом функциям f и f', то $\mathcal{C}$ доминирует над $\mathcal{C}'$ тогда и только тогда, когда $f \ne f'$ и $f'(x) \le f(x)$ для всех x. Тогда f' не является самодуальной, поскольку существует x, такое, что $f'(\bar{x}) = 0$, $f(\bar{x}) = 1$; а мы имеем $f(x) = 0$, следовательно, $f'(x) = 0$.

И наоборот, если f' не самодуальна, существует y, такое, что $f'(y) = f'(\bar{y}) = 0$. Если $y = 0\ldots0$, комитет $\mathcal{C}'$ пуст, и над ним доминирует каждый другой комитет. В противном случае определим $f(x) = f'(x) \vee [x \supseteq y]$. Тогда f является монотонной, и $f(\bar{x}) \le \overline{f(x)}$ для всех x; так что она соответствует комитету, доминирующему над $\mathcal{C}'$.

67. (а) Черный Y в t приводит к черному Y в t^*, поскольку соседние черные камни a — b — c в t дают два соседних черных камня в t^*. Аналогично, черный Y в t^* приводит к черному Y в t.

(б) Эта формула следует из (а) и из того факта, что $(t_{abc})_{def} = t_{(a+d)(b+e)(c+f)} = (t_{def})_{abc}$. [Шенстед (Schensted) сформулировал результат этого упражнения, а также упр. 62 и 69, в своем 28-страничном письме Мартину Гарднеру (Martin Gardner) от 21 января 1979 года. Милнор (Milnor) писал Гарднеру о соответствующей игре, называемой "Треугольник", 26 марта 1957 года.

68. На рисунке справа приведено одно из 258 594 решений для $n = 15$, содержащее 59 черных камней. (Ответы для $1 \le n \le 15$ представляют собой соответственно 2, 3, 4, 6, 8, 11, 14, 18, 23, 27, 33, 39, 45, 52 и 59. Простые импликанты для этих функций могут быть представлены довольно малыми ZDD (бинарными диаграммами принятия решений с отброшенными нулями); см. раздел 7.1.4.)

69. Доказательство теоремы P показывает, что необходимо доказать только $Y(T) \le f(x)$. Y в T означает, что мы должны получить как минимум одну переменную в каждой p_j. Следовательно, $f(\bar{x}_1, \ldots, \bar{x}_n) = 0$ и $f(x_1, \ldots, x_n) = 1$.

70. Самодуальность g очевидна для произвольных t, если f самодуальна: $\overline{g(\bar{x})} = (\overline{f(\bar{x})} \vee [\bar{x} = t]) \wedge [\bar{x} \ne \bar{t}] = (f(x) \vee [x = \bar{t}]) \wedge [x \ne t] = (f(x) \wedge [x \ne t]) \vee ([x = \bar{t}] \wedge [x \ne t]) = g(x)$.

Пусть $x = x_1 \ldots x_{j-1} 0 x_{j+1} \ldots x_n$ и $y = x_1 \ldots x_{j-1} 1 x_{j+1} \ldots x_n$; для монотонности мы должны доказать, что $g(x) \le g(y)$. Если $x = t$ или $y = t$, мы имеем $g(x) = 0$; если $x = \bar{t}$ или $y = \bar{t}$, мы имеем $g(y) = 1$; в противном случае $g(x) = f(x) \le f(y) = g(y)$. [*European J. Combinatorics* **16** (1995), 491–501; независимо открыто Я. К. Биохом (J. C. Bioch) и Т. Ибараки (T. Ibaraki), *IEEE Transactions on Parallel and Distributed Systems* **6** (1995), 905–914.]

71. $\langle\langle xyz\rangle uv\rangle = \langle\langle\langle xyz\rangle uv\rangle uv\rangle = \langle\langle\langle yuv\rangle x\langle zuv\rangle\rangle uv\rangle = \langle\langle yuv\rangle\langle xuv\rangle\langle\langle zuv\rangle uv\rangle\rangle = \langle\langle xuv\rangle\langle yuv\rangle \langle zuv\rangle\rangle$.

72. Для (58) $v = \langle uvu\rangle = u$. Для (59) $\langle uyv\rangle = \langle vu\langle xuy\rangle\rangle = \langle\langle vux\rangle uy\rangle = \langle xuy\rangle = y$. И для (60) $\langle xyz\rangle = \langle\langle xuv\rangle yz\rangle = \langle x\langle uyz\rangle\langle vyz\rangle\rangle = \langle xyy\rangle = y$.

73. (а) Если $d(u,v) = d(u,x) + d(x,v)$, то очевидно, что мы получим кратчайший путь в виде u — $\cdots$ — x — $\cdots$ — v. И наоборот, если $[uxv]$, то пусть u — $\cdots$ — x — $\cdots$ — v представляет собой кратчайший путь с l шагами до x, после чего следуют m шагов до v. Тогда $d(u,v) = l + m \ge d(u,x) + d(x,v) \ge d(u,v)$.

(б) Для всех z справедливо $\langle zxu\rangle = \langle z\langle vux\rangle\langle yux\rangle\rangle = \langle\langle zvy\rangle ux\rangle \in \{\langle yux\rangle, \langle vux\rangle\} = \{u, x\}$.

(в) Можно считать, что $d(x,u) \ge d(x,v) > 0$. Пусть u — $\cdots$ — y — v является кратчайшим путем и пусть $w = \langle xuy\rangle$. Тогда $\langle vxw\rangle = \langle v\langle vux\rangle\langle wux\rangle\rangle = \langle\langle vvw\rangle ux\rangle = \langle vux\rangle = x$, так что $x \in [w\,..\,v]$. Мы имеем $[uwy]$, поскольку $d(u,y) < d(u,v)$ и $w \in [u\,..\,y]$. Если $w \ne u$, мы имеем $d(w,v) < d(u,v)$; следовательно, $[wxv]$; следовательно, $[uxv]$. Если $w = u$, мы имеем x — u согласно (б). Но $d(x,u) \ge d(x,v)$; следовательно, x — v и $[uxv]$.

(г) Пусть $y = \langle uxv\rangle$. Поскольку $y \in [u\,..\,x]$, мы имеем $d(u,x) = d(u,y) + d(y,x)$ согласно (а) и (в). Аналогично $d(u,v) = d(u,y) + d(y,v)$ и $d(x,v) = d(x,y) + d(y,v)$. Но эти три уравнения вместе с $d(u,v) = d(u,x) + d(x,v)$ дают $d(x,y) = 0$. [*Proc. Amer. Math. Soc.* **12** (1961), 407–414.]

74. $w = \langle yxw\rangle = \langle yx\langle zxw\rangle\rangle = \langle yx\langle zx\langle yzw\rangle\rangle\rangle = \langle\langle yxz\rangle x\langle yzw\rangle\rangle = \langle x\langle xyz\rangle\langle wyz\rangle\rangle = \langle\langle xxw\rangle yz\rangle = \langle xyz\rangle$ согласно (55), (55), (55), (52), (51), (53) и (50).

75. (а) Если $w = \langle xxy\rangle$, мы имеем $[xwx]$ согласно (iii), следовательно, $w = x$ согласно (i).

(б) Аксиома (iii) и п. (а) говорят нам о том, что $[xxy]$ всегда истинно. Так что мы можем установить $y = x$ в (ii), чтобы заключить, что $[uxv] \iff [vxu]$. Определение $\langle xyz\rangle$ в (iii) является, таким образом, идеально симметричным относительно x, y и z.

(в) Согласно определению $\langle uxv\rangle$ в (iii) мы имеем $x = \langle uxv\rangle$ тогда и только тогда, когда $[uxx]$, $[uxv]$ и $[xxv]$. Но мы знаем, что $[uxx]$ и $[xxv]$ всегда истинны.

(г) На этом и последующих шагах мы будем строить одну или несколько вспомогательных точек M, а затем использовать алгоритм C для вывода каждого известного следствия отношений промежуточности. (Аксиомы имеют удобный вид дизъюнктов Хорна.) Например, здесь мы определим $z = \langle xyv\rangle$, так что мы знаем, что $[uxy]$, $[uyv]$, $[xzy]$, $[xzv]$ и $[yzv]$. Из этих гипотез мы выводим $[uzy]$ и $[uzv]$. Так что $z = \langle uyv\rangle = y$.

(д) Из построения из указания вытекает среди прочего, что $[utv]$, $[utz]$, $[vtz]$, $[uwv]$, $[uwz]$, $[vwz]$; следовательно, $t = w$. (Здесь может помочь компьютерная программа.) Добавление гипотез $[rws]$, $[rwz]$, $[swz]$ дает, как и требовалось, $[xyz]$; оказывается также, что $r = p$ и $s = q$.

(е) Пусть $r = \langle yuv\rangle$, $s = \langle zuv\rangle$, $t = \langle xyz\rangle$, $p = \langle xrs\rangle$, $q = \langle tuv\rangle$; тогда получается $[pqp]$. [*Proc. Amer. Math. Soc.* **5** (1954), 801–807. В качестве начальной работы по изучению

аксиом промежуточности см. E. V. Huntington и J. R. Kline, *Trans. Amer. Math. Soc.* **18** (1917), 301–325.]

76. Аксиома (i) выполняется очевидным образом, а аксиома (ii) следует из коммутативности и (52). В ответе к упр. 74 из тождества $\langle xyz\rangle = \langle x\langle xyz\rangle\langle wyz\rangle\rangle$ выводится (iii); так что нужно только проверить формулу $\langle x\langle xyz\rangle\langle wyz\rangle\rangle = \langle\langle yxz\rangle x\langle wyz\rangle\rangle = \langle\langle\langle yxz\rangle xz\rangle x\langle wyz\rangle\rangle = \langle\langle yxz\rangle x\langle zx\langle wyz\rangle\rangle\rangle = \langle x\langle xyz\rangle\langle z\langle xyz\rangle w\rangle\rangle = \langle\langle x\langle xyz\rangle z\rangle\langle xyz\rangle w\rangle = \langle\langle xyz\rangle\langle xyz\rangle w\rangle$.

Примечания. В исходной работе по алгебре медиан Биркхоффа (Birkhoff) и Кисса (Kiss) *Bull. Amer. Math. Soc.* **53** (1947), 749–752, принимаются (50), (51) и короткий закон дистрибутивности (53). Тот факт, что дистрибутивность вытекает из ассоциативности (52), в течение длительного срока не был понят; M. Колибяр (M. Kolibiar) и T. Маркизова (T. Marcisová), *Matematický Casopis* **24** (1974), 179–185, доказали это с использованием аксиом Шоландера, как в данном упражнении. Механический вывод (53) из (50)–(52) был обнаружен в 2005 году R. Вероффом (R. Veroff) и У. Мак-Кьюном (W. McCune) с применением расширенной программы автоматического доказательства теорем Otter.

77. (а) Предположим, что в координате r — s меток метка $l(r)$ имеет 0, а метка $l(s)$ имеет 1; тогда в этой координате левые вершины имеют 0. Если u — v — u', где u и u' находятся слева, а v справа, то $\langle uu'v\rangle$ лежит слева. Но $[u\,..\,v] \cap [u'\,..\,v] = \{v\}$, если только не $u = u'$.

(б) Это утверждение очевидно в соответствии со следствием C.

(в) Предположим, что u — v и u' — v', где u и u' находятся слева, а v и v' находятся справа. Пусть $v = v_0$ — $\cdots$ — $v_k = v'$ является кратчайшим путем и пусть $u_0 = u$, $u_k = u'$. Все вершины v_j лежат справа, согласно п. (б). Левая вершина $u_1 = \langle u_0 v_1 u_k\rangle$ должна быть общим соседом для u_0 и v_1, поскольку расстояние $d(u_0, v_1) = 2$. (У нас не может быть $u_1 = u_0$, поскольку это привело бы к существованию более краткого пути от v до v', проходящего через левую вершину u.) Следовательно, v_1 имеет ранг 1; то же самое справедливо для v_2, ..., v_{k-1}, в соответствии с такими же рассуждениями. [L. Nebeský, *Commentationes Mathematic? Universitatis Carolin?* **12** (1971), 317–325; M. Mulder, *Discrete Math.* **24** (1978), 197–204.]

(г) Эти шаги посещают все вершины v ранга 1 в порядке их расстояния $d(v, s)$ от s. Если такая v имеет пока еще непросмотренного позднего соседа u, то ранг u должен быть равен 1 или 2. Если ранг равен 1, u будет иметь как минимум два ранних соседа, а именно v и будущего `MATE(u)`. На шаге I8 решение основывается на произвольном раннем соседе w вершины u, таком, что $w \neq v$. Вершина $x = \langle svw\rangle$ имеет ранг 1 согласно п. (в). Если $x = v$, u имеет ранг 2, если только w не имеет ранг 0. Если w имеет ранг 0, то $x = v$; так что u имеет ранг 1. В противном случае $d(x, s) < d(v, s)$ и ранг w был корректно определен при посещении x. Если w имеет ранг 1, u лежит на кратчайшем пути от v до w; если w имеет ранг 2, w лежит на кратчайшем пути от u до s. В обоих случаях u и w имеет один и тот же ранг согласно п. (в).

(д) Алгоритм удаляет все ребра, эквивалентные r — s, согласно пп. (а) и (г). Ясно, что это удаление разъединяет граф; получившиеся две части остаются выпуклыми согласно п. (б), так что они связные и в действительности представляют собой медианные графы. На шаге I7 записываются все существенные отношения между двумя частями, поскольку здесь рассматриваются все исчезнувшие 4-циклы. Каждая часть корректно маркируется по индукции по числу вершин.

78. Каждый раз, когда на шаге I4 появляется вершина v, она теряет одного из своих соседей u_j. Каждое из этих ребер v — u_j соответствует отличной координате меток, так что можно считать, что $l(v)$ имеет вид $\alpha 1^k$ для некоторой бинарной строки α. Тогда метками u_1, u_2, ..., u_k являются $\alpha 01^{k-1}$, $\alpha 101^{k-2}$, ..., $\alpha 1^{k-1}0$. Покомпонентно беря

медианы, мы можем теперь доказать, что у вершин графа встречаются все 2^k меток вида $\alpha\beta$, поскольку $\langle(\alpha\beta)(\alpha\beta')(0\ldots0)\rangle$ является битовой строкой $\alpha(\beta \mathbin{\&} \beta')$.

79. (а) Если $l(v) = k$, то ровно $\nu(k)$ меньших вершин являются соседями v.

(б) В битовой позиции j для $0 \le j < \lceil \lg n \rceil$ встречается не более $\lfloor n/2 \rfloor$ единиц.

(в) Предположим, что ровно k вершин имеют метки, начинающиеся с 0. Этой битовой позиции соответствует не более $\min(k, n-k)$ ребер, и имеется не более $f(k)+f(n-k)$ других ребер. Но

$$f(n) = \max_{0\le k\le n} \bigl(\min(k, n-k) + f(k) + f(n-k)\bigr),$$

поскольку функция $g(m,n) = f(m+n) - m - f(m) - f(n)$ удовлетворяет рекуррентному соотношению

$$g(2m+a, 2n+b) = ab + g(m+a, n) + g(m, n+b) \qquad \text{для } 0 \le a, b \le 1.$$

Отсюда по индукции следует, что $g(m,m) = g(m,m+1) = 0$ и что $g(m,n) \ge 0$ при $m \le n$. [*Annals of the New York Academy of Sciences* **175** (1970), 170–186; D. E. Knuth, *Proc. IFIP Congress 1971* (1972), 24.]

80. (а) (Решение В. Имриха (W. Imrich).) Граф с метками вершин 0000, 0001, 0010, 0011, 0100, 0110, 0111, 1100, 1101, 1110, 1111 не может быть помечен любым существенно иным способом; но расстояние от 0001 до 1101 равно 4, а не 2.

(б) Цикл C_{2m} представляет собой неполный куб, поскольку его вершины можно пометить как $l(k) = 1^k0^{m-k}$, $l(m+k) = 0^k1^{m-k}$ для $0 \le k < m$. Но побитовая медиана от $l(0)$, $l(m-1)$ и $l(m+1)$ равна $01^{m-2}0$; в действительности при $m > 2$ эти вершины не имеют медианы.

81. Да. Медианный граф является порожденным подграфом гиперкуба, являющегося двудольным.

82. Общий случай сводится к простому случаю, когда G имеет только две вершины $\{0, 1\}$, поскольку можно работать с медианными метками покомпонентно и поскольку $d(u,v)$ представляет собой расстояние Хэмминга между $l(u)$ и $l(v)$.

В простом случае упомянутое правило устанавливает $u_k \leftarrow v_k$, за исключением $u_{k-1} = v_{k-1} = v_{k+1} \ne v_k$, и его оптимальность можно легко доказать. (Существуют, однако, и другие варианты достижения оптимума; например, если $v_0v_1v_2v_3 = 0110$, можно установить $u_0u_1u_2u_3 = 0000$.)

[Эта задача возникла при изучении самоорганизующихся структур данных. В *Discrete Algorithms and Complexity* (Academic Press, 1987), 351–387, Ф. Р. К. Чанг (F. R. K. Chung), Р. Л. Грэхем (R. L. Graham) и М. Э. Сакс (M. E. Saks) доказали, что медианные графы являются *единственными* графами, для которых u_k всегда можно выбрать оптимальным образом, как функцию от $(v_0, v_1, \ldots, v_{k+1})$, независимо от последующих значений $(v_{k+2}, \ldots, v_t)$. В *Combinatorica* **9** (1989), 111–131, они также охарактеризовали все случаи, для которых достаточно заданного конечного количества предпросмотров.]

83. Рассмотрим сначала булев (двухвершинный) случай, и пусть оптимальное решение получается с помощью рекурсивных правил $u_0 \leftarrow v_0$ и $u_j \leftarrow f_{t+2-j}(u_{j-1}, v_j, \ldots, v_t)$ для $1 \le j \le t$, где каждое f_k представляет собой подходящую булеву функцию от k переменных. Первая функция $f_{t+1}(v_0, v_1, \ldots, v_t)$ на самом деле зависит от ее "наиболее удаленной" переменной v_t, поскольку, когда $\rho = 1-\epsilon$ и $k \ge 2$, должно выполняться $f_{2k+1}(0,1,1,0,1,0,1,\ldots,0,1,0,x) = x$.

Одна подходящая функция f_{t+1} может быть получена следующим образом: пусть $f_{t+1}(0, v_1, \ldots, v_t) = 0$, если $v_1 = 0$. В противном случае пусть "сериями" входной последовательности будут

$$v_0v_1\ldots v_t = 01^{a_k}0^{a_{k-1}}\ldots1^{a_2}0^{a_1} \qquad \text{или} \qquad 01^{a_k}0^{a_{k-1}}\ldots1^{a_3}0^{a_2}1^{a_1},$$

где $a_k, \ldots, a_1 \geq 1$, и пусть $\alpha_j = 2 \dot{-} a_j\rho = \max(0, 2 - a_j\rho)$ для $1 \leq j \leq k$. Тогда

$$f_{t+1}(0, v_1, \ldots, v_t) = \bigl[\alpha_k \dot{-} (\alpha_{k-1} \dot{-} (\cdots \dot{-} (\alpha_2 \dot{-} (1 \dot{-} a_1\rho))\cdots)) = 0\bigr].$$

Пусть также $f_{t+1}(1, v_1, \ldots, v_t) = \bar{f}_{t+1}(0, \bar{v}_1, \ldots, \bar{v}_t)$, так что функция f_{t+1} самодуальна.

Несколько утонченное доказательство позволяет показать, что функция f_{t+1} является также монотонной.

Следовательно, согласно теореме P можно применить f_{t+1} покомпонентно к меткам произвольного медианного графа, всегда оставаясь в пределах графа.

84. Имеется 81 такая функция, каждая из которых может быть представлена как медиана нечетного количества элементов. Встречаются вершины семи типов.

Тип	Типичная вершина	Количество случаев	Смежная с	Степень
1	$\langle z\rangle$	5	$\langle vwxyzzz\rangle$	1
2	$\langle vwxyzzz\rangle$	5	$\langle z\rangle, \langle wxyzz\rangle$	5
3	$\langle wxyzz\rangle$	20	$\langle vwxyzzz\rangle, \langle vwxxyyzzz\rangle$	4
4	$\langle vwxxyyzzz\rangle$	30	$\langle xyz\rangle, \langle wxyzz\rangle, \langle vwxyyzz\rangle$	5
5	$\langle vwxyyzz\rangle$	10	$\langle vwxxyyzzz\rangle, \langle vwxyz\rangle$	7
6	$\langle vwxyz\rangle$	1	$\langle vwxyyzz\rangle$	10
7	$\langle xyz\rangle$	10	$\langle vwxxyyzzz\rangle$	3

[Фон Нейманн (von Neumann) и Моргенштерн (Morgenstern) перечислили эти семь типов в своей книге *Theory of Games and Economic Behavior* (1944), §52.5, в связи с изучением эквивалентной задачи о системах побеждающих коалиций, которые они именовали *простыми играми*. Граф для функций от шести переменных с 2646 вершинами 30 типов описан в статье Мейеровица (Meyerowitz), упоминавшейся в упр. 70. Только 21 из этих типов может быть представлен в виде простой медианы нечетного количества элементов; вершина наподобие $\langle\langle abd\rangle\langle ace\rangle\langle bcf\rangle\rangle$, например, такого представления не имеет. Пусть соответствующий граф для n переменных имеет M_n вершин; П. Эрдеш (P. Erdös) и Н. Хиндман (N. Hindman) в *Discrete Math.* **48** (1984), 61–65, показали, что $\lg M_n$ асимптотически стремится к $\binom{n-1}{\lfloor n/2\rfloor}$. Д. Кляйтман (D. Kleitman) в *J. Combin. Theory* **1** (1966), 153–155, показал, что вершины для различных проецирующих функций наподобие x и y в этом графе всегда удалены одна от другой.]

85. Каждый сильно связный компонент должен состоять из одной вершины; в противном случае две координаты всегда были бы равны, или всегда были бы дополнительными одна к другой. Таким образом, ориентированный граф должен быть ацикличным.

Кроме того, не должно быть пути от вершины к ее дополнению; в противном случае координата должна быть константой.

Когда эти два условия удовлетворены, можно доказать, что не имеется вершины x, являющейся избыточной, путем присвоения значения 0 всем вершинам, которые предшествуют x или $\bar{x}$; значения 1 — всем вершинам, которые следуют за ними, и подходящих значений — всем остальным вершинам.

(В результате мы получаем совершенно отличный способ представления медианного графа. Например, показанный ориентированный граф соответствует медианному графу, метками которого являются $\{0000, 0001, 0010, 0011, 0111, 1010\}$.)

86. Да. Согласно теореме P *любая* монотонная самодуальная функция отображает элементы X в X.

87. В этом случае (72) может заменить топологическое упорядочение $7\,6\,5\,4\,3\,2\,1\,\overline{1}\,\overline{2}\,\overline{3}\,\overline{4}\,\overline{5}\,\overline{6}\,\overline{7}$; так что мы получаем

(Последовательные инверторы на одной линии, конечно же, могут быть удалены.)

88. Данное значение d вносит не более $6\lceil t/d\rceil$ единиц задержки (для $2\lceil t/d\rceil$ кластеров).

89. Предположим сначала, что новое условие представляет собой $i \to j$, в то время как старым условием является $i' \to j'$, где $i < j$ и $i' < j'$ и нет дополненных литералов. Новый модуль изменяет $x_1 \ldots x_t$ на $y_1 \ldots y_t$, где $y_i = x_i \wedge x_j$, $y_j = x_i \vee x_j$, а для прочих значений индексов $y_k = x_k$. Мы, безусловно, имеем $y_{i'} \le y_{j'}$, когда $\{i', j'\} \cap \{i, j\} = \emptyset$. Никаких проблем при $i = i'$ не возникает, поскольку $y_{i'} = y_i \le x_i = x_{i'} \le x_{j'} = y_{j'}$. Но случай $i = j'$ более сложный: здесь отношения $i' \to i$ и $i \to j$ влекут за собой также $i' \to j$; а это отношение обеспечивается *предыдущими* модулями, поскольку модули добавляются в порядке уменьшения расстояния d в топологическом упорядочении $u_1 \ldots u_{2t}$. Следовательно, $y_{i'} = x_{i'} \le x_j$ и $y_{i'} \le x_{j'} = x_i$; следовательно, $y_{i'} \le x_i \wedge x_j = y_i = y_{j'}$. Аналогичное доказательство работает, когда $j = i'$ или $j = j'$.

Наконец, в случае дополненных литералов, построение искусно сводит общий случай к недополненному путем инвертирования и последующего обратного инвертирования битов.

90. Когда $t = 2$, работу выполняет сеть . Общий случай получается из этого "кирпичика" рекурсивно, сведением t к $\lceil t/2\rceil$.

[Изучение CI-сетей, а также других сетей большей общности, было инициировано работой E. W. Mayr and A. Subramanian, *J. Computer and System Sci.* **44** (1992), 302–323.]

91. Ответ пока что не кажется известным даже для частного случая, когда медианный граф является свободным деревом (с $t+1$ вершинами), или в монотонном случае, когда он представляет собой дистрибутивную решетку, как в следствии F. В последнем случае инверторы могут оказаться ненужными.

93. Пусть $d_X(u,v)$ — количество ребер в кратчайшем пути между u и v, когда путь полностью лежит в X. Ясно, что $d_X(u,v) \ge d_G(u,v)$. А если $u = u_0 — u_1 — \cdots — u_k = v$ является кратчайшим путем в G, то путь $u = f(u_0) — f(u_1) — \cdots — f(u_k) = v$ лежит в X, когда f является ретракцией из G в X; следовательно, $d_X(u,v) \le d_G(u,v)$.

94. Если f является ретракцией t-мерного куба в X, две различные позиции координат не могут быть всегда равны или всегда быть дополнительными одна к другой для всех $x \in X$, если только они не являются константами. Если, скажем, все элементы X имеют вид $00{*}\ldots{*}$ или $11{*}\ldots{*}$, то не должно существовать пути между вершинами этих двух типов, что противоречит тому факту, что X является изометрическим подграфом (следовательно, связным).

Пусть для заданных $x, y, z \in X$ их медианой в t-мерном кубе является $w = \langle xyz\rangle$. Тогда $f(w) \in [x\,..\,y] \cap [x\,..\,z] \cap [y\,..\,z]$, поскольку, например, $f(w)$ лежит на кратчайшем пути от x до y в X. Так что $f(w) = w$, и мы должны доказать, что $w \in X$. [Этот результат и его гораздо более тонкое обращение найдены Г. Ю. Бандельтом (H. J. Bandelt), *J. Graph Theory* **8** (1984), 501–510.]

95. Ложно (хотя автор и надеялся на истинность этого утверждения); показанная справа сеть получает $0001 \mapsto 0000$, $0010 \mapsto 0011$, $1101 \mapsto 0110$, но ничего $\mapsto 0010$.

(Множество всех возможных выходных данных, похоже, не имеет простого описания, даже если не использовать инверторы. Например, чисто сравнивающая сеть слева, построенная Томасом Федером (Tomás Feder), отображает $000000 \mapsto 000000$, $010101 \mapsto 010101$ и $101010 \mapsto 011001$, но ничего $\mapsto 010001$. См. также упр. 5.3.4–50 и 5.3.4–52.)

96. Нет. Пусть f — пороговая функция, основанная на действительных параметрах $w = (w_1, \ldots, w_n)$ и t, и пусть $\max\{w \cdot x \mid f(x) = 0\} = t - \epsilon$. Тогда $\epsilon > 0$ и f определяется 2^n неравенствами $w \cdot x - t \geq 0$, когда $f(x) = 1$, и $t - w \cdot x - \epsilon \geq 0$, когда $f(x) = 0$. Если A является любой целочисленной матрицей размером $M \times N$, для которой система линейных неравенств $Av \geq (0, \ldots, 0)^T$ имеет решение в действительных числах $v = (v_1, \ldots, v_N)^T$, где $v_N > 0$, существует решение и в целых числах. (Доказывается по индукции по N.) Так что можно считать, что $w_1, \ldots, w_n$, t и ϵ являются целыми числами.

[Более полный анализ с применением неравенства Адамара (см. формулу 4.6.1–(25)) доказывает, что в действительности достаточно весов с абсолютными значениями, не превышающими $(n+1)^{(n+1)/2}/2^n$; см. С. Мурога (S. Muroga), И. Тода (I. Toda) и С. Такасу (S. Takasu), *J. Franklin Inst.* **271** (1961), 376–418, теорема 16. Кроме того, в упр. 112 показано, что иногда действительно требуются такие большие веса.]

97. $\langle 11111x_1x_2\rangle$, $\langle 111x_1x_2\rangle$, $\langle 1x_1x_2\rangle$, $\langle 0x_1x_2\rangle$, $\langle 000x_1x_2\rangle$, $\langle 00000x_1x_2\rangle$.

98. Можно считать, что $f(x_1, \ldots, x_n) = \langle y_1^{w_1} \ldots y_n^{w_n}\rangle$ с положительными целыми весами w_j и с нечетным значением $w_1 + \cdots + w_n$. Пусть δ — минимальное положительное значение среди 2^n сумм $\pm w_1 \pm \cdots \pm w_n$ с n независимо изменяющимися знаками. Переименуем все индексы так, что $w_1 + \cdots + w_k - w_{k+1} - \cdots - w_n = \delta$. Тогда $w_1y_1 + \cdots + w_ny_n > \frac{1}{2}(w_1 + \cdots + w_n) \iff w_1(y_1 - \frac{1}{2}) + \cdots + w_n(y_n - \frac{1}{2}) > 0 \iff w_1(y_1 - \frac{1}{2}) + \cdots + w_n(y_n - \frac{1}{2}) > -\delta/2 \iff w_1y_1 + \cdots + w_ny_n > \frac{1}{2}(w_1 + \cdots + w_n - (w_1 + \cdots + w_k - w_{k+1} - \cdots - w_n)) = w_{k+1} + \cdots + w_n \iff w_1y_1 + \cdots + w_ky_k - w_{k+1}\bar{y}_{k+1} - \cdots - w_n\bar{y}_n > 0$.

99. Имеем $[x_1 + \cdots + x_{2s-1} + s(y_1 + \cdots + y_{2t-2}) \geq st] = [\lfloor (x_1 + \cdots + x_{2s-1})/s\rfloor + y_1 + \cdots + y_{2t-2} \geq t]$; и $\lfloor (x_1 + \cdots + x_{2s-1})/s\rfloor = [x_1 + \cdots + x_{2s-1} \geq s]$.

(Например, $\langle\langle xyz\rangle uv\rangle = \langle xyzu^2v^2\rangle$, величина, которая, как мы знаем, согласно (53) и (54) равна $\langle x\langle yuv\rangle\langle zuv\rangle\rangle$ и $\langle\langle xuv\rangle\langle yuv\rangle\langle zuv\rangle\rangle$. *Источник*: C. C. Elgot, *FOCS* **2** (1961), 238.)

100. Истинно, исходя из предыдущего упражнения и (45).

101. (а) Когда $n = 7$, это $x_7 \wedge x_6$, $x_6 \wedge x_5$, $x_7 \wedge x_5 \wedge x_4$, $x_6 \wedge x_4 \wedge x_3$, $x_7 \wedge x_5 \wedge x_3 \wedge x_2$, $x_6 \wedge x_4 \wedge x_2 \wedge x_1$, $x_7 \wedge x_5 \wedge x_3 \wedge x_1$; а в общем случае имеется n простых импликант, образующих похожий шаблон. (Мы имеем либо $x_n = x_{n-1}$, либо $x_n = \bar{x}_{n-1}$. В первом случае $x_n \wedge x_{n-1}$, очевидно, является простой импликантой. Во втором случае $F_n(x_1, \ldots, x_{n-1}, \bar{x}_{n-1}) = F_{n-1}(x_1, \ldots, x_{n-1})$; так что мы используем простые импликанты последней и вставляем x_n, когда не встречается x_{n-1}.)

(б) Шаблон развертки (0000011, $000011\underline{0}$, $00011\underline{0}1$, $0011\underline{0}1\underline{0}$, $011\underline{0}1\underline{0}1$, $11\underline{0}1\underline{0}1\underline{0}$, $1\underline{0}1\underline{0}1\underline{0}1$) для $n = 7$ работает для всех n.

(в) Две из нескольких возможностей для $n = 7$ иллюстрируют общий случай:

$$F_7(x_1, \ldots, x_7) \;=\; Y\begin{pmatrix} x_6 \\ x_7\ x_5 \\ x_6\ x_6\ x_4 \\ x_7\ x_5\ x_7\ x_3 \\ x_6\ x_6\ x_4\ x_6\ x_2 \\ x_7\ x_5\ x_7\ x_3\ x_7\ x_1 \end{pmatrix} \;=\; Y\begin{pmatrix} x_6 \\ x_7\ x_5 \\ x_6\ x_6\ x_4 \\ x_7\ x_5\ x_5\ x_3 \\ x_6\ x_6\ x_4\ x_4\ x_2 \\ x_7\ x_5\ x_5\ x_3\ x_3\ x_1 \end{pmatrix}.$$

[Пороговая функция Фибоначчи была введена С. Мурогой (S. Muroga), который также нашел оптимальный результат в упр. 105; см. *IEEE Transactions* **EC-14** (1965), 136–148.]

102. (а) Согласно (11) и (12) $\hat{f}(\bar{x}_0, \bar{x}_1, \ldots, \bar{x}_n)$ является дополнением $\hat{f}(x_0, x_1, \ldots, x_n)$.

(б) Если f задана формулой (75), то $\hat{f}$ представляет собой $[(w+1-2t)x_0+w_1x_1+\cdots+w_nx_n \ge w+1-t]$, где $w=w_1+\cdots+w_n$. И обратно, если $\hat{f}$ является пороговой функцией, то таковой же является и $f(x_1,\ldots,x_n)=\hat{f}(1,x_1,\ldots,x_n)$. [E. Goto and H. Takahasi, *Proc. IFIP Congress* (1962), 747–752.]

103. [См. R. C. Minnick, *IRE Transactions* **EC-10** (1961), 6–16.] Мы хотим минимизировать $w_1+\cdots+w_n$ при условии ограничений $w_j \ge 0$ для $1 \le j \le n$ и $(2e_1-1)w_1+\cdots+(2e_n-1)w_n \ge 1$ для каждой простой импликанты $x_1^{e_1}\wedge\cdots\wedge x_n^{e_n}$. Например, если $n=6$, простая импликанта $x_2\wedge x_5\wedge x_6$ приводит к ограничению $-w_1+w_2-w_3-w_4+w_5+w_6 \ge 1$. Если минимум равен $+\infty$, данная функция не является пороговой. (Ответ к упр. 84 дает один из простейших примеров такого случая.) В противном случае, если решение $(w_1,\ldots,w_n)$ включает только целые числа, оно минимизирует требуемый размер. Если же решение оказывается нецелым, должны добавляться дополнительные ограничения, пока не будет найдено наилучшее решение, как в п. (в) следующего упражнения.

104. Сначала нам нужен алгоритм для генерации простых импликант $x_1^{e_1}\wedge\cdots\wedge x_n^{e_n}$ заданной функции мажоризации $\langle x_1^{w_1}\ldots x_n^{w_n}\rangle$ при $w_1 \ge \cdots \ge w_n$ и нечетном значении $w_1+\cdots+w_n$.

K1. [Инициализация.] Установить $t \leftarrow 0$. Затем для $j=n$, $n-1$, …, 1 (в указанном порядке) установить $a_j \leftarrow t$, $t \leftarrow t+w_j$, $e_j \leftarrow 0$. Наконец установить $t \leftarrow (t+1)/2$, $s_1 \leftarrow 0$ и $l \leftarrow 0$.

K2. [Вход на уровень l.] Установить $l \leftarrow l+1$, $e_l \leftarrow 1$, $s_{l+1} \leftarrow s_l+w_l$.

K3. [Ниже порога?] Если $s_{l+1} < t$, вернуться к шагу K2.

K4. [Посещение простой импликанты.] Посетить степени $(e_1,\ldots,e_n)$.

K5. [Уменьшение размера.] Установить $e_l \leftarrow 0$. Затем, если $s_l+a_l \ge t$, установить $s_{l+1} \leftarrow s_l$ и перейти к шагу K2.

K6. [Откат.] Установить $l \leftarrow l-1$. Завершить работу алгоритма, если $l=0$; в противном случае перейти к шагу K5, если $e_l=1$; в противном случае повторить этот шаг. ▮

(а) $\langle x_1x_2^2x_3^3x_4^5x_5^6x_6^8x_7^{10}x_8^{12}\rangle$ (21 простая импликанта).

(б) Оптимальными весами для $\langle x_0^{16-2t}x_1^8x_2^4x_3^2x_4\rangle$ являются $w_0w_1w_2w_3w_4=10000$, 31111, 21110, 32211, 11100, 23211, 12110, 13111, 01000 для $0 \le t \le 8$; прочие случаи дуальны.

(в) В этом случае оптимальными весами $(w_1,\ldots,w_{10})$ являются $(29,25,19,15,12,8,8,3,3,0)/2$; так что мы видим, что x_{10} не играет роли и мы должны работать с дробными весами. Ограничивая $w_8 \ge 2$, получаем целочисленные веса $(15,13,10,8,6,4,4,2,1,0)$, которые должны быть оптимальны, поскольку их сумма превышает предыдущую сумму на 2. (Только две из 175 428 самодуальных пороговых функций от девяти переменных имеют нецелые веса, минимизирующие $w_1+\cdots+w_n$; второй такой функцией является $\langle x_1^{17}x_2^{15}x_3^{11}x_4^9x_5^7x_6^5x_7^4x_8^2x_9\rangle$. Наибольшее w_1 в минимальном представлении встречается в $\langle x_1^{42}x_2^{22}x_3^{18}x_4^{15}x_5^{13}x_6^{10}x_7^8x_8^4x_9^3\rangle$; Наибольшее значение $w_1+\cdots+w_9$ встречается один раз в функции $\langle x_1^{34}x_2^{32}x_3^{28}x_4^{27}x_5^{24}x_6^{20}x_7^{18}x_8^{15}x_9^{11}\rangle$, которая также является примером наибольшего значения w_9. См. S. Muroga, T. Tsuboi, and C. R. Baugh, *IEEE Transactions* **C-19** (1970), 818–825.)

105. Когда $n=7$, неравенства, генерируемые в упр. 103, представляют собой $w_7+w_6-w_5-w_4-w_3-w_2-w_1 \ge 1$, $-w_7+w_6+w_5-w_4-w_3-w_2-w_1 \ge 1$, $w_7-w_6+w_5+w_4-w_3-w_2-w_1 \ge 1$, $-w_7+w_6-w_5+w_4+w_3-w_2-w_1 \ge 1$, $w_7-w_6+w_5-w_4+w_3+w_2-w_1 \ge 1$, $-w_7+w_6-w_5+w_4-w_3+w_2+w_1 \ge 1$, $w_7-w_6+w_5-w_4+w_3-w_2+w_1 \ge 1$. Умножим их соответственно на 1, 1, 2, 3, 5, 8, 5, чтобы получить $w_1+\cdots+w_7 \ge 1+1+2+3+5+8+5$. Эта же идея работает для всех $n \ge 3$.

106. (а) $\langle x_1^{2^{n-1}} x_2^{2^{n-2}} \ldots x_n\ \bar{y}_1^{2^{n-1}} \bar{y}_2^{2^{n-2}} \ldots \bar{y}_n \bar{z}\rangle$. (Согласно упр. 99 мы можем также вычислить n медиан трех элементов: $\langle\langle \ldots \langle x_n \bar{y}_n \bar{z}\rangle \ldots x_2 \bar{y}_2\rangle x_1 \bar{y}_1\rangle$.) (б) Если $\langle x_1^{u_1} x_2^{u_2} \ldots x_n^{u_n} \bar{y}_1^{v_1} \bar{y}_2^{v_2} \ldots \bar{y}_n^{v_n} \bar{z}^w\rangle$ решает поставленную задачу, должны выполняться $2^{n+1}-1$ базовых неравенств; например, при $n=2$ это неравенства $u_1+u_2-v_1+v_2-w \ge 1$, $u_1+u_2-v_1-v_2+w \ge 1$, $u_1-u_2+v_1-v_2-w \ge 1$, $u_1-u_2-v_1+v_2+w \ge 1$, $-u_1+u_2+v_1+v_2-w \ge 1$, $-u_1+u_2+v_1-v_2+w \ge 1$, $-u_1-u_2+v_1+v_2+w \ge 1$. Добавим их все, чтобы получить $u_1+u_2+\cdots+u_n+v_1+v_2+\cdots+v_n+w \ge 2^{n+1}-1$.

107.

f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$	f	$N(f)$	$\Sigma(f)$
$\bot$	0	$(0,0)$	$\overline{\subset}$	1	$(0,1)$	$\overline{\nabla}$	1	$(0,0)$	$\overline{\llcorner}$	2	$(0,1)$
$\wedge$	1	$(1,1)$	$\mathrel{\rotatebox{-90}{$\subset$}}$	2	$(1,2)$	$\equiv$	2	$(1,1)$	$\supset$	3	$(1,2)$
$\overline{\supset}$	1	$(1,0)$	$\oplus$	2	$(1,1)$	$\overline{\mathrel{\rotatebox{-90}{$\subset$}}}$	2	$(1,0)$	$\overline{\wedge}$	3	$(1,1)$
$\llcorner$	2	$(2,1)$	$\vee$	3	$(2,2)$	$\subset$	3	$(2,1)$	$\top$	4	$(2,2)$

Обратите внимание, что $\oplus$ и $\equiv$ имеют одни и те же параметры $N(f)$ и $\Sigma(f)$. Это единственные бинарные булевы операции, не являющиеся пороговыми функциями.

108. Если $\Sigma(g)=(s_0,s_1,\ldots,s_n)$, значение g равно 1 в s_0 случаях при $x_0=1$ и в 2^n-s_0 случаях при $x_0=0$. Мы также имеем $\Sigma(f_0)+\Sigma(f_1)=(s_1,\ldots,s_n)$, и

$$\begin{aligned}\Sigma(f_0) &= \sum_{x_1=0}^{1} \cdots \sum_{x_n=0}^{1} (\bar{x}_1,\ldots,\bar{x}_n)\, g(0,\bar{x}_1,\ldots,\bar{x}_n)\\ &= \sum_{x_1=0}^{1} \cdots \sum_{x_n=0}^{1} \big((1,\ldots,1)-(x_1,\ldots,x_n)\big)\big(1-g(1,x_1,\ldots,x_n)\big)\\ &= (2^{n-1}-s_0,\ldots,2^{n-1}-s_0)+\Sigma(f_1).\end{aligned}$$

Так что ответами при $n>0$ являются (а) $N(f_0)=2^n-s_0$, $\Sigma(f_0)=\frac{1}{2}(s_1-s_0+2^{n-1},\ldots,s_n-s_0+2^{n-1})$; (б) $N(f_1)=s_0$, $\Sigma(f_1)=\frac{1}{2}(s_1+s_0-2^{n-1},\ldots,s_n+s_0-2^{n-1})$. [Эквивалентные результаты были представлены Э. Гото (E. Goto) в лекциях в MIT в 1963 году.]

109. (а) $a_1+\cdots+a_k \ge b_1+\cdots+b_k$ тогда и только тогда, когда $k-a_1-\cdots-a_k \le k-b_1-\cdots-b_k$.

(б) Пусть $\alpha^+=(a_1,a_1+a_2,\ldots,a_1+\cdots+a_n)$. Тогда вектор $(c_1,\ldots,c_n)$, получаемый покомпонентной минимизацией α^+ и β^+, равен $(\alpha\wedge\beta)^+$. (Ясно, что $c_j=c_{j-1}+a_j$ или b_j.)

(в) Действуем, как в п. (б), но с покомпонентной *максимизацией*; или берем $\overline{\bar{\alpha}\wedge\bar{\beta}}$.

(г) Истинны, поскольку max и min удовлетворяют этим законам дистрибутивности. (В действительности мы получаем дистрибутивную *решетку мажоризации со смешанным основанием* аналогичным путем из множества всех n-кортежей $a_1\ldots a_n$ с $0\le a_j<m_j$ для $1\le j\le n$. Р. П. Стенли (R. P. Stanley) заметил, что рис. 8 представляет собой также решетку порядковых идеалов показанной справа треугольной сетки.)

(д) $\alpha 1$ покрывает $\alpha 0$, а $\alpha 10\beta$ покрывает $\alpha 01\beta$. [Эта характеристика принадлежит Р. О. Вайндеру (R. O. Winder), *IEEE Trans.* **EC-14** (1965), 315–325, но он не доказал свойство решетки. Решетка часто называется $M(n)$; см. B. Lindström, *Nordisk Mat. Tidskrift* **17** (1969), 61–70; R. P. Stanley, *SIAM J. Algebraic and Discrete Methods* **1** (1980), 177–179.]

(е) В силу (д) мы имеем $r(\alpha)=na_1+(n-1)a_2+\cdots+a_n$.

(ж) Суть в том, что $0\beta\succeq 0\alpha$ тогда и только тогда, когда $\beta\succeq\alpha$, и что $1\beta\succeq 0\alpha$ тогда и только тогда, когда $1\beta\succeq 10\ldots0\vee 0\alpha=1\alpha'$.

(з) Т. е. сколько $a_1\ldots a_n$ обладает тем свойством, что $a_1\ldots a_k$ содержит не больше единиц, чем нулей? Ответ равен $\binom{n}{\lfloor n/2\rfloor}$; см., например, упр. 2.2.1–4 или 7.2.1.6–42, (а).

110. (а) Если $x\subseteq y$, то $x\preceq y$; следовательно, $f(x)\le f(y)$; QED.

(б) Нет; пороговая функция не обязана быть монотонной (см. (79)). Но *можно* показать, что f регулярна, если потребовать также выполнения условия $w_n \geq 0$: если $f(x) = 1$ и y покрывает x, то мы имеем $w \cdot y \geq w \cdot x$.

(в) При условиях $f(x) = 1$ и $x_j < x_{j+1}$ мы имеем $f(y) = 1$, когда y покрывает x с $x_j \leftrightarrow x_{j+1}$; следовательно, $s_j \geq s_{j+1}$. (Эти рассуждения справедливы даже при $w_n < 0$.)

(г) Нет; рассмотрим, например, $\langle x_1 x_2^2 x_3^2 \rangle$, равную $\langle x_1 x_2 x_3 \rangle$. Контрпример может иметься, даже когда веса минимизируют $w_1 + \cdots + w_n$, поскольку решение линейной программы из упр. 103 не всегда единственное. Одним таким случаем, найденным Мурогой (Muroga), Цубои (Tsuboi) и Бофом (Baugh), является $\langle x_1^{17} x_2^9 x_3^8 x_4^6 x_5^7 x_6^5 x_7^3 x_8^2 x_9^2 \rangle$, функция, которая, как ни странно, симметрична относительно x_4 и x_5. Но если $s_j > s_{j+1}$, то мы должны иметь $w_j > w_{j+1}$ в соответствии с п. (в).

111. (а) Найдем оптимальную самодуальную функцию f поточечно, как в упр. 14; в случае неоднозначности выберем $f(x_1, \ldots, x_n) = x_1$. Таким образом, $f(x_1, \ldots, x_n) = [r_1^{x_1} \ldots r_n^{x_n} \geq \sqrt{r_1 \ldots r_n}]$, за исключением того, что '$\geq$' превращается в '$>$' при $x_1 = 0$. Эта функция регулярна при $r_1 \geq \cdots \geq r_n \geq 1$.

(б) Пусть g — регулярная самодуальная функция, построенная в п. (а). Если f представляет собой заданную регулярную, самодуальную функцию, то мы хотим проверить, что $f(x) \leq g(x)$ для всех векторов x; отсюда будет вытекать, что $f = g$, поскольку обе функции самодуальны.

Предположим, что $f(x) = 1$, и пусть $y \preceq x$ является минимальным вектором, таким, что $f(y) = 1$. Если мы проверили, что $g(y) = 1$, то, действительно, как и требовалось, $g(x) = 1$. [См. K. Makino and T. Kameda, *SIAM Journal on Discrete Mathematics* **14** (2001), 381–407.]

Например, при $n = 5$ имеется только семь самодуальных регулярных булевых функций, генерируемых следующими минимальными элементами на рис. 8: 10000; 01111, 10001; 01110, 10010; 01101, 10011, 10100; 01100; 01011, 11000; 00111. Таким образом, оптимальный комитет может быть найден путем исследования только небольшого количества значений функций.

(в) Предположим, что $1 > p_1 \geq \cdots \geq p_r \geq \frac{1}{2} > p_{r+1} \geq \cdots \geq p_n > 0$. Пусть $f_k(x_1, \ldots, x_n)$ — k-я монотонная самодуальная функция, а $F_k(x_1, \ldots, x_n)$ — ее целочисленное мультилинейное представление. Мы хотим найти оптимальную доступность $G(p_1, \ldots, p_n) = \max_k F_k(p_1, \ldots, p_n)$. Если $p_1 \leq p_1', \ldots, p_n \leq p_n'$, то мы имеем $F_k(p_1, \ldots, p_n) \leq F_k(p_1', \ldots, p_n')$ согласно упр. 12, (д); следовательно, $G(p_1, \ldots, p_n) \leq G(p_1', \ldots, p_n')$.

Таким образом, если $0 < r < n$, то мы имеем $G(p_1, \ldots, p_n) \leq G(p_1, \ldots, p_r, \frac{1}{2}, \ldots, \frac{1}{2})$. А последнее значение равно $F(p_1, \ldots, p_r, \frac{1}{2}, \ldots, \frac{1}{2})$, порожденному этими бóльшими вероятностями, как в п. (а). Эта функция не зависит от $(x_{r+1}, \ldots, x_n)$, так что она дает нам оптимум.

Если $r = 0$, задача кажется более глубокой. Мы имеем $G(p_1, \ldots, p_n) \leq G(p_1, \ldots, p_1)$; так что мы можем сделать вывод, что в этом случае оптимальный комитет представляет собой $f(x_1, \ldots, x_n) = x_1$, если мы можем показать, что $F_k(p, \ldots, p) \leq p$ для всех k при $p < \frac{1}{2}$. В общем случае $F_k(p, \ldots, p) = \sum_m c_m p^m (1-p)^{n-m}$, где c_m равно количеству векторов x, таких что $f_k(x) = 1$ и $\nu x = m$. Поскольку f_k самодуальна, мы имеем $c_m + c_{n-m} = \binom{n}{m}$ для всех k. А в теореме Эрдеша–Ко–Радо (упр. 7.2.1.3–111) говорится о том, что мы имеем $c_m \leq \binom{n-1}{m-1}$ для любого перекрывающегося семейства m-множеств, когда $m \leq n/2$. Отсюда следует искомый результат.

[См. Y. Amir and A. Wool, *Information Processing Letters* **65** (1998), 223–228.]

112. (а) Ведущими членами являются соответственно 0, $+xy$, $-xy$, $+x$, $-xy$, $+y$, $-2xy$, $-xy$, $+xy$, $+2xy$, $-y$, $+xy$, $-x$, $+xy$, $-xy$, 1; так что $F(f) = 1$, когда f представляет собой $\wedge$, $\llcorner$, R, ∇, $\equiv$, $\subset$, $\supset$, $\top$.

(б) Коэффициент, соответствующий, скажем, степеням 01101, равен f_{0**0*} в обозначениях из ответа к упр. 12; это линейная комбинация элементов таблицы истинности, всегда лежащая в диапазоне $\lceil -2^{k-1}\rceil \le f_{0**0*} \le \lceil 2^{k-1}\rceil$ при наличии k звездочек. Таким образом, ведущий коэффициент положителен тогда и только тогда, когда число в смешанной системе счисления

$$\begin{bmatrix} f_{**\ldots*}, & f_{0*\ldots*}, & \ldots, & f_{*0\ldots0}, & f_{00\ldots0} \\ 2^m{+}1, & 2^{m-1}{+}1, & \ldots, & 2^1{+}1, & 2^0{+}1 \end{bmatrix}$$

положительно, где f упорядочено в обратном порядке последовательности Чейза, а основание 2^k+1 соответствует f с k звездочками. Например, при $m=2$ мы имеем $F(f)=1$ тогда и только тогда, когда сумма $18f_{**}+6f_{0*}+2f_{*0}+f_{00} = 18(f_{11}-f_{01}-f_{10}+f_{00})+6(f_{01}-f_{00})+2(f_{10}-f_{00})+f_{00} = 18f_{11}-12f_{01}-16f_{10}+11f_{00}$ положительна; так что пороговая функция может быть записана как $\langle f_{11}^{18}\bar f_{01}^{12}\bar f_{10}^{16} f_{00}^{11}\rangle$.

(В этом конкретном случае в действительности корректным является гораздо более простое выражение $\langle f_{11}f_{11}\bar f_{01}\bar f_{10}f_{00}\rangle$. Но в п. (в) будет показано, что при больших значениях m мы не можем существенно улучшить получаемый результат.)

(в) Предположим, что $F(f) = \bigl[\sum_\alpha v_\alpha(f_\alpha - \frac12) > 0\bigr]$, где суммирование выполняется по всем 2^m бинарным строкам α длиной m и где каждое v_α представляет собой целочисленный вес. Определим

$$w_\alpha = \sum_\beta (-1)^{\nu(\alpha\dot{-}\beta)} v_\beta \qquad \text{и} \qquad F_\alpha = \sum_\beta (-1)^{\nu(\alpha\dot{-}\beta)} f_\beta - 2^{m-1}[\alpha = 00\ldots0];$$

таким образом, например, $w_{01} = -v_{00}+v_{01}-v_{10}+v_{11}$ и $F_{11} = f_{00}-f_{01}-f_{10}+f_{11}$. Можно показать, что $F_{1^k0^l} = 2^l f_{*^k0^l}$, если $F_\alpha = 0$ при $\nu(\alpha) > k > 0$; следовательно, знаки преобразованных коэффициентов истинности F_α определяют знак ведущего коэффициента в мультилинейном представлении. Кроме того, теперь мы имеем $F(f) = \bigl[\sum_\alpha w_\alpha F_\alpha > 0\bigr]$.

Общая идея доказательства заключается в выборе тестовой функции f, из которой мы можем вывести свойства преобразованных весов w_α. Например, если $f(x_1,\ldots,x_m) = x_1\oplus\cdots\oplus x_k$, мы находим $F_\alpha = 0$ для всех α, за исключением $F_{1^k0^{m-k}} = (-1)^{k-1}2^{m-1}$. Мультилинейное представление $x_1\oplus\cdots\oplus x_k$ имеет ведущий член $\lceil(-2)^{k-1}\rceil x_1\ldots x_k$; следовательно, мы можем заключить, что $w_{1^k0^{m-k}} > 0$, а также аналогичным способом, что $w_\alpha > 0$ для всех α. В общем случае, если m изменяется на $m+1$, но f не зависит от x_{m+1}, мы имеем $F_{\alpha0} = 2F_\alpha$ и $F_{\alpha1} = 0$.

Тестовая функция $x_2\oplus\cdots\oplus x_m\oplus x_1\bar x_2\ldots\bar x_m$ доказывает, что

$$w_{1^m} > (2^{m-1}{-}1)w_{01^{m-1}} + \sum_{k=1}^{m-1} w_{1^k01^{m-1-k}} + \text{меньшие члены},$$

где меньшие члены включают только w_α с $\nu(\alpha)\le m-2$. В частности, $w_{11} > w_{01}+w_{10}+w_{00}$. Тестовая функция $x_1\oplus\cdots\oplus x_{m-1}\oplus\bar x_1\ldots\bar x_{m-2}(x_{m-1}\oplus\bar x_m)$ доказывает, что

$$w_{1^{m-2}01} > (2^{m-2}{-}1)w_{1^{m-2}10} + \sum_{k=0}^{m-3}(w_{1^k01^{m-3-k}10} + w_{1^k01^{m-3-k}01}) + \text{меньшие члены},$$

где в этот раз меньшие члены имеют $\nu(\alpha)\le m-3$. В частности, $w_{101} > w_{110}+w_{010}+w_{001}$. Перестанавливая индексы, мы получаем аналогичные неравенства, ведущие к

$$w_{\alpha_j} \;>\; (2^{\nu(\alpha_j)-1}-1)w_{\alpha_{j-1}} \qquad \text{для } 0<j<2^m,$$

поскольку w начинают быстро расти. Но мы имеем $v_\alpha = \sum_\beta(-1)^{\nu(\beta\dot{-}\alpha)}w_\beta/n$; следовательно, $|v_\alpha| = w_{11\ldots1}/n + O(w_{11\ldots1}/n^2)$. [*SIAM J. Discrete Math.* **7** (1994), 484–492. Важные обобщения этого результата были получены Н. Алоном (N. Alon) и В. Х. Ву (V. H. Vũ), *J. Combinatorial Theory* **A79** (1997), 133–160.]

113. Указанная g_3 равна $S_{2,3,6,8,9}$, поскольку указанная g_2 равна $S_{2,3,4,5,8,9,10,11,12}$.

Для вычисления более сложной функции $S_{1,3,5,8}$ положим $g_1 = [\nu x \geq 6]$; $g_2 = [\nu x \geq 3]$; $g_3 = [\nu x - 5g_1 - 2g_2 \geq 2] = S_{2,4,5,9,10,11,12}$; $g_4 = [2\nu x - 15g_1 - 9g_3 \geq 1] = S_{1,3,5,8}$. [См. M. A. Fischler and M. Tannenbaum, *IEEE Transactions* **C-17** (1968), 273–279.]

114. $[4x + 2y + z \in \{3, 6\}] = (\bar{x} \wedge y \wedge z) \vee (x \wedge y \wedge \bar{z})$. Таким же образом *любая* булева функция от n переменных представляет собой частный случай симметричной функции от $2^n - 1$ переменных. [См. W. H. Kautz, *IRE Transactions* **EC-10** (1961), 378.]

115. Обе части самодуальны, так что можно считать, что $x_0 = 0$. Тогда

$$s_j = [x_j + \cdots + x_{j+m-1} > x_{j+m} + \cdots + x_{j+2m-1}].$$

Если $x_1 + \cdots + x_{2m}$ нечетно, имеем $s_j = \bar{s}_{j+m}$; следовательно, $s_1 + \cdots + s_{2m} = m$ и результат равен 1. Но если $x_1 + \cdots + x_{2m}$ четно, разность $x_j + \cdots + x_{j+m-1} - x_{j+m} - \cdots - x_{j+2m-1}$ будет равна нулю как минимум для одного $j \leq m$; это делает $s_j = s_{j+m} = 0$, так что мы получим $s_1 + \cdots + s_{2m} < m$.

116. (а) Это импликанта тогда и только тогда, когда $f(x) = 1$ при $j \leq \nu x \leq n - k + j$. Это простая импликанта тогда и только тогда, когда, кроме того, $f(x) = 0$ при $\nu x = j - 1$ или $\nu x = n - k + j + 1$.

(б) Рассмотрим строку $v = v_0 v_1 \ldots v_n$, такую, что $f(x) = v_{\nu x}$. Согласно п. (а) при $v = 0^a 1^{b+1} 0^c$ имеется $\binom{a+b+c}{a,b,c}$ простых импликант. В указанном случае $a = b = c = 3$, так что простых импликант — 1680 штук.

(в) В случае обобщенной симметричной функции мы собираем вместе простые импликанты для каждой серии единиц в v. Ясно, что когда $a < c - 1$, их больше для $v = 0^{a+1} 1^{b+1} 0^{c-1}$, чем для $v = 0^a 1^{b+1} 0^c$; так что, когда достигается максимум, v не содержит двух последовательных нулей.

Пусть $\hat{b}(m, n)$ — максимально возможное число простых импликант, когда $v_m = 1$ и $v_j = 0$ для $m < j \leq n$. Тогда при $m \leq \frac{1}{2}n$ мы имеем

$$\begin{aligned}\hat{b}(m,n) &= \max_{0 \leq k \leq m} \left(\binom{n}{k, m-k, n-m} + \hat{b}(k-2, n) \right) \\ &= \binom{n}{\lceil m/2 \rceil, \lfloor m/2 \rfloor, n-m} + \hat{b}(\lceil m/2 \rceil - 2, n),\end{aligned}$$

с $\hat{b}(-2, n) = \hat{b}(-1, n) = 0$. А глобальный максимум равен

$$\hat{b}(n) = \binom{n}{n_0, n_1, n_2} + \hat{b}(n_1 - 2, n) + \hat{b}(n_2 - 2, n), \qquad n_j = \left\lfloor \frac{n+j}{3} \right\rfloor.$$

В частности, мы имеем $\hat{b}(9) = 1698$ с максимумом, достигаемым при $v = 1101111011$.

(г) Согласно приближению Стирлинга $\hat{b}(n) = 3^{n+3/2}/(2\pi n) + O(3^n/n^2)$.

(д) В этом случае соответствующее рекуррентное соотношение для $m < \lceil n/2 \rceil$ имеет вид

$$\begin{aligned}\tilde{b}(m,n) &= \max_{0 \leq k \leq m} \left(\binom{n}{k, m-k, n-m} + \binom{n}{k-1, 0, n-k+1} + \tilde{b}(k-2, n) \right) \\ &= \binom{n}{\lceil m/2 \rceil, \lfloor m/2 \rfloor, n-m} + \binom{n}{\lceil m/2 \rceil - 1} + \tilde{b}(\lceil m/2 \rceil - 2, n),\end{aligned}$$

а значение $\tilde{b}(n) = \tilde{b}(\lceil n/2 \rceil - 1, n)$ максимизирует функцию $\min(\text{простые импликанты}(f), \text{простые импликанты}(\bar{f}))$. Мы имеем $(\tilde{b}(1), \tilde{b}(2), \ldots) = (1, 1, 4, 5, 21, 31, 113, 177, 766, 1271, 4687, 7999, 34412, \ldots)$; например, $\tilde{b}(9) = 766$ соответствует $S_{0,2,3,4,8}(x_1, \ldots, x_9)$. Асимптотически $\tilde{b}(n) = 2^{(3n+3+(n \bmod 2))/2}/(2\pi n) + O(2^{3n/2}/n^2)$.

Ссылки. *Summaries, Summer Inst. for Symbolic Logic* (Dept. of Math., Cornell Univ., 1957), 211–212; B. Dunham and R. Fridshal, *J. Symbolic Logic* **24** (1959), 17–19; А. П. Викулин, *Проблемы кибернетики* **29** (1974), 151–166, где он сообщает о работе, выполненной в 1960 году; Y. Igarashi, *Transactions of the IEICE of Japan* **E62** (1979), 389–394.

117. Максимальное количество подкубов в n-мерном кубе, где ни один из них не содержится в другом, получается при выборе всех подкубов с размерностью $\lfloor n/3 \rfloor$. (Оно получается также путем выбора всех подкубов с размерностью $\lfloor (n+1)/3 \rfloor$; например, при $n = 2$ мы можем выбрать либо $\{0*, 1*, *0, *1\}$, либо $\{00, 01, 10, 11\}$.) Следовательно, $b^*(n) = \binom{n}{\lfloor n/3 \rfloor} 2^{n-\lfloor n/3 \rfloor} = 3^{n+1}/\sqrt{4\pi n} + O(3^n/n^{3/2})$. [См. статью А. П. Викулина из предыдущего упражнения, страницы 164–166; A. K. Chandra and G. Markowsky, *Discrete Math.* **24** (1978), 7–11; N. Metropolis and G. C. Rota, *SIAM J. Applied Math.* **35** (1978), 689–694.]

118. Будем рассматривать две функции как эквивалентные, если одну из них можно получить из другой с помощью дополнения и/или перестановки переменных, но не дополнения самого́ значения функции. Ясно, что такие функции имеют одинаковое количество простых импликант; это отношение эквивалентности изучается в ответе к упр. 125. Компьютерная программа на основе результатов упр. 30 дает следующие результаты.

m	Классы	Функции	m	Классы	Функции	m	Классы	Функции
0	1	1	5	87	17472	10	7	632
1	5	81	6	70	12696	11	1	96
2	18	1324	7	43	7408	12	2	24
3	46	6608	8	24	3346	13	1	16
4	87	14536	9	10	1296	14	0	0

А вот так выглядит соответствующая статистика для функций от пяти переменных.

m	Классы	Функции	m	Классы	Функции	m	Классы	Функции
0	1	1	11	186447	666555696	22	338	608240
1	6	243	12	165460	590192224	23	130	197440
2	37	14516	13	129381	459299440	24	71	75720
3	244	318520	14	91026	319496560	25	37	28800
4	1527	3319580	15	57612	199792832	26	15	10560
5	6997	19627904	16	33590	113183894	27	6	2880
6	23434	73795768	17	17948	58653984	28	4	1040
7	57048	190814016	18	8880	27429320	29	2	640
8	105207	362973410	19	3986	11597760	30	2	48
9	152763	538238660	20	1795	4548568	31	2	64
10	183441	652555480	21	720	1633472	32	1	16

119. Несколько авторов высказали гипотезу о том, что $b(n) = \hat{b}(n)$; М. М. Гаджиев доказал справедливость этого равенства при $n \le 6$ [*Дискретный анализ* **18** (1971), 3–24].

120. (а) Каждая простая импликанта является минитермом, поскольку в n-мерном кубе нет соседних точек с одинаковой четностью. Так что в этом случае единственной подходящей DNF является полная дизъюнктивная форма.

(б) Теперь все простые импликанты состоят из двух смежных точек. Мы должны включить 14 подкубов $0^j{*}0^{6-j}$ и $1^j{*}1^{6-j}$ для $0 \le j \le 6$, чтобы покрыть точки с $\nu x = 1$ и $\nu x = 6$. Другие $\binom{7}{3} + \binom{7}{4} = 70$ точек могут быть покрыты 35 тщательно подобранными простыми импликантами (см., например, упр. 6.5–1 или “шаблон рождественской елки” из раздела 7.2.1.6). Таким образом, кратчайшая дизъюнктивная нормальная форма имеет длину 49. [Изобретательное правдоподобное, но, увы, неверное доказательство необходимости 70 простых импликант было приведено в работе С. В. Яблонского *Проблемы кибернетики* **7** (1962), 229–230.]

(в) Для каждого из 2^{n-1} выборов $(x_1, \ldots, x_{n-1})$ нам нужно не более одной импликанты для учета поведения функции по отношению к x_n.

[Асимптотически почти все булевы функции от n переменных имеют кратчайшую дизъюнктивную нормальную форму с $\Theta(2^n/(\log n \log\log n))$ простыми импликантами. См. Р. Г. Нигматуллин, *Дискретный анализ* **10** (1967), 69–89; В.В, Глаголев, *Проблемы кибернетики* **19** (1967), 75–94; А. Д. Коршунов, *Методы дискретного анализа* **37** (1981), 9–41; N. Pippenger, *Random Structures & Algorithms* **22** (2003), 161–186.]

121. (а) Пусть $x = x_1 \ldots x_m$ и $y = y_1 \ldots y_n$. Поскольку f является функцией от $(\nu x, \nu y)$, всего имеется $2^{(m+1)(n+1)}$ возможных вариантов.

(б) В этом случае из $\nu x \le \nu x'$ и $\nu y \le \nu y'$ вытекает $f(x,y) \le f(x',y')$. Каждая такая функция соответствует зигзагообразному пути от $a_0 = (-\frac{1}{2}, n+\frac{1}{2})$ до $a_{m+n+2} = (m+\frac{1}{2}, -\frac{1}{2})$ с $a_j = a_{j-1} + (1,0)$ или $a_j = a_{j-1} - (0,1)$ для $1 \le j \le m+n+2$; мы имеем $f(x,y) = 1$ тогда и только тогда, когда точка $(\nu x, \nu y)$ лежит над путем. Так что количество возможностей равно количеству таких путей, а именно $\binom{m+n+2}{m+1}$.

(в) Дополнение x и y изменяет νx на $m - \nu x$ и νy на $n - \nu y$. Так что, когда и m, и n четны, таких функций нет; в противном случае их $2^{(m+1)(n+1)/2}$.

(г) Путь в (б) должен теперь удовлетворять условию $a_j + a_{m+n+2-j} = (m,n)$ для $0 \le j \le m+n+2$. Следовательно, количество таких функций равно $\binom{\lceil m/2\rceil + \lceil n/2\rceil}{\lceil m/2\rceil}$ [m нечетно или n нечетно]. Например, при $m = 3$ и $n = 6$ имеются следующие десять случаев.

122. Функция такого рода, в которой все x находятся слева от y, регулярна тогда и только тогда, когда зигзагообразный путь не содержит двух точек, (x,y) и $(x+2,y)$, с $0 < y < n$; если все y находятся слева от x, она регулярна тогда и только тогда, когда зигзагообразный путь не содержит ни $(x, y+2)$, ни (x,y), где $0 < x < m$. Это пороговая функция тогда и только тогда, когда имеется прямая линия, проходящая через точку $(m/2, n/2)$ и обладающая тем свойством, что (s,t) находится над этой линией тогда и только тогда, когда (s,t) находится над путем, где $0 \le s \le m$ и $0 \le t \le n$. Так что случаи 5 и 8, проиллюстрированные в ответе к предыдущему упражнению, не являются регулярными; случаи 1, 2, 3, 7, 9 и 10 являются пороговыми функциями. Остающиеся непороговые функции могут также быть выражены следующим образом: $((x_1 \lor x_2 \lor x_3) \land \langle x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle) \lor (x_1 \land x_2 \land x_3)$ (случай 4); $\langle 00x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle \lor (\langle x_1x_2x_3\rangle \land \langle 11x_1x_2x_3y_1y_2y_3y_4y_5y_6\rangle)$ (случай 6).

123. Самодуальные *регулярные* функции относительно просто перечислить для малых значений n, но их количество быстро растет: при $n = 9$ имеется 319 124 такие функции, найденные в 1967 году Мурогой (Muroga), Цубои (Tsuboi) и Бофом (Baugh); когда $n = 10$, этих функций — 1 214 554 343 (см. упр. 7.1.4–75). Соответствующие числа для $n \le 6$ приведены в табл. 5, поскольку все такие функции являются пороговыми функциями при $n < 9$; при $n = 7$ их 135, а при $n = 8$ — 2470.

Условие пороговости функции может быть быстро проверено для любой такой функции путем усовершенствования метода из упр. 103, поскольку ограничения необходимы только для *минимальных* векторов x (по отношению к мажоризации), таких, что $f(x) = 1$.

Известно, что количество θ_n пороговых функций от n переменных удовлетворяет уравнению $\lg \theta_n = n^2 - O(n^2/\log n)$; см. Ю. А. Зуев, *Математические вопросы кибернетики* **5** (1994), 5–61.

124. 222 класса эквивалентности, перечисленные в табл. 5, включают 24 класса размером $2^{n+1}n! = 768$; так что имеется $24 \times 768 = 18\,432$ ответа на поставленный вопрос. Одним из них является функция $(w \land (x \lor (y \land z))) \oplus z$.

125. 0; x; $x \wedge y$; $x \wedge y \wedge z$; $x \wedge (y \vee z)$; $x \wedge (y \oplus z)$. (Эти функции представляют собой $x \wedge f(y,z)$, где f пробегает классы эквивалентности функций от двух переменных и/или дополнений переменных, но *не* значений функций. В общем случае пусть $f \simeq g$ означает, что f эквивалентна g в этом слабом смысле, но если они эквивалентны в смысле табл. 5, будем писать $f \cong g$. Тогда $x \wedge f \cong x \wedge g$ тогда и только тогда, когда $f \simeq g$, в предположении, что f и g не зависят от переменной x. Легко увидеть, что $(x \wedge f) \simeq (\bar{x} \vee \bar{g})$ невозможно. А если $(x \wedge f) \simeq (x \wedge g)$, можно доказать, что $f \simeq g$, показав, что если σ представляет собой знаковую перестановку $\{x_0, \ldots, x_n\}$ и если $x = x_1 \ldots x_n$, то из тождества $x_0 \wedge f(x) = (x_0\sigma) \wedge g(x\sigma)$ вытекает $f(x) = g(x\sigma\tau)$, где τ обменивает $x_0 \leftrightarrow x_0\sigma$. В результате нижняя строка табл. 5 перечисляет классы эквивалентности по отношению к $\simeq$, но с n, увеличенным на 1; имеется, например, 402 таких класса функций от четырех переменных.)

126. (а) Функция является канализирующей тогда и только тогда, когда она имеет простую импликанту не более чем с одним литералом, или простой дизъюнкт не более чем с одним литералом.

(б) Функция является канализирующей тогда и только тогда, когда как минимум один из компонентов $\Sigma(f)$ равен 0, 2^{n-1}, $N(f)$ или $N(f) - 2^{n-1}$. [См. I. Shmulevich, H. Lähdesmäki, and K. Egiazarian, *IEEE Signal Processing Letters* **11** (2004), 289–292, Proposition 6.]

(в) Если, скажем, $\vee(f) = y_1 \ldots y_n$, где $y_j = 0$, то $f(x) = 0$ при $x_j = 1$. Следовательно, функция f канализирующая тогда и только тогда, когда мы не имеем $\vee(f) = \vee(\bar{f}) = 1 \ldots 1$ и $\wedge(f) = \wedge(\bar{f}) = 0 \ldots 0$. С помощью такого теста можно доказать, что многие функции не являются канализирующими, зная их значения только в нескольких точках.

127. (а) Поскольку самодуальная функция $f(x_1, \ldots, x_n)$ истинна ровно в 2^{n-1} точках, она является канализирующей по отношению к переменной x_j или $\bar{x}_j$ тогда и только тогда, когда $f(x_1, \ldots, x_n) = x_j$.

(б) Определенная функция Хорна, очевидно, является канализирующей, если (i) она содержит любой дизъюнкт с единственным литералом или если (ii) некоторый литерал входит в каждый дизъюнкт. В противном случае она не является канализирующей. Ибо мы имеем $f(0, \ldots, 0) = f(1, \ldots, 1) = 1$, поскольку (i) ложно; а если x_j — любая переменная, то имеется дизъюнкт C_0, не содержащий $\bar{x}_j$, и дизъюнкт C_1, не содержащий x_j, в силу ложности (ii). Выбирая соответствующие значения других переменных, можно сделать $C_0 \wedge C_1$ ложным как при $x_j = 0$, так и при $x_j = 1$.

128. Например, $(x_1 \wedge \cdots \wedge x_n) \vee (\bar{x}_1 \wedge \cdots \wedge \bar{x}_n)$.

129. $\sum_{k=1}^{n} (-1)^{k+1} \binom{n}{k} 2^{2^{n-k}+k+1} - 2(n-1) - 4(n \bmod 2) = n 2^{2^{n-1}+2} + O(n^2 2^{2^{n-2}})$. [См. W. Just, I. Shmulevich, and J. Konvalina, *Physica* **D197** (2004), 211–221.]

130. (а) Если имеется a_n функций от n или меньшего количества переменных, и при этом b_n функций оказываются ровно от n переменных, то $a_n = \sum_k \binom{n}{k} b_k$. Следовательно, $b_n = \sum_k (-1)^{n-k} \binom{n}{k} a_k$. (Этот закон, отмеченный К. Э. Шенноном (C. E. Shannon) в *Trans. Amer. Inst. Electrical Engineers* **57** (1938), 713–723, §4, применим ко всем строкам табл. 3, *за исключением* случая симметричных функций.) В частности, искомый ответ равен $168 - 4 \cdot 20 + 6 \cdot 6 - 4 \cdot 3 + 2 = 114$.

(б) Если имеется a'_n существенно различных функций от n или меньшего количества переменных, и при этом b'_n функций оказываются ровно от n переменных, то мы имеем $a'_n = \sum_{k=0}^{n} b'_k$. Следовательно, $b'_n = a'_n - a'_{n-1}$, и в этом случае ответ равен $30 - 10 = 20$.

131. Пусть имеется $h(n)$ функций Хорна и $k(n)$ функций Крома. Ясно, что $\lg h(n) \geq \binom{n}{\lfloor n/2 \rfloor}$ и $\lg k(n) \geq \binom{n}{2}$. В. Б. Алексеев [*Дискретная математика* **1** (1989), 129–136] доказал, что $\lg h(n) = \binom{n}{\lfloor n/2 \rfloor}(1 + O(n^{-1/4} \log n))$. Б. Боллобас (B. Bollobás), Г. Брайтвелл (G. Brightwell) и И. Лидер (I. Leader) [*Israel J. Math.* **133** (2003), 45–60] доказали, что $\lg k(n) \sim \frac{1}{2} n^2$.

132. (а) Указание истинно, так как $\sum_y s(y)s(y\oplus z) = \sum_{w,x,y}(-1)^{f(w)+w\cdot y+f(x)+x\cdot(y+z)} = 2^n\sum_{w,x}(-1)^{f(w)+f(x)+x\cdot z}[x=w]$. Теперь предположим, что $f(x)=g(x)$ для $2^{n-1}+k$ значений x; тогда $f(x)=g(x)\oplus 1$ для $2^{n-1}-k$ значений x. Но если $|k|<2^{n/2-1}$ для всех аффинных g, то мы должны получить $|s(y)|<2^{n/2}$ для всех y, что противоречит указанию при $z=0$.

(б) Для заданных y_0, y_1, …, y_n имеется ровно $2^{n/2}((y_1y_2+y_3y_4+\cdots+y_{n-1}y_n+1+y_0+h(y_1,y_3,\ldots,y_{n-1}))\bmod 2)$ решений уравнения $f(x)=(y_0+x\cdot y)\bmod 2$ при $x_{2k}=y_{2k-1}$ для $1\le k\le n/2$ и $2^{n/2-1}$ решений для каждого из других $2^{n/2}-1$ значений $(x_2,x_4,\ldots,x_n)$. Так что всего имеется $2^{n-1}\pm 2^{n/2-1}$ решений. (Эти рассуждения в действительности доказывают, что, когда $g(x_1,x_3,\ldots,x_{2n-1})$ представляет собой перестановку всех $2^{n/2}$-битовых векторов, функция $(g(x_1,x_3,\ldots,x_{2n-1})\cdot(x_2,x_4,\ldots,x_{2n})+h(x_2,x_4,\ldots,x_{2n}))\bmod 2$ изогнутая.)

(в) Рассуждения в п. (а) доказывают, что $f(x)$ является изогнутой тогда и только тогда, когда $s(y)=2^{n/2}(-1)^{g(y)}$ для некоторой булевой функции $g(y)$. Эта функция g, которая представляет собой преобразование Фурье/Адамара функции f, также изогнута, так как $\sum_y(-1)^{g(y)+w\cdot y}=2^{-n/2}\sum_{x,y}(-1)^{f(x)+x\cdot y+w\cdot y}=2^{n/2}\sum_x(-1)^{f(x)}[x=w]=2^{n/2}(-1)^{f(w)}$ для всех w. Теперь указание говорит о том, что $\sum_y(-1)^{g(y)+g(y\oplus z)}=0$ для всех ненулевых z, и то же самое выполняется для f.

И обратно, предположим, что $f(x)$ удовлетворяет указанному условию. Тогда мы имеем

$$s(y)^2=\sum_{x,t}(-1)^{f(x)+x\cdot y+f(x\oplus t)+(x\oplus t)\cdot y}=\sum_t(-1)^{t\cdot y}\sum_x(-1)^{f(x)+f(x\oplus t)}=2^n$$

для всех y.

(г) Согласно упр. 11 член $x_1\ldots x_r$ присутствует тогда и только тогда, когда уравнение $f(x_1,\ldots,x_r,0,\ldots,0)=1$ имеет нечетное количество решений, а эквивалентным условием является $(\sum_{x_1,\ldots,x_r}(-1)^{f(x_1,\ldots,x_r,0,\ldots,0)})\bmod 4=2$. В п. (в) мы видели, что эта сумма равна

$$2^{-n}\sum_{x_1,\ldots,x_r,y}s(y)(-1)^{x_1y_1+\cdots+x_ry_r}\;=\;2^{r-n}\sum_{y_{r+1},\ldots,y_n}s(0,\ldots,0,y_{r+1},\ldots,y_n).$$

Если $r=n$, последняя сумма равна $\pm 2^{n/2}$; в противном случае она содержит четное число слагаемых, каждое из которых равно $\pm 2^{r-n/2}$. Так что результат представляет собой кратное 4.

[Изогнутые функции были введены О. С. Ротхаусом (O. S. Rothaus) в 1966 году; его статья, распространяемая частным образом, в конце концов была опубликована в *J. Combinatorial Theory* **A20** (1976), 300–305. Д. Ф. Диллон (J. F. Dillon) в *Congressus Numerantium* **14** (1975), 237–249, открыл дополнительные семейства изогнутых функций; впоследствии были найдены многие другие примеры для $n\ge 8$ и четных n. Изогнутые функции не существуют при нечетных n, но функция наподобие $g(x_1,\ldots,x_{n-1})\oplus x_n\wedge h(x_1,\ldots,x_{n-1})$ имеет расстояние $2^{n-1}-2^{(n-1)/2}$ от всех аффинных функций, когда изогнуты g и $g\oplus h$. Для случая $n=15$ Н. Д. Паттерсоном (N. J. Patterson) и Д. Г. Видеманном (D. H. Wiedemann) было найдено лучшее построение [*IEEE Transactions* **IT-29** (1983), 354–356, **IT-36** (1990), 443], дающее расстояние $2^{14}-108$. С. Кавут (S. Kavut) и М. Дикер Юцель (M. Diker Yücel) [*Information and Computation* **208** (2010), 341–350] добились расстояния 2^8-14 при $n=9$.]

133. Пусть $p_k=1/(2^{2^{n-k}}+1)$, так что $\bar p_k=2^{2^{n-k}}/(2^{2^{n-k}}+1)$. [Диссертация (MIT, 1994).]

РАЗДЕЛ 7.1.2

1. $((x_1\vee x_4)\wedge x_2)\equiv(x_1\vee x_3)$.

2. (а) $(w\oplus(x\wedge y))\oplus((x\oplus y)\wedge z)$; (б) $(w\wedge(x\vee y))\wedge((x\wedge y)\vee z)$.

3. [*Доклады Академии наук СССР* **115** (1957), 247–248.] Построим матрицу размером $k \times n$, строки которой представляют собой векторы x, где $f(x) = 1$. С учетом перестановки и/или дополнения переменных можно считать, что верхняя строка имеет вид $1 \ldots 1$ и что столбцы отсортированы. Предположим, что имеется l различных столбцов. Тогда $f = g \wedge h$, где g представляет собой И выражений $(x_{j-1} \equiv x_j)$ по всем $1 < j \le n$ таким, что столбец $j-1$ равен столбцу j, а h представляет собой ИЛИ всех k минитермов длиной l с использованием одной переменной из каждой группы равных столбцов. Например, если $n = 8$ и если f равна 1 в $k = 3$ точках 11111111, 00001111, 00110111, то $l = 4$ и $f(x)$ равна $(x_1 \equiv x_2) \wedge (x_3 \equiv x_4) \wedge (x_6 \equiv x_7) \wedge (x_7 \equiv x_8) \wedge \big((x_1 \wedge x_3 \wedge x_5 \wedge x_6) \vee (\bar{x}_1 \wedge \bar{x}_3 \wedge x_5 \wedge x_6) \vee (\bar{x}_1 \wedge x_3 \wedge \bar{x}_5 \wedge x_6)\big)$. Длина этой формулы в общем случае равна $2n + (k-2)l - 1$, и мы имеем $l \le 2^{k-1}$.

Заметим, что если k велико, мы получаем более короткие формулы, записывая $f(x)$ как дизъюнкцию $f_1(x) \vee \cdots \vee f_r(x)$, где каждое f_j имеет не более $\lceil k/r \rceil$ единиц. Таким образом,

$$L(f) \;\le\; \min_{r \ge 1}\big(r - 1 + (2n + \lceil k/r - 2 \rceil 2^{\lceil k/r - 1 \rceil})\, r\big).$$

4. Первое неравенство очевидно, поскольку бинарное дерево глубиной d имеет не более $1 + 2 + \cdots + 2^{d-1} = 2^d - 1$ внутренних узлов.

Указание следует из того, что мы можем считать f_t формулой размера $L(f) - L(g) - 1$, возникающей, когда g заменяется на t. Для $1 \le k < L(f)$ полагаем g_k минимальной подформулой размера $\ge k$. Тогда $g_k ?\ f_{k1} : f_{k0}$ получается из дерева, которое содержит g_k, f_{k1} и f_{k0} на уровне 2.

Пусть $d_r = \max\{\, D(f) \mid L(f) = r \,\}$. Поскольку дочерние по отношению к g_k узлы находятся на уровне 3 и имеют размер $< k$, мы имеем $d_r \le \min_{k=1}^{r-1} \max(3 + d_{k-1}, 2 + d_{r-k-1})$ для $r \ge 3$. По индукции по r следует, что $d_r \le l$ при $r \le b_l$, где $b_l = l$ для $0 \le l \le 2$ и $b_l = b_{l-2} + b_{l-3} + 2$ для $l \ge 3$. Мы также имеем $b_l + 2 = (8P_l + 18P_{l+1} + 11P_{l+2})/23 = c\chi^l + O(0.87^l)$ с применением чисел Перрина из упр. 7.1.4–15, где $c = (2 + 4\chi + 3\chi^2)/(3 + 2\chi) \approx 2.224$. Следовательно, $d_r < \alpha \lg r$ при $r > 1$. [См. P. M. Spira, *Hawaii Int. Conf. Syst. Sci.* **4** (1971), 525–527; R. Brent, D. Kuck and K. Maruyama, *IEEE* **C-22** (1973), 532–534. In *JACM* **23** (1976), 534–543, Д. Ю. Мюллер (D. E. Muller) и Ф. П. Препарата (F. P. Preparata) доказали, что $D(f) \le \beta \lg L(f) + O(1)$, где $\beta = 1/\lg z \approx 2.0807$, $z^4 = 2z + 1$. Является ли β оптимальным?]

5. Пусть $g_0 = 0$, $g_1 = x_1$ и $g_j = x_j \wedge (x_{j-1} \vee g_{j-2})$ для $j \ge 2$. Тогда $F_n = g_n \vee g_{n-1}$, со стоимостью $2n - 2$ и с глубиной n. [Эти функции g_j также играют заметную роль в бинарном сложении; см. способ их вычисления с глубиной $O(\log n)$ в упр. 42 и 44.]

6. Истинно: рассмотрите случаи $y = 0$ и $y = 1$.

7. $\hat{x}_5 = x_1 \vee x_4$, $\hat{x}_6 = x_2 \wedge \hat{x}_5$, $\hat{x}_7 = x_1 \vee x_3$, $\hat{x}_8 = \hat{x}_6 \oplus \hat{x}_7$. (Исходная цепочка вычисляет "случайную" функцию (6); см. упр. 1. Новая цепочка вычисляет нормализацию этой функции, а именно ее дополнение.)

8. Требуемая таблица истинности состоит из блоков из 2^{n-k} нулей, чередующихся с блоками из 2^{n-k} единиц, как в (7). Следовательно, при умножении на $2^{2^{n-k}} + 1$ мы получим $x_k + (x_k \ll 2^{n-k})$, что представляет собой сплошные единицы.

9. При поиске $L(f) = \infty$ на шаге L6 мы можем сохранить g и h в записи, связанной с f. Тогда рекурсивная процедура сможет строить формулу с минимальной длиной для f из соответствующих формул для g и h.

10. На шаге L3 используйте $k = r - 1$ вместо $k = r - 1 - j$. Кроме того, везде замените L на D.

11. Единственной тонкостью является то, что j на шаге U3 должно *уменьшаться*; тогда мы никогда не получим $\phi(g)\ \&\ \phi(h) \ne 0$ при $j = 0$, так что все случаи со стоимостью $r - 1$ будут открыты до того, как мы начнем рассмотрение списка $r - 1$.

U1. [Инициализация.] Установить $U(0) \leftarrow \phi(0) \leftarrow 0$ и $U(f) \leftarrow \infty$ для $1 \le f < 2^{2^n-1}$. Затем установить $U(x_k) \leftarrow \phi(x_k) \leftarrow 0$ и поместить x_k в список 0, как на шаге L1. Установить также $U(x_j \circ x_k) \leftarrow 1$, установить $\phi(x_j \circ x_k)$ равным его уникальному вектору отпечатка (который содержит только одну единицу), и поместить $x_j \circ x_k$ в список 1 для $1 \le j < k \le n$ и всех пяти нормальных операторов $\circ$. Наконец установить $c \leftarrow 2^{2^n-1} - 5\binom{n}{2} - n - 1$.

U2. [Цикл по r.] Выполнять шаг U3 для $r = 2, 3, \ldots$ до тех пор, пока $c > 0$.

U3. [Цикл по j и k.] Выполнять шаг U4 для $j = \lfloor (r-1)/2 \rfloor$, $\lfloor (r-1)/2 \rfloor - 1$, $\ldots$ и $k = r - 1 - j$, до тех пор, пока $j \ge 0$.

U4. [Цикл по g и h.] Выполнять шаг U5 для всех g в списке j и всех h в списке k; если $j = k$, ограничить h функциями, которые *следуют за* g в списке k.

U5. [Цикл по f.] Если $\phi(g) \mathbin{\&} \phi(h) \ne 0$, установить $u \leftarrow r - 1$ и $v \leftarrow \phi(g) \mathbin{\&} \phi(h)$; в противном случае установить $u \leftarrow r$ и $v \leftarrow \phi(g) \mid \phi(h)$. Затем выполнить шаг U6 для $f = g \mathbin{\&} h$, $f = \bar{g} \mathbin{\&} h$, $f = g \mathbin{\&} \bar{h}$, $f = g \mid h$ и $f = g \oplus h$.

U6. [Обновление $U(f)$ и $\phi(f)$.] Если $U(f) = \infty$, установить $c \leftarrow c - 1$, $\phi(f) \leftarrow v$, $U(f) \leftarrow u$ и поместить f в список u. В противном случае если $U(f) > u$, переместить f из списка $U(f)$ в список u и установить $\phi(f) \leftarrow v$, $U(f) \leftarrow u$. В противном случае если $U(f) = u$, установить $\phi(f) \leftarrow \phi(f) \mid v$. ▮

12. $x_4 = x_1 \oplus x_2$, $x_5 = x_3 \wedge x_4$, $x_6 = x_2 \wedge \bar{x}_4$, $x_7 = x_5 \vee x_6$.

13. $f_5 = 01010101$ (x_3); $f_4 = 01110111$ $(x_2 \vee x_3)$; $f_3 = 01110101$ $((\bar{x}_1 \wedge x_2) \vee x_3)$; $f_2 = 00110101$ $(x_1 ?\ x_3 : x_2)$; $f_1 = 00010111$ $(\langle x_1 x_2 x_3 \rangle)$.

14. Для $1 \le j \le n$ сначала вычислите $t \leftarrow (g \oplus (g \gg 2^{n-j})) \mathbin{\&} x_j$, $t \leftarrow t \oplus (t \ll 2^{n-j})$, где x_j представляет собой таблицу истинности (11); тогда для $1 \le k \le n$ и $k \ne j$ требуемая таблица истинности, соответствующая $x_j \leftarrow x_j \circ x_k$, представляет собой $g \oplus (t \mathbin{\&} ((x_j \circ x_k) \oplus x_j))$.

($5n(n-1)$ масок $(x_j \circ x_k) \oplus x_j$ не зависят от g и могут быть вычислены заранее. Та же идея применима и тогда, когда мы допускаем более общие вычисления вида $x_{j(i)} \leftarrow x_{k(i)} \circ_i x_{l(i)}$ с $5n^2(n-1)$ масками $(x_k \circ x_l) \oplus x_j$.)

15. Замечательно асимметричные способы вычисления симметричных функций.

(а)
$$\begin{aligned} x_1 &\leftarrow x_1 \oplus x_2, \\ x_1 &\leftarrow x_1 \oplus x_3, \\ x_2 &\leftarrow x_2 \wedge x_3, \\ x_1 &\leftarrow x_1 \wedge \bar{x}_2. \end{aligned}$$

(б)
$$\begin{aligned} x_1 &\leftarrow x_1 \oplus x_2, \\ x_3 &\leftarrow x_3 \oplus x_4, \\ x_1 &\leftarrow x_1 \oplus x_3, \\ x_2 &\leftarrow x_2 \oplus x_4, \\ x_3 &\leftarrow x_3 \vee x_2, \\ x_3 &\leftarrow x_3 \wedge \bar{x}_1. \end{aligned}$$

(в)
$$\begin{aligned} x_1 &\leftarrow x_1 \oplus x_2, \\ x_2 &\leftarrow x_2 \wedge \bar{x}_1, \\ x_3 &\leftarrow x_3 \oplus x_4, \\ x_4 &\leftarrow x_4 \wedge x_1, \\ x_2 &\leftarrow \bar{x}_2 \wedge x_3, \\ x_2 &\leftarrow x_2 \oplus x_1, \\ x_2 &\leftarrow x_2 \wedge \bar{x}_4. \end{aligned}$$

(г)
$$\begin{aligned} x_1 &\leftarrow x_1 \oplus x_2, \\ x_2 &\leftarrow x_2 \oplus x_3, \\ x_2 &\leftarrow x_2 \vee x_1, \\ x_1 &\leftarrow x_1 \oplus x_4, \\ x_1 &\leftarrow x_1 \wedge x_3, \\ x_2 &\leftarrow x_2 \wedge \bar{x}_1, \\ x_2 &\leftarrow x_2 \oplus x_4. \end{aligned}$$

16. Вычисление, которое использует только $\oplus$ и дополнение, не производит ничего, кроме аффинных функций (см. упр. 7.1.1–132). Предположим, что $f(x) = f(x_1, \ldots, x_n)$ является неаффинной функцией, вычислимой с использованием минимальной памяти. Тогда $f(x)$ имеет вид $g(Ax+c)$, где $g(y_1, y_2, \ldots, y_n) = g(y_1 \wedge y_2, y_2, \ldots, y_n)$, для некоторой несингулярной матрицы A размером $n \times n$ из нулей и единиц, где x и c — векторы столбцов, а операции над векторами выполняются по модулю 2; в этой формуле матрица A и вектор c вычисляются для всех операций $x_i \leftarrow x_i \oplus x_j$ и/или перестановок и дополнений координат, которые встречаются после последней выполненной неаффинной операции. Мы воспользуемся тем фактом, что $g(0, 0, y_3, \ldots, y_n) = g(1, 0, y_3, \ldots, y_n)$.

Пусть α и β — первые две строки A; пусть также a и b — первые два элемента c. Тогда, если $Ax + c \equiv y$ (по модулю 2), мы имеем $y_1 = y_2 = 0$ тогда и только тогда, когда $\alpha \cdot x \equiv a$ и $\beta \cdot x \equiv b$. Этому условию удовлетворяют ровно 2^{n-2} векторов x, и для всех таких векторов мы имеем $f(x) = f(x \oplus w)$, где $Aw \equiv (1, 0, \ldots, 0)^T$.

Для заданных α, β, a, b и w с $\alpha \neq (0,\ldots,0)$, $\beta \neq (0,\ldots,0)$, $\alpha \neq \beta$ и $\alpha \cdot w \equiv 1$ (по модулю 2) имеется $2^{2^n - 2^{n-2}}$ функций f, обладающих тем свойством, что $f(x) = f(x \oplus w)$ при $\alpha \cdot x \bmod 2 = a$ и $\beta \cdot x \bmod 2 = b$. Следовательно, общее количество функций, вычислимых с использованием минимального количества памяти, не превышает 2^{n+1} (для аффинных функций) плюс

$$(2^n - 1)(2^n - 2)2^2(2^{n-1})(2^{2^n - 2^{n-2}}) < 2^{2^n - 2^{n-2} + 3n + 1}.$$

17. Пусть $f(x_1,\ldots,x_n) = g(x_1,\ldots,x_{n-1}) \oplus (h(x_1,\ldots,x_{n-1}) \wedge x_n)$, как в 7.1.1–(16). Представляя h в конъюнктивной нормальной форме, будем образовывать дизъюнкты по одному в x_0, и выполнять над ними операцию И с размещением результата в x_n, получая $h \wedge x_n$. Представляя g в виде суммы конъюнкций по модулю 2, образуем последовательные конъюнкции в x_0 и по завершении работы выполняем операцию ЛИБО с x_n.

(Похоже, что мы не сможем вычислить все функции в рамках $n+1$ регистра, если откажемся от неканализирующих операторов $\oplus$ и $\equiv$. Однако ясно, что $n+2$ регистров будет достаточно, даже если ограничиться единственным оператором $\overline{\wedge}$.)

18. Как упоминалось в ответе к упр. 14, мы должны расширить данное в разделе определение вычислений с использованием минимального количества памяти, позволяя также шаги наподобие $x_{j(i)} \leftarrow x_{k(i)} \circ_i x_{l(i)}$ с $k(i) \neq j(i)$ и $l(i) \neq j(i)$, поскольку это даст лучшие результаты для некоторых функций, зависящих только от четырех из пяти переменных. Тогда мы найдем $C_m(f) = (0, 1, \ldots, 13, 14)$ для соответственно (2, 2, 5, 20, 93, 389, 1960, 10459, 47604, 135990, 198092, 123590, 21540, 472, 0) классов функций... оставляя в стороне 75 908 классов (и 575 963 136 функций), для которых $C_m(f) = \infty$, поскольку они не могут быть вычислены с использованием минимального количества памяти *вообще*. Самой интересной функцией такого рода является, вероятно, $(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_3 \wedge x_4) \vee (x_4 \wedge x_5) \vee (x_5 \wedge x_1)$, которая имеет $C(f) = 7$, но при этом $C_m(f) = \infty$. Еще одним интересным случаем является функция $(((x_1 \vee x_2) \oplus x_3) \vee ((x_2 \vee \bar{x}_4) \wedge x_5)) \wedge ((x_1 \equiv x_2) \vee x_3 \vee x_4)$, для которой $C(f) = 8$ и $C_m(f) = 13$. Одним из способов вычисления этой функции за восемь шагов является $x_6 = x_1 \vee x_2$, $x_7 = x_1 \vee x_4$, $x_8 = x_2 \oplus x_7$, $x_9 = x_3 \oplus x_6$, $x_{10} = x_4 \oplus x_9$, $x_{11} = x_5 \vee x_9$, $x_{12} = x_8 \wedge x_{10}$, $x_{13} = x_{11} \wedge \bar{x}_{12}$.

19. Если это не так, то левое и правое поддеревья корня должны перекрываться, поскольку случай (i) не выполняется. Согласно гипотезе каждая переменная должна встречаться в качестве листа по меньшей мере один раз. Как минимум две переменные должны встречаться в качестве листьев дважды, поскольку не выполняется случай (ii). Но у нас не может быть $n+2$ листьев с $r \leq n+1$ внутренними узлами, если только поддеревья не являются неперекрывающимися.

20. Сейчас алгоритм L (с пропущенным '$f = g \oplus h$' на шаге L5) показывает, что некоторые формулы должны иметь длину 15; и даже метод отпечатков из упр. 11 не дает ничего лучшего, чем 14. Чтобы получить истинно минимальные цепочки, к 25 специальным цепочкам для $r = 6$ в разделе следует добавить пять других, которые больше нельзя исключать, а именно

и когда $r = (7, 8, 9)$, мы должны также рассмотреть соответственно $(653, 12387, 225660)$ дополнительных потенциальных цепочек, которые не являются частными случаями нисходящего или восходящего построения. Вот как выглядит получающаяся в результате статистика (ср. с табл. 1).

$C_c(f)$	Классы	Функции	$U_c(f)$	Классы	Функции	$L_c(f)$	Классы	Функции	$D_c(f)$	Классы	Функции
0	2	10	0	2	10	0	2	10	0	2	10
1	1	48	1	1	48	1	1	48	1	1	48
2	2	256	2	2	256	2	2	256	2	7	684
3	7	940	3	7	940	3	7	940	3	59	17 064
4	9	2 336	4	9	2 336	4	7	2 048	4	151	47 634
5	24	6 464	5	21	6 112	5	20	5 248	5	2	96
6	30	10 616	6	28	9 664	6	23	8 672	6	0	0
7	61	18 984	7	45	15 128	7	37	11 768	7	0	0
8	45	17 680	8	40	14 296	8	27	10 592	8	0	0
9	37	7 882	9	23	8 568	9	33	11 536	9	0	0
10	4	320	10	28	5 920	10	16	5 472	10	0	0
11	0	0	11	6	1 504	11	30	6 304	11	0	0
12	0	0	12	5	576	12	3	960	12	0	0
13	0	0	13	3	144	13	8	1 472	13	0	0
14	0	0	14	2	34	14	2	96	14	0	0
15	0	0	15	0	0	15	4	114	15	0	0

Два класса функций с глубиной 5 представлены функциями $S_{2,4}(x_1, x_2, x_3, x_4)$ и $x_1 \oplus S_2(x_2, x_3, x_4)$; и эти две функции, вместе с $S_2(x_1, x_2, x_3, x_4)$ и функцией четности $S_{1,3}(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, имеют длину 15. Кроме того, $U_c(S_{2,4}) = U_c(S_{1,3}) = 14$. Четыре класса со стоимостью 10 представлены функциями $S_{1,4}(x_1, x_2, x_3, x_4)$, $S_{2,4}(x_1, x_2, x_3, x_4)$, $(x_4?\ x_1 \oplus x_2 \oplus x_3\colon \langle x_1x_2x_3\rangle)$ и $[(x_1x_2x_3x_4)_2 \in \{0, 1, 4, 7, 10, 13\}]$. (Третья из них, кстати, эквивалентна (20), "Гарвардскому труднейшему случаю".)

21. (Авторы указали, что записи в их таблице "должны рассматриваться только как наиболее экономичные операторы, известные авторам".) Минимальная стоимость их самой трудной функции (20) все еще неизвестна, но Дэвид Стивенсон (David Stevenson) показал, что $V(f) \le 17$:

$$\begin{aligned} g &= \text{И}\big(\text{НЕ-И}(w, x), \text{НЕ-И}(\bar{w}, \bar{x})\big);\\ f &= \text{ИЛИ}\big(\text{И}\big(\text{НЕ}(g), \text{НЕ-И}(w, \bar{z}), \text{НЕ-И}(y, z)\big),\\ &\qquad \text{И}\big(\text{НЕ}(\text{НЕ}(g)), \text{НЕ-И}(y, \bar{z}), \text{НЕ-И}(\bar{y}, z)\big)\big). \end{aligned}$$

И хотя гарвардские исследователи и не нашли этой конкретной конструкции, они очень хорошо поработали, в ряде случаев обогнав эвристику на основе отпечатков на целых 6 сеток!

22. $\nu(x_1x_2x_3x_4x_5) = 3$ тогда и только тогда, когда $\nu(x_1x_2x_3x_4) \in \{2, 3\}$, а значение $\nu(x_1x_2x_3x_4x_5)$ нечетно. Аналогично $S_2(x_1, x_2, x_3, x_4, x_5) = S_3(\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_5)$ включает $S_{1,2}(x_1, x_2, x_3, x_4)$.

23. Нам необходимо рассмотреть только 32 нормальных случая, как на рис. 9, поскольку дополнение симметричной функции является симметричным. Затем для вывода большинства оставшихся случаев мы можем использовать отражение наподобие $S_{1,2}(x) = S_{3,4}(\bar{x})$, возможно, вместе с дополнением наподобие $S_{2,3,4,5}(x) = \bar{S}_{0,1}(x) = \bar{S}_{4,5}(\bar{x})$. Конечно, S_1, $S_{1,3,5}$ и $S_{1,2,3,4,5}$ тривиально имеют стоимость 4. Так что остается только $S_{1,2,3,4}(x_1, x_2, x_3, x_4, x_5) = (x_1 \oplus x_2) \vee (x_2 \oplus x_3) \vee (x_3 \oplus x_4) \vee (x_4 \oplus x_5)$, рассматривающаяся для обобщенного значения n в упр. 79.

24. Как отмечалось в разделе, эта гипотеза справедлива при $n \le 5$.

25. Это $2^{2^n-1}-n-1$, количество нетривиальных нормальных функций. (В любой нормальной цепочке длиной r, которая не включает все эти функции, $x_j \circ x_k$ будет новой функцией для некоторых j и k в диапазоне $1 \le j,k \le n+r$ и некоторого нормального бинарного оператора $\circ$; так что мы можем вычислять новую функцию на каждом новом шаге, пока не получим их все.)

26. Ложно. Например, если $g = S_{1,3}(x_1,x_2,x_3)$ и $h = S_{2,3}(x_1,x_2,x_3)$, то $C(gh) = 5$ представляет собой стоимость полного сумматора; но $f = S_{2,3}(x_0,x_1,x_2,x_3)$ имеет стоимость 6 согласно рис. 9.

27. Да: операции '$x_2 \leftarrow x_2 \oplus x_1$, $x_1 \leftarrow x_1 \oplus x_3$, $x_1 \leftarrow x_1 \wedge \bar{x}_2$, $x_1 \leftarrow x_1 \oplus x_3$, $x_2 \leftarrow x_2 \oplus x_3$' преобразуют (x_1,x_2,x_3) в (z_1,z_0,x_3).

28. Пусть $v' = v'' = v \oplus (x \oplus y)$; $u' = ((v \oplus y) \overline{\subset} (x \oplus y)) \oplus u$, $u'' = ((v \oplus y) \vee (x \oplus y)) \oplus u$. Таким образом, мы можем установить $u_0 = 0$, $v_0 = x_1$, $u_j = ((v_{j-1} \oplus x_{2j+1}) \vee (x_{2j} \oplus x_{2j+1})) \oplus u_{j-1}$, если j нечетно, $u_j = ((v_{j-1} \oplus x_{2j+1}) \overline{\subset} (x_{2j} \oplus x_{2j+1})) \oplus u_{j-1}$, если j четно, и $v_j = v_{j-1} \oplus (x_{2j} \oplus x_{2j+1})$, получая тем самым $(u_j v_j)_2 = (x_1 + \cdots + x_{2j+1}) \bmod 4$ для $1 \le j \le \lfloor n/2 \rfloor$. Установим $x_{n+1} = 0$, если n четно. Таким образом, $[(x_1 + \cdots + x_n) \bmod 4 = 0] = \bar{u}_{\lfloor n/2 \rfloor} \wedge \bar{v}_{\lfloor n/2 \rfloor}$ вычисляется за $\lfloor 5n/2 \rfloor - 2$ шагов.

Это построение предложено Л. Д. Стокмейером (L. J. Stockmeyer), который доказал, что оно близко к оптимальному. Действительно, результат упр. 80 вместе с рис. 9 и 10 показывает, что оно не более чем на один шаг длиннее наилучшей возможной цепочки для всех $n \ge 5$.

Кстати, аналогичная формула $u''' = ((v \oplus y) \wedge (x \oplus y)) \oplus u$ дает $(u'''v')_2 = ((uv)_2 + x - y) \bmod 4$. Более просто выглядящая функция $((uv)_2 + x + y) \bmod 4$ имеет стоимость 6, а не 5.

29. Чтобы получить верхнюю границу, будем считать, что каждый полный сумматор или полусумматор увеличивают глубину на 3. Если имеется a_{jd} бит весом 2^j и глубиной $3d$, мы планируем не более $\lceil a_{jd}/3 \rceil$ последовательных битов весом $\{2^j, 2^{j+1}\}$ и глубиной $3(d+1)$. По индукции следует, что $a_{jd} \le \binom{d}{j} 3^{-d} n + 4$. Следовательно, $a_{jd} \le 5$, когда $d \ge \log_{3/2} n$, и общая глубина не превышает $3 \log_{3/2} n + 3$. (Забавно, что общая глубина при $n = 10^7$ оказывается равной ровно 100.)

30. Как обычно, пусть νn обозначает контрольное суммирование битов в бинарном представлении n. Тогда $s(n) = 5n - 2\nu n - 3\lfloor \lg n \rfloor - 3$.

31. После контрольного суммирования за $s(n) < 5n$ шагов произвольная функция от $(z_{\lfloor \lg n \rfloor}, \ldots, z_0)$ может быть вычислена не более чем за $\sim 2n/\lg n$ шагов согласно теореме L. [См. О. Б. Лупанов, *Доклады Академии наук СССР* **140** (1961), 322–325.]

32. Метод раскрутки: сначала докажите по индукции по n, что $t(n) \le 2^{n+1}$.

33. Ложно по техническим причинам: если, скажем, $N = \sqrt{n}$, требуется как минимум n шагов. Однако можно доказать корректную асимптотическую формулу $N + O(\sqrt{N}) + O(n)$, заметив сначала, что метод из раздела дает $N + O(\sqrt{N})$ при $N \ge 2^{n-1}$; в противном случае, если $\lfloor \lg N \rfloor = n-k-1$, мы можем использовать $O(n)$ операций, чтобы получить И величины $\bar{x}_1 \wedge \cdots \wedge \bar{x}_k$ с другими переменными $x_{k+1}, \ldots, x_n$, а затем работать с n, уменьшенным на k.

(Одним из следствий является то, что мы можем вычислять симметричные функции $\{S_1, S_2, \ldots, S_n\}$ со стоимостью $s(n) + n + O(\sqrt{n}) = 6n + O(\sqrt{n})$ и глубиной $O(\log n)$.)

34. Будем говорить, что *расширенный* приоритетный шифратор имеет $n+1 = 2^m$ входов $x_0 x_1 \ldots x_n$ и $m+1$ выходов $y_0 y_1 \ldots y_m$, где $y_0 = x_0 \vee x_1 \vee \cdots \vee x_n$. Если Q'_m и Q''_m представляют собой расширенные шифраторы для $x'_0 \ldots x'_n$ и $x''_0 \ldots x''_n$, то Q_{m+1} работает для $x'_0 \ldots x'_n x''_0 \ldots x''_n$, если мы определим $y_0 = y'_0 \vee y''_0$, $y_1 = y''_0$, $y_2 = y_1 ?\ y''_1 : y'_1$, $\ldots$, $y_{m+1} = y_1 ?\ y''_m : y'_m$. Если P'_m представляет собой обычный приоритетный шифратор для $x'_1 \ldots x'_n$, мы аналогичным образом получим P_{m+1} для $x'_1 \ldots x'_n x''_0 \ldots x''_n$.

Начиная с $m = 2$ и $y_2 = x_3 \vee (x_1 \wedge \bar{x}_2)$, $y_1 = x_2 \vee x_3$, $y_0 = x_0 \vee x_1 \vee y_1$, это построение дает P_m и Q_m со стоимостями p_m и q_m, где $p_2 = 3$, $q_2 = 5$ и $p_{m+1} = 3m + p_m + q_m$, $q_{m+1} = 3m + 1 + 2q_m$ для $m \geq 2$. Следовательно, $p_m = q_m - m$ и $q_m = 15 \cdot 2^{m-2} - 3m - 4 \approx 3.75n$.

35. Если $n = 2m$, вычислим $x_1 \wedge x_2, \ldots, x_{n-1} \wedge x_n$, затем рекурсивно образуем $x_1 \wedge \cdots \wedge x_{2k-2} \wedge x_{2k+1} \wedge \cdots \wedge x_n$ для $1 \leq k \leq m$ и завершим работу еще за n шагов. Если $n = 2m - 1$, используем эту цепочку для $n+1$ элемента; три шага могут быть устранены, если положить $x_{n+1} \leftarrow 1$. [I. Wegener, *The Complexity of Boolean Functions* (1987), упр. 3.25. Та же идея может быть использована с *любым* ассоциативным и коммутативным оператором вместо $\wedge$.]

36. Рекурсивно построим $P_n(x_1, \ldots, x_n)$ и $Q_n(x_1, \ldots, x_n)$ так, как описано далее, где P_n имеет $D(y_j) \leq \lceil \lg n \rceil$ для $1 \leq j \leq n$, а Q_n имеет $D(y_j) \leq \lceil \lg n \rceil + [j \neq n]$. Случай $n = 1$ тривиален; в противном случае P_n получается из $Q'_r(x_1, \ldots, x_r)$ и $P''_s(x_{r+1}, \ldots, x_n)$, где $r = \lceil n/2 \rceil$ и $s = \lfloor n/2 \rfloor$, путем установки $y_j = y'_j$ для $1 \leq j \leq r$, $y_j = y'_r \wedge y''_{j-r}$ для $r < j \leq n$. А Q_n получается либо из $P'_r(x_1 \wedge x_2, \ldots, x_{n-1} \wedge x_n)$, либо из $P'_r(x_1 \wedge x_2, \ldots, x_{n-2} \wedge x_{n-1}, x_n)$, путем установки $y_1 = x_1$, $y_{2j} = y'_j$, $y_{2j+1} = y'_j \wedge x_{2j+1}$ для $1 \leq j < s$ и $y_{2s} = y'_s$, $y_n = y'_r$.

Эти вычисления могут быть выполнены с использованием минимального *количества памяти*, если устанавливать $x_{k(i)} \leftarrow x_{j(i)} \wedge x_{k(i)}$ на шаге i для некоторых индексов $j(i) < k(i)$. Таким образом, мы можем проиллюстрировать построение с диаграммами, аналогичными диаграммам для сортирующих сетей. Например,

$P_8 =$

; $Q_8 =$

.

Стоимости p_n и q_n удовлетворяют соотношениям $p_n = \lfloor n/2 \rfloor + q_{\lceil n/2 \rceil} + p_{\lfloor n/2 \rfloor}$, $q_n = 2\lfloor n/2 \rfloor - 1 + p_{\lceil n/2 \rceil}$ при $n > 1$; например, $(p_1, \ldots, p_7) = (q_1, \ldots, q_7) = (0, 1, 2, 4, 5, 7, 9)$. Установка $\bar{p}_n = 4n - p_n$ и $\bar{q}_n = 3n - q_n$ приводит к более простым формулам, которые доказывают, что $p_n < 4n$ и $q_n < 3n$: $\bar{q}_n = \bar{p}_{\lceil n/2 \rceil} + [n \text{ четно}]$; $\bar{p}_{4n} = \bar{p}_{2n} + \bar{p}_n + 1$, $\bar{p}_{4n+1} = \bar{p}_{2n} + \bar{p}_{n+1} + 1$, $\bar{p}_{4n+2} = \bar{p}_{2n+1} + \bar{p}_{n+1}$, $\bar{p}_{4n+3} = \bar{p}_{4n+2} + 2$. В частности, $1 + \bar{p}_{2^m} = F_{m+5}$ является числом Фибоначчи.

[См. *JACM* **27** (1980), 831–834. Немного лучшие цепочки получатся, если мы заменим Q_{2n+1} на (Q_{2n} и $y_{2n+1} = y_{2n} \wedge x_{2n+1}$), когда n является степенью 2, если заменить P_5 и P_6 на Q_5 и Q_6 и если затем заменить $(P_9, P_{10}, P_{11}, P_{17})$ на $(Q_9, Q_{10}, Q_{11}, Q_{17})$.]

Заметим, что это построение работает в общем случае, если заменить '$\wedge$' *любым* ассоциативным оператором. В частности, последовательность префиксов $x_1 \oplus \cdots \oplus x_k$ для $1 \leq k \leq n$ определяет преобразование бинарного кода Грея в целые числа в двоичной системе счисления 7.2.1.1–(10).

37. Случай $m = 15$, $n = 16$ проиллюстрирован на рисунке справа.

(а) Пусть $x_{i..j}$ обозначает исходное значение $x_i \wedge \cdots \wedge x_j$. Можно показать, что когда алгоритм устанавливает $x_k \leftarrow x_j \wedge x_k$, то предыдущее значение x_k было равно $x_{j+1..k}$. После шага S1 x_k равно $x_{f(k)+1..k}$, где $f(k) = k \,\&\, (k-1)$ для $1 \leq k < m$ и $f(m) = 0$. После шага S2 x_k равно $x_{1..k}$ для $1 \leq k \leq m$.

(б) Стоимость S1 равна $m - 1$, стоимость S2 равна $m - 1 - \lceil \lg m \rceil$, а стоимость S3 равна $n - m$. Последняя задержка x_k равна $\lfloor \lg k \rfloor + \nu k - 1$ для $1 \leq k < m$ и равна $\lceil \lg m \rceil + k - m$ для $m \leq k \leq n$. Так что максимальная задержка для $\{x_1, \ldots, x_{m-1}\}$ оказывается равной $g(m) = m - 1$ для $m < 4$, $g(m) = \lfloor \lg m \rfloor + \lfloor \lg \frac{m}{3} \rfloor$ для $m \geq 4$. Мы имеем $c(m, n) = m + n - 2 - \lceil \lg m \rceil$, $d(m, n) = \max(g(m), \lceil \lg m \rceil + n - m)$. Следовательно, $c(m, n) + d(m, n) = 2n - 2$ при $n \geq m + g(m) - \lceil \lg m \rceil$.

(в) Таблица значений показывает, что $d(n) = \lceil \lg n \rceil$ для $n < 8$, и $d(n) = \lfloor \lg(n - \lfloor \lg n \rfloor + 3) \rfloor + \lfloor \lg \frac{2}{3}(n - \lfloor \lg n \rfloor + 3) \rfloor - 1$ для $n \geq 8$. Формулируя иначе, мы имеем $d(n) > d(n-1)$ и $n > 2$ тогда и только тогда, когда $n = 2^k + k - 3$ или $2^k + 2^{k-1} + k - 3$ для некоторого $k > 1$. Минимум с минимальной стоимостью получается для $m = n$, когда $n < 8$; в противном случае он получается для $m = n - \lfloor \lg \frac{2}{3}(n - \lfloor \lg n \rfloor + 3) \rfloor + 2 - [n = 2^k + k - 3$ для некоторого $k]$.

(г) Установим $m \leftarrow m(n, d)$, где $m(n, d(n))$ определено в предыдущем пункте, а при $d > d(n)$ имеем $m(n, d) = m(n-1, d-1)$. [См. *J. Algorithms* **7** (1986), 185–201.]

38. (а) Сверху вниз $f_k(x_1, \ldots, x_n)$ представляет собой элементарную симметричную функцию, называемую также пороговой функцией $S_{\geq k}(x_1, \ldots, x_n)$. (См. упр. 5.3.4–28, формулу 7.1.1–(90).)

(б) После вычисления $\{S_1, \ldots, S_n\}$ за $\approx 6n$ шагов, как в ответе к упр. 33, мы можем применить метод из упр. 37 для завершения вычислений за $2n$ дополнительных шагов.

Но более интересно разработать булеву цепочку специально для вычисления $2^m + 1$ пороговых функций $g_k(x_1, \ldots, x_m) = [(x_1 \ldots x_m)_2 \geq k]$ для $0 \leq k \leq 2^m$. Поскольку $[(x'x'')_2 \geq (y'y'')_2] = [(x')_2 \geq (y')_2 + 1] \vee ([(x')_2 \geq (y')_2] \wedge [(x'')_2 \geq (y'')_2])$, построение "разделяй и властвуй", аналогичное бинарному декодеру, решает эту задачу со стоимостью, не превышающей $2t(m)$.

Кроме того, если $2^{m-1} \leq n < 2^m$, стоимость $u(n)$ вычисления $\{g_1, \ldots, g_n\}$ этим методом оказывается равной $2n + O(\sqrt{n})$, что вполне приемлемо при малом n:

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$u(n) =$	0	1	2	4	7	7	8	12	15	17	19	19	20	21	22	27	32	34	36	36

Начав с контрольного суммирования, мы можем отсортировать n булевых значений за $s(n) + u(n) \approx 7n$ шагов. Сортирующая сеть со стоимостью $2\hat{S}(n)$ оказывается лучшей при $n = 4$, но проигрывает при $n \geq 8$. [См. 5.3.4–(11); D. E. Muller and F. P. Preparata, *JACM* **22** (1975), 195–201.]

39. [*IEEE Transactions* **C-29** (1980), 737–738.] Тождество

$$M_{r+s}(x_1, \ldots, x_r, x_{r+1}, \ldots, x_{r+s}; y_0, \ldots, y_{2^{r+s}-1}) = M_r(x_1, \ldots, x_r; y'_0, \ldots, y'_{2^r-1}),$$

где $y'_j = \bigvee_{k=0}^{2^s-1}(d_k \wedge y_{2^s j+k})$ и d_k является k-м выходом s-к-2^s декодера, примененного к $(x_{r+1}, \ldots, x_{r+s})$, показывает, что $C(M_{r+s}) \leq C(M_r) + 2^{r+s} + 2^r(2^s - 1) + t(s)$, где $t(s)$ является стоимостью (30) декодера. Глубина равна $D(M_{r+s}) = \max(D_x(M_{r+s}), D_y(M_{r+s}))$, где D_x и D_y обозначают максимальные глубины переменных x и y; мы имеем $D_x(M_{r+s}) \leq \max(D_x(M_r), 1 + s + \lceil \lg s \rceil + D_y(M_r))$ и $D_y(M_{r+s}) \leq 1 + s + D_y(M_r)$.

Беря $r = \lceil m/2 \rceil$ и $s = \lfloor m/2 \rfloor$, мы получаем $C(M_m) \leq 2^{m+1} + O(2^{m/2})$, $D_y(M_m) \leq m + 1 + \lceil \lg m \rceil$ и $D_x(M_m) \leq D_y(M_m) + \lceil \lg m \rceil$.

40. Мы можем, например, положить $f_{nk}(x) = \bigvee_{j=1}^{n+1-k}(l_j(x) \wedge r_{j+k-1}(x))$, где

$$l_j(x) = \begin{cases} x_j, & \text{если } j \bmod k = 0, \\ x_j \wedge l_{j+1}(x), & \text{если } j \bmod k \neq 0, \end{cases} \quad \text{для } 1 \leq j \leq n - (n \bmod k);$$

$$r_j(x) = \begin{cases} 1, & \text{если } j \bmod k = 0, \\ x_j \wedge r_{j-1}(x), & \text{если } j \bmod k \neq 0, \end{cases} \quad \text{для } k \leq j \leq n.$$

Стоимость равна $4n - 3k - 3\lfloor \frac{n}{k} \rfloor - \lfloor \frac{n-1}{k} \rfloor + 2 - (n \bmod k)$.

При малом n или k предпочтительнее рекурсивное решение. Заметим, что

$$f_{nk}(x) = \begin{cases} x_{n-k+1} \wedge \cdots \wedge x_k \wedge \\ \qquad f_{(2n-2k)(n-k)}(x_1, \ldots, x_{n-k}, x_{k+1}, \ldots, x_n), & \text{для } k < n < 2k; \\ f_{\lfloor (n+k)/2 \rfloor k}(x_1, \ldots, x_{\lfloor (n+k)/2 \rfloor}) \vee \\ \qquad f_{\lfloor (n+k-1)/2 \rfloor k}(x_{\lfloor (n-k)/2 \rfloor + 1}, \ldots, x_n), & \text{для } n \geq 2k. \end{cases}$$

Можно показать, что стоимость этого решения равна $n-1+\sum_{j=1}^{n-k}\lfloor \lg j\rfloor$ при $k\le n<2k$, и асимптотически, при $n\to\infty$, она находится между $(m+\alpha_k-1)n+O(km)$ и $(m+2-2/\alpha_k)n+O(km)$, где $m=\lfloor \lg k\rfloor$ и $1<\alpha_k=(k+1)/2^m\le 2$.

Наилучшим решением является объединение этих методов; оптимальная стоимость пока что неизвестна.

41. Пусть $c(m)$ — стоимость вычисления обоих значений, $(x)_2+(y)_2$ и $(x)_2+(y)_2+1$, с помощью метода условного суммирования, когда x и y имеют по $n=2^m$ бит, и пусть $c'(m)$ — стоимость более простой задачи вычисления $(x)_2+(y)_2$. Тогда $c(m+1)=2c(m)+6\cdot 2^m+2$, $c'(m+1)=c(m)+c'(m)+3\cdot 2^m+1$. (Бит z_n суммы стоит 1; но биты z_k для $n<k\le 2n+1$ стоят 3, поскольку они имеют вид $c?\ a_k{:}\ b_k$, где c — бит переноса.) Если начать с $n=1$ и $c(0)=3$, $c'(0)=2$, то решение имеет вид $c(m)=(3m+5)2^m-2$, $c'(m)=(3m+2)2^m-m$. Однако усовершенствованное построение для случая $n=2$ позволяет начать с $c(1)=11$ и $c'(1)=7$; тогда решением является $c(m)=(3m+\frac{7}{2})2^m-2$, $c'(m)=(3m+\frac{1}{2})2^m-m+1$. В любом случае глубина равна $2m+1$. [См. J. Sklansky, *IRE Transactions* **EC-9** (1960), 226–231.]

42. (а) Поскольку $\langle x_k y_k c_k\rangle = u_k\vee(v_k\wedge c_k)$, можно использовать (26) и индукцию.

(б) Заметьте, что $U_k^{k+1}=u_k$ и $V_k^{k+1}=v_k$; затем воспользуйтесь индукцией по $j-i$. [См. A. Weinberger and J. L. Smith, *IRE Transactions* **EC-5** (1956), 65–73; R. P. Brent and H. T. Kung, *IEEE Transactions* **C-31** (1982), 260–264.]

(в) Сначала для $l=1,2,\ldots,m-1$ и для $1\le k\le n$ вычислите V_i^k для всех $h(l)$, кратных i, в диапазоне $k_l\ge i\ge k_{l+1}$, где $k_l=h(l)\lfloor (k-1)/h(l)\rfloor$ обозначает наибольшее кратное $h(l)$, меньшее k. Например, когда $l=3$ и $k=99$, мы вычисляем V_{96}^{99}, $V_{88}^{99}=V_{96}^{99}\wedge V_{88}^{96}$, $V_{80}^{99}=V_{88}^{99}\wedge V_{80}^{88},\ldots,V_{64}^{99}=V_{72}^{99}\wedge V_{64}^{72}$; это префиксное вычисление с использованием значений V_{96}^{99}, V_{88}^{96}, $V_{80}^{88},\ldots,V_{64}^{72}$, которые были вычислены при $l=2$. Применяя метод из упр. 36, шаг l добавляет не более l уровней к глубине и требует логических элементов общим числом $(p_1+p_2+\cdots+p_{2^l})n/2^l=O(2^l n)$.

Тогда, вновь для $l=1,2,\ldots,m-1$ и для $1\le k\le n$, вычислим U_i^k для $i=k_{l+1}$ с использованием "развернутой" формулы

$$U_{k_{l+1}}^k \;=\; U_{k_l}^k \vee \bigvee_{\substack{k_l>j\ge k_{l+1}\\ h(l)\backslash j}} (V_{j+h(l)}^k\wedge U_j^{j+h(l)}).$$

Например, развернутая формула для $l=3$ и $k=99$ имеет вид

$$U_{64}^{99} \;=\; U_{96}^{99}\vee(V_{96}^{99}\wedge U_{88}^{96})\vee(V_{88}^{99}\wedge U_{80}^{88})\vee(V_{80}^{99}\wedge U_{72}^{80})\vee(V_{72}^{99}\wedge U_{64}^{72}).$$

Каждое такое U_i^k представляет собой объединение не более чем 2^l членов, так что оно может быть вычислено с глубиной $\le l$ в дополнение к глубине каждого члена. Общая стоимость этой фазы для $1\le k\le n$ равна $(0+2+4+\cdots+(2^l-2))n/2^l=O(2^l n)$.

Общая стоимость для вычисления всех необходимых U и V равна, таким образом, $\sum_{l=1}^{m-1}O(2^l n)=O(2^m n)$. (Кроме того, величины V_0^k в действительности не нужны, так что мы экономим стоимость $\sum_{l=1}^{m-1}h(l)p_{2^l}$ логических элементов.) Например, когда $m=(2,3,4,5)$, мы получаем булевы цепочки для сложения $(2,8,64,1024)$-битовых чисел соответственно с общими глубинами $(3,7,11,16)$ и стоимостями $(7,64,1254,48470)$.

[Это построение принадлежит В. М. Храпченко, *Проблемы кибернетики* **19** (1967), 107–122, который также показал, как комбинировать его с другими методами так, чтобы общая стоимость составляла $O(n)$ при глубине $\lg n+O(\sqrt{\log n})$. Однако его комбинированный метод представляет чисто теоретический интерес, поскольку, чтобы глубина стала меньше $2\lg n$, требуется $n>2^{64}$. Еще один способ получения малой глубины с применением рекуррентности в (б) может основываться на числах Фибоначчи. Метод Фибоначчи вычисляет переносы с глубиной $\log_\phi n+O(1)\approx 1.44\lg n$ и стоимостью $O(n\log n)$. Например,

он дает цепочки для бинарного сложения со следующими характеристиками.

n =	4	8	16	32	64	128	256	512	1024
Глубина	6	7	9	10	12	13	15	16	18
Стоимость	24	71	186	467	1125	2648	6102	13775	30861

См. D. E. Knuth, *The Stanford GraphBase* (1994), 276–279.

Чарльз Бэббидж (Charles Babbage) нашел изобретательное механическое решение аналогичной задачи для сложения в десятичной системе счисления, заявляя, что его устройство в состоянии суммировать числа произвольной точности за константное время; чтобы его решение работало, ему потребовались бы идеализированные жесткие компоненты с отсутствием зазоров. См. H. P. Babbage, *Babbage's Calculating Engines* (1889), 334–335. Забавно, что эквивалентная идея хорошо работает с физическими транзисторами, хотя ее нельзя выразить через булевы цепочки; см. P. M. Fenwick, *Comp. J.* **30** (1987), 77–79.]

43. (а) Пусть $A = B = Q = \{0, 1\}$ и $q_0 = 0$. Определим $c(q, a) = d(q, a) = \bar{q} \wedge a$.

(б) Ключевая идея состоит в построении функций $d_1(q) \ldots d_{n-1}(q)$, где $d_1(q) = d(q, a_1)$ и $d_j(q) = d(d_{j-1}(q), a_j)$. Другими словами, $d_1 = d^{(a_1)}$ и $d_j = d_{j-1} \circ d^{(a_j)}$, где $d^{(a)}$ является функцией, которая отображает $q \mapsto d(q, a)$, и где $\circ$ обозначает композицию функций. Каждая функция d_j может быть закодирована в бинарном представлении, а $\circ$ представляет собой ассоциативную операцию над этими бинарными представлениями. Следовательно, функции $d_1 d_2 \ldots d_{n-1}$ являются префиксами $d^{(a_1)}$, $d^{(a_1)} \circ d^{(a_2)}$, $\ldots$, $d^{(a_1)} \circ \cdots \circ d^{(a_{n-1})}$; и $q_1 q_2 \ldots q_n = q_0 d_1(q_0) \ldots d_{n-1}(q_0)$.

(в) Представим функцию $f(q)$ с помощью ее таблицы истинности $f_0 f_1$. Тогда композиция $f_0 f_1 \circ g_0 g_1$ представляет собой $h_0 h_1$, где функции $h_0 = f_0 ?\, g_1 \colon g_0$ и $h_1 = f_1 ?\, g_1 \colon g_0$ являются мультиплексорными функциями, которые могут быть вычислены со стоимостью 3 и глубиной 2. (Объединенная стоимость $C(h_0 h_1)$ равна лишь 5, но мы пытаемся оставить малой глубину.) Таблица истинности для $d^{(a)}$ представляет собой $a0$. Воспользовавшись упр. 36, мы, таким образом, сможем вычислить таблицы истинности $d_{10} d_{11} d_{20} d_{21} \ldots d_{(n-1)0} d_{(n-1)1}$ со стоимостью $\le 6p_{n-1} < 24n$ и глубиной $\le 2\lceil \lg(n-1) \rceil$; тогда $b_1 = a_1$, и $b_j = \bar{q}_j \wedge a_j = \bar{d}_{(j-1)0} \wedge a_j$ для $j > 1$. (Эти оценки стоимости достаточно консервативны; из-за нулей в исходных таблицах истинности $d^{(a_j)}$ и из-за того, что многие промежуточные значения d_{j1} никогда не используются, возможны существенные упрощения. Например, когда $n = 5$, фактическая стоимость равна 10, а не $6p_{n-1} + (n-1) = 28$; фактическая глубина равна 4, а не $2\lceil \lg(n-1) \rceil + 1 = 5$. Заметим, что прямая цепочка $b_j = a_j \wedge \bar{b}_{j-1}$ для $1 < j \le n$ также решает задачу (а); она выигрывает в стоимости, но имеет глубину $n - 1$.)

44. Входные данные можно рассматривать как строку $x_0 y_0\, x_1 y_1\, \ldots\, x_{n-1} y_{n-1}$, элементы которой принадлежат четырехбуквенному алфавиту $A = \{00, 01, 10, 11\}$; имеются два состояния $Q = \{0, 1\}$, представляющие возможный бит переноса, с $q_0 = 0$; выходной алфавит имеет вид $B = \{0, 1\}$; и мы имеем $c(q, xy) = q \oplus x \oplus y$, $d(q, xy) = \langle qxy \rangle$. Таким образом, в этом случае конечный преобразователь, по сути, описывается полным сумматором.

Когда мы составляем отображения $d^{(xy)}$, встречаются только три из четырех возможных функций q. Их можно закодировать как $u \vee (q \wedge v)$. Исходные функции $d^{(xy)}$ имеют $u = x \wedge y$, $v = x \oplus y$; а композиция $(uv) \circ (u'v')$ представляет собой $u''v''$, где $u'' = u' \vee (v' \wedge u)$ и $v'' = v \wedge v'$.

Например, когда $n = 4$, цепочка имеет следующий вид с использованием обозначений из упр. 42: $U_k^{k+1} = x_k \wedge y_k$, $V_k^{k+1} = x_k \oplus y_k$, для $0 \le k < 4$; $U_0^2 = U_1^2 \vee (V_1^2 \wedge U_0^1)$, $U_2^4 = U_3^4 \vee (V_3^4 \wedge U_2^3)$, $V_2^4 = V_2^3 \wedge V_3^4$; $U_0^3 = U_2^3 \vee (V_2^3 \wedge U_0^2)$, $U_0^4 = U_2^4 \vee (V_2^4 \wedge U_0^2)$; $z_0 = V_0^1$, $z_1 = U_0^1 \oplus V_1^2$, $z_2 = U_0^2 \oplus V_2^3$, $z_3 = U_0^3 \oplus V_3^4$, $z_4 = U_0^4$. Общая стоимость равна 20; максимальная глубина, 6, встречается в вычислении z_3.

В общем случае стоимость в обозначениях из упр. 36 будет равна $2n + 3p_n$, поскольку нам требуется $2n$ логических элементов для исходных u и v, затем $3p_n$ логических элементов для вычисления префикса; еще $n - 1$ дополнительных логических элементов, необходимых для образования z_j для $0 < j < n$, компенсируются тем фактом, что нам не нужно вычислять V_0^j для $1 < j \le n$. Следовательно, общая стоимость равна $14 \cdot 2^m - 3F_{m+5} + 3$, что превосходит метод условного суммирования (который, однако, имеет глубину $2m + 1$, а не $2m + 2$):

$n =$	2	4	8	16	32	64	128	256	512	1024
Стоимость цепочки условного суммирования	7	25	74	197	492	1179	2746	6265	14072	31223
Ладнера–Фишера	7	20	52	125	286	632	1363	2888	6040	12509

[Джордж Буль (George Boole) разработал свою алгебру для того, чтобы показать, что логика может быть понята через арифметику. В конечном итоге логика стала настолько понятной, что ситуация стала обратной: такие исследователи, как Шеннон (Shannon) и Цузе (Zuse), начали в 1930-х годах разрабатывать схемы для арифметических вычислений посредством логики, и с тех пор было открыто немало подходов к задаче параллельного сложения. Первые булевы цепочки со стоимостью $O(n)$ и глубиной $O(\log n)$ были разработаны Ю. П. Офманом, *Доклады Академии наук СССР* **145** (1962), 48–51. Его цепочки подобны построенным выше, но их глубина — около $4m$.]

45. Этот аргумент действительно проще, но он недостаточно строг для доказательства требующегося результата. (Множество цепочек с нулевым ветвлением по выходу завышают простую оценку.) Доказательство, приведенное в тексте раздела, дано Д. Э. Сэведжем (J. E. Savage) в его книге *The Complexity of Computing* (New York: Wiley, 1976), теорема 3.4.1.

46. Когда $r = 2^n/n + O(1)$, мы имеем $\ln(2^{2r+1}(n+r-1)^{2r}/(r-1)!) = r\ln r + (1+\ln 4)r + O(n) = (2^n/n)(n\ln 2 - \ln n + 1 + \ln 4) + O(n)$. Так что $\alpha(n) \le (n/(4e))^{-2^n/n+O(n/\log n)}$, что достаточно быстро стремится к нулю при $n > 4e$.

(В действительности (32) дает $\alpha(11) < 7.6\times 10^7$, $\alpha(12) < 4.2\times 10^{-6}$, $\alpha(13) < 1.2\times 10^{-38}$.)

47. Ограничим перестановки $(r-m)!$ случаями, где $i\pi = i$ для $1 \le i \le n$, а $(n+r+1-k)\pi$ представляет собой k-й выход. Тогда мы получим $(r-m)!\,c(m,n,r) \le 2^{2r+1}(n+r-1)^{2r}$ вместо (32). Следовательно, как в упр. 46, почти все такие функции имеют стоимость, превосходящую $2^n m/(n + \lg m)$ при $m = O(2^n/n^2)$.

48. (а) Не удивительно, что эта нижняя граница $C(n)$ достаточно грубая при малых n:

$n =$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$r(n) =$	1	1	2	3	5	9	16	29	54	99	184	343	639	1196	2246	4229

(б) Метод раскрутки (см. *Concrete Mathematics* §9.4) дает

$$r(n) = \frac{2^n}{n}\Bigl(1 + \frac{\lg n - 2 - 1/\ln 2}{n} + O\Bigl(\frac{\log n}{n^2}\Bigr)\Bigr).$$

49. Количество нормальных булевых функций, которые могут быть представлены формулой длиной $\le r$, не превышает $5^r n^{r+1} g_r$, где g_r — количество ориентированных бинарных деревьев с r внутренними узлами. Положим в этой формуле $r = 2^n/\lg n - 2^{n+2}/(\lg n)^2$ и разделим на 2^{2^n-1}, чтобы получить верхнюю границу доли функций, у которых $L(f) \le r$. Результат быстро стремится к нулю согласно упр. 2.3.4.4–7, поскольку он представляет собой $O((5\alpha/16)^{2^n/\lg n})$, где $\alpha \approx 2.483$.

[Д. Риордан (J. Riordan) и К. Э. Шеннон (C. E. Shannon) получили аналогичную нижнюю границу для последовательно-параллельных коммутируемых сетей в *J. Math. and Physics* **21** (1942), 83–93; такие сети эквивалентны формулам, в которых используются

только канализирующие операторы. Р. Е. Кричевский получил более общие результаты в *Проблемы кибернетики* **2** (1959), 123–138, а О. Б. Лупанов дал асимптотическую верхнюю границу в *Проблемы кибернетики* **3** (1960), 61–80.]

50. (а) Если воспользоваться обозначениями подкубов, как в упр. 7.1.1–30, то простыми импликантами являются 00001∗, (0001∗1), 0100∗1, 0111∗1, 1010∗1, 101∗11, 00∗011, 00∗101, (01∗111), 11∗101, (0∗1101), (1∗0101), 1∗1011, 0∗0∗11, ∗00101, (∗01011), (∗11101), где подкубы в скобках в кратчайшей дизъюнктивной нормальной форме пропущены. (б) Аналогично простые дизъюнкты и кратчайшая конъюнктивная нормальная форма задаются как 00111∗, 01010∗, 10110∗, 0110∗∗, 00∗00∗, 11∗00∗, 11∗11∗, (0∗100∗), (1∗00∗∗), 1∗0∗1∗, (1∗∗∗∗0), ∗0000∗, (∗1100∗), ∗1∗∗∗0, ∗∗1∗∗0, ∗∗∗1∗0 и (∗∗∗∗00). (Таким образом, конъюнктивная нормальная форма имеет вид $(x_1 \lor x_2 \lor \bar x_3 \lor \bar x_4 \lor \bar x_5) \land (x_1 \lor \bar x_2 \lor x_3 \lor \bar x_4 \lor x_5) \land \cdots \land (\bar x_4 \lor x_6)$.)

51. $f = \big([x_5x_6 \in \{01\}] \land [(x_1x_2x_3x_4)_2 \in \{1,3,4,7,9,10,13,15\}]\big) \lor \big([x_5x_6 \in \{10,11\}] \land [x_1x_2x_3x_4 = 0000]\big) \lor \big([x_5x_6 \in \{11\}] \land [(x_1x_2x_3x_4)_2 \in \{1,2,4,5,7,10,11,14\}]\big)$.

52. Результаты для малых n существенно отличаются от асимптотических:

n	k	l	(38)	n	k	l	(38)	n	k	l	(38)	n	k	l	(38)
5	2	2	39	8	3	2	175	11	4	4	803	14	5	5	4045
6	2	2	67	9	3	2	279	12	4	3	1329	15	5	5	7141
7	2	1	109	10	4	4	471	13	5	6	2355	16	5	4	12431

(Эти верхние границы достаточно слабы при малых n. Например, мы знаем, что $C(n) = (0,1,4,7,12)$, когда $n = (1,2,3,4,5)$; а 7.1.1–(16) дает $C(n+1) \le 2C(n)+2$, так что $C(6) \le 26$, $C(7) \le 54$, и т. д.)

53. Сначала заметим, что $2^k/l \le n - 3\lg n$, следовательно, $m_i \le n - 3\lg n + 1$ и $2^{m_i} = O(2^n/n^3)$. Также $l = O(n)$ и $t(n-k) = O(2^n/n^2)$. Так что (38) сводится к $l \cdot 2^{n-k} + O(2^n/n^2) = 2^n/(n - 3\lg n) + O(2^n/n^2)$.

54. Эвристика жадных отпечатков дает цепочку длиной 14:

$$\begin{array}{rrr}
x_5 = x_1 \oplus x_3, & x_{10} = x_4 \land \bar x_5, & f_3 = x_{15} = \bar x_8 \land x_9, \\
x_6 = x_2 \oplus x_3, & x_{11} = x_4 \oplus x_5, & f_4 = x_{16} = x_4 \land x_8, \\
x_7 = x_1 \land x_2, & x_{12} = x_6 \land x_{11}, & f_5 = x_{17} = x_7 \land x_9, \\
x_8 = x_1 \land \bar x_6, & f_1 = x_{13} = \bar x_7 \land x_{12}, & f_6 = x_{19} = x_6 \land x_{10}. \\
x_9 = x_4 \land x_5, & f_2 = x_{14} = \bar x_6 \land x_{10}, &
\end{array}$$

Метод первоначального вычисления минитермов соответствует цепочке длиной 22, после того как мы удалили никогда не использующиеся шаги:

$$\begin{array}{rrr}
x_5 = \bar x_1 \land \bar x_2, & x_{13} = x_5 \land x_{10}, & x_{20} = x_8 \land x_{11}, \\
x_6 = \bar x_1 \land x_2, & x_{14} = x_5 \land x_{11}, & f_6 = x_{21} = x_{15} \lor x_{18}, \\
x_7 = x_1 \land \bar x_2, & x_{15} = x_6 \land x_9, & f_1 = x_{22} = x_{13} \lor x_{21}, \\
x_8 = x_1 \land x_2, & x_{16} = x_6 \land x_{11}, & f_2 = x_{23} = x_{12} \lor x_{20}, \\
x_9 = \bar x_3 \land x_4, & x_{17} = x_7 \land x_9, & x_{24} = x_{14} \lor x_{16}, \\
x_{10} = x_3 \land \bar x_4, & x_{18} = x_7 \land x_{11}, & f_3 = x_{25} = x_{24} \lor x_{19}, \\
x_{11} = x_3 \land x_4, & f_5 = x_{19} = x_8 \land x_9, & f_4 = x_{26} = x_{17} \lor x_{20}. \\
x_{12} = x_5 \land x_9, & &
\end{array}$$

(Закон дистрибутивности мог бы заменить вычисление x_{14}, x_{16} и x_{24} двумя шагами.)

Кстати, три функции в ответе к упр. 51 могут быть вычислены всего за десять шагов:

$$\begin{array}{rrr}
x_5 = x_2 \lor x_4, & f_3 = x_9 = x_6 \oplus x_8, & x_{12} = x_2 \oplus x_3, \\
x_6 = \bar x_1 \land x_5, & x_{10} = x_1 \oplus x_8, & x_{13} = \bar x_{10} \land x_{12}, \\
x_7 = x_2 \land x_4, & \bar f_2 = x_{11} = x_9 \lor x_{10}, & f_1 = x_{14} = x_4 \oplus x_{13}. \\
x_8 = x_3 \land \bar x_7, & &
\end{array}$$

55. Оптимальные двухуровневые представления в дизъюнктивной нормальной форме и конъюнктивной нормальной форме в ответе к упр. 50 имеют стоимости 53 и 43 соответственно. Формула (37) имеет стоимость 29, будучи оптимизированной, как в упр. 54. Альтернативное решение в упр. 51 имеет стоимость всего лишь 17. Но каталог оптимальных цепочек для пяти переменных предлагает для данной функции от шести переменных следующую цепочку:

$$\begin{array}{llll}
x_7 = \bar{x}_1 \land x_2, & x_{11} = x_5 \land x_{10}, & x_{15} = x_{13} \oplus x_{14}, & x_{18} = \bar{x}_4 \land x_{17}, \\
x_8 = x_3 \oplus x_7, & x_{12} = x_5 \lor x_{10}, & x_{16} = x_5 \land \bar{x}_{10}, & x_{19} = x_6 \land x_{15}, \\
x_9 = x_2 \land x_8, & x_{13} = x_4 \land \bar{x}_{11}, & x_{17} = \bar{x}_3 \land x_{16}, & x_{20} = x_{18} \lor x_{19}. \\
x_{10} = x_1 \oplus x_9, & x_{14} = x_8 \land x_{12}, & &
\end{array}$$

Существует ли лучшее решение?

56. Если нас интересуют не более двух значений, то функция представляет собой либо константу, либо x_j или $\bar{x}_j$.

57. Таблицы истинности для значений от x_5 до x_{15} в шестнадцатеричной записи представляют собой соответственно `0fff`, `3ccc`, `30c0`, `75d5`, `4919`, `7000`, `0606`, `4808`, `2000`, `5d5d`, `3ece`. Так что мы получаем

$$1010 \mapsto \mathrm{0}, \quad 1011 \mapsto \mathrm{7}, \quad 1100 \mapsto \mathrm{4}, \quad 1101 \mapsto \mathrm{5}, \quad 1110 \mapsto \mathrm{6}, \quad 1111 \mapsto \mathrm{7}.$$

[Кори Пловер (Corey Plover), полагающий, что было бы лучше иметь решение, в котором не цифры никогда не принимают вид цифр, открыл цепочку длиной 12 (с не жадным выбором x_7)

$$\begin{array}{lll}
x_5 = x_1 \oplus x_2, & x_9 = x_2 \oplus x_3, & \bar{b} = x_{13} = x_2 \land \bar{x}_{11}, \\
x_6 = x_3 \land \bar{x}_4, & g = x_{10} = x_7 \lor x_9, & \bar{c} = x_{14} = x_7 \land x_9, \\
x_7 = x_1 \oplus x_6, & \bar{d} = x_{11} = x_8 \oplus x_{10}, & \bar{e} = x_{15} = x_4 \lor x_{12}, \\
x_8 = x_4 \lor x_7, & \bar{a} = x_{12} = \bar{x}_3 \land x_{11}, & \bar{f} = x_{16} = \bar{x}_5 \land x_8,
\end{array}$$

при которой a, ..., g имеют таблицы истинности `b7ff`, `f9f0`, `dfe3`, `b6df`, `a2aa`, `8ff2`, `3efd` и

$$1010 \mapsto \mathrm{C}, \quad 1011 \mapsto \mathrm{7}, \quad 1100 \mapsto \mathrm{5}, \quad 1101 \mapsto \mathrm{1}, \quad 1110 \mapsto \mathrm{u}, \quad 1111 \mapsto \mathrm{3}.$$

Он также показал, что все 11-шаговые решения (44) отображают не цифры на один из вариантов $(0,7,4,5,6,7)$, $(0,7,6,1,6,3)$, $(0,7,6,1,C,3)$, $(2,3,6,7,6,7)$, $(2,1,6,7,4,5)$ или $(2,4,6,7,4,5)$.]

58. Таблицы истинности всех функций со стоимостью 7 с ровно восемью единицами в их таблицах истинности равны `0779`, `169b` либо `179a`. Комбинируя их всеми возможными способами, получаем 9656 решений, различных относительно перестановок и/или дополнения $\{x_1, x_2, x_3, x_4\}$, так же, как и относительно перестановок и/или дополнения $\{f_1, f_2, f_3, f_4\}$.

59. Жадная эвристика на основе отпечатков дает следующую 17-шаговую цепочку:

$$\begin{array}{lll}
x_5 = x_2 \oplus x_3, & x_{11} = x_2 \lor x_7, & x_{17} = \bar{x}_6 \land x_8, \\
x_6 = x_1 \oplus x_4, & x_{12} = x_2 \land \bar{x}_6, & f_1 = x_{18} = x_{11} \oplus x_{17}, \\
x_7 = x_1 \oplus x_3, & x_{13} = x_3 \land x_4, & f_2 = x_{19} = x_{10} \land \bar{x}_{14}, \\
x_8 = x_4 \lor x_5, & x_{14} = x_4 \land x_5, & f_3 = x_{20} = x_9 \oplus x_{16}, \\
x_9 = x_6 \land x_8, & x_{15} = x_5 \land x_{10}, & f_4 = x_{21} = x_{12} \oplus x_{15}. \\
x_{10} = x_7 \lor x_9, & x_{16} = x_2 \land \bar{x}_{13}, &
\end{array}$$

Все исходные функции имеют больши́е отпечатки, так что получить $C(f_1 f_2 f_3 f_4) = 28$ мы не можем; но, вероятно, может существовать немного более трудный S-блок.

60. Одним из способов является $u_1 = x_1 \oplus y_1$, $u_2 = x_2 \oplus y_2$, $v_1 = y_2 \oplus u_1$, $v_2 = y_1 \oplus u_2$, $z_1 = v_1 \land \bar{u}_2$, $z_2 = v_2 \land \bar{u}_1$.

61. Приведенное решение Дэвида Стивенсона (David Stevenson) с 17 логическими элементами обобщается на $8m+1$ логических элементов для суммирования по модулю 2^m+1: $u_0 = x_0 \wedge y_0$, $v_0 = x_0 \oplus y_0$, $u_1 = x_1 \wedge y_1$, $v_1 = x_1 \oplus y_1$, $t_1 = v_1 \wedge u_0$, $t_2 = v_1 \oplus u_0$, $c_2 = u_1 \vee t_1$; $u_2 = x_2 \wedge y_2$, $t_3 = x_2 \vee y_2$, $t_4 = t_3 \vee c_2$; $t_5 = t_2 \vee v_0$, $t_6 = t_5 \wedge t_4$, $t_7 = t_6 \vee u_2$; $t_8 = t_7 \wedge \bar{v}_0$, $z_0 = t_7 \oplus v_0$, $z_1 = t_2 \oplus t_8$; $z_2 = t_4 \oplus t_7$. (Обратите внимание, что $(x_2x_1x_0)_2 + (y_2y_1y_0)_2 = (u_2t_4t_2v_0)_2 - 4[x=y=4]$. Гильберт Ли (Gilbert Lee) нашел другое 17-шаговое решение, если входные данные представлены значениями 000, 001, 011, 101 и 111.)

62. Имеется $\binom{2^n}{2^nd}2^{2^nc}$ таких функций, не более $\binom{2^n}{2^nd}c(n,r)$ из которых имеют стоимость $\le r$. Так что мы можем рассуждать, как в упр. 46, чтобы из (32) заключить, что доля со стоимостью $\le r = \lfloor 2^nc/n \rfloor$ не превышает $2^{2r+1-2^nc}(n+r-1)^{2r}/(r-1)! = 2^{-r\lg n + O(r)}$.

63. [*Проблемы кибернетики* **21** (1969), 215–226.] Поместим таблицу истинности в массив $2^k \times 2^{n-k}$, как в методе Лупанова, и допустим, что в столбце j для $0 \le j < 2^{n-k}$ имеется c_j определенных значений. Разобъем этот столбец на $\lfloor c_j/m \rfloor$ подстолбцов, каждый из которых содержит m определенных значений, плюс (возможно, пустой) подстолбец внизу, который содержит менее m из них. Указание говорит нам, что не более 2^{m+k} векторов столбцов достаточно для соответствия нулям и единицам каждого подстолбца с определенной верхней строкой i_0 и нижней строкой i_1. Таким образом, с помощью $O(m2^{m+3k})$ операций можно построить $O(2^{m+3k})$ функций $g_t(x_1,\ldots,x_k)$ из минитермов из $\{x_1,\ldots,x_k\}$, так что каждый подстолбец соответствует некоторому типу t. А для каждого типа t можно построить функции $h_t(x_{k+1},\ldots,x_n)$ из минитермов из $\{x_{k+1},\ldots,x_n\}$, определяющих столбцы, которые соответствуют t; соответствующая стоимость не превышает $\sum_j(\lfloor c_j/m \rfloor + 1) \le 2^nc/m + 2^{n-k}$. Наконец $f = \bigvee_t(g_t \wedge h_t)$ требует $O(2^{m+3k})$ дополнительных шагов. Выбор $k = \lfloor 2\lg n \rfloor$ и $m = \lceil n - 9\lg n \rceil$ делает общую стоимость не превышающей $(2^nc/n)(1 + 9n^{-1}\lg n + O(n^{-1}))$.

Конечно, мы должны доказать указание, что сделано Э. И. Нечипоруком [*Доклады Академии наук СССР* **163** (1965), 40–42]. В действительности достаточно $2^m(1+\lceil k\ln 2\rceil)$ векторов (см. S. K. Stein, *J. Combinatorial Theory* **A16** (1974), 391–397): если мы выберем $q = 2^m\lceil k\ln 2\rceil$ векторов случайным образом, не обязательно различных, то ожидаемое количество несоприкасающихся подкубов будет равно $\binom{k}{m}2^m(1-2^{-m})^q < \binom{k}{m}2^me^{-q2^{-m}} < 2^m$. (Явное построение было бы более красивым.)

Существенное обобщение можно найти в работе N. Pippenger, *Mathematical Systems Theory* **10** (1977), 129–167.

64. Эта игра — не что иное, как игра в крестики-нулики, если мы перенумеруем клетки $\begin{smallmatrix}6&1&8\\7&5&3\\2&9&4\end{smallmatrix}$, как в древнем китайском магическом квадрате. [Берлекамп (Berlekamp), Конвей (Conway) и Гай (Guy) использовали эту схему нумерации, чтобы представить полный анализ игры в крестики-нулики в своей книге *Winning Ways* **3** (2003), 732–736.]

65. Одним из решений является замена "обороняющихся" ходов d_j "атакующими" ходами a_j и "контратакующими" ходами c_j и включение их только для угловых клеток $j \in \{1,3,9,7\}$. Пусть $j \cdot k = (jk) \bmod 10$; тогда

$$\begin{array}{ccc} j\cdot 1 & j\cdot 2 & j\cdot 3 \\ j\cdot 4 & j\cdot 5 & j\cdot 6 \\ j\cdot 7 & j\cdot 8 & j\cdot 9 \end{array}$$

дает нам другой способ взглянуть на диаграмму игры в крестики-нулики, когда j является углом, поскольку $j \perp 10$. Точное определение a_j и c_j в таком случае имеет вид

$$\begin{aligned} a_j = m_j \wedge \big((x_{j\cdot 3} \wedge \beta_{(j\cdot 8)(j\cdot 9)} \wedge (o_{j\cdot 4} \oplus o_{j\cdot 6})\big) \vee \big(x_{j\cdot 7} \wedge \beta_{(j\cdot 6)(j\cdot 9)} \wedge (o_{j\cdot 2} \oplus o_{j\cdot 8})\big) \\ \vee \big(m_{j\cdot 9} \wedge \big((m_{j\cdot 8} \wedge x_{j\cdot 2} \wedge \overline{(o_{j\cdot 3} \oplus o_{j\cdot 6})}) \vee (m_{j\cdot 6} \wedge x_{j\cdot 4} \wedge \overline{(o_{j\cdot 7} \oplus o_{j\cdot 8})})\big)\big)\big); \\ c_j = d_j \wedge \overline{(x_{j\cdot 6} \wedge o_{j\cdot 7})} \wedge \overline{(x_{j\cdot 8} \wedge o_{j\cdot 3})} \wedge \bar{d}_{j\cdot 9}; \end{aligned}$$

здесь вместо (51) используется $d_j = m_j \wedge \beta_{(j\cdot 2)(j\cdot 3)} \wedge \beta_{(j\cdot 4)(j\cdot 7)}$. Мы также определяем

$$\begin{aligned} u &= (x_1 \oplus x_3) \oplus (x_7 \oplus x_9), \\ v &= (o_1 \oplus o_3) \oplus (o_7 \oplus o_9), \\ t &= m_2 \wedge m_6 \wedge m_8 \wedge m_4 \wedge (u \vee \bar{v}), \end{aligned} \qquad z_j = \begin{cases} m_j \wedge \bar{t}, & \text{если } j = 5, \\ m_j \wedge \bar{d}_{j\cdot 9}, & \text{если } j \in \{1,3,9,7\}, \\ m_j, & \text{если } j \in \{2,6,8,4\}, \end{cases}$$

для того, чтобы охватить несколько больше исключительных случаев. Наконец последовательность упорядоченных по рангу ходов $d_5 d_1 d_3 d_9 d_7 d_2 d_6 d_8 d_4 m_5 m_1 m_3 m_9 m_7 m_2 m_6 m_8 m_4$ в (53) заменяется последовательностью $a_1 a_3 a_9 a_7 c_1 c_3 c_9 c_7 z_5 z_1 z_3 z_9 z_7 z_2 z_6 z_8 z_4$; и мы заменяем $(d_j \wedge \bar{d}'_j) \vee (m_j \wedge \bar{m}'_j)$ в (55) на $(a_j \wedge \bar{a}'_j) \vee (c_j \wedge \bar{c}'_j) \vee (z_j \wedge \bar{z}'_j)$, когда j является угловой клеткой, а в противном случае просто на $(z_j \wedge \bar{z}'_j)$.

(Заметим, что эта машина должна ходить корректно из *любой* допустимой позиции, даже когда эти позиции не могут возникнуть после предыдущих ходов машины. Мы, по сути, разрешаем человеку играть до тех пор, пока он не попросит совета у машины. В противном случае были бы возможны существенные упрощения. Например, если $\times$ всегда ходит первым, можно было бы захватывать центральную клетку и убрать тем самым огромное количество будущих вариантов; при этом может получиться менее чем $8 \times 6 \times 4 \times 2 = 384$ партии. Даже если первым ходит $\circ$, имеется менее чем $9 \times 7 \times 5 \times 3 = 945$ возможных сценариев. В действительности реальное количество различных партий с определенной здесь стратегией оказывается равным $76 + 457$, из которых в $72 + 328$ выигрывает машина, а в остальных случаях получается ничья.)

66. Булева цепочка в ответе к предыдущему упражнению выполняет поставленную перед ней задачу и делает корректные ходы из всех 4520 допустимых позиций, где корректность, по сути, определена как наилучший ход в наихудшем случае. Но настоящий игрок в крестики-нулики поступает не так. Например, в позиции машина захватит центр, , и $\circ$, вероятно, походит в угловую клетку. Но ход или оставит $\circ$ только два шанса избежать поражения. [См. Martin Gardner, *Hexaflexagons and Other Mathematical Diversions*, Chapter 4.]

Кроме того, лучшим ходом в позиции наподобие является , а не немедленная победа. Затем, если ответным ходом является , выполняется ход . Таким образом, вы все равно выигрываете, но без унизительного для противника "киндермата".

Наконец неверна сама концепция единственного "наилучшего хода", поскольку хороший игрок, как заметил Бэббидж (Babbage), в разных играх выбирает разные ходы.

> *Можно решить, что программирование компьютера для игры в крестики-нулики, или разработка специальной схемы для машины, играющей в крестики-нулики, — простое дело. Это так и есть, если только ваша цель не заключается в создании робота, который выигрывал бы максимальное количество игр у неопытного игрока.*
>
> — МАРТИН ГАРДНЕР (MARTIN GARDNER),
> *Математические головоломки и развлечения*
> *(The Scientific American Book of*
> *Mathematical Puzzles & Diversions) (1959)*

67. Наилучшее известное в настоящее время решение принадлежит Дэвиду Стивенсону (David Stevenson). Оно было получено им в 2010 году и использует всего 818 логических элементов (472 И, 327 ИЛИ, 13 НИ, 6 НО-НЕ); см. подробную информацию по адресу

`http://www-cs-faculty.stanford.edu/~knuth/818-gate-solution.`

После обработки таких ходов, как w_j и b_j, и искусной оптимизации безразличных значений Стивенсон, по сути, выполняет ИЛИ для около 200 специальных позиций (таких, как),

которые делают $c = 1$, около 200 других (таких, как $\begin{smallmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{smallmatrix}$), которые делают $s = 1$, и около 50 (таких, как $\begin{smallmatrix} \times & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{smallmatrix}$), которые делают $m = 1$; затем он экономит логические элементы путем поиска общих подвыражений среди И, которые определяют специальные позиции, а также с помощью закона дистрибутивности и т. п.

[Это упражнение вызвано дискуссией в книге John Wakerly *Digital Design* (Prentice–Hall, 3rd edition, 2000), §6.2.7. Кстати, Бэббидж (Babbage) планировал выбирать среди k возможных ходов с помощью значения $N \bmod k$, где N — количество игр, выигранных к настоящему моменту; он не понимал, что последовательные ходы будут сильно коррелированы, пока не изменится N. Гораздо лучше положить N равным количеству сделанных к этому времени *ходов*.]

68. Нет. Этот метод дает "однородную" цепочку с ясной структурой, но ее стоимость равна $\Omega(n2^n)$. Существует схема с примерно $2^n/n$ логическими элементами, построенная в соответствии с теоремой L, однако ее гораздо сложнее создать. (Кстати, $C(\pi_5) = 10$.)

69. (а) Можно проверить этот результат, например, испытав все 64 случая.

(б) Если x_m лежит в той же строке или столбце, что и x_i, а также в той же строке или столбце, что и x_j, мы имеем $\alpha_{111} = \alpha_{101} = \alpha_{011} = 0$, так что пары являются хорошими. В противном случае имеются три существенно разных возможности, и все они плохие: если $(i, j, m) = (1, 2, 4)$, то $\alpha_{101} = 0$, $\alpha_{100} = x_5x_9 \oplus x_6x_8$, $\alpha_{011} = x_9$; если $(i, j, m) = (1, 2, 6)$, то $\alpha_{010} = x_4x_9$, $\alpha_{011} = x_7$, $\alpha_{100} = x_5x_9$, $\alpha_{101} = x_8$; если $(i, j, m) = (1, 5, 9)$, то $\alpha_{111} = 1$, $\alpha_{110} = 0$, $\alpha_{010} = x_3x_7$.

70. (а) $x_1 \wedge ((x_5 \wedge x_9) \oplus (x_6 \wedge x_8)) \oplus x_2 \wedge ((x_6 \wedge x_7) \oplus (x_4 \wedge x_9)) \oplus x_3 \wedge ((x_4 \wedge x_8) \oplus (x_5 \wedge x_7))$.

(б) $x_1 \wedge ((x_5 \wedge x_9) \vee (x_6 \wedge x_8)) \vee x_2 \wedge ((x_6 \wedge x_7) \vee (x_4 \wedge x_9)) \vee x_3 \wedge ((x_4 \wedge x_8) \vee (x_5 \wedge x_7))$.

(в) Пусть $y_1 = x_1 \wedge x_5 \wedge x_9$, $y_2 = x_1 \wedge x_6 \wedge x_8$, $y_3 = x_2 \wedge x_6 \wedge x_7$, $y_4 = x_2 \wedge x_4 \wedge x_9$, $y_5 = x_3 \wedge x_4 \wedge x_8$, $y_6 = x_3 \wedge x_5 \wedge x_7$. Функция $f(y_1, \ldots, y_6) = [y_1 + y_2 + y_3 > y_4 + y_5 + y_6]$ может быть вычислена пятнадцатью последующими шагами с двумя полными сумматорами и компаратором; однако имеется и 14-шаговое решение. Пусть $z_1 = (y_1 \oplus y_2) \oplus y_3$, $z_2 = (y_1 \oplus y_2) \vee (y_1 \oplus y_3)$, $z_3 = (y_4 \oplus y_5) \oplus y_6$, $z_4 = (y_4 \oplus y_5) \vee (y_4 \oplus y_6)$. Тогда $f = (z_1 \oplus (z_2 \wedge (\bar z_4 \oplus (z_1 \vee z_3)))) \wedge (\bar z_3 \vee z_4)$. Кроме того, $y_1y_2y_3 = 111 \iff y_4y_5y_6 = 111$; так что имеются безразличные значения, приводящие к 11-шаговому решению: $f = ((\bar z_1 \wedge z_3) \vee \bar z_4) \wedge z_2$. Общая стоимость равна $12 + 11 = 23$.

(Автору неизвестен способ, который позволил бы компьютеру найти такую эффективную цепочку за приемлемое время по одной лишь таблице истинности f. Но, вероятно, лучшая цепочка все же имеется.)

71. (а) $P(p) = 1 - 12p^2 + 24p^3 + 12p^4 - 96p^5 + 144p^6 - 96p^7 + 24p^8$, что равно $\frac{11}{32} + \frac{9}{2}\epsilon^2 - 3\epsilon^4 - 24\epsilon^6 + 24\epsilon^8$ при $p = \frac{1}{2} + \epsilon$.

(б) Имеется $N = 2^{n-3}$ множеств из восьми значений $(f_0, \ldots, f_7)$, каждое из которых дает хорошие пары с вероятностью $P(p)$. Так что ответ равен $1 - P(p)^N$.

(в) Вероятность того, что удачными окажутся ровно r множеств, равна $\binom{N}{r} P(p)^r (1 - P(p))^{N-r}$; а в подобном случае t испытаний будут находить хорошие пары с вероятностью $(r/N)^t$. Следовательно, ответ имеет вид $1 - \sum_{r=0}^{N} \binom{N}{r} P(p)^r (1 - P(p))^{N-r} (r/N)^t = 1 - P(p)^t + O(t^2/N)$.

(г) $\sum_{r=0}^{N} \binom{N}{r} P(p)^r (1 - P(p))^{N-r} \sum_{j=0}^{t-1} (r/N)^j = (1 - P(p)^t)/(1 - P(p)) + O(t^3/N)$.

72. Вероятность в ответе к упр. 71, (а) становится равной $P(p) + (72p^3 - 264p^4 + 432p^5 - 336p^6 + 96p^7)r + (60p^2 - 240p^3 + 456p^4 - 432p^5 + 144p^6)r^2 + (-48p^2 + 144p^3 - 216p^4 + 96p^5)r^3 + (-36p^2 + 24p^3 + 12p^4)r^4 + (48p^2 - 24p^3)r^5 - 12p^2r^6$. Если $p = q = (1 - r)/2$, то это значение равно $(11 + 48r + 36r^2 - 144r^3 - 30r^4 + 336r^5 - 348r^6 + 144r^7 - 21r^8)/32$; например, при $r = 1/2$ это $7739/8192 \approx 0.94$.

73. Рассмотрим дизъюнкты Хорна $1\wedge 2\Rightarrow 3$, $1\wedge 3\Rightarrow 4$, $\ldots$, $1\wedge(n-1)\Rightarrow n$, $1\wedge n\Rightarrow 2$ и $i\wedge j\Rightarrow 1$ для $1<i<j\le n$. Предположим, что в разложении $|Z|>1$, и пусть i представляет собой минимум, такой, что $x_i\in Z$. Пусть также j представляет собой минимум, такой, что $j>i$ и $x_j\in Z$. Мы не можем иметь $i>1$, поскольку в этом случае $i\wedge j\Rightarrow 1$. Таким образом, $i=1$, и $x_j\in Z$ для $2\le j\le n$.

74. Предположим, что мы знаем, что не существует нетривиального разложения с $x_1\in Z$ или $\cdots$ или $x_{i-1}\in Z$; изначально $i=1$. Мы надеемся исключить и $x_i\in Z$ путем искусного выбора j и m. Дизъюнкты Хорна $i\wedge j\Rightarrow m$ сводятся к дизъюнктам Крома $j\Rightarrow m$ при принятии i. Так что, по сути, мы хотим использовать поиск в глубину Таржана (Tarjan) для сильно связных компонентов в ориентированном графе с дугами $j\Rightarrow m$, которые могут как существовать, так и отсутствовать.

При исследовании от вершины j сначала испытаем $m=1$, $\ldots$, $m=i-1$; если любая такая импликация $i\wedge j\Rightarrow m$ успешна, мы можем удалить j и всех его предшественников из ориентированного графа для i. В противном случае проверяем $j\Rightarrow m$ для всех таких удаленных вершин m. В противном случае проверяем неисследованные вершины m. В противном случае проверяем вершины m, которые уже были просмотрены, оказывая предпочтение тем, которые находятся ближе к корню дерева поиска вглубь.

В примере $f(x)=(\det X)\bmod 2$ мы поочередно находим $1\wedge 2\not\Rightarrow 3$, $1\wedge 2\Rightarrow 4$, $1\wedge 4\Rightarrow 3$, $1\wedge 3\Rightarrow 5$, $1\wedge 5\Rightarrow 6$, $1\wedge 6\Rightarrow 7$, $1\wedge 7\Rightarrow 8$, $1\wedge 8\Rightarrow 9$, $1\wedge 9\Rightarrow 2$ (теперь $i\leftarrow 2$); $2\wedge 3\not\Rightarrow 1$, $2\wedge 3\Rightarrow 4$, $2\wedge 4\not\Rightarrow 1$, $2\wedge 4\not\Rightarrow 5$, $2\wedge 4\Rightarrow 6$, $2\wedge 6\Rightarrow 1$ (теперь 3, 4 и 6 удалены из ориентированного графа для 2), $2\wedge 5\Rightarrow 1$ (удалено и 5), $2\wedge 7\not\Rightarrow 1$, $2\wedge 7\Rightarrow 3$ (удалено 7), $2\wedge 8\Rightarrow 1$, $2\wedge 9\Rightarrow 1$ (теперь $i\leftarrow 3$); $3\wedge 4\not\Rightarrow 1$, $3\wedge 4\Rightarrow 2$, $3\wedge 5\Rightarrow 1$, и т. д.

75. Эта функция равна 1 только в двух точках, являющихся дополнениями одна для другой. Так что эта функция неразложима; пары (58) *никогда* не могут быть плохими при $n>3$. Каждое разбиение (Y,Z) будет, таким образом, кандидатом на разложение.

Аналогично, если f является разложимой по отношению к (Y,Z), то неразложимая функция $f(x)\oplus S_{0,n}(x)$ будет действовать при этих проверках, по сути, как f. (В программе проверки разложимости общего назначения, вероятно, должен присутствовать метод для работы с *почти разложимыми функциями*.)

76. (а) Пусть $a_l=[i\ge l]$ для $0\le l\le 2^m$. Как было замечено в ответе к упр. 38, (б), стоимость $\le 2t(m)$; и действительно, стоимость может быть снижена до $2^{m+1}-2m-2$ при глубине $\Theta(m)$. Кроме того, функция $[i\le j]=(\bar i_1\wedge j_1)\vee((i_1\equiv j_1)\wedge[i_2\ldots i_m\le j_2\ldots j_m])$ может быть вычислена с помощью $4m-3$ логических элементов. После вычисления $x\oplus y$ каждое z_l стоит $2^{m+1}+1=O(n)$.

(б) Здесь стоимость не превышает $C(g_0)+\cdots+C(g_{2^m})\le(2^m+1)(2^{2^m}/(2^m-O(m)))$ согласно теореме L, поскольку каждое g_l представляет собой функцию от 2^m входных данных.

(в) Если $i\le j$, мы имеем $z_l=x$ для $l\le i$ и $z_l=y$ для $l>i$; следовательно, $f_i(x)=c_0\oplus\cdots\oplus c_i$ и $f_j(y)=c_{j+1}\oplus\cdots\oplus c_{2^m}$. Если $i>j$, мы имеем $z_l=y$ для $l\le i$ и $z_l=x$ для $l>i$; следовательно, $f_j(y)=c_0\oplus\cdots\oplus c_j$ и $f_i(x)=c_{i+1}\oplus\cdots\oplus c_{2^m}$.

(г) Функции $b_l=[j<l]$ могут быть вычислены для $0\le l\le 2^m$ за $O(2^m)$ шагов, как в п. (а). Так что мы можем вычислить F из $(c_0,\ldots,c_{2^m})$ с $O(2^m)$ дополнительными логическими элементами. Таким образом, шаг (б) доминирует в стоимости при больших m.

(д) $a_0=1$, $a_1=i$, $a_2=0$; $b_0=0$, $b_1=j$, $b_2=1$; $d=[i\le j]=\bar i\vee j$; $m_l=a_l\oplus d$, $z_{l0}=x_0\oplus(m_l\wedge(x_0\oplus y_0))$, $z_{l1}=x_1\oplus(m_l\wedge(x_1\oplus y_1))$ для $l=0$, 1, 2; $c_0=z_{01}$; $c_1=z_{10}\wedge\bar z_{11}$; $c_2=z_{20}\vee z_{21}$; $c'_l=c_l\wedge(d\equiv a_l)$, $c''_l=c_l\wedge(d\equiv b_l)$ для $l=0$, 1, 2; и наконец $F=(c'_0\oplus c'_1\oplus c'_2)\vee(c''_0\oplus c''_1\oplus c''_2)$.

Себестоимость (29 после очевидных упрощений), конечно, вопиюще велика для такого малого примера. Удивит это вас или нет, но современный оптимизатор оказывается способным сократить эту цепочку всего до 5 логических элементов.

[Этот результат является частным случаем более общих теорем из *Математические заметки* **15** (1974), 937–944; *London Math. Soc. Lecture Note Series* **169** (1992), 165–173.]

77. Пусть задана такая кратчайшая цепочка для f_n или $\bar f_n$. Пусть $U_l = \{i \mid l = j(i)$ или $l = k(i)\}$ — количество "использований" x_l и пусть $u_l = |U_l|$. Пусть $t_i = 1$, если $x_i = x_{j(i)} \vee x_{k(i)}$; в противном случае $t_i = 0$. Мы покажем, что имеется цепочка длиной $\le r - 4$, которая вычисляет либо f_{n-1}, либо $\bar f_{n-1}$, воспользовавшись следующей идеей. Если переменной x_m присвоено значение 0 или 1 для любого m, мы можем получить цепочку для f_{n-1} или $\bar f_{n-1}$, удаляя все шаги U_m и соответствующим образом изменяя другие шаги. Кроме того, если $x_i = x_{j(i)} \circ x_{k(i)}$ и если известно, что либо $x_{j(i)}$, либо $x_{k(i)}$ равно t_i при значении x_m, установленном равным 0 или 1, то мы можем также удалить шаги U_i. (На протяжении нашего решения буква m будет означать индекс в диапазоне $1 \le m \le n$.)

Случай 1. $u_m = 1$ для некоторого m. Этот случай не может осуществиться в кратчайшей цепочке. Если единственное использование x_m представляет собой $x_i = \bar x_m$, устранение этого шага изменит $f_n \leftrightarrow \bar f_n$; в противном случае мы могли бы установить значения x_1, $\ldots$, x_{m-1}, x_{m+1}, $\ldots$, x_n, чтобы сделать x_i независимым от x_m, что противоречит $x_{n+r} = f_n$ или $\bar f_n$. Таким образом, каждая переменная должна быть использована как минимум дважды.

Случай 2. $x_l = \bar x_m$ для некоторых l и m, где $u_m > 1$. Тогда $x_i = x_l \circ x_k$ для некоторых i и k, и мы можем установить $x_m \leftarrow \bar t_i$, чтобы сделать x_i независимым от x_k. Тогда удаление шагов U_m, U_l и U_i приведет к удалению как минимум 4 шагов, за исключением случая, когда $u_l = u_i = 1$ и $u_m = 2$, и $x_j = x_m \circ x_i$; но в этом случае мы можем также удалить U_j.

Случай 3. $u_m \ge 3$ для некоторого m, но случай 2 не осуществляется. Если $i, j, k \in U_m$ и $i < j < k$, установим $x_m \leftarrow t_k$ и удалим шаги i, j, k, U_k.

Случай 4. $u_1 = u_2 = \cdots = u_n = 2$, но случай 2 не осуществляется. Можно считать, что первым шагом является $x_{n+1} = x_1 \circ x_2$ и что $x_l = x_1 \circ x_k$ для некоторого $k < l$.

Случай 4.1. $k > n$. Тогда $k > n + 1$. Если $u_k = 1$, установим $x_1 \leftarrow t_l$ и удалим шаги $n+1$, k, l, U_l. В противном случае установим $x_2 \leftarrow t_{n+1}$; это обеспечит $x_k = \bar t_l$, и мы можем удалить $n + 1$, k, l, U_k.

Случай 4.2. $x_l = x_1 \circ x_m$.Тогда мы должны иметь $m = 2$; если $m > 2$, мы могли бы установить $x_2 \leftarrow t_{n+1}$, $x_m \leftarrow t_l$ и сделать x_{n+r} независимым от x_1. Следовательно, мы можем считать, что $x_{n+1} = x_1 \wedge x_2$, $x_{n+2} = x_1 \vee x_2$. Установка $x_1 \leftarrow 0$ позволяет нам удалить U_1 и U_{n+1}; установка $x_1 \leftarrow 1$ позволяет нам удалить U_1 и U_{n+2}. Таким образом мы действуем, пока не получаем $u_{n+1} = u_{n+2} = 1$.

Если $x_p = \bar x_{n+1}$, установим $x_1 \leftarrow 0$ и удалим $n + 1$, $n + 2$, p, U_p; если $x_q = \bar x_{n+2}$, установим $x_1 \leftarrow 1$ и удалим $n + 1$, $n + 2$, q, U_q. В противном случае $x_p = x_{n+1} \circ x_u$ и $x_q = x_{n+2} \circ x_v$, где x_u и x_v не зависят от x_1 или x_2. Но это невозможно; это позволило бы нам установить x_3, $\ldots$, x_n, чтобы сделать $x_u = t_p$, а затем $x_2 \leftarrow 1$, чтобы сделать x_{n+r} независимым от x_1.

[*Проблемы кибернетики* **23** (1970), 83–101; **28** (1974), 4. С помощью аналогичного доказательства Редькин показал, что кратчайшие И-ИЛИ-НЕ-цепочки для функций '$x_1 \ldots x_n < y_1 \ldots y_n$' и '$x_1 \ldots x_n = y_1 \ldots y_n$' имеют длины $5n - 3$ и $5n - 1$ соответственно.]

78. [*SICOMP* **6** (1977), 427–430.] Будем говорить, что y_k *активно*, если $k \in S$. Мы можем считать, что цепочка нормальная и что $\|S\| > 1$; доказательство при этом подобно доказательству Редькина из ответа к упр. 77.

Случай 1. Некоторое активное y_k используется более одного раза. Установка $y_k \leftarrow 0$ экономит как минимум два шага и дает цепочку для функции с $\|S\| - 1$ активных значений.

Случай 2. Некоторое активное y_k появляется только в логическом элементе И. Установка $y_k \leftarrow 0$ удаляет как минимум два шага, если только это И не является конечным шагом. Но оно не может быть конечным шагом, поскольку $y_k = 0$ делает результат не зависящим от каждого другого активного y_j.

Случай 3. Подобен случаю 2, но с логическим элементом ИЛИ, или НЕ-НО, или НО-НЕ. Установка $y_k \leftarrow c$ для некоторой подходящей константы c дает требуемый эффект.

Случай 4. Подобен случаю 2, но с логическим элементом ЛИБО. Этот логический элемент не может быть последним, поскольку результат должен быть независим от y_k, когда $(x_1 \ldots x_m)_2$ адресует другое активное значение y_j. Так что мы можем устранить два шага, установив значение y_k равным функции, определяемой *другим* входом логического элемента ЛИБО.

79. (а) Предположим, что стоимость равна $r < 2n-2$; тогда $n > 1$. Если каждая переменная используется ровно один раз, два листа должны быть напарниками. Следовательно, некоторые переменные используются как минимум дважды. Обрезка этих ветвей дает цепочку стоимостью $\le r-2$ от $n-1$ переменных, не имеющую напарников.

(Кстати, стоимость составляет не менее $2n-1$, если каждая переменная используется как минимум два раза, поскольку не менее $2n$ использований переменных должны быть связаны в цепочке вместе.)

(б) Заметим, что $S_{0,n} = \bigwedge_{u-v}(u \equiv v)$, когда ребра u — v образуют свободное дерево на множестве $\{x_1, \ldots, x_n\}$. Так что имеется много способов получить стоимость $2n-3$.

Любая цепочка со стоимостью $r < 2n-3$ должна иметь $n > 2$ и должна содержать напарников u и v. С помощью переименования и возможного дополнения промежуточных результатов можно считать, что $u = 1$, $v = 2$ и что $f(x_1, \ldots, x_n) = g(x_1 \circ h(x_3, \ldots, x_n), x_2, \ldots, x_n)$, где $\circ$ представляет собой $\wedge$ или $\oplus$.

Случай 1. $\circ$ представляет собой И. Мы должны иметь $h(0, \ldots, 0) = h(1, \ldots, 1) = 1$, поскольку в противном случае $f(x_1, x_2, y, \ldots, y)$ не будет зависеть от x_1. Следовательно, $f(x_1, \ldots, x_n) = h(x_3, \ldots, x_n) \wedge g(x_1, x_2, \ldots, x_n)$ может быть вычислена с помощью цепочки той же стоимости, что и цепочка, в которой 1 и 2 являются напарниками и в которой путь между ними короче.

Случай 2. $\circ$ представляет собой ЛИБО. Тогда $f = f_0 \vee f_1$, где $f_0(x_1, \ldots, x_n) = (x_1 \equiv h(x_3, \ldots, x_n)) \wedge g(0, x_2, \ldots, x_n)$ и $f_1(x_1, \ldots, x_n) = (x_1 \oplus h(x_3, \ldots, x_n)) \wedge g(1, x_2, \ldots, x_n)$. Но $f = S_{0,n}$ имеет только две простые импликанты; так что имеется только четыре возможности.

Случай 2, а. $f_0 = f$. Тогда мы можем заменить $x_1 \oplus h$ на 0, чтобы получить цепочку стоимостью $\le r-2$ для функции $g(0, x_2, \ldots, x_n) = S_{0,n-1}(x_2, \ldots, x_n)$.

Случай 2, б. $f_1 = f$. Аналогичен случаю 2, а.

Случай 2, в. $f_0(x) = x_1 \wedge \cdots \wedge x_n$ и $f_1(x) = \bar{x}_1 \wedge \cdots \wedge \bar{x}_n$. В этом случае мы должны иметь $g(0, x_2, \ldots, x_n) = x_2 \wedge \cdots \wedge x_n$ и $g(1, x_2, \ldots, x_n) = \bar{x}_2 \wedge \cdots \wedge \bar{x}_n$. Замена h на 1 дает, таким образом, цепочку, которая вычисляет f за $< r$ шагов.

Случай 2, г. $f_0(x) = \bar{x}_1 \wedge \cdots \wedge \bar{x}_n$ и $f_1(x) = x_1 \wedge \cdots \wedge x_n$. Аналогичен случаю 2, в.

Многократное применение этих приведений приведет к противоречию. Аналогично можно показать, что $C(S_0 S_n) = 2n-2$. [*Theoretical Computer Science* **1** (1976), 289–295.]

80. (а) Без потери общности $a_0 = 0$, а цепочка является нормальной. Определим U_l и u_l так, как это было сделано в ответе к упр. 77. Исходя из соображений симметрии мы можем считать, что $u_1 = \max(u_1, \ldots, u_n)$.

Мы должны иметь $u_1 \ge 2$. Если $u_1 = 1$, мы можем далее считать, что $x_{n+1} = x_1 \circ x_2$; следовательно, две из трех функций $S_\alpha(0, 0, x_3, \ldots, x_n) = S_{\alpha''}$, $S_\alpha(0, 1, x_3, \ldots, x_n) = S_{'\alpha'}$, $S_\alpha(1, 1, x_3, \ldots, x_n) = S_{''\alpha}$ должны быть равны. Но тогда S_α должна быть функцией четности или $S_{'\alpha'}$ должно быть константой.

Следовательно, установка $x_1 = 0$ позволяет нам убрать логические элементы U_1, давая цепочку для $S_{\alpha'}$ с как минимум на 2 меньшим количеством логических элементов. Отсюда

следует, что $C(S_\alpha) \geq C(S_{\alpha'}) + 2$. Аналогично установка $x_1 = 1$ доказывает, что $C(S_\alpha) \geq C(S_{'\alpha}) + 2$.

При дальнейшем изучении ситуации возникают три случая.

Случай 1. $u_1 \geq 3$. Установка $x_1 = 0$ доказывает, что $C(S_\alpha) \geq C(S_{\alpha'}) + 3$.

Случай 2. $U_1 = \{i, j\}$ и оператор $\circ_j$ канализирующий (а именно И, НО-НЕ, НЕ-НО или ИЛИ). Установка переменной x_1 равной соответствующей константе обеспечивает значение x_j и позволяет нам устранить $U_1 \cup U_j$; заметим, что $i \notin U_j$ находится в оптимальной цепочке. Так что либо $C(S_\alpha) \geq C(S_{\alpha'}) + 3$, либо $C(S_\alpha) \geq C(S_{'\alpha}) + 3$.

Случай 3. $U_1 = \{i, j\}$ и $\circ_i = \circ_j = \oplus$. Мы можем считать, что $x_i = x_1 \oplus x_2$ и $x_j = x_1 \oplus x_k$. Если $u_j = 1$ и $x_l = x_j \oplus x_p$, можно реструктуризировать цепочку, полагая $x_j = x_k \oplus x_p$, $x_l = x_1 \oplus x_j$; следовательно, мы можем считать, что либо $u_j \neq 1$, либо $x_l = x_j \circ x_p$ для некоторого канализирующего оператора $\circ$. Если $U_2 = \{i, j'\}$, можно аналогично считать, что $x_{j'} = x_2 \oplus x_{k'}$ и что либо $u_{j'} \neq 1$, либо $x_{l'} = x_{j'} \circ' x_{p'}$ для некоторого канализирующего оператора $\circ'$. Кроме того, из соображений симметрии можно считать, что x_j не зависит от $x_{j'}$.

Если x_k не зависит от x_i, пусть $f(x_3, \ldots, x_n) = x_k$; в противном случае пусть $f(x_3, \ldots, x_n)$ является значением x_k при $x_i = 1$. Установкой $x_1 = f(x_3, \ldots, x_n)$ и $x_2 = \bar{f}(x_3, \ldots, x_n)$ (или наоборот) мы делаем x_i и x_j константами и получаем цепочку для неконстантной функции $S_{'\alpha'}$. В действительности мы можем обеспечить, чтобы x_l было константой в случае $u_j = 1$. Мы утверждаем, что как минимум пять логических элементов этой цепочки (включая x_i и x_j) могут быть удалены; следовательно, $C(S_\alpha) \geq C(S_{'\alpha'}) + 5$. Это утверждение, безусловно, истинно, если $|U_i \cup U_j| \geq 3$.

Мы должны иметь $|U_i \cup U_j| > 1$. В противном случае мы бы имели $p = i$, и x_k не зависело бы от x_i, так что S_α было бы независимо от x_1 при нашем выборе x_2. Следовательно, $|U_i \cup U_j| = 2$.

Случай 3, а. $U_j = \{l\}$. Тогда x_l является константой; мы можем убрать x_i, x_j и $U_i \cup U_j \cup U_l$. Если последнее множество содержит только два элемента, то $x_q = x_i \circ x_l$ также является константой и мы удаляем U_q. Поскольку $S_{'\alpha'}$ константой не является, мы не можем убрать выходной логический элемент.

Случай 3, б. $U_i \subseteq U_j$, $|U_j| = 2$. Тогда $x_q = x_i \circ x_j$ для некоторого q; мы можем убрать x_i, x_j и $U_j \cup U_q$. Наше утверждение доказано.

(б) По индукции $C(S_k) \geq 2n + \min(k, n-k) - 3 - [n = 2k]$ для $0 < k < n$; $C(S_{\geq k}) \geq 2n + \min(k, n+1-k) - 4$ для $1 < k < n$. Простыми случаями являются $C(S_0) = C(S_n) = C(S_{\geq 1}) = C(S_{\geq n}) = n - 1$; $C(S_{\geq 0}) = 0$. (Согласно рис. 9 и 10 эти границы являются оптимальными при $n = 4$ за исключением для S_1, S_3, $S_{\geq 2}$ и $S_{\geq 3}$, и оптимальными при $n = 5$ за исключением для S_1, S_4, $S_{\geq 2}$ и $S_{\geq 4}$. Все известные результаты согласуются с гипотезой о том, что $C(S_k) = C(S_{\geq k})$ для $k > n/2$.)

Источник: *Mathematical Systems Theory* **10** (1977), 323–336.

81. Если некоторая переменная используется более одного раза, мы можем положить ее равной константе, уменьшая n на 1 и уменьшая c на ≥ 2. В противном случае первая операция должна включать x_1, поскольку $y_1 = x_1$ является единственным выходом, который не требует вычислений; делая x_1 константой, мы уменьшаем n на 1, c на ≥ 1, а d на ≥ 1. [*J. Algorithms* **7** (1986), 185–201.]

82. (62) Ложно.

(63) Читается как "для всех чисел m существует число n, такое, что $m < n + 1$"; высказывание истинно, поскольку мы можем взять $m = n$.

(64) Ложно при $n = 0$ или $n = 1$, поскольку числа в формулах должны быть неотрицательными целыми числами.

(65) Гласит, что если b превосходит a не менее чем на 2, то между этими числами имеется число ab. Конечно, это высказывание истинно, поскольку мы можем положить $ab = a + 1$.

(66) Пояснялось в разделе, и оно также истинно. Заметим, что в (65) '$\wedge$' имеет больший приоритет, чем '$\vee$', а '$\equiv$' — больший приоритет, чем '$\Leftrightarrow$', так же как '$+$' имеет более высокий приоритет, чем '$\geq$', а '$<$' имеет более высокий приоритет, чем '$\wedge$'; эти соглашения снижают необходимость в скобках в предложениях L.

(67) Гласит, что если множество A содержит как минимум один элемент n, оно должно содержать минимальный элемент m (элемент, который не превышает все элементы множества). Истинно.

(68) Аналогично, но m теперь представляет собой максимальный элемент. Также истинно, поскольку все множества считаются конечными.

(69) Говорит о существовании множества P, обладающего следующими свойствами: $[0 \in P] = [3 \notin P]$, $[1 \in P] = [4 \notin P]$, $\ldots$, $[999 \in P] = [1002 \notin P]$, $[1000 \in P] \neq [1003 \notin P]$, $[1001 \in P] \neq [1004 \notin P]$, и т. д. Истинно тогда (и только тогда), когда $P = \{x \mid x \bmod 6 \in \{1, 2, 3\}$ и $0 \leq x < 1000\}$.

Наконец, подформула $\forall n\,(n \in C \Leftrightarrow n + 1 \in C)$ в (70) представляет собой другой способ сказать, что $C = \emptyset$, поскольку множество C конечно. Следовательно, формула в скобках после $\forall A\,\forall B$ — это просто хитрый способ сказать, что $A = \emptyset$ и $B \neq \emptyset$. (Стокмейер (Stockmeyer) и Мейер (Meyer) использовали этот трюк для сокращения предложений L, которые включают длинные подформулы более одного раза.) Высказывание (70) истинно, поскольку пустое множество не равно непустому множеству.

83. Мы можем считать, что цепочка — нормальная. Пусть канализирующими шагами являются $y_1, \ldots, y_p$. Тогда $y_k = \alpha_k \circ \beta_k$ и $f = \alpha_{p+1}$, где α_k и β_k представляют собой $\oplus$ некоторых подмножеств $\{x_1, \ldots, x_n, y_1, \ldots, y_{k-1}\}$; для их вычисления с первоначальной комбинацией общих членов требуется не более $n + k - 2$ операторов $\oplus$. Следовательно, $C(f) \leq p + \sum_{k=1}^{p+1}(n + k - 2) = (p + 1)(n + p/2) - 1$.

84. Рассуждаем, как и в предыдущем упражнении, с $\vee$ или $\wedge$ вместо $\oplus$. [N. Alon and R. B. Boppana, *Combinatorica* **7** (1987), 15–16.]

85. (а) Простая компьютерная программа показывает, что 13 744 являются законными, а 19 024 — нет. (Незаконное семейство такого вида имеет как минимум 8 членов; одним из них является $\{\texttt{00}, \texttt{0f}, \texttt{33}, \texttt{55}, \texttt{ff}, \texttt{15}, \texttt{3f}, \texttt{77}\}$. Действительно, если функции $x_1 \vee x_2$ (`3f`), $x_2 \vee x_3$ (`77`) и $(x_1 \vee x_2) \wedge x_3$ (`15`) присутствуют в законном семействе L, то $x_2 \sqcup \texttt{15} = \texttt{33} \mid \texttt{15} = \texttt{37}$ также должно быть в L.)

(б) Проецирующие и константные функции, очевидно, присутствуют. Определим $A^* = \bigcap\{B \mid B \supseteq A$ и $B \in \mathcal{A}\}$ или $A^* = \infty$, если такое множество B не существует. Тогда мы имеем $\lceil A \rceil \sqcap \lceil B \rceil = \lceil A \cap B \rceil$ и $\lceil A \rceil \sqcup \lceil B \rceil = \lceil (A \cup B)^* \rceil$.

(в) Сократим формулы до $\hat{x}_l \subseteq x_l \vee \bigvee_{i=n+1}^{l} \delta_i$, $x_l \subseteq \hat{x}_l \vee \bigvee_{i=n+1}^{l} \epsilon_i$ и применим метод математической индукции. Если шаг l является шагом И, $\hat{x}_l = \hat{x}_j \sqcap \hat{x}_k \subseteq \hat{x}_j \wedge \hat{x}_k \subseteq (x_j \vee \bigvee_{i=n+1}^{l} \delta_i) \wedge (x_k \vee \bigvee_{i=n+1}^{l} \delta_i) = x_l \vee \bigvee_{i=n+1}^{l} \delta_i$; $x_l = x_j \wedge x_k \subseteq (\hat{x}_j \vee \bigvee_{i=n+1}^{l-1} \epsilon_i) \wedge (\hat{x}_k \vee \bigvee_{i=n+1}^{l-1} \epsilon_i) = (\hat{x}_j \wedge \hat{x}_k) \vee \bigvee_{i=n+1}^{l-1} \epsilon_i$, и $\hat{x}_j \wedge \hat{x}_k = \hat{x}_l \vee \epsilon_l$. Рассуждения в случае, когда шаг l представляет собой шаг ИЛИ, аналогичны.

86. (а) Если S является r-семейством, содержащимся в $(r + 1)$-семействе S', то ясно, что $\Delta(S) \subseteq \Delta(S')$.

(б) В соответствии с принципом голубиного гнезда $\Delta(S)$ содержит элементы u и v каждой части, когда S представляет собой r-семейство. И если $\Delta(S) = \{u, v\}$, мы, определенно, имеем u — v.

(в) Результат очевиден при $r = 1$. Имеется не более $r - 1$ ребер, содержащих любую заданную вершину u, в силу "сильности" свойства. А если u — v, ребра, *не имеющие*

общих элементов с $\{u, v\}$, являются сильно $(r-1)$-замкнутыми; так что по индукции их имеется не более $(r-2)^2$. Таким образом, всего имеется не более $1 + 2(r-2) + (r-2)^2$ ребер.

(г) Да, согласно упр. 85, (б), если $r > 1$, поскольку сильно r-замкнутые графы являются замкнутыми относительно пересечения. Все графы с ≤ 1 ребер являются сильно r-замкнутыми, когда $r > 1$, поскольку они не имеют r-семейств, содержащих различные ребра.

(д) Имеется $\binom{n}{3}$ треугольников $x_{ij} \wedge x_{ik} \wedge x_{jk}$, только $n-2$ из которых содержатся в любом члене x_{uv} в $\hat{f}$. Следовательно, минитермы для не более чем $(r-1)^2(n-2)$ треугольников содержатся в $\hat{f}$, а другие должны содержаться в объединении членов $\epsilon_i = \hat{x}_i \oplus (\hat{x}_{j(i)} \wedge \hat{x}_{k(i)})$ для p шагов И. Такой член имеет вид $T = (\lceil G \rceil \sqcap \lceil H \rceil) \oplus (\lceil G \rceil \wedge \lceil H \rceil) = (\lceil G \rceil \wedge \lceil H \rceil) \wedge \overline{\lceil G \cap H \rceil}$, где G и H являются сильно r-замкнутыми; мы докажем, что T содержит не более $2(r-1)^3$ треугольников.

Треугольник $x_{ij} \wedge x_{ik} \wedge x_{jk}$ в T должен включать некоторую переменную (скажем, x_{ij}) из $\lceil G \rceil$ и некоторую переменную (скажем, x_{ik}) из $\lceil H \rceil$, но не переменную из $\lceil G \cap H \rceil$. Имеется не более $(r-1)^2$ вариантов выбора для ij; а тогда имеется не более $2(r-1)$ вариантов выбора для k, поскольку H имеет не более $r-1$ ребер, соприкасающихся с i, и не более $r-1$ ребер, соприкасающихся с j.

(е) Имеется 2^{n-1} полных биграфов, получаемых раскрашиванием 1 красным, а других вершин — красным или синим. Полагаем u — v тогда и только тогда, когда u и v имеют противоположные цвета. Согласно первой формуле из упр. 85, (в), минитермы B для каждого такого графа должны содержаться в одном из членов $T = \delta_i = \hat{x}_i \oplus (\hat{x}_{j(i)} \vee x_{k(i)}) = \lceil (G \cup H)^* \rceil \wedge \overline{\lceil G \cup H \rceil}$. (Например, если $n = 4$ и вершинами $(2, 3, 4)$ являются (красный, синий, синий), то $B = \bar{x}_{12} \wedge x_{13} \wedge x_{14} \wedge x_{23} \wedge x_{24} \wedge \bar{x}_{34}$.) Минитерм B содержится в T тогда и только тогда, когда в раскраске для B некоторое ребро из $(G \cup H)^*$ имеет вершины противоположных цветов, но все ребра $G \cup H$ монохроматичны. Мы докажем, что T включает не более $2^{n-r} r^2$ таких B.

Пусть G — произвольный граф, а $T = \lceil G^* \rceil \wedge \overline{\lceil G \rceil}$. Для поиска G^* может использоваться следующий (неэффективный) алгоритм. Если имеется r-семейство S с $|\Delta(S)| < 2$, завершить работу алгоритма с $G^* = \infty$. В противном случае, если $\Delta(S) = \{u, v\}$ и $u \not\text{—} v$, добавить ребро u — v к G и повторить алгоритм.

Не более 2^{n-r} двудольных минитермов B имеют монохроматичные $\{u_j, v_j\}$ для $1 \le j \le r$ при $|\Delta(S)| < 2$. А когда $\Delta(S) = \{u, v\}$, имеется 2^{n-r-1} двудольных минитермов с монохроматичными $\{u_j, v_j\}$ и бихроматичными $\{u, v\}$. Так что мы хотим показать, что алгоритм для G выполняет менее $2r^2$ итераций, когда G сильно r-замкнутый.

Пусть для $k \ge 1$ u_k — v_k представляет собой первое новое ребро, добавленное к G, которое не имеет общих элементов с $\{u_j, v_j\}$ для $1 \le j < k$. Таких ребер имеется не более r согласно свойству "сильности"; и за каждым из них следует не более $2r-3$ новых ребер, которые соприкасаются с u_j или v_j. Так что общее количество шагов для поиска G^* не превышает $r(2r-2) + 1 < 2r^2$.

(ж) Упр. 84 говорит нам, что $q < \binom{p}{2} + (p+1)\binom{n}{2}$. Таким образом, мы имеем либо $2(r-1)^3 p \ge \binom{n}{3} - (r-1)^2(n-2)$, либо $\binom{p}{2} + (p+1)\binom{n}{2} > 2^{r-1}/r^2$. Обе нижние границы для p представляют собой

$$\frac{1}{12}\left(\frac{n}{6\lg n}\right)^3\left(1 + O\left(\frac{\log\log n}{\log n}\right)\right) \qquad \text{при} \qquad r = \left\lceil \lg\left(\frac{n^6}{186624(\lg n)^4}\right)\right\rceil.$$

[В работе Noga Alon and Ravi B. Boppana, *Combinatorica* **7** (1987), 1–22, доказана таким способом, помимо прочего, нижняя граница $\Omega(n/\log n)^s$ для количества $\wedge$ в любой монотонной цепочке, которая решает, имеет ли граф G клику фиксированного размера $s \ge 3$.]

87. Элементы в X^3, где X представляет собой матрицу из нулей и единиц, не превышают n^2. Булева цепочка с $O(n^{\lg 7}(\log n)^2)$ логическими элементами может реализовать алгоритм умножения матриц Штрассена (Strassen) 4.6.4–(36) над целыми числами по модулю $2^{\lfloor \lg n^2 \rfloor + 1}$.

88. Всего имеется 1 422 564 таких функций в 716 классах по отношению к перестановке переменных. Алгоритм L и другие методы данного раздела легко расширяются на тернарные операции, и мы получаем следующие результаты для оптимальных медианных вычислений:

$C_c(f)$	Классы	Функции	$U_c(f)$	Классы	Функции	$L_c(f)$	Классы	Функции	$D_c(f)$	Классы	Функции
0	1	7	0	1	7	0	1	7	0	1	7
1	1	35	1	1	35	1	1	35	1	1	35
2	2	350	2	2	350	2	2	350	2	13	5 670
3	9	3 885	3	9	3 885	3	8	3 745	3	700	1 416 822
4	48	42 483	4	48	42 483	4	38	35 203	4	1	30
5	201	406 945	5	188	391 384	5	139	270 830	5	0	0
6	353	798 686	6	253	622 909	6	313	699 377	6	0	0
7	99	169 891	7	69	134 337	7	176	367 542	7	0	0
8	2	282	8	2	2 520	8	34	43 135	8	0	0
9	0	0	9	0	0	9	3	2 310	9	0	0
10	0	0	10	0	0	10	0	0	10	0	0
11	0	0	∞	143	224 654	11	1	30	11	0	0

В работе S. Amarel, G. E. Cooke and R. O. Winder [*IEEE Trans.* **EC-13** (1964), 4–13, Fig. 5b] выдвинута гипотеза о том, что формула с девятью операциями

$$\langle x_1x_2x_3x_4x_5x_6x_7\rangle = \langle x_1\langle\langle x_2x_3x_5\rangle\langle x_2x_4x_6\rangle\langle x_3x_4x_7\rangle\rangle\langle\langle x_2x_5x_6\rangle\langle x_3x_5x_7\rangle\langle x_4x_6x_7\rangle\rangle\rangle$$

представляет собой наилучший способ вычисления медианы семи значений через медианы трех значений. Но "волшебная" формула

$$\langle x_1\langle x_2\langle x_3x_4x_5\rangle\langle x_3x_6x_7\rangle\rangle\langle x_4\langle x_2x_6x_7\rangle\langle x_3x_5\langle x_5x_6x_7\rangle\rangle\rangle\rangle$$

требует только восьми операций; а в действительности кратчайшая цепочка использует только семь операций:

$$\langle x_1x_2x_3x_4x_5x_6x_7\rangle = \langle x_1\langle x_2\langle x_5x_6x_7\rangle\langle x_3\langle x_5x_6x_7\rangle x_4\rangle\rangle\langle x_5\langle x_2x_3x_4\rangle\langle x_6\langle x_2x_3x_4\rangle x_7\rangle\rangle\rangle.$$

Интересная функция $f(x_1,\ldots,x_7) = (x_1\wedge x_2\wedge x_4)\vee(x_2\wedge x_3\wedge x_5)\vee(x_3\wedge x_4\wedge x_6)\vee(x_4\wedge x_5\wedge x_7)\vee(x_5\wedge x_6\wedge x_1)\vee(x_6\wedge x_7\wedge x_2)\vee(x_7\wedge x_1\wedge x_3)$, простые импликанты которой соответствуют проективной плоскости с семью точками, оказывается самой трудной: ее минимальная длина $L(f) = 11$ и минимальная глубина $D(f) = 4$ достигаются замечательной формулой

$$\langle\langle x_1x_4\langle x_4x_5x_6\rangle\rangle\langle x_3x_6\langle x_1\langle x_2x_3x_7\rangle\langle x_2x_5x_6\rangle\rangle\rangle\langle x_2x_7\langle x_1\langle x_5x_2x_4\rangle\langle x_5x_3x_7\rangle\rangle\rangle\rangle.$$

А следующая еще более удивительная цепочка вычисляет ее оптимальным образом:

$$x_8 = \langle x_1x_2x_3\rangle,\quad x_9 = \langle x_1x_4x_6\rangle,\quad x_{10} = \langle x_1x_5x_8\rangle,\quad x_{11} = \langle x_2x_7x_8\rangle,$$
$$x_{12} = \langle x_3x_9x_{10}\rangle,\quad x_{13} = \langle x_4x_5x_{12}\rangle,\quad x_{14} = \langle x_6x_{11}x_{12}\rangle,\quad x_{15} = \langle x_7x_{13}x_{14}\rangle.$$

РАЗДЕЛ 7.1.3

1. Эти операции обменивают биты x и y в позициях, где биты m равны 1. (В частности, если $m = -1$, шаг '$y \leftarrow y \oplus (x \mathbin{\&} m)$' превращается просто в '$y \leftarrow y \oplus x$', и эти три присваивания выполняют обмен $x \leftrightarrow y$ без использования дополнительного регистра. Г. С. Уоррен-мл. (H. S. Warren, Jr.) обнаружил этот трюк в примечаниях к курсу IBM по программированию, датированному 1961 годом.)

2. Все три соотношения выполняются для неотрицательных x и y, или если мы рассматриваем x и y как "беззнаковые 2-адические целые числа", для которых $0 < 1 < 2 < \cdots < -3 < -2 < -1$. Но если отрицательные целые числа меньше неотрицательных, (i) не выполняется тогда и только тогда, когда $x < 0$ и $y < 0$; (ii) и (iii) не выполняются тогда и только тогда, когда $x \oplus y < 0$, а именно тогда и только тогда, когда $x < 0$ и $y \geq 0$ или $x \geq 0$ и $y < 0$.

3. Заметим, что $x - y = (x \oplus y) - 2(\bar{x} \mathbin{\&} y)$ (см. упр. 93). Удаляя общие для x и y биты в левой части, можем считать, что $x_{n-1} = 1$ и $y_{n-1} = 0$. Тогда $2(\bar{x} \mathbin{\&} y) \leq 2((x \oplus y) - 2^{n-1}) = (x \oplus y) - (x \oplus y)^M - 1$.

4. $x^{CN} = x + 1 = x^S$ согласно (16). Следовательно, $x^{NC} = x^{NCSP} = x^{NCCNP} = x^{NNP} = x^P$.

5. (а) Опровержение: пусть $x = (\ldots x_2 x_1 x_0)_2$. Тогда бит l числа $x \ll k$ равен $x_{l-k}[l \geq k]$. Так что бит l в левой части равен $x_{l-k-j}[l \geq k][l - k \geq j]$, в то время как бит l в правой части равен $x_{l-j-k}[l \geq j + k]$. Эти выражения совпадают, когда $j \geq 0$ или $k \leq 0$. Но если $j < 0 < k$, они отличаются при $l = \max(0, j + k)$ и $x_{l-j-k} = 1$.

(Однако во всех случаях мы имеем $(x \ll j) \ll k \subseteq x \ll (j + k)$.)

(б) Доказательство: бит l во всех трех формулах равен $x_{l+j}[l \geq -j] \wedge y_{l-k}[l \geq k]$.

6. Поскольку $x \ll y \geq 0$ тогда и только тогда, когда $x \geq 0$, мы должны иметь $x \geq 0$ тогда и только тогда, когда $y \geq 0$. Очевидно, что $x = y$ всегда является решением. Решениями с $x > y$ являются (а) $x = -1$ и $y = -2$, или $2^y > x > y > 0$; (б) $x = 2$ и $y = 1$, или $2^{-x} \geq -y > -x > 0$.

7. Установите $x' \leftarrow (x + \bar{\mu}_0) \oplus \bar{\mu}_0$, где μ_0 — константа из (47). Тогда $x' = (\ldots x'_2 x'_1 x'_0)_2$, поскольку $(x' \oplus \bar{\mu}_0) - \bar{\mu}_0 = (\ldots \bar{x}'_3 x'_2 \bar{x}'_1 x'_0)_2 - (\ldots 1010)_2 = (\ldots 0 x'_2 0 x'_0)_2 - (\ldots x'_3 0 x'_1 0)_2 = x$.

[Это хак 128 из НАКМЕМ; см. ответ к упр. 20. Д. П. Агравалем (D. P. Agrawal), *IEEE Trans.* **C-29** (1980), 1032–1035, была также предложена альтернативная формула $x' \leftarrow (\mu_0 - x) \oplus \mu_0$. Результаты корректны по модулю 2^n для всех n, но может произойти переполнение или потеря значимости. Например, бинарные числа в дополнительном коде в n-битовом регистре находятся в диапазоне от -2^{n-1} до $2^{n-1} - 1$ включительно, а негабинарные — от $-\frac{2}{3}(2^n - 1)$ до $\frac{1}{3}(2^n - 1)$ при четном n. В общем случае формула $x' \leftarrow (x + \mu) \oplus \mu$ выполняет преобразование из бинарной записи в обобщенную числовую систему с бинарным базисом $\langle 2^n(-1)^{m_n} \rangle$, которая рассматривалась в упр. 4.1–30(с), при $\mu = (\ldots m_2 m_1 m_0)_2$.]

8. Во-первых, $x \oplus y \notin (S \oplus y) \cup (x \oplus T)$. Во-вторых, предположим, что $0 \leq k < x \oplus y$, и пусть $x \oplus y = (\alpha 1 \alpha')_2$, $k = (\alpha 0 \alpha'')_2$, где α, α' и α'' являются строками из нулей и единиц, обладающие тем свойством, что $|\alpha'| = |\alpha''|$. Исходя из симметрии будем считать, что $x = (\beta 1 \beta')_2$ и $y = (\gamma 0 \gamma')_2$, где $|\alpha'| = |\beta'| = |\gamma'|$. Тогда $k \oplus y = (\beta 0 \gamma'')_2$ меньше, чем x. Следовательно, $k \oplus y \in S$ и $k = (k \oplus y) \oplus y \in S \oplus y$. [См. R. P. Sprague, *Tôhoku Math. J.* **41** (1936), 438–444; P. M. Grundy, *Eureka* **2** (1939), 6–8.]

9. Теорема Спрэга–Гранди из предыдущего упражнения показывает, что две кучки с x и y камнями эквивалентны одной кучке с $x \oplus y$ камнями. (Неотрицательное целое число $k < x \oplus y$ существует тогда и только тогда, когда существует либо неотрицательное $i < x$ с $i \oplus y < x \oplus y$, либо неотрицательное $j < y$, с $x \oplus j < x \oplus y$.) Так что k кучек эквивалентны одной кучке размером $a_1 \oplus \cdots \oplus a_k$. [См. C. L. Bouton, *Annals of Math.* (2) **3** (1901–1902), 35–39.]

10. Для ясности и краткости будем писать просто xy вместо $x \otimes y$ и $x + y$ вместо $x \oplus y$, но *только в частях от (i) до (iv) данного ответа.*

(i) Ясно, что $0y = 0$, $x + y = y + x$ и $xy = yx$. По индукции по y, кроме того, выполняется $1y = y$.

(ii) Если $x \neq x'$ и $y \neq y'$, то $xy + xy' + x'y + x'y' \neq 0$, поскольку определение xy гласит, что $xy' + x'y + x'y' \neq xy$ при $0 \leq x' < x$ и $0 \leq y' < y$. В частности, если $x \neq 0$ и $y \neq 0$,

то $xy \neq 0$. Другое следствие заключается в том, что, если $x = \operatorname{mex}(S)$ и $y = \operatorname{mex}(T)$ для произвольных конечных множеств S и T, то мы имеем $xy = \operatorname{mex}\{xj + iy + ij \mid i \in S, j \in T\}$.

(iii) Следовательно, по индукции по (обычной) сумме x, y и z, $(x+y)z$ равно

$$\operatorname{mex}\{(x+y)z' + (x'+y)z + (x'+y)z', (x+y)z' + (x+y')z + (x+y')z' \\ \mid\ 0 \le x' < x,\ 0 \le y' < y,\ 0 \le z' < z\},$$

что представляет собой $\operatorname{mex}\{xz'+x'z+x'z'+yz, xz+yz'+y'z+y'z'\} = xz+yz$. В частности, действует закон сокращения: если $xz = yz$, то $(x+y)z = 0$, так что $x = y$ или $z = 0$.

(iv) По аналогичной индукции $(xy)z = \operatorname{mex}\{(xy)z' + (xy' + x'y + x'y')(z+z')\} = \operatorname{mex}\{(xy)z' + (xy')z + (xy')z' + \cdots\} = \operatorname{mex}\{x(yz') + x(y'z) + x(y'z') + \cdots\} = \operatorname{mex}\{(x + x')(yz' + y'z + y'z') + x'(yz)\} = x(yz)$.

(v) Докажем, что если $0 \le x, y < 2^{2^n}$, то $x \otimes y < 2^{2^n}$, $2^{2^n} \otimes y = 2^{2^n} y$ и $2^{2^n} \otimes 2^{2^n} = \frac{3}{2} 2^{2^n}$. Согласно закону дистрибутивности (iii) достаточно рассмотреть случай $x = 2^a$ и $y = 2^b$ для $0 \le a, b < 2^n$. Пусть $a = 2^p + a'$ и $b = 2^q + b'$, где $0 \le a' < 2^p$ и $0 \le b' < 2^q$; тогда $x = 2^{2^p} \otimes 2^{a'}$ и $y = 2^{2^q} \otimes 2^{b'}$ по индукции по n.

Если $p < n-1$ и $q < n-1$, мы уже доказали, что $x \otimes y < 2^{2^{n-1}}$. Если $p < q = n-1$, то $x \otimes 2^{b'} < 2^{2^q}$, следовательно, $x \otimes y < 2^{2^n}$. А если $p = q = n-1$, мы имеем $x \otimes y = 2^{2^p} \otimes 2^{2^p} \otimes 2^{a'} \otimes 2^{b'} = (\frac{3}{2} 2^{2^p}) \otimes z$, где $z < 2^{2^p}$. Таким образом, $x \otimes y < 2^{2^n}$ во всех случаях.

Согласно закону сокращения неотрицательные целые числа, меньшие, чем 2^{2^n}, образуют подполе. Следовательно, в формуле

$$2^{2^n} \otimes y \;=\; \operatorname{mex}\{2^{2^n} y' \oplus x'(y \oplus y') \mid 0 \le x' < 2^{2^n}, 0 \le y' < y\}$$

мы можем выбрать x' для каждого y', чтобы исключить все числа между $2^{2^n} y'$ и $2^{2^n}(y' + 1) - 1$; но $2^{2^n} y$ никогда не исключается.

Наконец в $2^{2^n} \otimes 2^{2^n} = \operatorname{mex}\{2^{2^n}(x' \oplus y') \oplus (x' \otimes y') \mid 0 \le x', y' < 2^{2^n}\}$ выбор $x' = y'$ исключит все числа до $2^{2^n} - 1$ включительно, поскольку из $x \otimes x = y \otimes y$ вытекает, что $(x \oplus y) \otimes (x \oplus y) = 0$, а следовательно, $x = y$. Выбор $x' = y' \oplus 1$ исключает числа от 2^{2^n} до $\frac{3}{2} 2^{2^n} - 1$, поскольку из $(x \otimes x) \oplus x = (y \otimes y) \oplus y$ вытекает, что $x = y$ или $x = y \oplus 1$, и поскольку старший бит $x \otimes x$ тот же, что и старший бит x. Это же наблюдение показывает, что $\frac{3}{2} 2^{2^n}$ *не* исключается. QED.

Рассмотрим, например, подполе $\{0, 1, \ldots, 15\}$. Согласно закону дистрибутивности мы можем свести $x \otimes y$ к сумме $x \otimes 1$, $x \otimes 2$, $x \otimes 4$, и/или $x \otimes 8$. Мы имеем $2 \otimes 2 = 3$, $2 \otimes 4 = 8$, $4 \otimes 4 = 6$; а умножение на 8 может быть выполнено путем умножения сначала на 2, а потом на 4, или наоборот, поскольку $8 = 2 \otimes 4$. Таким образом, $2 \otimes 8 = 12$, $4 \otimes 8 = 11$, $8 \otimes 8 = 13$.

В общем случае представим $n > 0$ в виде $n = 2^m + r$, где $0 \le r < 2^m$. Существует матрица Q_n размером $2^{m+1} \times 2^{m+1}$, такая, что умножение на 2^n эквивалентно применению Q_n к блокам из 2^{m+1} бит при работе по модулю 2. Например, $Q_1 = \left(\begin{smallmatrix} 1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$, и $(\ldots x_4 x_3 x_2 x_1 x_0)_2 \otimes 2^1 = (\ldots y_4 y_3 y_2 y_1 y_0)_2$, где $y_0 = x_1$, $y_1 = x_1 \oplus x_0$, $y_2 = x_3$, $y_3 = x_3 \oplus x_2$, $y_4 = x_5$ и т. д. Матрицы образуются рекурсивно следующим образом: пусть $Q_0 = R_0 = (1)$ и

$$Q_{2^m+r} = \begin{pmatrix} I & R_m \\ I & 0 \end{pmatrix} \begin{pmatrix} Q_r & & 0 \\ & \ddots & \\ 0 & & Q_r \end{pmatrix}, \qquad R_{m+1} = \begin{pmatrix} R_m & R_m^2 \\ R_m & 0 \end{pmatrix} = Q_{2^{m+1}-1},$$

где Q_r реплицирована достаточное количество раз, чтобы образовать 2^{m+1} строк и столбцов. Например,

$$Q_2 = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}; \qquad Q_3 = Q_2 \begin{pmatrix} Q_1 & 0 \\ 0 & Q_1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = R_2.$$

Если в регистре x хранится любое 64-битовое число и если $0 \le j \le 7$, то MMIX-команда MXOR y,q_j,x будет вычислять $y = x \otimes 2^j$ для данных шестнадцатеричных матричных констант

$$\begin{aligned} q_0 &= \texttt{8040201008040201}, & q_3 &= \texttt{d0b0c0800d0b0c08}, & q_6 &= \texttt{b9678d4bb0608040}, \\ q_1 &= \texttt{c08030200c080302}, & q_4 &= \texttt{8d4b2c1880402010}, & q_7 &= \texttt{deb9c68dd0b0c080}. \\ q_2 &= \texttt{b06080400b060804}, & q_5 &= \texttt{c68d342cc0803020}, \end{aligned}$$

[Д. Х. Конвей (J. H. Conway), *On Numbers and Games* (1976), глава 6, показал, что эти определения действительно дают алгебраически замкнутое поле над порядковыми числами.]

11. Пусть $m = 2^{a_s} + \cdots + 2^{a_1}$ с $a_s > \cdots > a_1 \ge 0$ и $n = 2^{b_t} + \cdots + 2^{b_1}$ с $b_t > \cdots > b_1 \ge 0$. Тогда $m \otimes n = mn$ тогда и только тогда, когда $(a_s \mid \cdots \mid a_1)\ \&\ (b_t \mid \cdots \mid b_1) = 0$.

12. Если $x = 2^{2^n}a + b$, где $0 \le a, b < 2^{2^n}$, положим $x' = x \otimes (x \oplus a)$. Тогда

$$x' = ((2^{2^n} \otimes a) \oplus b) \otimes ((2^{2^n} \otimes a) \oplus a \oplus b) \;=\; (2^{2^n-1} \otimes a \otimes a) \oplus (b \otimes (a \oplus b)) \;<\; 2^{2^n}.$$

Для ним-деления на x можно, таким образом, ним-поделить на x' и умножить на $x \oplus a$. [Этот алгоритм принадлежит Х. В. Ленстре-мл. (H. W. Lenstra, Jr.); см. *Séminaire de Théorie des Nombres* (Université de Bordeaux, 1977–1978), exposé 11, exercise 5.]

13. Если $a_2 \oplus \cdots \oplus a_k = a_1 \oplus a_3 \oplus \cdots \oplus ((k-2) \otimes a_k) = 0$, каждый ход нарушает это условие; мы не можем иметь $(a \otimes x) \oplus (b \otimes y) = (a \otimes x') \oplus (b \otimes y')$ при $x \oplus y = x' \oplus y'$ и $a \ne b$, если только не выполняется условие $(x, y) = (x', y')$.

И обратно, если $a_2 \oplus \cdots \oplus a_k \ne 0$, можно уменьшить некоторое a_j с $j \ge 2$, чтобы сделать эту сумму нулевой; тогда a_1 можно установить равным $a_3 \oplus \cdots \oplus ((k-2) \otimes a_k)$. Если $a_2 \oplus \cdots \oplus a_k = 0$ и $a_1 \ne a_3 \oplus \cdots \oplus ((k-2) \otimes a_k)$, мы просто уменьшаем a_1, если оно слишком велико. В противном случае имеется $j \ge 3$, такое, что равенство будет выполняться, если $(j-2) \otimes a_j$ заменить подходящим меньшим значением $((j-2) \otimes a'_j) \oplus ((i-2) \otimes (a_j \oplus a'_j))$ для некоторого $2 \le i < j$ и $0 \le a'_j < a_j$ из-за определения ним-умножения; следовательно, оба искомых равенства достигаются путем установки $a_j \leftarrow a'_j$ и $a_i \leftarrow a_i \oplus a_j \oplus a'_j$. [Эта игра была представлена в *Winning Ways* Берлекампом (Berlekamp), Конвеем (Conway) и Гаем (Guy) в конце главы 14.]

14. (а) Каждое $y = (\ldots y_2 y_1 y_0)_2 = x^T$ единственным образом определяет $x = (\ldots x_2 x_1 x_0)_2$, поскольку $x_0 = y_0 \oplus t$ и $\lfloor y/2 \rfloor = \lfloor x/2 \rfloor^{T x_0}$.

(б) Когда $k > 0$, это выражение представляет собой функцию ветвления с метками $t_{\alpha a \beta} = a$ для $|\beta| = k-1$ и $t_\alpha = 0$ для $|\alpha| < k$. Но когда $k \le 0$, отображение не является перестановкой; в действительности, когда $k < 0$, оно отображает 2^{-k} различных 2-адических целых чисел в 0.

[Случай $k = 1$ представляет особенный интерес: тогда x^T отображает неотрицательные целые числа в неотрицательные целые числа с четной четностью, отрицательные целые числа в неотрицательные целые числа с нечетной четностью и $-1/3 \mapsto -1$. Кроме того, $\lfloor x^T/2 \rfloor$ представляет собой "бинарный код Грея" 7.2.1.1–(9).]

(в) Если $\rho(x \oplus y) = k$, мы имеем $T(x) \equiv T(y)$ и $x \equiv y + 2^k$ (по модулю 2^{k+1}). Следовательно, $\rho(x^T \oplus y^T) = \rho(x \oplus y \oplus T(x) \oplus T(y)) = k$. И обратно, если $\rho(x^T \oplus y^T) = k$ при $y = x + 2^k$, мы получаем подходящую маркировку битами, полагая $t_\alpha = (x^T \gg |\alpha|) \bmod 2$ при $x = (\alpha^R)_2$.

(г) Это утверждение следует непосредственно из (а) и (в). Если мы всегда имеем $\rho(x \oplus y) = \rho(x^U \oplus y^U) = \rho(x^V \oplus y^V)$, то $\rho(x \oplus y) = \rho(x^U \oplus y^U) = \rho(x^{UV} \oplus y^{UV})$. А если $x^{TU} = x$ для всех x, $\rho(x^U \oplus y^U) = \rho(x \oplus y)$ эквивалентно $\rho(x \oplus y) = \rho(x^T \oplus y^T)$.

Мы можем также построить маркировку явно: если $W = UV$, заметим, что при $a, b, c \in \{0, 1\}$ мы имеем $W_a = U_a V_{a'}$, $W_{ab} = U_{ab} V_{a'b'}$ и $W_{abc} = U_{abc} V_{a'b'c'}$, где $a' = a \oplus u$, $b' = b \oplus u_a$, $c' = c \oplus u_{ab}$ и т. д.; следовательно, $w = u \oplus v$, $w_a = u_a \oplus v_{a'}$, $w_{ab} = u_{ab} \oplus v_{a'b'}$ и т. д.

Маркировка T, обратная U, получается путем обмена левого и правого поддеревьев всех узлов с меткой 1; таким образом, $t = u$, $t_{a'} = u_a$, $t_{a'b'} = u_{ab}$ и т. д.

(д) Явное построение в (г) демонстрирует, что условие сбалансированности при композициях и инверсиях сохраняется, поскольку на каждом уровне $\{0', 1'\} = \{0, 1\}$.

Примечания. Хендрик Ленстра (Hendrik Lenstra) заметил, что функции ветвления могут с пользой рассматриваться как *изометрии* (перестановки, сохраняющие расстояния) 2-адических целых чисел, при использовании для определения "расстояния" между 2-адическими целыми числами x и y формулы $1/2^{\rho(x\oplus y)}$. Кроме того, функции ветвления по модулю 2^d оказываются 2-подгруппами Сылова группы всех перестановок $\{0, 1, \ldots, 2^d - 1\}$, а именно уникальной (до изоморфизма) подгруппой, которая имеет максимальный порядок степени двойки среди всех подгрупп данной группы. Они также эквивалентны автоморфизмам полного бинарного дерева с 2^d листьями.

15. Эквивалентно $(x + 2a) \oplus b = (x \oplus b) + 2a$; так что мы можем найти также все b и c, такие, что $(x \oplus b) + c = (x + c) \oplus b$. Из установок $x = 0$ и $x = -c$ вытекает, что $b + c = b \oplus c$ и $b - c = b \oplus (-c)$; следовательно, $b \,\&\, c = b \,\&\, (-c) = 0$ согласно (89), и мы имеем $b < 2^{\rho c}$. Это условие является также достаточным. Таким образом, $0 \le b < 2^{\rho a+1}$ — необходимое и достаточное условие для решения исходной задачи.

16. (а) Если $\rho(x \oplus y) = k$, мы имеем $x \equiv y + 2^k$ (по модулю 2^{k+1}); следовательно, $x + a \equiv y + a + 2^k$ и $\rho((x + a) \oplus (y + a)) = k$. А $\rho((x \oplus b) \oplus (y \oplus b))$ очевидно равно k.

(б) Маркировка из указания, назовем ее $P(c)$, имеет единицы на пути, соответствующем c, и нули в остальных местах; таким образом, она является сбалансированной. Обобщенная анимирующая функция может быть записана как

$$x^{P(c_0)^{-a_1} P(c_1)^{-a_2} \ldots P(c_{m-1})^{-a_m}} \oplus c_m, \qquad \text{где } c_j = b_1 \oplus \cdots \oplus b_j;$$

так что она сбалансирована тогда и только тогда, когда $c_m = 0$.

[Кстати, множество $S = \{P(0)\} \cup \{P(k) \oplus P(k + 2^e) \mid k \ge 0 \text{ и } 2^e > k\}$ предоставляет интересный *базис* для всех возможных сбалансированных маркировок: маркировка сбалансирована тогда и только тогда, когда она представляет собой $\bigoplus\{q \mid q \in Q\}$ для некоторого $Q \subseteq S$. Эта операция исключающего или хорошо определена, несмотря на то, что Q может быть бесконечным, поскольку в каждом узле может присутствовать только конечное количество единиц.]

(в) Функция $P(c)$ в (б) имеет указанный вид, поскольку $x^{P(c)} = x \oplus \lfloor x \oplus c \rceil$. Обратная к ней $x^{S(c)} = ((x \oplus c) + 1) \oplus c$ представляет собой $x \oplus \lfloor x \oplus \bar{c} \rceil = x^{P(\bar{c})}$. Кроме того, мы имеем $x^{P(c)P(d)} = x^{P(c)} \oplus \lfloor x^{P(c)} \oplus d \rceil = x \oplus \lfloor x \oplus c \rceil \oplus \lfloor x \oplus d^{S(c)} \rceil$, поскольку $\lfloor x \oplus y \rceil = \lfloor x^T \oplus y^T \rceil$ для любой функции ветвления x^T. Аналогично $x^{P(c)P(d)P(e)} = x \oplus \lfloor x \oplus c \rceil \oplus \lfloor x \oplus d^{S(c)} \rceil \oplus \lfloor x \oplus e^{S(d)S(c)} \rceil$ и т. д. После отбрасывания равных членов мы получаем требуемый вид. Получающиеся в результате числа p_j уникальны, поскольку они являются единственными значениями x, в которых функция меняет знак.

(г) Мы имеем, например, $x \oplus \lfloor x \oplus a \rceil \oplus \lfloor x \oplus b \rceil \oplus \lfloor x \oplus c \rceil = x^{P(a')P(b')P(c')}$, где $a' = a$, $b' = b^{P(a')}$ и $c' = c^{P(a')P(b')}$.

[Теория анимирующих функций была разработана Д. Х. Конвеем (J. H. Conway) в главе 13 его книги *On Numbers and Games* (1976) и основывалась на предшествующей работе К. П. Велтера (C. P. Welter) в *Indagationes Math.* **14** (1952), 304–314; **16** (1954), 194–200.]

17. (Решение М. Сланины (M. Slanina).) Такие уравнения разрешимы, даже если мы допустим такие операции, как $x \& y$, $\bar{x}$, $x \ll 1$, $x \gg 1$, $2^{\rho x}$ и $2^{\lambda x}$, и даже если мы разрешим булевы комбинации утверждений и квантификацию над целочисленными переменными путем их трансляции в формулы монадической логики второго порядка с одним преемником (S1S). Каждая 2-адическая переменная $x = (\ldots x_2 x_1 x_0)_2$ соответствует переменной множества

S1S X, где $j \in X$ означает $x_j = 1$:

$$\begin{array}{lll} z = \bar{x} & \text{становится} & \forall t(t \in Z \Leftrightarrow t \notin X); \\ z = x \,\&\, y & \text{становится} & \forall t(t \in Z \Leftrightarrow (t \in X \wedge t \in Y)); \\ z = 2^{\rho x} & \text{становится} & \forall t(t \in Z \Leftrightarrow (t \in X \wedge \forall s(s < t \Rightarrow s \notin X))); \\ z = x + y & \text{становится} & \exists C \forall t(0 \notin C \ \wedge\ (t \in Z \Leftrightarrow (t{\in}X) \oplus (t{\in}Y) \oplus (t{\in}C)) \\ & & \qquad \wedge\ (t{+}1 \in C \Leftrightarrow \langle (t{\in}X)(t{\in}Y)(t{\in}C) \rangle)). \end{array}$$

Тождество, такое как $x \,\&\, (-x) = 2^{\rho x}$, эквивалентно трансляции

$$\forall X \forall Y \forall Z((\text{integer}(X) \ \wedge\ 0 = x + y \ \wedge\ z = x \,\&\, y) \ \Rightarrow\ z = 2^{\rho x}),$$

где $\text{integer}(X)$ означает $\exists t \forall s(s > t \Rightarrow (s \in X \Leftrightarrow t \in X))$. Можно также включить 2-адические константы, если они представляют собой, скажем, отношения целых чисел; например, $z = \mu_0$ эквивалентно формуле $0 \in Z \wedge \forall t(t \in Z \Leftrightarrow t + 1 \notin Z)$. Но, конечно, мы не можем включать произвольные (невычислимые) константы.

Ю. Р. Бюхи (J. R. Büchi) доказал, что все формулы S1S являются разрешимыми, в *Logic, Methodology, and Philosophy of Science: Proceedings* (Stanford, 1960), 1–11. Если мы ограничимся равенствами, то можно показать, что в действительности экспоненциального времени будет достаточно.

С другой стороны, M. Гамбург (M. Hamburg) показал, что задача может быть неразрешимой, если добавить в набор операций ρx, λx или $1 \ll x$; тогда может быть закодировано умножение.

Кстати, имеется много нетривиальных тождеств, даже если мы используем только лишь операции $x \oplus y$ и $x + 1$. Например, К. П. Велтер (C. P. Welter) заметил в 1952 году, что

$$((x \oplus (y + 1)) + 1) \oplus (x + 1) \ = \ ((((x + 1) \oplus y) + 1) \oplus x) + 1.$$

18. Конечно, строка x полностью пуста, когда x кратно 64. Мелкие детали этого изображения, по всей видимости, "хаотичны" и сложны, но имеется достаточно простой способ понять, что происходит вблизи точек, где прямые линии $x = 64\sqrt{j}$ пересекаются с гиперболами $xy = 2^{11}k$, для целых не слишком больших значений $j, k \geq 1$.

Действительно, когда x и y представляет собой целые числа, значение $x^2y \gg 11$ нечетно тогда и только тогда, когда $x^2y/2^{12} \bmod 1 \geq \frac{1}{2}$. Таким образом, если $x = 64\sqrt{j} + \delta$ и $xy = 2^{11}(k + \epsilon)$, мы имеем

$$\frac{x^2y}{2^{12}} \bmod 1 = \left(\frac{128\sqrt{j}\delta + \delta^2}{4096}\right) y \bmod 1 = \left(\frac{2\delta x - \delta^2}{4096}\right) y \bmod 1 = \left((k + \epsilon)\delta - \frac{\delta^2 y}{4096}\right) \bmod 1,$$

и эта величина имеет известное отношение к $\frac{1}{2}$, когда, скажем, δ близкó к небольшому целому числу. [См. C. A. Pickover and A. Lakhtakia, *J. Recreational Math.* **21** (1989), 166–169.]

19. (a) Когда $n = 1$, $f(A, B, C)$ имеет одно и то же значение при всех расположениях, за исключением случая, когда $a_0 \neq a_1$, $b_0 \neq b_1$ и $c_0 \neq c_1$; и тогда оно не может превышать 1. Для бóльших значений n доказательство можно провести по индукции, полагая $n = 3$, чтобы избежать громоздких обозначений. Пусть $A_0 = (a_0, a_1, a_2, a_3)$, $A_1 = (a_4, a_5, a_6, a_7)$, $\ldots$, $C_1 = (c_4, c_5, c_6, c_7)$. В таком случае по индукции $f(A, B, C) = \sum_{j \oplus k \oplus l = 0} f(A_j, B_k, C_l) \leq \sum_{j \oplus k \oplus l = 0} f(A_j^*, B_k^*, C_l^*)$. Таким образом, можно считать, что $a_0 \geq a_1 \geq a_2 \geq a_3$, $a_4 \geq a_5 \geq a_6 \geq a_7$, $\ldots$, $c_4 \geq c_5 \geq c_6 \geq c_7$. Можно также отсортировать подвекторы $A_0' = (a_0, a_1, a_4, a_5)$, $A_1' = (a_2, a_3, a_6, a_7)$, $\ldots$, $C_1' = (c_2, c_3, c_6, c_7)$ более простым способом. Наконец, можно отсортировать $A_0'' = (a_0, a_1, a_6, a_7)$, $A_1'' = (a_2, a_3, a_4, a_5)$, $\ldots$, $C_1'' = (c_2, c_3, c_4, c_5)$, поскольку в каждом члене $a_j b_k c_l$ количество индексов $\{j, k, l\}$ со старшими битами 01, 10 и 11 должно удовлетворять соотношению $s_{01} \equiv s_{10} \equiv s_{11}$ (по модулю 2). А эти три операции сортировки делают A, B и C полностью отсортированными согласно

упр. 5.3.4–48. (Для всех $n \geq 2$ необходимы ровно три сортировки подвекторов длиной 2^{n-1}.)

(б) Предположим, что $A = A^*$, $B = B^*$ и $C = C^*$. Тогда мы имеем $a_j = \sum_{t=0}^{2^n-1} \alpha_t[j \leq t]$, где $\alpha_j = a_j - a_{j+1} \geq 0$, и устанавливаем $a_{2^n} = 0$; аналогичные формулы выполняются для b_k и c_l. Пусть $A_{(p)}$ обозначает вектор $(a_{p(0)}, \ldots, a_{p(2^n-1)})$, где p представляет собой перестановку $\{0, 1, \ldots, 2^n - 1\}$. Тогда согласно п. (а) мы имеем

$$\begin{aligned} f(A_{(p)}, B_{(q)}, C_{(r)}) &= \textstyle\sum_{j\oplus k\oplus l=0} \sum_{t,u,v} \alpha_t\beta_u\gamma_v[p(j) \leq t][q(k) \leq u][r(l) \leq v] \\ &\leq \textstyle\sum_{j\oplus k\oplus l=0} \sum_{t,u,v} \alpha_t\beta_u\gamma_v[j \leq t][k \leq u][l \leq v] = f(A, B, C). \end{aligned}$$

[Это доказательство принадлежит Харди (Hardy), Литтлвуду (Littlewood) и Пойа (Pólya), *Inequalities* (1934), §10.3.]

(в) Этот же метод доказательства распространяется на любое количество векторов. [R. E. A. C. Paley, *Proc. London Math. Soc.* (2) **34** (1932), 265–279, Theorem 15.]

20. Приведенные шаги вычисляют наименьшее целое y, большее x, такое, что $\nu y = \nu x$. Эта функция полезна для генерации всех сочетаний из n объектов по m (т. е. всех m-элементных подмножеств n-элементного множества, где элементы представлены единичными битами).

[Это хак 175 из HAKMEM, Massachusetts Institute of Technology Artificial Intelligence Laboratory Memo No. 239 (29 February 1972).]

21. Установить $t \leftarrow y + 1$, $u \leftarrow t \oplus y$, $v \leftarrow t \,\&\, y$, $x \leftarrow v - (v \,\&\, -v)/(u + 1)$. Если $y = 2^m - 1$ является *первым* m-сочетанием, эти восемь операций установят x равным нулю. (Тот факт, что $x = \overline{f(\bar{y})}$, похоже, не приводит к какой-либо более короткой схеме.)

22. Контрольное суммирование позволяет избежать деления: `SUBU t,x,1; ANDN u,x,t; SADD k,t,x; ADDU v,x,u; XOR t,v,x; ADDU k,k,2; SRU t,t,k; ADDU y,v,t`. Но в действительности можно сэкономить один шаг, если благоразумно воспользоваться константой `mone` $= -1$: `SUBU t,x,1; XOR u,t,x; ADDU y,x,u; SADD k,t,y; ANDN y,y,u; SLU t,mone,k; ORN y,y,t`.

23. (а) $(0\ldots01\ldots1)_2 = 2^m - 1$ и $(0101\ldots01)_2 = (2^{2m} - 1)/3$.

(б) Это решение использует 2-адическую константу $\mu_0 = (\ldots010101)_2 = -1/3$:

$$t \leftarrow x \oplus \mu_0, \quad u \leftarrow (t{-}1) \oplus t, \quad v \leftarrow x \mid u, \quad w \leftarrow v + 1, \quad y \leftarrow w + \left\lfloor \frac{v \,\&\, \overline{w}}{\sqrt{u+1}} \right\rfloor.$$

Если $x = (2^{2m} - 1)/3$, операции приводят к странному результату, поскольку $u = 2^{2m+1} - 1$.

(в) `XOR t,x,m0; SUBU u,t,1; XOR u,t,u; OR v,x,u; SADD y,u,m0; ADDU w,v,1; ANDN t,v,w; SRU y,t,y; ADDU y,w,y`. [Это упражнение основано на работах Йорга Арндта (Jörg Arndt).]

24. Имеет смысл "залить воды в насос", т. е. с самого начала инициализировать массив состоянием, которое он будет иметь после отсеивания всех кратных 3, 5, 7 и 11. Можно скомбинировать 3 с 11 и 5 с 7, как предложил Э. Вада (E. Wada).

```
      LOC Data_Segment
qbase GREG @ ;N IS 3584 ;n GREG N ;one GREG 1
Q     OCTA #816d129a64b4cb6e                    Q0 (прямой порядок).
      LOC Q+N/16
qtop  GREG @                                    Конец таблицы Q.
Init  OCTA #9249249249249249|#4008010020040080  Кратные 3 или 11 в [129..255].
      OCTA #8421084210842108|#0408102040810204  Кратные 5 или 7.
t IS $255 ;x33 IS $0 ;x35 IS $1 ;j IS $4
      LOC #100
Main  LDOU x33,Init; LDOU x35,Init+8
```

```
     LDA j,qbase,8; SUB j,j,qtop                  Подготовка к установке Q_1.
 1H  NOR t,x33,x33; ANDN t,t,x35; STOU t,qtop,j   Инициализация 64 бит решета.
     SLU t,x33,2; SRU x33,x33,31; OR x33,x33,t    Подготовка следующих 64 значений.
     SLU t,x35,6; SRU x35,x35,29; OR x35,x35,t
     ADD j,j,8; PBN j,1B                          Повторять до достижения qtop. ▌
```

Затем отбрасываются составные числа p^2, p^2+2p, ... для $p = 13, 17, \ldots$, пока $p^2 \le N$:

```
p IS $0 ;pp IS $1 ;m IS $2 ;mm IS $3 ;q IS $4 ;s IS $5
     LDOU q,qbase,0; LDA pp,qbase,8
     SET p,13; NEG m,13*13,n; SRU q,q,6           Начать с p = 13.
 1H  SR m,m,1                                     m ← ⌊(p^2 − N)/2⌋.
 2H  SR mm,m,3; LDOU s,qtop,mm; AND t,m,#3f;
     SLU t,one,t; ANDN s,s,t; STOU s,qtop,mm      Обнуление бита.
     ADD m,m,p; PBN m,2B                          Продвижение на p бит.
     SRU q,q,1; PBNZ q,3F                         Переход к следующему
                                                  потенциальному простому.
 2H  LDOU q,pp,0; INCL pp,8                       Чтение другого пакета
     OR p,p,#7f; PBNZ q,3F                          потенциально простых.
     ADD p,p,2; JMP 2B                            Пропуск последних 128 составных.
 2H  SRU q,q,1
 3H  ADD p,p,2; PBEV q,2B                         Установка p ← p + 2, пока p
                                                    не станет простым.
     MUL m,p,p; SUB m,m,n; PBN m,1B               Повтор, пока p^2 ≤ N. ▌
```

Время работы, $1172\mu + 5166\upsilon$, конечно же, гораздо меньшее, чем требуется для шагов P1–P8 программы 1.3.2′P, а именно $10037\mu + 641543\upsilon$ (улучшено до $10096\mu + 215351\upsilon$ в упр. 1.3.2′–14). [См. некоторые поучительные вариации в P. Pritchard, *Science of Computer Programming* **9** (1987), 17–35. На практике программы наподобие приведенной катастрофически замедляются, когда решето оказывается слишком большим для размещения в кэше. Лучшие результаты получаются при работе с сегментированным решетом, содержащим биты для чисел между $N_0 + k\delta$ и $N_0 + (k+1)\delta$, как предложено в работах L. J. Lander and T. R. Parkin, *Math. Comp.* **21** (1967), 483–488; C. Bays and R. H. Hudson, *BIT* **17** (1977), 121–127. Здесь N_0 может быть достаточно большим, но δ ограничено размером кэша; вычисления выполняются отдельно для $k = 0, 1, \ldots$. Сегментированные решета хорошо проработаны; см., например, T. R. Nicely, *Math. Comp.* **68** (1999), 1311–1315, и цитируемые там ссылки. Автор в 2006 году использовал такую программу для обнаружения необычно большого промежутка длиной 1370 чисел между 418 032 645 936 712 127 и следующим простым числом.]

25. $(1+1+25+1+1+25+1+1 = 56)$ мм; червь никогда не увидит страницы 2–500 тома 1 или 1–499 тома 4. (Если только книги не выстроены на полке в духе “прямого порядка байтов”; в этом случае ответ — 106 мм.) Эта классическая головоломка имеется в книге Сэма Лойда (Sam Loyd) *Cyclopedia* (New York: 1914), страницы 327 и 383.

26. Вместо деления на 12 можно прибегнуть к умножению на `#aa...ab` (см. упр. 1.3.1′–17); но умножение также является слишком медленной операцией. Можно работать с “плоской” последовательностью 12000000×5 последовательных битов (= 7.5 мегабайт), игнорируя границы между словами. Еще одна возможность заключается в применении схемы, использующей не прямой и не обратный порядок байтов, а *транспонированный*: поместим элемент k в октет $8(k \bmod 2^{20})$, где он сдвинут влево на $5\lfloor k/2^{20} \rfloor$. Поскольку $k < 12000000$, величина сдвига всегда будет меньше 60. Код `MMIX` для размещения элемента k в регистре $1 имеет вид `AND $0,k,[#fffff]; SLU $0,$0,3; LDOU $1,base,$0; SRU $0,k,20; 4ADDU $0,$0,$0; SRU $1,$1,$0; AND $1,$1,#1f`.

[Это решение использует 8 больши́х мегабайтов (2^{23} байт). Будет работать *любая* удобная схема для преобразования номеров элементов в адреса октетов и величины сдвигов, лишь бы был согласованно использован один и тот же метод. Конечно, простое '`LDBU $1,base,k`' было бы быстрее.]

27. (а) $((x-1) \oplus x) + x$. [Это упражнение основано на идее Лютера Вудрами (Luther Woodrum), который заметил, что $((x-1) \mid x) + 1 = (x \mathbin{\&} -x) + x$.]

(б) $(y + x) \mid y$, где $y = (x-1) \oplus x$.

(в, г, д) $((z \oplus x) + x) \mathbin{\&} z$, $((z \oplus x) + x) \oplus z$ и $\overline{((z \oplus x) + x)} \mathbin{\&} z$, где $z = x-1$.

(е) $x \oplus$(а); или, в качестве альтернативного решения, $t \oplus (t+1)$, где $t = x \mid (x-1)$. [Число $(0^\infty 01^a 11^b)_2$ выглядит проще, но, очевидно, требует *пяти* операций: $((t+1) \mathbin{\&} \bar{t}) - 1$.]

Все эти построения дают разумные результаты в исключительных случаях $x = -2^b$.

28. Бит 1, указывающий крайний справа 0 в x (например, $(101011)_2 \mapsto (000100)_2$); $-1 \mapsto 0$.

29. $\mu_k = \mu_{k+1} \oplus (\mu_{k+1} \ll 2^k)$ [см. *STOC* **6** (1974), 125]. Это отношение выполняется также для констант $\mu_{d,k}$ из (48) при $0 \le k < d$, если начать с $\mu_{d,d} = 2^{2^d} - 1$. (Однако не имеется простого пути перейти от μ_k к μ_{k+1}, если только не использовать операцию "zip"; см. (77).)

30. Добавляем '`CSZ rho,x,64`' к (50), что приводит к увеличению времени вычисления на 1υ; или заменяем последние две строки на `SRU t,y,rho; SLU t,t,2; SRU t,[#300020104],t; AND t,t,#f; ADD rho,rho,t`, что экономит 1υ. В случае (51) нам просто нужно обеспечить $rhotab[0] = 8$.

31. Прежде всего, его код зациклится при $x = 0$. Но даже после исправления этой ошибки предположение о случайности x весьма сомнительно. Во многих приложениях, когда мы хотим вычислить ρx для ненулевого 64-битового числа x, более разумным предположением является равновероятность получающихся результатов $\{0, 1, \ldots, 63\}$. При этом среднее значение и стандартное отклонение становятся равными 31.5 и ≈ 18.5.

32. '`NEGU y,x; AND y,x,y; MULU y,debruijn,y; SRU y,y,58; LDB rho,decode,y`' имеет оценочную стоимость $\mu + 14\upsilon$, хотя умножение на степень 2 может оказаться быстрее типичного умножения. Добавьте 1υ для исправления из упр. 30.

33. В действительности исчерпывающее вычисление показывает, что ровно 94 727 подходящих констант a дают "идеальную хеш-функцию" для данной задачи, 90 970 из них также распознают случаи степени двойки $y = 2^j$; 90 918 из них способны также распознавать случай $y = 0$. Множитель `#208b2430c8c82129` является единственным наилучшим в том смысле, что не требует обращения к записям таблицы выше $decode[32400]$, когда известно, что y является корректным входом.

34. Тождество (а) неверно при $x = 5$, $y = 6$; но (б) истинно, причем также и тогда, когда $xy = 0$. Доказательство (в): если $x \ne y$ и $\rho x = \rho y = k$, мы имеем $x = \alpha 10^k$ и $y = \beta 10^k$; следовательно, $x \oplus y = (\alpha \oplus \beta)00^k = (x-1) \oplus (y-1)$. Если $\rho x > \rho y = k$, мы имеем $(x \oplus y) \bmod 2^{k+2} \ne ((x-1) \oplus (y-1)) \bmod 2^{k+2}$.

35. Пусть $f(x) = x \oplus 3x$. Ясно, что $f(2x) = 2f(x)$ и $f(4x+1) = 4f(x) + 2$. Мы также имеем $f(4x-1) = 4f(x) + 2$ согласно упр. 34(в). Отсюда следует тождество из указания.

Для заданного n установим $u \leftarrow n \gg 1$, $v \leftarrow u + n$, $t \leftarrow u \oplus v$, $n^+ \leftarrow v \mathbin{\&} t$ и $n^- \leftarrow u \mathbin{\&} t$. Ясно, что $u = \lfloor n/2 \rfloor$ и $v = \lfloor 3n/2 \rfloor$, так что $n^+ - n^- = v - u = n$. А это и есть представление Райтвизнера, поскольку $n^+ \mid n^-$ не имеет последовательных единиц. [H. Prodinger, *Integers* **0** (2000), A08:1–A08:14. Кстати, мы также имеем $f(-x) = f(x)$.]

36. (i) Команды $x \leftarrow x \oplus (x \ll 1)$, $x \leftarrow x \oplus (x \ll 2)$, $x \leftarrow x \oplus (x \ll 4)$, $x \leftarrow x \oplus (x \ll 8)$, $x \leftarrow x \oplus (x \ll 16)$, $x \leftarrow x \oplus (x \ll 32)$ заменяют x на $x^{\oplus}$. (ii) $x^{\&} = x \mathbin{\&} {\sim}(x+1)$.

(Применения $x^{\oplus}$ описаны в упр. 66 и 70; см. также упр. 128 и 209.)

37. Вставка '`CSZ y,x,half`' после `FLOTU` в (55), где `half` = `#3fe0000000000000`; обратите внимание, что (55) гласит '`SR`' (не '`SRU`'). Если $lamtab[0] = -1$, изменения в (56) не нужны.

38. '`SRU t,x,1; OR y,x,t; SRU t,y,2; OR y,y,t; SRU t,y,4; OR y,y,t; ...; SRU t,y,32; OR y,y,t; SRU t,y,1; SUBU y,y,t`' выполняется за время 14υ.

39. (Решение Г. С. Уоррена-мл. (H. S. Warren, Jr.)) Пусть $\sigma(x)$ обозначает результат распространения x вправо, как в первой строке (57). Вычислите $x \mathbin{\&} \sigma((x \gg 1) \mathbin{\&} \bar{x})$.

40. Предположим, что $\lambda x = \lambda y = k$. Если $x = y = 0$, (58), определенно, выполняется, независимо от того, как мы определяем $\lambda 0$. В противном случае $x = (1\alpha)_2$ и $y = (1\beta)_2$ для некоторых бинарных строк α и β с $|\alpha| = |\beta| = k$; и $x \oplus y < 2^k \le x \mathbin{\&} y$. С другой стороны, если $\lambda x < \lambda y = k$, мы имеем $x \oplus y \ge 2^k > x \mathbin{\&} y$. А Г. С. Уоррен-мл. (H. S. Warren, Jr.) заметил, что $\lambda x < \lambda y$ тогда и только тогда, когда $x < y \mathbin{\&} \bar{x}$.

41. (а) $\sum_{n=1}^{\infty}(\rho n)z^n = \sum_{k=1}^{\infty} z^{2^k}/(1-z^{2^k}) = \sum_{k=1}^{\infty} kz^{2^k}/(1-z^{2^{k+1}}) = z/(1-z) - \sum_{k=0}^{\infty} z^{2^k}/(1+z^{2^k})$. Производящая функция Дирихле проще: $\sum_{n=1}^{\infty}(\rho n)/n^z = \zeta(z)/(2^z-1)$.

(б) $\sum_{n=1}^{\infty}(\lambda n)z^n = \sum_{k=1}^{\infty} z^{2^k}/(1-z)$.

(в) $\sum_{n=1}^{\infty}(\nu n)z^n = \sum_{k=0}^{\infty} z^{2^k}/((1-z)(1+z^{2^k})) = \sum_{k=0}^{\infty} z^{2^k}\mu_k(z)$, где $\mu_k(z) = (1 + z + \cdots + z^{2^k-1})/(1-z^{2^{k+1}})$. ("Магические маски" из (47) соответствуют $\mu_k(2)$.)

[В *Automatic Sequences* Ж.-П. Аллуша (J.-P. Allouche) и Д. Шаллита (J. Shallit) (2003), главе 3, имеется дополнительная информация о функциях ρ и ν, которые они обозначают как ν_2 и s_2.]

42. $e_1 2^{e_1-1} + (e_2+2)2^{e_2-1} + \cdots + (e_r+2r-2)2^{e_r-1}$ по индукции по r. [D. E. Knuth, *Proc. IFIP Congress* (1971), **1**, 19–27. Фрактальные аспекты этой суммы проиллюстрированы на рис. 3.1 и 3.2 книги Аллуша и Шаллита.] Рассмотрена также $S'_n(1)$, где

$$S_n(z) \;=\; \sum_{k=0}^{n-1} z^{\nu k} \;=\; (1+z)^{e_1} + z(1+z)^{e_2} + \cdots + z^{r-1}(1+z)^{e_r}.$$

43. Непосредственная реализация (63), '`SET nu,0; SET y,x; BZ y,Done; 1H ADD nu,nu,1; SUBU t,y,1; AND y,y,t; PBNZ y,1B`', имеет стоимость $(5+4\nu x)\upsilon$; она превосходит реализацию (62) при $\nu x < 4$, такая же при $\nu x = 4$ и проигрывает при $\nu x > 4$.

Однако в реализации (62) можно сэкономить 4υ, если заменить последнее умножение со сдвигом на '$y \leftarrow y + (y \gg 8)$, $y \leftarrow y + (y \gg 16)$, $y \leftarrow y + (y \gg 32)$, $\nu \leftarrow y \mathbin{\&} {}^{\#}\mathtt{ff}$'. [Конечно, гораздо лучше использовать отдельную команду `MMIX` '`SADD nu,x,0`'.]

44. Пусть эта сумма равна $\nu^{(2)}x$. Если мы можем решить задачу для 2^d-битных чисел, то можем решить ее и для 2^{d+1}-битных чисел, поскольку $\nu^{(2)}(2^{2^d}x + x') = \nu^{(2)}x + \nu^{(2)}x' + 2^d\nu x$. Таким образом, на 64-битной машине само собой напрашивается решение, аналогичное (62):

Установить $z \leftarrow (x \gg 1) \mathbin{\&} \mu_0$ и $y \leftarrow x - z$.
Установить $z \leftarrow ((z + (z \gg 2)) \mathbin{\&} \mu_1) + ((y \mathbin{\&} \bar{\mu}_1) \gg 1)$ и $y \leftarrow (y \mathbin{\&} \mu_1) + ((y \gg 2) \mathbin{\&} \mu_1)$.
Установить $z \leftarrow ((z + (z \gg 4)) \mathbin{\&} \mu_2) + ((y \mathbin{\&} \bar{\mu}_2) \gg 2)$ и $y \leftarrow (y + (y \gg 4)) \mathbin{\&} \mu_2$.
Наконец $\nu^{(2)} \leftarrow (((Az) \bmod 2^{64}) \gg 56) + ((((By) \bmod 2^{64}) \gg 56) \ll 3)$,
где $A = (11111111)_{256}$ и $B = (01234567)_{256}$.

Но на машине `MMIX` со встроенным контрольным суммированием есть лучшее решение, предложенное Д. Даллосом (J. Dallos).

```
SADD  nu2,x,m5     SADD  t,x,m3       2ADDU nu2,nu2,t    SADD  t,x,m0
SADD  t,x,m4       2ADDU nu2,nu2,t    SADD  t,x,m1       2ADDU nu2,nu2,t  ▌
2ADDU nu2,nu2,t    SADD  t,x,m2       2ADDU nu2,nu2,t
```

[В общем случае $\nu^{(2)}x = \sum_k 2^k \nu(x \mathbin{\&} \bar{\mu}_k)$. См. *Dr. Dobb's Journal* **8**, 4 (April 1983), 24–37.]

45. Пусть $d = (x - y) \mathbin{\&} (y - x)$; следует проверить выполнение соотношения $d \mathbin{\&} y \neq 0$. [Рокицки (Rokicki) обнаружил, что эта идея, именуемая *солексным упорядочением*, может использоваться с адресами узлов для псевдослучайных бинарных деревьев поиска или декартовых деревьев, как если бы они были дерамидами (treaps), без необходимости в дополнительном случайном "приоритетном ключе" в каждом узле. См. *U.S. Patent 6347318* (12 February 2002).]

46. `SADD t,x,m; NXOR y,x,m; CSOD x,t,y;` маска `m` равна `~(1<<i|1<<j)`. (В общем случае эти команды дополняют биты, указанные $\overline{\mathtt{m}}$, если эти биты имеют нечетную четность.)

47. $y \leftarrow (x \gg \delta) \mathbin{\&} \theta$, $z \leftarrow (x \mathbin{\&} \theta) \ll \delta$, $x \leftarrow (x \mathbin{\&} m) \mid y \mid z$, где $\overline{m} = \theta \mid (\theta \ll \delta)$.

48. Для заданного δ имеется $s_\delta = \prod_{j=0}^{\delta-1} F_{\lfloor (n+j)/\delta \rfloor + 1}$ различных δ-обменов, включая тождественную перестановку. (См. упр. 4.5.3–32.) Суммирование по δ дает общее количество, равное $1 + \sum_{\delta=1}^{n-1}(s_\delta - 1)$.

49. (а) Множество $S = \{a_1\delta_1 + \cdots + a_m\delta_m \mid \{a_1, \ldots, a_m\} \subseteq \{-1, 0, +1\}\}$ для перемещений δ_1, ..., δ_m должно содержать $\{n-1, n-3, \ldots, 1-n\}$, поскольку k-й бит должен быть обменен с $(n+1-k)$-м битом для $1 \le k \le n$. Следовательно, $|S| \ge n$. А S содержит не более 3^m чисел, не более $2 \cdot 3^{m-1}$ из которых нечетны.

(б) Ясно, что $s(mn) \le s(m) + s(n)$, поскольку можно обратить порядок битов m полей по n бит каждое. Таким образом, $s(3^m) \le m$ и $s(2 \cdot 3^m) \le m + 1$. Кроме того, обращение порядка 3^m бит использует только δ-обменов с четными значениями δ; соответствующие $(\delta/2)$-обмены доказывают, что мы имеем $s((3^m \pm 1)/2) \le m$. Эти верхние границы соответствуют нижним границам (а) при $m > 1$.

(в) Строка $\alpha a \beta \theta \psi z \omega$ с $|\alpha| = |\beta| = |\theta| = |\psi| = |\omega| = n$ может быть изменена на $\omega z \psi \theta \beta a \alpha$ с помощью $(3n+1)$-обмена с последующим $(n+1)$-обменом. Затем $s(n)$ дальнейших обменов обращают весь порядок. Следовательно, $s(32) \le s(6) + 2 = 4$, а $s(64) \le 5$. А равенство выполняется согласно (а).

Кстати, $s(63) = 4$, поскольку $s(7) = s(9) = 2$. Нижняя граница в (а) оказывается точным значением $s(n)$ для $1 \le n \le 22$, с тем исключением, что $s(16) = 4$.

50. Выразим $n = (t_m \ldots t_1 t_0)_3$ в сбалансированной тернарной записи. Пусть $n_j = (t_m \ldots t_j)_3$ и $\delta_j = 2n_j + t_{j-1}$, так что $n_{j-1} - \delta_j = n_j$ и $2\delta_j - n_{j-1} = n_j + t_{j-1}$ для $1 \le j \le m$. Пусть $E_0 = \{0\}$ и $E_{j+1} = E_j \cup \{t_j - x \mid x \in E_j\}$ для $0 \le j < m$. (Таким образом, например, $E_1 = \{0, t_0\}$ и $E_2 = \{0, t_0, t_1, t_1 - t_0\}$.) Заметим, что из $\varepsilon \in E_j$ вытекает $|\varepsilon| \le j$.

Допустим по индукции по j, что δ-обмены для $\delta = \delta_1, \ldots, \delta_j$ заменяют n-битовое слово $\alpha_1 \ldots \alpha_{3j}$ на $\alpha_{3j} \ldots \alpha_1$, где каждое подслово α_k имеет длину $n_j + \varepsilon_k$ для некоторого $\varepsilon_k \in E_j$. Если $n_{j+1} > j$, δ_{j+1}-обмен внутри каждого подслова будет сохранять это допущение. В противном случае каждое подслово α_k имеет $|\alpha_k| \le n_j + j \le 3n_{j+1} + 1 + j \le 4j + 1 < 4m$. Следовательно, 2^k-обмены для $\lfloor \lg 4m \rfloor \ge k \ge 0$ обратят порядок их всех. (Заметим, что 2^k-обмен подслова размером t, где $2^k < t \le 2^{k+1}$, сводит его к трем подсловам размером $t - 2^k$, $2^{k+1} - t$ и $t - 2^k$.)

51. (а) Если $c = (c_{d-1} \ldots c_0)_2$, мы должны иметь $\theta_{d-1} = c_{d-1}\mu_{d,d-1}$. Но для $0 \le k < d-1$ мы можем взять $\theta_k = c_k \mu_{d,k} \oplus \hat\theta_k$, где $\hat\theta_k$ — *любая* маска $\subseteq \mu_{d,k}$.

(б) Пусть $\Theta(d, c)$ — множество всех таких последовательностей масок. Ясно, что $\Theta(1, c) = \{c\}$. Когда $d > 1$, мы рекурсивно получим

$$\Theta(d, c) = \bigl\{(\theta_0, \ldots, \theta_{d-2}, \theta_{d-1}, \hat\theta_{d-2}, \ldots, \hat\theta_0) \mid \theta_k = \theta'_{k-1} \mathbin{\ddagger} \theta''_{k-1},\ \hat\theta_k = \hat\theta'_{k-1} \mathbin{\ddagger} \hat\theta''_{k-1}\bigr\},$$

с помощью "zip-операции" последовательностей $(\theta'_0, \ldots, \theta'_{d-3}, \theta'_{d-2}, \hat\theta'_{d-3}, \ldots, \hat\theta'_0) \in \Theta(d-1, c')$ и $(\theta''_0, \ldots, \theta''_{d-3}, \theta''_{d-2}, \hat\theta''_{d-3}, \ldots, \hat\theta''_0) \in \Theta(d-1, c'')$ для некоторых подходящих θ_0, $\hat\theta_0$, c' и c''.

Когда c нечетно, двудольный граф, соответствующий (75), имеет только один цикл; так что $(\theta_0, \hat\theta_0, c', c'')$ имеет вид либо $(\mu_{d,0}, 0, \lceil c/2 \rceil, \lfloor c/2 \rfloor)$, либо $(0, \mu_{d,0}, \lfloor c/2 \rfloor, \lceil c/2 \rceil)$. Но

когда c четно, двудольный граф имеет 2^{d-1} двойных связей; так что $\theta_0 = \hat\theta_0$ представляет собой произвольную маску $\subseteq \mu_{d,0}$, и $c' = c'' = c/2$. [Кстати, $\lg|\Theta(d,c)| = 2^{d-1}(d-1) - \sum_{k=1}^{d-1}(2^{d-k}-1)\,(2^{k-1} - |2^{k-1} - c \bmod 2^k|)$.]

В обоих случаях мы можем, таким образом, положить $\hat\theta_{d-2} = \cdots = \hat\theta_0 = 0$ и опустить вторую половину (71) полностью. Конечно, в случае (б) мы должны выполнять циклический сдвиг непосредственно, вместо применения (71). Но упр. 58 доказывает, что с помощью (71) с $\hat\theta_k = 0$ для всех k могут быть обработаны многие другие полезные перестановки, следующие за циклическим сдвигом. Перестановки, *обратные* к ним, могут быть обработаны с $\theta_k = 0$ для $0 \le k < d-1$.

52. В следующих решениях, где это возможно, делается $\hat\theta_j = 0$. Маски θ мы выразим через μ, например, записывая $\mu_{6,5} \mathbin{\&} \mu_0$ вместо указания запрошенной шестнадцатеричной константы #55555555; запись через μ короче и более поучительна.

(a) $\theta_k = \mu_{6,k} \mathbin{\&} \mu_5$ и $\hat\theta_k = \mu_{6,k} \mathbin{\&} (\mu_{k+1} \oplus \mu_{k-1})$ для $0 \le k < 5$; $\theta_5 = \theta_4$. (Здесь $\mu_{-1} = 0$. Чтобы получить "другое" идеальное тасование, $(x_{31}x_{63}\ldots x_1x_{33}x_0x_{32})_2$, положим $\hat\theta_0 = \mu_{6,0} \mathbin{\&} \bar\mu_1$.)

(b) $\theta_0 = \theta_3 = \hat\theta_0 = \mu_{6,0} \mathbin{\&} \mu_3$; $\theta_1 = \theta_4 = \hat\theta_1 = \mu_{6,1} \mathbin{\&} \mu_4$; $\theta_2 = \theta_5 = \hat\theta_2 = \mu_{6,2} \mathbin{\&} \mu_5$; $\hat\theta_3 = \hat\theta_4 = 0$. [См. общую теорию в J. Lenfant, *IEEE Trans.* **C-27** (1978), 637–647.]

(в) $\theta_0 = \mu_{6,0} \mathbin{\&} \mu_4$; $\theta_1 = \mu_{6,1} \mathbin{\&} \mu_5$; $\theta_2 = \theta_4 = \mu_{6,2} \mathbin{\&} \mu_4$; $\theta_3 = \theta_5 = \mu_{6,3} \mathbin{\&} \mu_5$; $\hat\theta_0 = \mu_{6,0} \mathbin{\&} \mu_2$; $\hat\theta_1 = \mu_{6,1} \mathbin{\&} \mu_3$; $\hat\theta_2 = \hat\theta_0 \oplus \theta_2$; $\hat\theta_3 = \hat\theta_1 \oplus \theta_3$; $\hat\theta_4 = 0$.

(г) $\theta_k = \mu_{6,k} \mathbin{\&} \mu_{5-k}$ для $0 \le k \le 5$; $\hat\theta_k = \theta_k$ для $0 \le k \le 2$; $\hat\theta_3 = \hat\theta_4 = 0$.

53. Можно записать ψ как произведение $d-t$ транспозиций, $(u_1v_1)\ldots(u_{d-t}v_{d-t})$ (см. упр. 5.2.2–2). Перестановка, вызванная одной транспозицией (uv) индексных цифр, при $u < v$ соответствует $(2^v - 2^u)$-обмену с маской $\mu_{d,v} \mathbin{\&} \bar\mu_u$. Мы должны выполнить такой обмен для (u_1v_1) первым, …, $(u_{d-t}v_{d-t})$ — последним.

В частности, идеальное тасование в 2^d-битовом регистре соответствует случаю, где перестановка $\psi = (01\ldots(d{-}1))$ является одноцикловой; так что оно может быть достигнуто выполнением таких $(2^v - 2^u)$-обменов для $(u,v) = (0,1), \ldots, (0,d-1)$. Например, когда $d = 3$, двушаговая процедура представляет собой $12345678 \mapsto 13245768 \mapsto 15263748$. [Гай Стил (Guy Steele) предложил альтернативную $(d-1)$-шаговую процедуру: можно выполнить 2^k-обмен с маской $\mu_{d,k+1} \mathbin{\&} \bar\mu_k$ для $d-1 > k \ge 0$. Когда $d = 3$, его метод отображает $12345678 \mapsto 12563478 \mapsto 15263748$.]

Транспонирование матрицы в упр. 52, (б) соответствует $d = 6$ и $(u,v) = (0,3)$, $(1,4)$, $(2,5)$. Эти операции являются шагами, которые представляют собой 7-, 14- и 28-обмен для транспонирования матрицы размером 8×8, проиллюстрированного в тексте раздела; они могут быть выполнены в любом порядке.

В упр. 52, (в) используйте $d = 6$ и $(u,v) = (0,2)$, $(1,3)$, $(0,4)$, $(1,5)$. Упр. 52, (г) выполняется так же легко, как и 52, (б), с $(u,v) = (0,5)$, $(1,4)$, $(2,3)$.

54. Транспонирование заключается в обращении порядка битов побочных диагоналей (направленных с юго-запада на северо-восток). Последовательные элементы этих диагоналей находятся в регистре на расстоянии $m-1$ один от другого. Одновременное обращение порядка битов во всех диагоналях соответствует одновременному обращению порядка битов в подсловах размером 1, …, m, которое может быть выполнено с помощью 2^k-обменов для $0 \le k < \lceil \lg m \rceil$ (поскольку такое транспонирование легко выполняется при m, являющемся степенью 2, как показано в разделе). Вот как выглядит процедура для $m = 7$.

Задано	6-обмен	12-обмен	24-обмен
00 01 02 03 04 05 06	00 **10** 02 **12** 04 **14** 06	00 10 **20** **30** 04 14 **24**	00 10 20 30 **40** **50** **60**
10 11 12 13 14 15 16	**01** 11 **03** 13 **05** 15 **25**	01 11 **21** **31** 05 15 25	01 11 21 31 **41** **51** **61**
20 21 22 23 24 25 26	20 **30** 22 **32** 24 **16** 26	**02** **12** 22 32 **06** 16 26	02 12 22 32 **42** **52** **62**
30 31 32 33 34 35 36	**21** 31 **23** 33 **43** 35 **45**	**03** **13** 23 33 43 **53** **63**	03 13 23 33 43 53 63
40 41 42 43 44 45 46	40 **50** 42 **34** 44 **36** 46	40 50 **60** 34 44 **54** **64**	**04** **14** **24** 34 44 54 64
50 51 52 53 54 55 56	**41** 51 **61** 53 **63** 55 **65**	41 51 61 **35** **45** 55 65	**05** **15** **25** 35 45 55 65
60 61 62 63 64 65 66	60 **52** 62 **54** 64 **56** 66	**42** 52 62 **36** **46** 56 66	**06** **16** **26** 36 46 56 66

55. Для заданных x и y сначала устанавливаем $x \leftarrow x \mid (x \ll 2^k)$ и $y \leftarrow y \mid (y \ll 2^k)$ для $2d \le k < 3d$. Затем устанавливаем $x \leftarrow (2^{2d+k} - 2^k)$-обмен x с маской $\mu_{2d+k} \,\&\, \bar{\mu}_k$ и $y \leftarrow (2^{2d+k} - 2^{d+k})$-обмен y с маской $\mu_{2d+k} \,\&\, \bar{\mu}_{d+k}$ для $0 \le k < d$. Наконец устанавливаем $z \leftarrow x \,\&\, y$, затем либо $z \leftarrow z \mid (z \gg 2^k)$, либо $z \leftarrow z \oplus (z \gg 2^k)$ для $2d \le k < 3d$ и $z \leftarrow z \,\&\, (2^{n^2} - 1)$. [Идея заключается в формировании двух массивов размером $n \times n \times n$ — $x = (x_{000} \ldots x_{(n-1)(n-1)(n-1)})_2$ и $y = (y_{000} \ldots y_{(n-1)(n-1)(n-1)})_2$ с $x_{ijk} = a_{jk}$ и $y_{ijk} = b_{jk}$, затем — в транспонировании координат так, чтобы $x_{ijk} = a_{ji}$ и $y_{ijk} = b_{ik}$; после этого операция $x \,\&\, y$ выполняет все n^3 битовых умножений за один раз. Этот метод представлен в V. R. Pratt and L. J. Stockmeyer, *J. Computer and System Sci.* **12** (1976), 210–213.]

56. Используйте (71) с $\theta_0 = \hat{\theta}_0 = 0$, $\theta_1 = {}^{\#}\texttt{0010201122113231}$, $\theta_2 = {}^{\#}\texttt{00080e0400080c06}$, $\theta_3 = {}^{\#}\texttt{00000092008100a2}$, $\theta_4 = {}^{\#}\texttt{0000000000000f16}$, $\theta_5 = {}^{\#}\texttt{0000000003199c26}$, $\hat{\theta}_4 = {}^{\#}\texttt{00000c9f0000901a}$, $\hat{\theta}_3 = {}^{\#}\texttt{003a00b50015002b}$, $\hat{\theta}_2 = {}^{\#}\texttt{000103080c0d0f0c}$ и $\hat{\theta}_1 = {}^{\#}\texttt{0020032033233333}$.

57. Два варианта выбора для каждого цикла при $d > 1$ имеют дополняющие установки. Так что можно выбрать настройку, в которой не менее половины модулей перекрещиваний не активны, за исключением среднего столбца. (См. дополнительную информацию по перестановочным сетям в упр. 5.3.4–55.)

58. (а) Каждая отличная от других настройка модулей перекрещиваний дает свою, отличную от других перестановку, поскольку имеется ровно один путь от входной линии i к выходной линии j для всех $0 \le i, j < N$. (Сеть с таким свойством называется сетью с автоблокиратором ("banyan").) Единственный такой путь переносит вход i на линию $l(i, j, k) = ((i \gg k) \ll k) + (j \bmod 2^k)$ после выполнения k обменивающих шагов.

(б) Мы имеем $l(i\varphi, i, k) = l(j\varphi, j, k)$ тогда и только тогда, когда $i \bmod 2^k = j \bmod 2^k$ и $i\varphi \gg k = j\varphi \gg k$; так что $(*)$ является необходимым. Это также достаточное условие, поскольку отображение φ, которое удовлетворяет $(*)$, всегда может быть направлено таким путем, что $j\varphi$ появляется на линии $l = l(j\varphi, j, k)$ после k шагов: если $k > 0$, $j\varphi$ появится на линии $l(j\varphi, j, k-1)$, которая является одним из входов для l. Условие $(*)$ говорит, что его можно маршрутизировать на линию l без конфликтов, даже если l представляет собой $l(i\varphi, i, k)$.

[В *IEEE Transactions* **C-24** (1975), 1145–1155, Дункан Ловри (Duncan Lawrie) доказал, что условие $(*)$ необходимое и достаточное для произвольного *отображения* φ множества в себя, когда модули перекрещиваний могут быть обобщенными отображающими модулями 2×2, как в упр. 75. Кроме того, отображение φ может быть только частично определенным, с $j\varphi = *$ ("безразличное значение") для некоторых значений j. Доказательство, приведенное в предыдущем абзаце, в действительности демонстрирует более общую теорему Ловри.]

(в) $i \bmod 2^k = j \bmod 2^k$ тогда и только тогда, когда $k \le \rho(i \oplus j)$; $i \gg k = j \gg k$ тогда и только тогда, когда $k > \lambda(i \oplus j)$; а $i\varphi = j\varphi$ тогда и только тогда, когда $i = j$, если φ представляет собой перестановку.

(г) $\lambda(i\varphi \oplus j\varphi) \ge \rho(i \oplus j)$ для всех $i \ne j$ тогда и только тогда, когда $\lambda(i\tau\varphi \oplus j\tau\varphi) \ge \rho(i\tau \oplus j\tau) = \rho(i \oplus j)$ для всех $i \ne j$, поскольку τ является перестановкой. [Заметим, что обозначения могут вводить в заблуждение: бит $j\tau\varphi$ появляется в позиции j, если перестановка φ применяется первой, а *затем* τ. Группа Сылова T включает много интересных и важных перестановок, в том числе обращение порядка битов и циклические сдвиги.

Она соответствует настройкам омега-сети, когда перекрещивания длиной 2^j, являющиеся конгруэнтными по модулю 2^{j+1}, либо все выполняют переключение сигналов, либо все пропускают сигналы без изменения, как единый узел.]

(д) Поскольку $l(j,j,k) = j$ для $0 \le k \le d$, перестановка Ω фиксирует j тогда и только тогда, когда каждый из ее обменов фиксирует j. Таким образом, обмены, выполняемые φ и ψ, оперируют непересекающимися элементами. Объединение этих обменов дает $\varphi\psi$.

(е) Любая установка перекрещиваний соответствует перестановке, которая заставляет модули компаратора Батчера выполнять эквивалентное переключение.

59. Это количество равно $2^{M_d(a,b)}$, где $M_d(a,b)$ представляет собой число перекрещиваний, обе конечные точки которых лежат в $[a\,..\,b]$. Для их подсчета положим $k = \lambda(a \oplus b)$, $a' = a \bmod 2^k$ и $b' = b \bmod 2^k$; заметим, что $b - a = 2^k + b' - a'$ и $M_d(a,b) = M_{k+1}(a', 2^k + b')$. Подсчет перекрещиваний в верхней половине и в нижней половине, а также тех, которые проходят между половинами, дает $M_{k+1}(a', 2^k+b') = M_k(a', 2^k-1) + M_k(0,b') + ((b'+1) \dot{-} a')$. Наконец мы имеем $M_k(0,b') = S(b'+1)$; и $M_k(a', 2^k-1) = M_k(0, 2^k-1-a') = S(2^k - a') = k2^{k-1} - ka' + S(a')$, где $S(n)$ вычисляется, как в упр. 42.

60. Цикл длиной $2l$ соответствует шаблону $u_0 \leftarrow v_0 \leftrightarrow v_1 \rightarrow u_1 \leftrightarrow u_2 \leftarrow v_2 \leftrightarrow \cdots \leftrightarrow v_{2l-1} \rightarrow u_{2l-1} \leftrightarrow u_{2l}$, где $u_{2l} = u_0$ и '$u \leftarrow v$' или '$v \rightarrow u$' обозначает, что перестановка отправляет u в v, а '$x \leftrightarrow y$' обозначает, что $x = y \oplus 1$.

Мы можем генерировать случайную перестановку следующим образом: для данного u_0 имеется $2n$ выборов v_0, затем $2n-1$ выборов для u_1, только один из которых дает $u_2 = u_0$, затем $2n-2$ выборов v_2, затем $2n-3$ выборов для u_3, только один из которых закрывает цикл, и т. д.

Следовательно, производящая функция представляет собой $G(z) = \prod_{j=1}^{n} \frac{2n-2j+z}{2n-2j+1}$. Ожидаемое количество циклов, k, равно $G'(1) = H_{2n} - \frac{1}{2}H_n = \frac{1}{2}\ln n + \ln 2 + \frac{1}{2}\gamma + O(n^{-1})$. Среднее значение 2^k равно $G(2) = (2^n n!)^2/(2n)! = \sqrt{\pi n} + O(n^{-1/2})$; а дисперсия составляет $G(4) - G(2)^2 = (n+1-G(2))G(2) = \sqrt{\pi}\,n^{3/2} + O(n)$.

62. Установку перекрещиваний в $P(2^d)$ можно сохранить в $(2d-1)2^{d-1} = Nd - \frac{1}{2}N$ бит. Для получения обратной перестановки следует работать справа налево. [См. P. Heckel and R. Schroeppel, *Electronic Design* **28**, 8 (12 April 1980), 148–152. Заметим, что *любой* способ представления произвольной перестановки требует как минимум $\lg N! > Nd - N/\ln 2$ бит памяти; так что с точки зрения используемой памяти это представление близко к оптимальному.]

63. (i) $x = y$. (ii) Либо z четно, либо $x \oplus y < 2^{\max(0,(z-1)/2)}$. (Когда z нечетно, мы имеем $(x \ddagger y) \gg z = (y \gg \lceil z/2 \rceil) \ddagger (x \gg \lfloor z/2 \rfloor)$, даже когда $z < 0$.) (iii) Это тождество выполняется для всех w, x, y и z (а также для любого другого булева оператора вместо &).

64. $(((z \,\&\, \mu_0) + (z' \mid \bar{\mu}_0)) \,\&\, \mu_0) \mid (((z \,\&\, \bar{\mu}_0) + (z' \mid \mu_0)) \,\&\, \bar{\mu}_0)$. (См. (86).)

65. $xu(x^2) + v(x^2) = xu(x)^2 + v(x)^2$.

66. (а) $v(x) = (u(x)/(1+x^\delta)) \bmod x^n$; это единственный полином степени, меньшей чем n, такой, что $(1+x^\delta)v(x) \equiv u(x)$ (по модулю x^n). (Или, что то же самое, v является единственным n-битовым целым числом, таким, что $(v \oplus (v \ll \delta)) \bmod 2^n = u$.)

(б) Можно считать, что $n = 64m$ и что $u = (u_{m-1} \ldots u_1 u_0)_{2^{64}}$, $v = (v_{m-1} \ldots v_1 v_0)_{2^{64}}$. Установить $c \leftarrow 0$; затем, используя упр. 36, установить $v_j \leftarrow u_j^{\oplus} \oplus (-c)$ и $c \leftarrow v_j \gg 63$ для $j = 0, 1, \ldots, m-1$.

(в) Установить $c \leftarrow v_0 \leftarrow u_0$; затем $v_j \leftarrow u_j \oplus c$ и $c \leftarrow v_j$ для $j = 1, 2, \ldots, m-1$.

(г) Начать с $c \leftarrow 0$ и для $j = 0, 1, \ldots, m-1$ выполнить следующее: установить $t \leftarrow u_j$, $t \leftarrow t \oplus (t \ll 3)$, $t \leftarrow t \oplus (t \ll 6)$, $t \leftarrow t \oplus (t \ll 12)$, $t \leftarrow t \oplus (t \ll 24)$, $t \leftarrow t \oplus (t \ll 48)$, $v_j \leftarrow t \oplus c$, $c \leftarrow (t \gg 61) \times {}^{\#}\texttt{9249249249249249}$.

(д) Начать с $v \leftarrow u$. Затем для $j = 1, 2, \ldots, m-1$, установить $v_j \leftarrow v_j \oplus (v_{j-1} \ll 3)$ и (если $j < m-1$) $v_{j+1} \leftarrow v_{j+1} \oplus (v_{j-1} \gg 61)$.

67. Пусть $n = 2l-1$ и $m = n-2d$. Если $\frac{1}{2}n < k < n$, мы имеем $x^{2k} \equiv x^{m+t} + x^t$ (по модулю $x^n + x^m + 1$), где $t = 2k-n$ нечетно. Следовательно, если $v = (v_{n-1} \ldots v_1 v_0)_2$, число

$$w \;=\; u \oplus (((u \gg d) \oplus (u \gg 2d) \oplus (u \gg 3d) \oplus \cdots) \mathbin{\&} -2^{l-d})$$

оказывается равным $(v_{n-2} \ldots v_3 v_1 v_{n-1} \ldots v_2 v_0)_2$. Например, когда $l = 4$ и $d = 2$, квадрат $u_6x^6 + \cdots + u_1x + u_0$ по модулю $(x^7 + x^3 + 1)$ равен $u_6x^5 + u_5x^3 + (u_6 \oplus u_4)x^1 + (u_5 \oplus u_3)x^6 + (u_6 \oplus u_4 \oplus u_2)x^4 + u_1x^2 + u_0$. Для вычисления v мы, таким образом, выполняем идеальное тасование, $v = \lfloor w/2^l \rfloor \mathbin{\ddagger} (w \bmod 2^l)$. Число w может быть вычислено методами, подобными использованными в предыдущем упражнении. [См. R. P. Brent, S. Larvala and P. Zimmermann, *Math. Comp.* **72** (2003), 1443–1452; **74** (2005), 1001–1002.]

68. `SRU t,x,delta; PUT rM,theta; MUX x,t,x`.

69. Заметим, что процедура может работать некорректно, если мы попытаемся выполнить 2^{d-1}-сдвиг первым, а не последним. Ключ к доказательству того, что стратегия, заключающаяся в том, чтобы сначала выполнять малые сдвиги, работает корректно, состоит в рассмотрении промежутков *между* выбранными битами; мы докажем, что длины этих промежутков после 2^k-сдвига кратны 2^{k+1}.

Рассмотрим бесконечную строку $\chi_k = \ldots 1^{t_4} 0^{2^k} 1^{t_3} 0^{2^k} 1^{t_2} 0^{2^k} 1^{t_1} 0^{2^k} 1^{t_0}$, которая представляет ситуацию, где $t_l \geq 0$ элементов необходимо сдвинуть на $2^k l$ позиций вправо. 2^k-сдвиг с любой маской вида $\theta_k = \ldots 0^{t_4} *^{2^{k+1}} 1^{t_3} 0^{t_2} *^{2^{k+1}} 1^{t_1} 0^{t_0}$ оставляет нас в ситуации, представленной строкой $\chi_{k+1} = \ldots 1^{T_2} 0^{2^{k+1}} 1^{T_1} 0^{2^{k+1}} 1^{T_0}$, в которой ровно $T_l = t_{2l} + t_{2l+1}$ элементов необходимо сдвинуть вправо на $2^{k+1} l$ позиций. Так что исходное утверждение выполняется по индукции по k.

70. Пусть $\psi_k = \theta_k \oplus (\theta_k \ll 1)$, так что $\theta_k = \psi_k^{\oplus}$ в обозначениях из упр. 36. Если в ответе к предыдущему упражнению взять $*^{2^{k+1}} = 0^{2^k} 1^{2^k}$, мы имеем $\psi_0 = \bar{\chi}$ и $\psi_{k+1} = (\psi_k \mathbin{\&} \bar{\theta}_k) \gg 2^k$. Следовательно, мы можем действовать следующим образом.

Установить $\psi \leftarrow \bar{\chi}$, $k \leftarrow 0$ и повторять следующие шаги до тех пор, пока $\psi \neq 0$: установить $x \leftarrow \psi$ и затем $x \leftarrow x \oplus (x \ll 2^l)$ для $0 \leq l < d$, затем $\theta_k \leftarrow x$, $\psi \leftarrow (\psi \mathbin{\&} \bar{x}) \gg 2^k$ и $k \leftarrow k+1$.

Вычисления завершаются при $k = \lambda\nu\bar{\chi}+1$; остальные маски $\theta_k, \ldots, \theta_{d-1}$, если таковые имеются, нулевые, и соответствующие шаги в (80) можно опустить. "Минимальные" маски, для которых $*^{2^{k+1}} = 0^{2^{k+1}}$ в ответе к упр. 69, получаются, если операции '$\theta_k \leftarrow x$, $\psi \leftarrow (\psi \mathbin{\&} \bar{x}) \gg 2^k$' в приведенном выше цикле заменить на '$\psi \leftarrow (\psi \mathbin{\&} \bar{x}) \gg 2^k$, $\theta_k \leftarrow x \mathbin{\&} (x + \psi)$'.

[См. раздел "*Сжатие, или обобщенное извлечение*" в H. S. Warren, Jr., *Hacker's Delight* (Addison–Wesley, 2002), §7–4 (Генри С. Уоррен, *Алгоритмические трюки для программистов* (М.: Издательский дом "Вильямс", 2003)); а также G. L. Steele Jr., *U.S. Patent 6715066* (30 March 2004). В ЭВМ БЭСМ-6, разработанной в 1965 году, реализована команда *сжатия* под именем "сборка". Ее команда "разборка" действует в противоположном направлении.]

71. Начать с $x \leftarrow y$. Выполнить (-2^k)-сдвиг x с маской $\theta_k \ll 2^k$ для $k = d-1, \ldots, 1, 0$, используя маски из упр. 70. Наконец, установить $z \leftarrow x$ (или $z \leftarrow x \mathbin{\&} \chi$, если нужен "чистый" результат).

72. Будем считать, что крайний слева бит маски, χ_{N-1}, равен нулю, поскольку это не играет никакой роли. Тогда результат $(z_{(N-1)\varphi} \ldots z_{1\varphi} z_{0\varphi})_2$ любого сбора с переворотом соответствует перестановке с $0\varphi < \cdots < k\varphi > \cdots > (N-1)\varphi$, где $k = \nu\chi$. Например, если $N = 8$ и $\chi = (00101100)_2$, результат равен $(z_0 z_1 z_4 z_6 z_7 z_5 z_3 z_2)_2$. Так что $\varphi \in \Omega$ согласно упр. 5.3.4–11 и 58, (е).

Кроме того, маски θ_0, θ_1, ..., θ_{d-1} для 1-, 2-, ..., 2^{d-1}-обмена могут быть вычислены следующим образом: перестановка $\psi = \varphi^-$ удовлетворяет соотношению $j\psi = (N-1-j)\bar{\chi}_j + s_j$, где $s_j = \chi_{j-1} + \cdots + \chi_1 + \chi_0$ подсчитывает единицы, следующие за битом χ_j маски. Пусть $\psi_0 = \psi$ и $\theta_k = (\lfloor \psi_k/2^k \rfloor \bmod 2)\ \&\ \mu_k$, где ψ_{k+1} представляет собой 2^k-обмен ψ_k с маской θ_k. (В нашем примере $s_7 \ldots s_1 s_0 = 33221000$ и $(0\bar{\chi}_7)\ldots(6\bar{\chi}_1)(7\bar{\chi}_0) = 01030067$; следовательно, $\psi_0 = (7\psi)\ldots(1\psi)(0\psi) = 33221000 + 01030067 = 34251067$. Тогда $\theta_0 = (10011001)_2\ \&\ \mu_0 = (00010001)_2$; $\psi_1 = 34521076$; $\theta_1 = (10010011)_2\ \&\ \mu_1 = (00010011)_2$; $\psi_2 = 32547610$; $\theta_2 = (00111100)_2\ \&\ \mu_2 = (00001100)_2$. В общем случае $j\psi_k \equiv j$ (по модулю 2^k).) Представим каждую перестановку ψ_k как множество d битовых векторов, а именно как "битовые срезы" $\psi_k \bmod 2$, $\lfloor \psi_k/2 \rfloor \bmod 2$, и т. д. Тогда для этого вычисления будет достаточно $O(d^2)$ битовых операций.

Операция *разброса с переворотом* (*scatter-flip*), которая обращает действие сбора с переворотом, получается с помощью той же сети с перекрещиваниями, но работающей справа налево (первым выполняется 2^{d-1}-обмен, а последним — 1-обмен).

[См. *Journal of Signal Processing Systems* **53** (2008), 145–169.]

73. (а) Эквивалентно d операций sheep-and-goats должны быть способны преобразовать слово $x^\pi = (x_{(2^d-1)\pi} \ldots x_{1\pi} x_{0\pi})_2$ в $(x_{2^d-1} \ldots x_1 x_0)_2$ для любой перестановки π множества $\{0, 1, \ldots, 2^d-1\}$. Это может быть сделано с помощью поразрядной сортировки в двоичной системе счисления (алгоритм 5.2.5R): сначала перенести биты с нечетными номерами влево, затем перенести влево биты j для нечетных $\lfloor j/2 \rfloor$ и т. д. Например, когда $d = 3$ и $x^\pi = (x_3x_1x_0x_7x_5x_2x_6x_4)_2$, три операции последовательно дают $(x_3x_1x_7x_5x_0x_2x_6x_4)_2$, $(x_3x_7x_2x_6x_1x_5x_0x_4)_2$, $(x_7x_6x_5x_4x_3x_2x_1x_0)_2$. [См. Z. Shi and R. Lee, *Proc. IEEE Conf. ASAP'00* (IEEE CS Press, 2000), 138–148.]

(б) В случае сбора с переворотом та же стратегия всегда дает $(x_{g(2^d-1)} \ldots x_{g(1)} x_{g(0)})_2$, где $g(k)$ представляет собой бинарный код Грея 7.2.1.1–(9). Так, пример из п. (а) теперь имеет вид $(x_5x_7x_1x_3x_0x_2x_6x_4)_2$, $(x_6x_2x_3x_7x_5x_1x_0x_4)_2$, $(x_4x_5x_7x_6x_2x_3x_1x_0)_2$.

74. Если $\left|\sum c_{2l} - \sum c_{2l+1}\right| = 2\Delta > 0$, мы должны забрать Δ у "богатой" половины и отдать "бедной". В бедной части имеется позиция l с $c_l = 0$; в противном случае эта половина должна дать сумму, не меньшую 2^{d-1}. Циклический 1-сдвиг, который изменяет позиции от l до $(l+t) \bmod 2^d$, делает $c'_{l+k} = c_{l+k+1}$ для $0 \le k < t$, $c'_{l+t} = c_{l+t+1} - \delta$, $c'_{l+t+1} = \delta$ и $c'_{l+k} = c_{l+k}$ для всех прочих k; здесь δ может быть любым требуемым числом в диапазоне $0 \le \delta \le c_{l+t+1}$. (В этих формулах мы рассматриваем все индексы по модулю 2^d.) Так что мы можем использовать наименьшее четное t, такое, что $c_{l+1} + c_{l+3} + \cdots + c_{l+t+1} = c_l + c_{l+2} + \cdots + c_{l+t} + \Delta + \delta$ для некоторого $\delta \ge 0$.

(1-сдвиг не обязан быть циклическим, если мы допустим сдвиг влево вместо сдвига вправо. Но свойство цикличности может потребоваться в последующих шагах.)

75. Эквивалентно, для данных индексов $0 \le i_0 < i_1 < \cdots < i_{s-1} < i_s = 2^d$ и $0 = j_0 < j_1 < \cdots < j_{s-1} < j_s = 2^d$ мы хотим отобразить $(x_{2^d-1} \ldots x_1 x_0)_2 \mapsto (x_{(2^d-1)\varphi} \ldots x_{1\varphi} x_{0\varphi})_2$, где $j\varphi = i_r$ для $j_r \le j < j_{r+1}$ и $0 \le r < s$. Если $d = 1$, это выполняет модуль отображения.

Когда $d > 1$, мы можем настроить левые перекрещивания таким образом, чтобы они маршрутизировали вход i_r в линию $i_r \oplus ((i_r + r) \bmod 2)$. Если s четно, мы рекурсивно требуем от одной из сетей $P(2^{d-1})$ внутри $P(2^d)$ решения задачи для индексов $\lfloor \{i_0, i_2, \ldots, i_s\}/2 \rfloor$ и $\lfloor \{j_0, j_2, \ldots, j_s\}/2 \rfloor$, в то время как другая решает ее для $\lfloor \{i_1, i_3, \ldots, i_{s-1}, 2^d\}/2 \rfloor$ и $\lceil \{j_0, j_2, \ldots, j_s\}/2 \rceil$. Теперь справа от $P(2^d)$ можно проверить, что, когда $j_r \le j < j_{r+1}$, модуль отображения для линий j и $j \oplus 1$ имеет вход i_r на линии j, если $j \equiv r$ (по модулю 2), в противном случае i_r находится на линии $j \oplus 1$. Аналогичное доказательство работает для нечетного s. Например, если $(i_0, \ldots, i_5) = (j_0, \ldots, j_5) = (0, 1, 3, 5, 7, 8)$, в подзадачах $i = j = (0, 1, 3, 4)$ и $(0, 2, 4)$; $x_7 \ldots x_0 \mapsto x_6x_7x_5x_4x_2x_3x_1x_0 \mapsto \cdots \mapsto x_5x_7x_5x_3x_1x_3x_1x_0 \mapsto x_7x_5x_5x_3x_3x_1x_1x_0$.

Примечания. Эта сеть представляет собой немного усовершенствованную конструкцию Ю. П. Офмана, *Труды моск. мат. общества* **14** (1965), 186–199. Можно реализовать соответствующую сеть, подставляя "δ-отображения" вместо δ-обменов; вместо (69) мы используем две маски и выполняем семь операций вместо шести: $y \leftarrow x \oplus (x \gg \delta)$, $x \leftarrow x \oplus (y \mathbin{\&} \theta) \oplus ((y \mathbin{\&} \theta') \ll \delta)$. Это расширение (71), таким образом, требует только d дополнительных единиц времени.

76. Когда отображающая сеть осуществляет перестановку, все ее модули должны действовать как перекрещивания; следовательно, $G(n) \geq \lg n!$. Офман доказал, что $G(n) \leq 2.5n \lg n$, и отметил в сноске, что константа 2.5 может быть улучшена (не приводя никаких дополнительных деталей). Мы видели, что в действительности $G(n) \leq 2n \lg n$. Заметим, что $G(3) = 3$.

77. Представим n-сеть как $(x_{2^n-1} \ldots x_1 x_0)_2$, где $x_k =$ [бинарное представление k является возможной конфигурацией нулей и единиц при применении сети ко всем 2^n последовательностям нулей и единиц], для $0 \leq k < 2^n$. Таким образом, пустая сеть представлена как $2^{2^n} - 1$, а сортирующая сеть для $n = 3$ имеет представление $(10001011)_2$. В общем случае x представляет сортирующую сеть для n элементов тогда и только тогда, когда оно представляет n-сеть и $\nu x = n+1$, тогда и только тогда, когда $x = 2^0+2^1+2^3+2^7+\cdots+2^{2^n-1}$.

Если в соответствии с этими соглашениями x представляет α, то представлением $\alpha[i{:}j]$ является $(x \oplus y) \mid (y \gg (2^{n-i} - 2^{n-j}))$, где $y = x \mathbin{\&} \bar{\mu}_{n-i} \mathbin{\&} \mu_{n-j}$.

[См. V. R. Pratt, M. O. Rabin, and L. J. Stockmeyer, *STOC* **6** (1974), 122–126.]

78. Если $k \geq \lg(m-1)$, тест корректен, поскольку мы всегда имеем $x_1 + x_2 + \cdots + x_m \geq x_1 \mid x_2 \mid \cdots \mid x_m$, причем равенство достигается тогда и только тогда, когда множества являются непересекающимися. Кроме того, мы имеем $(x_1 + \cdots + x_m) - (x_1 \mid \cdots \mid x_m) \leq (m-1)(2^{n-k-1} + \cdots + 1) < (m-1)2^{n-k} \leq 2^n$.

Обратно, если $m \geq 2^k + 2$ и $n > 2k$, тест некорректен. Например, мы можем иметь $x_1 + \cdots + x_m = (2^k+1)(2^{n-k} - 2^{n-2k-1}) + 2^{n-k-1} = 2^n + (2^{n-k} - 2^{n-2k-1})$.

Но если $n \leq 2k$, тест остается корректным при $m = 2^k + 2$, поскольку наше доказательство показывает, что в этом случае $x_1 + \cdots + x_m - (x_1 \mid \cdots \mid x_m) \leq (2^k+1)(2^{n-k}-1) < 2^n$.

79. $x_\prime = (x-1) \mathbin{\&} \chi$. (А формула $x_\prime = ((x-b-1) \mathbin{\&} a) + b$ соответствует (85).) Эти методы для x' и $x_\prime$ являются частью подпрограмм "битового колдовства" Йорга Арндта (Jörg Arndt) (2001); их происхождение неизвестно.

80. Вероятно, наиболее красивый способ состоит в том, чтобы начать с $x \leftarrow \chi - 1$ как знакового числа; затем, пока $x \geq 0$, устанавливать $x \leftarrow x \mathbin{\&} \chi$, посещать x и устанавливать $x \leftarrow 2x - \chi$. (Операция $2x - \chi$ фактически может быть выполнена одной командой `MMIX`, '`2ADDU x,x,minuschi`'.)

Но этот трюк не работает, если χ так велико, что *уже* "отрицательно". Немного более медленный, но и более общий метод начинает с $x \leftarrow \chi$ и, пока $x \neq 0$, выполняет следующее: устанавливает $t \leftarrow x \mathbin{\&} -x$, посещает $\chi - t$ и устанавливает $x \leftarrow x - t$.

81. $((z \mathbin{\&} \chi) - (z' \mathbin{\&} \chi)) \mathbin{\&} \chi$. (Один из способов проверить эту формулу — воспользоваться (18).)

82. Да, положив $z = z'$ в (86): $w \mid (z \mathbin{\&} \bar{\chi})$, где $w = ((z \mathbin{\&} \chi) + (z \mid \bar{\chi})) \mathbin{\&} \chi$.

83. (Приведенная далее итеративная процедура передает биты y вправо, в зазоры рассеянного аккумулятора t. Вспомогательные переменные u и v помечают соответственно левую и правую границы каждого зазора; они удваиваются в размере, пока не оказываются стертыми переменной w.) Установить $t \leftarrow z \mathbin{\&} \chi$, $u' \leftarrow (\chi \gg 1) \mathbin{\&} \bar{\chi}$, $v \leftarrow ((\chi \ll 1) + 1) \mathbin{\&} \bar{\chi}$, $w \leftarrow 3(u' \mathbin{\&} v)$, $u \leftarrow 3u'$, $v \leftarrow 3v$ и $k \leftarrow 1$. Затем, пока $u \neq 0$, выполнять следующие действия: $t \leftarrow t \mid ((t \gg k) \mathbin{\&} u')$, $k \leftarrow k \ll 1$, $u \leftarrow u \mathbin{\&} \overline{w}$, $v \leftarrow v \mathbin{\&} \overline{w}$, $w \leftarrow ((v \mathbin{\&} (u \gg 1) \mathbin{\&} \bar{u}) \ll (k+1)) - ((u \mathbin{\&} (v \ll 1) \mathbin{\&} \bar{v}) \gg k)$, $u' \leftarrow (u \mathbin{\&} \bar{v}) \gg k$, $v \leftarrow v + ((v \mathbin{\&} \bar{u}) \ll k)$, $u \leftarrow u + u'$. Наконец вернуть ответ $((t \gg 1) \mathbin{\&} \chi) \mid (z \mathbin{\&} \bar{\chi})$.

84. $z \leftharpoonup \chi = w - (z \mathbin{\&} \chi)$, где $w = (((z \mathbin{\&} \chi) \ll 1) + \bar{\chi}) \mathbin{\&} \chi$ появляется в ответе к упр. 82; $z \rightharpoonup \chi$ представляет собой величину t, вычисляемую (с бóльшими трудностями) в ответе к упр. 83.

85. (а) Если $x = \texttt{LOC}(a[i,j,k])$ представляет собой позицию барабана, соответствующую чередующимся, как было показано, битам, то $\texttt{LOC}(a[i+1,j,k]) = x \oplus ((x \oplus ((x \mathbin{\&} \chi) - \chi)) \mathbin{\&} \chi)$ и $\texttt{LOC}(a[i-1,j,k]) = x \oplus ((x \oplus ((x \mathbin{\&} \chi) - 1)) \mathbin{\&} \chi)$, где $\chi = (11111)_8$ согласно (84) и ответу к упр. 79. Формулы для $\texttt{LOC}(a[i,j \pm 1,k])$ и $\texttt{LOC}(a[i,j,k \pm 1])$ аналогичны, с масками 2χ и 4χ.

(б) Для произвольного доступа будем надеяться, что у нас окажется достаточно места для таблицы длиной 32, дающей $f[(i_4i_3i_2i_1i_0)_2] = (i_4i_3i_2i_1i_0)_8$. Тогда $\texttt{LOC}(a[i,j,k]) = (((f[k] \ll 1) + f[j]) \ll 1) + f[i]$. (На древних машинах битовое вычисление f было бы гораздо хуже табличного поиска, поскольку используемые операции с регистрами у них столь же медленные, как и выборка из памяти.)

(в) Пусть p — позиция страницы, находящейся в настоящее время в быстрой памяти, и пусть $z = -128$. Если при доступе к позиции x оказывается $x \mathbin{\&} z \neq p$, необходимо считать 128 слов с позиции $x \mathbin{\&} z$ барабана (после сохранения текущих данных (если они были изменены) в позицию барабана p); затем нужно установить $p \leftarrow x \mathbin{\&} z$. [См. *J. Royal Stat. Soc.* **B-16** (1954), 53–55. Эта схема размещения массива во внешней памяти была разработана около 1960 года Э. В. Дейкстрой (E. W. Dijkstra), который назвал ее методом “застежки-молнии” (“zip-fastener”). Она часто переоткрывалась, например, в 1966 году Г. М. Мортоном (G. M. Morton), а позже разработчиками квадрадеревьев (quadtrees); см. Hanan Samet, *Applications of Spatial Data Structures* (Addison–Wesley, 1990). См. также современное состояние дел в R. Raman and D. S. Wise, *IEEE Trans.* **C57** (2008), 567–573. Георг Кантор (Georg Cantor) рассмотрел чередующиеся цифры десятичных дробей в *Crelle* **84** (1878), 242–258, §7; но он заметил, что эта идея *не* приводит к простому взаимно однозначному соответствию между единичным отрезком $[0\,..\,1]$ и единичным квадратом $[0\,..\,1] \times [0\,..\,1]$.]

86. Если (p', q', r') крайних справа битов и (p'', q'', r'') прочих битов (i, j, k) находятся в части адреса, которая не влияет на номер страницы, то общее количество ошибок отсутствия страницы составляет $2\bigl((2^{p-p'} - 1)2^{q+r} + (2^{q-q'} - 1)2^{p+r} + (2^{r-r'} - 1)2^{p+q}\bigr)$. Следовательно, мы хотим минимизировать $2^{-p'} + 2^{-q'} + 2^{-r'}$ по всем неотрицательным целым числам $(p', q', r', p'', q'', r'')$, таким, что $p' + p'' \le p$, $q' + q'' \le q$, $r' + r'' \le r$, $p' + q' + r' + p'' + q'' + r'' = s$. Поскольку $2^a + 2^b > 2^{a-1} + 2^{b+1}$, когда a и b представляют собой целые числа, такие, что $a > b + 1$, минимум (для всех s) осуществляется, когда мы выбираем биты справа налево циклически до их исчерпания. Например, когда $(p, q, r) = (2, 6, 3)$, функция адресации должна иметь вид $(j_5j_4j_3k_2j_2k_1j_1i_1k_0j_0i_0)_2$. В частности, схема Точера (Tocher) является оптимальной.

[Но такое отображение не обязательно наилучшее, когда размер страниц не равен степени 2. Например, рассмотрим матрицу размером 16×16; функция адресации $(j_3i_3i_2i_1i_0j_2j_1j_0)_2$ лучше функции $(j_3i_3j_2i_2j_1i_1j_0i_0)_2$ для всех размеров страниц от 17 до 62, за исключением размера 32, для которого обе эти функции одинаково хороши.]

87. Установить $x \leftarrow x \mathbin{\&} {\sim}((x \mathbin{\&} \texttt{'@@@@@@@@'}) \gg 1)$; каждый байт $(a_7 \ldots a_0)_2$ тем самым изменяется на $(a_7a_6(a_5 \wedge \bar{a}_6)a_4 \ldots a_0)_2$. То же самое преобразование работает и для 30 дополнительных букв в дополнении “Latin-1” к ASCII (например, ? $\mapsto$?); но есть и один сбой — y $\mapsto$?.

[Дон Вудс (Don Woods) использовал этот трюк в своей игровой программе “Adventure” (1976), переводя пользовательский ввод в верхний регистр перед его поиском в словаре.]

88. Установить $z \leftarrow (x \oplus \bar{y}) \mathbin{\&} h$, затем $z \leftarrow ((x \mid h) - (y \mathbin{\&} \bar{h})) \oplus z$.

89. $t \leftarrow x \mid \bar{y}$, $t \leftarrow t \,\&\, (t \gg 1)$, $z \leftarrow (x \,\&\, \bar{y} \,\&\, \bar{\mu}_0) \mid (t \,\&\, \mu_0)$. [Из тестовой программы "`nasty`" для компилятора SWARC Г. Г. Дитца (H. G. Dietz) и Р. Д. Фишера (R. J. Fisher) (1998), оптимизированного Т. Далхаймером (T. Dahlheimer).]

90. Вставьте '$z \leftarrow z \mid ((x \oplus y) \,\&\, l)$' либо до, либо после '$z \leftarrow (x \,\&\, y) + z$'. (Порядок не имеет значения, поскольку $x + y \equiv x \oplus y$ (по модулю 4), когда $x + y$ нечетно. Следовательно, `MMIX` может выполнять округление к нечетному числу без дополнительной стоимости, используя команду `MOR`. Округление к четному числу в неоднозначных ситуациях более сложное и в случае арифметики с фиксированной точкой не имеет преимуществ.)

91. Если $\frac{1}{2}[x, y]$ обозначает среднее, как в (88), то искомый результат получается с помощью повторения следующих операций семь раз, с последующим выполнением $z \leftarrow \frac{1}{2}[x, y]$ еще один раз:

$$z \leftarrow \frac{1}{2}[x, y], \quad t \leftarrow \alpha \,\&\, h, \quad m \leftarrow (t \ll 1) - (t \gg 7),$$
$$x \leftarrow (m \,\&\, z) \mid (\overline{m} \,\&\, x), \quad y \leftarrow (\overline{m} \,\&\, z) \mid (m \,\&\, y), \quad \alpha \leftarrow \alpha \ll 1.$$

Хотя ошибки округления накапливаются на восьми уровнях, получающаяся абсолютная ошибка никогда не превышает 807/255. Более того, она составляет ≈ 1.13 при усреднении по всем 256^3 случаям, и она меньше 2 с вероятностью $\approx 94.2\%$. При округлении к нечетному, как в упр. 90, максимальная и средняя ошибки снижаются до 616/255 и ≈ 0.58; вероятность ошибки < 2 вырастает до $\approx 99.9\%$. Такое несмещенное округление используется следующим `MMIX`-кодом:

```
x GREG ;y GREG ;z GREG                  ⎧XOR  t,x,y     MOR m,ffhi,alf⎫
alf GREG ;m GREG ;t IS $255             ⎪MOR  z,rodd,t  PUT rM,m      ⎪
                      Повторить семь раз: ⎨AND  t,x,y     MUX x,z,x     ⎬,
rodd GREG #4020100804020101             ⎪ADDU z,z,t     MUX y,y,z     ⎪
ffhi GREG -1<<56                        ⎩               SLU alf,alf,1 ⎭
```

но с пропуском последнего `SLU`, а затем следует вновь повторить первые четыре команды. Общее время работы для восьми α-смешиваний (66υ) меньше стоимости восьми умножений.

92. Мы получим $z_j = \lceil (x_j + y_j)/2 \rceil$ для каждого j. (Этот факт, замеченный Г. С. Уорреном-мл. (H. S. Warren, Jr.), следует из тождества $x + y = ((x \mid y) \ll 1) - (x \oplus y)$. См. также следующее упражнение.)

93. $x - y = (x \oplus y) - ((\bar{x} \,\&\, y) \ll 1)$. ("Заемы" вместо "переносов".)

94. $(x - l)_j = (x_j - 1 - b_j) \bmod 256$, где b_j представляет собой "заем" из полей справа. Так что t_j ненулевое тогда и только тогда, когда $(x_j \ldots x_0)_{256} < (1 \ldots 1)_{256} = (256^{j+1} - 1)/255$. (Соответственно, ответами к заданным вопросам являются "да" и "нет".)

В общем случае, если константа l может иметь *любое* значение $(l_7 \ldots l_1 l_0)_{256}$, операция (90) делает $t_j \neq 0$ тогда и только тогда, когда $(x_j \ldots x_0)_{256} < (l_j \ldots l_0)_{256}$ и $x_j < 128$.

95. Использовать (90): выполнить проверку $h \,\&\, \big(t(x \oplus ((x \gg 8) + (x \ll 56))) \mid t(x \oplus ((x \gg 16) + (x \ll 48))) \mid t(x \oplus ((x \gg 24) + (x \ll 40))) \mid t(x \oplus ((x \gg 32) + (x \ll 32)))\big) = 0$, где $t(x) = (x - l) \,\&\, \bar{x}$. (Эти 28 шагов сводятся к 20, если доступен циклический сдвиг, или к 11 при использовании `MXOR` и `BDIF`.)

96. Предположим, что $0 \le x, y < 256$, $x_h = \lfloor x/128 \rfloor$, $x_l = x \bmod 128$, $y_h = \lfloor y/128 \rfloor$, $y_l = y \bmod 128$. Тогда $[x < y] = \langle \bar{x}_h y_h [x_l < y_l] \rangle$; см. упр. 7.1.1–106. А $[x_l < y_l] = [y_l + 127 - x_l \ge 128]$. Следовательно, $[x < y] = \lfloor \langle \bar{x} y z \rangle / 128 \rfloor$, где $z = (\bar{x} \,\&\, 127) + (y \,\&\, 127)$.

Отсюда следует, что $t = h \,\&\, \langle \bar{x} y z \rangle$ обладает требующимися свойствами, когда $z = (\bar{x} \,\&\, \bar{h}) + (y \,\&\, \bar{h})$. Эта формула может также быть записана как $t = h \,\&\, {\sim}\langle x \bar{y} \bar{z} \rangle$, где $\bar{z} = {\sim}((\bar{x} \,\&\, \bar{h}) + (y \,\&\, \bar{h})) = (x \mid h) - (y \,\&\, \bar{h})$ согласно (18).

Чтобы получить подобную тестовую функцию для $[x_j \le y_j] = 1 - [y_j < x_j]$, мы просто обмениваем $x \leftrightarrow y$ и берем дополнение: $t \leftarrow h \,\&\, {\sim}\langle x\bar{y}z\rangle = h \,\&\, \langle \bar{x}y\bar{z}\rangle$, где $z = (x \,\&\, \bar{h}) + (\bar{y} \,\&\, \bar{h})$.

97. Установить $x' \leftarrow x \oplus$ `'********'`, $y' \leftarrow x \oplus y$, $t \leftarrow h \,\&\, \big(x \mid ((x \mid h) - l)\big) \,\&\, \big(y' \mid ((y' \mid h) - l)\big)$, $m \leftarrow (t \ll 1) - (t \gg 7)$, $t \leftarrow t \,\&\, \big(x' \mid ((x' \mid h) - l)\big)$, $z \leftarrow (m \,\&\,$ `'********'`$) \mid (\overline{m} \,\&\, y)$. (20 шагов.)

98. Установить $u \leftarrow x \oplus y$, $z \leftarrow (\bar{x} \,\&\, \bar{h}) + (y \,\&\, \bar{h})$, $t \leftarrow h \,\&\, \big(x \oplus (u \mid (x \oplus z))\big)$, $v \leftarrow ((t \ll 1) - (t \gg 7)) \,\&\, u$, $z \leftarrow x \oplus v$, $w \leftarrow y \oplus v$. [Эта 14-шаговая процедура включает ответ к упр. 96 для вычисления $t = h \,\&\, \langle \bar{x}yz\rangle$ с использованием метода отпечатка из раздела 7.1.2 для вычисления медианы только за три шага, когда известно $x \oplus y$. Конечно, `MMIX`-решение гораздо быстрее: `BDIF t,x,y; ADDU z,y,t; SUBU w,x,t`.]

99. В этом попурри для каждого из восьми байтов решается задача иного вида; мы должны переформулировать условия так, чтобы они укладывались в общую схему: $f_0 = [x_0 \oplus$ `'!'` $\le 0]$, $f_1 = [x_1 \oplus$ `'*'` $> 0]$, $f_2 = [x_2 \le$ `'A'` $- 1]$, $f_3 = [x_3 >$ `'z'`$]$, $f_4 = [x_4 >$ `'a'` $- 1]$, $f_5 = [x_5 \oplus$ `h'` $\le 9]$, $f_6 = [x_6 \oplus 255 > 86]$, $f_7 = [x_7 \oplus$ `'?'` $\le 3]$. Ага! Мы можем использовать формулы из ответа к упр. 96, при необходимости приспосабливая d для переключения между $\le$ и $>$: $a = ($`'?'`(255)`h'000'*'''!'`$)_{256} =$ `#3fff300000002a21`; $b = \bar{h} =$ `#7f7f7f7f7f7f7f7f`; $c = \bar{h} \,\&\, {\sim}(3(86)9($`'a'` $- 1)$`'z'`$($`'A'` $- 1)00)_{256} =$ `#7c29761f053f7f7f` (самое сложное); $d =$ `#8000800000800080`; и $e = h =$ `#8080808080808080`.

100. Мы хотим получить $u_j = x_j + y_j + c_j - 10c_{j+1}$ и $v_j = x_j - y_j - b_j + 10b_{j+1}$, где c_j и b_j представляет собой "перенос" и "заем" в цифровой позиции j. Установим $u' \leftarrow (x + y + (6\ldots66)_{16}) \bmod 2^{64}$ и $v' \leftarrow (x - y) \bmod 2^{64}$. Затем найдем $u'_j = x_j + y_j + c_j + 6 - 16c_{j+1}$ и $v'_j = x_j - y_j - b_j + 16b_{j+1}$ для $0 \le j < 16$ по индукции по j. Следовательно, u' и v' имеют один и тот же шаблон переносов и заемов при работе в десятичной системе счисления, и мы имеем $u = u' - 6(\bar{c}_{16}\ldots\bar{c}_2\bar{c}_1)_{16}$, $v = v' - 6(b_{16}\ldots b_2b_1)_{16}$. Таким образом, требуемые результаты дают следующие схемы вычислений (10 операций для сложения, 9 для вычитания):

$$\begin{aligned} &y' \leftarrow y + (6\ldots66)_{16},\ u' \leftarrow x + y', && v' \leftarrow x - y,\\ &t \leftarrow \langle \bar{x}\bar{y}'u'\rangle \,\&\, (8\ldots88)_{16}, && t \leftarrow \langle \bar{x}yv'\rangle \,\&\, (8\ldots88)_{16},\\ &u \leftarrow u' - t + (t \gg 2); && v \leftarrow v' - t + (t \gg 2). \end{aligned}$$

101. Для вычитания установите $z \leftarrow x - y$; для сложения установите $z \leftarrow x + y +$ `#e8c4c4fc18`, где эта константа строится из $256 - 24 =$ `#e8`, $256 - 60 =$ `#c4` и $65536 - 1000 =$ `#fc18`. Заемы и переносы между полями осуществляются так, как если бы выполнялось вычитание или сложение в системе счисления со смешанным основанием. Остается задача коррекции для случаев, когда имелись заемы или не было переносов. Это можно легко сделать путем проверки отдельных цифр, поскольку основания счисления оказываются меньше половины размеров полей: установим $t \leftarrow z \,\&\,$ `#8080808000`, $t \leftarrow (t \ll 1) - (t \gg 7) - ((t \gg 15) \,\&\, 1)$, $z \leftarrow z - (t \,\&\,$ `#e8c4c4fc18`$)$. [См. Stephen Soule, *CACM* **18** (1975), 344–346. Нам повезло, что '`c`' в '`fc18`' четно.]

102. (a) Мы считаем, что $x = (x_{15}\ldots x_0)_{16}$ и $y = (y_{15}\ldots y_0)_{16}$ с $0 \le x_j, y_j < 5$; цель заключается в вычислении $u = (u_{15}\ldots u_0)_{16}$ и $v = (v_{15}\ldots v_0)_{16}$, с компонентами $u_j = (x_j + y_j) \bmod 5$ и $v_j = (x_j - y_j) \bmod 5$. Вот как это делается:

$$\begin{aligned} &u \leftarrow x + y, && v \leftarrow x - y + 5l,\\ &t \leftarrow (u + 3l) \,\&\, h, && t \leftarrow (v + 3l) \,\&\, h,\\ &u \leftarrow u - ((t - (t \gg 3)) \,\&\, 5l); && v \leftarrow v - ((t - (t \gg 3)) \,\&\, 5l). \end{aligned}$$

Здесь $l = (1\ldots1)_{16} = (2^{64} - 1)/15$, $h = 8l$. (Сложение выполняется за 7 операций, вычитание — за 8.)

(б) Теперь $x = (x_{20} \ldots x_0)_8$ и т. д., и мы должны быть более осторожны с ограничением переносов:

$$\begin{aligned} &t \leftarrow x + \bar{h}, \\ &z \leftarrow (t \mathbin{\&} \bar{h}) + (y \mathbin{\&} \bar{h}), \\ &t \leftarrow (y \mid z) \mathbin{\&} t \mathbin{\&} h, \\ &u \leftarrow x + y - (t + (t \gg 2)); \end{aligned} \qquad \begin{aligned} &z \leftarrow (x \mid h) - (y \mathbin{\&} \bar{h}), \\ &t \leftarrow (y \mid \bar{z}) \mathbin{\&} \bar{x} \mathbin{\&} h, \\ &v \leftarrow x - y + t + (t \gg 2). \end{aligned}$$

Здесь $h = (4 \ldots 4)_8 = (2^{65} - 4)/7$. (Сложение выполняется за 11 операций, вычитание — за 10.)

Аналогичные процедуры, конечно, работают и для других модулей. В действительности в общем случае мы можем работать с многобайтной арифметикой с использованием координат торов, с различными модулями в каждом компоненте (см. 7.2.1.3–(66)).

103. Пусть h и l представляют собой константы в (87) и (88). Сложение выполняется просто: $u \leftarrow x \mid ((x \& \bar{h}) + y)$. В случае вычитания необходимо отнять 1 и добавить $x_j \& (1 - y_j)$: $t \leftarrow (x \mathbin{\&} \bar{l}) \gg 1$, $v \leftarrow t \mid (t + (x \mathbin{\&} (y \oplus l)))$.

104. Да, это можно сделать за 19 операций. Пусть $a = (((1901 \ll 4) + 1) \ll 5) + 1$, $b = (((2099 \ll 4) + 12) \ll 5) + 28$. Установим $m \leftarrow (x \gg 5) \mathbin{\&} {}^{\#}\texttt{f}$ (месяц), $c \leftarrow {}^{\#}\texttt{10} \mathbin{\&} \sim((x \mid (x \gg 1)) \gg 5)$ (коррекция високосного года), $u \leftarrow b + {}^{\#}\texttt{3} \mathbin{\&} (({}^{\#}\texttt{3bbeecc} + c) \gg (m + m))$ (коррекция max_day) и $t \leftarrow ((x \oplus a \oplus (x - a)) \mid (x \oplus u \oplus (u - x))) \mathbin{\&} {}^{\#}\texttt{1000220}$ (проверка нежелательных переносов).

105. В упр. 98 поясняется, как вычислить битовые min и max; простая модификация будет вычислять min в одних позициях байтов, и max — в других. Таким образом, мы можем "сортировать путем идеального тасования", как в разделе 5.3.4, рис. 57, если можем переставлять соответствующим образом байты между x и y. Такая перестановка проста согласно упр. 1. [Конечно, есть гораздо более простые и быстрые пути сортировки 16 байт. Об асимптотических последствиях такого подхода можно прочесть в S. Albers and T. Hagerup, *Inf. and Computation* **136** (1997), 25–51, и в M. Thorup, *J. Algorithms* **42** (2002), 205–230.]

106. n бит рассматриваются как g полей по g бит каждое. Сначала обнаруживаются ненулевые поля (t_1), и мы образуем слово y, которое имеет $(y_{g-1} \ldots y_0)_2$ в каждом g-битовом поле, где $y_j =$ [поле j в x ненулевое]. Затем мы сравниваем каждое поле с константами $2^{g-1}, \ldots, 2^0$ (t_2) и образуем маску m, которая идентифицирует старшее ненулевое поле в x. После размещения g копий этого поля в z мы тестируем z, как тестировали y (t_3). Наконец соответствующее контрольное сложение t_2 и t_3 (по-g-битово) дает λ. (Испытайте случай $g = 4$, $n = 16$.)

Для вычисления 2^λ без сдвига влево замените '$t_2 \ll 1$' на '$t_2 + t_2$' и замените последнюю строку на $w \leftarrow (((a \cdot (t_3 \oplus (t_3 \gg g))) \bmod 2^n) \gg (n - g)) \cdot l$; тогда $w \& m$ представляет собой $2^{\lambda x}$.

107.
```
h  GREG #8000800080008000      SLU  q,t,16      OR   t,t,y
ms GREG #00ff0f0f33335555      ADDU t,t,q       AND  t,t,h
1H SRU  q,x,32                 SLU  q,t,32   5H SLU  q,t,15
   ZSNZ lam,q,32               ADDU t,t,q       ADDU t,t,q
   ADD  t,lam,16            3H ANDN y,t,ms      SLU  q,t,30
   SRU  q,x,t               4H XOR  t,t,y       ADDU t,t,q
   CSNZ lam,q,t                OR   q,y,h    6H SRU  q,t,60
2H SRU  t,x,lam                SUBU t,q,t       ADDU lam,lam,q  ▮
```

Общее время работы равно $22v$ (без обращений к памяти). [Имеется также версия (56) без обращений к памяти со стоимостью только $16v$, если заменить последнюю строку на `ADD t,lam,4; SRU y,x,t; CSNZ lam,y,t; SRU y,x,lam; SLU t,y,1; SRU t,[#ffffaa50],t; AND t,t,3; ADD lam,lam,t`.]

108. Например, пусть e представляет собой минимальное значение, для которого $n \le 2^e \cdot 2^{2^e}$. Если n кратно 2^e, мы можем использовать 2^e полей размером $n/2^e$ с e уменьшениями на шаге B1; в противном случае мы можем использовать 2^e полей размером $2^{\lceil \lg n \rceil - e - 1}$ с $e+1$ уменьшениями на шаге B1. В любом случае имеется e итераций в шагах B2 и B5, так что общее время работы составляет $O(e) = O(\log\log n)$.

109. Начните с $x \leftarrow x \mathbin{\&} -x$ и примените алгоритм B. (Шаг B4 этого алгоритма может быть немного упрощен в данном частном случае путем использования константы l вместо $x \oplus y$.)

110. Пусть $s = 2^d$, где $d = 2^e - e$. Мы будем использовать s-битовые поля в n-битовых словах.

K1. [Растяжение $x \bmod s$.] Установить $y \leftarrow x \mathbin{\&} (s-1)$. Затем установить $t \leftarrow y \mathbin{\&} \bar{\mu}_j$ и $y \leftarrow y \oplus t \oplus (t \ll 2^j(s-1))$ для $e > j \ge 0$. Наконец установить $y \leftarrow (y \ll s) - y$. [Если $x = (x_{2^e-1} \ldots x_0)_2$, мы имеем $y = (y_{2^e-1} \ldots y_0)_{2^s}$, где $y_j = (2^s - 1)x_j[j < d]$.]

K2. [Подготовка минитермов.] Установить $y \leftarrow y \oplus (a_{2^e-1} \ldots a_0)_{2^s}$, где $a_j = \mu_{d,j}$, для $0 \le j < d$ и $a_j = 2^s - 1$ для $d \le j < 2^e$.

K3. [Сжатие.] Установить $y \leftarrow y \mathbin{\&} (y \gg 2^j s)$ для $e > j \ge 0$. [Теперь $y = 1 \ll (x \bmod s)$. Это ключевой момент, делающий алгоритм работоспособным.]

K4. [Завершение.] Установить $y \leftarrow y \mid (y \ll 2^j s)$ для $0 \le j < e$. Наконец установить $y \leftarrow y \mathbin{\&} (\mu_{2^e,j} \oplus -((x \gg j) \mathbin{\&} 1))$ для $d \le j < 2^e$. ▮

111. n бит разделяются на поля по s бит, хотя крайнее слева поле может быть и короче. Сначала y устанавливается так, чтобы помечать все минус одно поле. Затем $t = (\ldots t_1 t_0)_{2^s}$ содержат биты-кандидаты для q, включая "ложные попадания" для некоторых шаблонов 01^k с $s \le k < r$. Мы всегда имеем $\nu t_j \le 1$, и из $t_j \ne 0$ вытекает $t_{j-1} = 0$. Биты u и v подразделяют t на две части так, что мы можем безопасно вычислить $m = (t \gg 1) \mid (t \gg 2) \mid \cdots \mid (t \gg r)$, перед тем как выполнить последнюю проверку для устранения ложных попаданий.

112. Заметим, что если $q = x \mathbin{\&} (x \ll 1) \mathbin{\&} \cdots \mathbin{\&} (x \ll (r-1)) \mathbin{\&} {\sim}(x \ll r)$, то мы имеем $x \mathbin{\&} \overline{x+q} = x \mathbin{\&} (x \ll 1) \mathbin{\&} \cdots \mathbin{\&} (x \ll (r-1))$.

Если мы можем решить поставленную задачу за $O(1)$ шагов, то мы можем также выделить старший бит r-битового числа за $O(1)$ шагов: для этого следует применить случай $n = 2r$ к числу $2^n - 1 - x$. И обратно, можно показать, что решение задачи выделения дает решение задачи $1^r 0$. Таким образом, из упр. 110 вытекает решение за $O(\log\log r)$ шагов.

113. Пусть $0' = 0$, $x'_0 = x_0$, и построим $x'_{i'} = x_i$ для $1 \le i \le r$ следующим образом: если $x_i = a \circ_i b$ и $\circ_i \notin \{+, -, \ll\}$, положим $i' = (i-1)' + 1$ и $x'_{i'} = a' \circ_i b'$, где $a' = x'_{j'}$, если $a = x_j$, и $a' = a$, если $a = c_i$. Если $x_i = a \ll c$, положим $i' = (i-1)' + 2$ и $(x'_{i'-1}, x'_{i'}) = (a' \mathbin{\&} (\lceil 2^{n-c} \rceil - 1), x'_{i'-1} \ll c)$. Если $x_i = a + b$, положим $i' = (i-1)' + 6$ и пусть $(x'_{(i-1)'+1}, \ldots, x'_{i'})$ вычисляет $((a' \mathbin{\&} \bar{h}) + (b' \mathbin{\&} \bar{h})) \oplus ((a' \oplus b') \mathbin{\&} h)$, где $h = 2^{n-1}$. А если $x_i = a - b$, выполняем аналогичное вычисление $((a' \mid h) - (b' \mathbin{\&} \bar{h})) \oplus ((a' \equiv b') \mathbin{\&} h)$. Ясно, что $r' \le 6r$.

114. Просто положим $X_i = X_{j(i)} \circ_i X_{k(i)}$ при $x_i = x_{j(i)} \circ_i x_{k(i)}$, $X_i = C_i \circ_i X_{k(i)}$ при $x_i = c_i \circ_i x_{k(i)}$ и $X_i = X_{j(i)} \circ_i C_i$ при $x_i = x_{j(i)} \circ_i c_i$, где $C_i = c_i$, когда c_i является величиной сдвига; в противном случае $C_i = (c_i \ldots c_i)_{2^n} = (2^{mn} - 1)c_i/(2^n - 1)$. Это построение возможно благодаря тому факту, что сдвиги переменной длины запрещены.

[Заметим, что если $m = 2^d$, мы можем воспользоваться этой идеей для моделирования 2^d экземпляров $f(x, y_i)$; тогда $O(d)$ дальнейших операций разрешают "квантификацию".]

115. (а) $z \leftarrow (\bar{x} \ll 1) \mathbin{\&} (x \ll 2)$, $y \leftarrow x \mathbin{\&} (x + z)$. [Эта задача была предложена автору Воганом Праттом (Vaughan Pratt) в 1977 году.]

(б) Сначала найдем $x_l \leftarrow (x \ll 1) \mathbin{\&} \bar{x}$ и $x_r \leftarrow x \mathbin{\&} (\bar{x} \ll 1)$, левые и правые концы блоков x, и установим $x'_r \leftarrow x_r \mathbin{\&} (x_r - 1)$. Тогда $z_e \leftarrow x'_r \mathbin{\&} (x'_r - (x_l \mathbin{\&} \bar{\mu}_0))$ и $z_o \leftarrow x'_r \mathbin{\&} (x'_r - (x_l \mathbin{\&} \mu_0))$

являются правыми концами, за которыми левый конец следует соответственно в четной или нечетной позиции. Ответ имеет вид $y \leftarrow x \mathbin{\&} (x + (z_e \mathbin{\&} \bar{\mu}_0) + (z_o \mathbin{\&} \mu_0))$; его можно упростить до $y \leftarrow x \mathbin{\&} (x + (z_e \oplus (x'_r \mathbin{\&} \mu_0)))$.

(в) В этом случае решение невозможно согласно следствию I.

116. Язык L является вполне определенным согласно лемме A (за исключением того, что наличие или отсутствие пустой строки не имеет значения). Язык является регулярным тогда и только тогда, когда он может быть определен с помощью конечного автомата, а 2-адическое целое число является рациональным тогда и только тогда, когда оно может быть определено с помощью конечного автомата, игнорирующего входные данные. Тождественная функция соответствует языку $L = 1(0 \cup 1)^*$, а простое построение будет определять автомат, который соответствует сумме, разности или булевой комбинации чисел, определенных любыми двумя заданными автоматами, работающими с последовательностью $x_0x_1x_2\ldots$. Следовательно, L является регулярным.

В упр. 115 L представляет собой (а) $11^*(000^*1(0\cup1)^*\cup0^*)$; (б) $11^*(00(00)^*1(0\cup1)^*\cup0^*)$.

117. Кстати, упомянутый язык L соответствует обратному бинарному коду Грея: он определяет функцию, обладающую тем свойством, что $f(2x) = {\sim}f(2x+1)$ и $g(f(2x)) = g(f(2x+1)) = x$, где $g(x) = x \oplus (x \gg 1)$ (см. 7.2.1.1–(9)).

118. Если $x = (x_{n-1}\ldots x_1x_0)_2$ и $0 \le a_j \le 2^j$ для $0 \le j < n$, мы имеем $\sum_{j=0}^{n-1} a_jx_j = \sum_{j=0}^{n-1}(a_j \mathbin{\dot{-}} (\bar{x} \mathbin{\&} 2^j))$. Возьмем $a_j = \lfloor 2^{j-1} \rfloor$, чтобы получить $x \gg 1$.

И обратно, приведенное далее рассуждение М. С. Патерсона (M. S. Paterson) доказывает, что монус должен использоваться как минимум $n-1$ раз. Рассмотрим любую цепочку для $f(x)$, которая использует сложение, вычитание, битовые булевы операции, и k операций "антипереполнения" $y \lhd z = (2^n-1)[y<z]$. Если $k < n-1$, должно иметься два n-битовых числа, x' и x'', таких, что $x' \bmod 2 = x'' \bmod 2 = 0$, и таких, что все k антипереполнений $\lhd$ дают один и тот же результат как для x', так и для x''. Тогда $f(x') \bmod 2^j = f(x'') \bmod 2^j$ при $j = \rho(x' \oplus x'')$. Так что $f(x)$ не является функцией $x \gg 1$.

119. $z \leftarrow x \oplus y$, $f \leftarrow 2^p \mathbin{\&} \bar{z} \mathbin{\&} (z-1)$. (См. (90).)

120. Обобщая следствие W, это функции, такие, что $f(x_1,\ldots,x_m) \equiv f(y_1,\ldots,y_m)$ (по модулю 2^k) при $x_j \equiv y_j$ (по модулю 2^k) для $1 \le j \le m$, $0 \le k \le n$. Младший бит представляет собой бинарную функцию от m переменных, так что имеется 2^{2^m} возможных вариантов. Следующий по старшинству бит является бинарной функцией от $2m$ переменных, а именно битов $(x_1 \bmod 4,\ldots,x_m \bmod 4)$, так что имеется $2^{2^{2^m}}$ вариантов; и т. д. Таким образом, окончательный ответ имеет вид $2^{2^m+2^{2m}+\cdots+2^{nm}}$.

121. (а) Если f имеет период длиной pq, где $q > 1$ нечетно, ее p-кратная итерация $f^{[p]}$ имеет период длиной q, скажем, $y_0 \mapsto y_1 \mapsto \cdots \mapsto y_q = y_0$, где $y_{j+1} = f^{[p]}(y_j)$ и $y_1 \ne y_0$. Но тогда согласно следствию W мы должны иметь $y_0 \bmod 2^{n-1} \mapsto y_1 \bmod 2^{n-1} \mapsto \cdots \mapsto y_q \bmod 2^{n-1}$ в соответствующей $(n-1)$-битовой цепочке. Следовательно, $y_1 \equiv y_0$ (по модулю 2^{n-1}) по индукции по n. Следовательно, $y_1 = y_0 \oplus 2^{n-1}$, $y_2 = y_0$ и т. д., так что мы получаем противоречие.

(б) $x_1 = x_0 + x_0$, $x_2 = x_0 \gg (p-1)$, $x_3 = x_1 \mid x_2$; период длиной p начинается со значения $x_0 = (1 + 2^p + 2^{2p} + \cdots) \bmod 2^n$.

122. Вычитание аналогично сложению; булевы операции еще проще; а константы имеют только один битовый шаблон. Единственный оставшийся случай — $x_r = x_j \gg c$, где мы имеем $S_r = S_j + c$; сдвиг направлен влево при $c < 0$. Тогда $V_{pqr} = V_{(p+c)(q+c)j}$, и

$$x_r \mathbin{\&} \lfloor 2^p - 2^q \rfloor \;=\; \big((x_j \mathbin{\&} \lfloor 2^{p+c} - 2^{q+c} \rfloor) \gg c\big) \mathbin{\&} (2^n - 1).$$

Следовательно, по индукции $|X_{pqr}| \le |X_{(p+c)(q+c)j}| \le B_j = B_r$.

123. Если $x = (x_{g-1} \ldots x_0)_2$, заметим сначала, что в (104) $t = 2^{g-1}(x_0 \ldots x_{g-1})_{2^g}$; следовательно, $y = (x_0 \ldots x_{g-1})_2$, как утверждалось. Из теоремы P теперь следует, что требуется $\lfloor \frac{1}{3} \lg g \rfloor$ широкословных шагов для умножения на a_{g+1} и на a_{g-1}. Как минимум одно из этих умножений должно требовать $\lfloor \frac{1}{6} \lg g \rfloor$ или более шагов.

124. Изначально $t \leftarrow 0$, $x_0 = x$, $U_0 = \{2^0, 2^1, \ldots, 2^{n-1}\}$ и $1' \leftarrow 0$. При продвижении $t \leftarrow t+1$, если текущей командой является $r_i \leftarrow r_j \pm r_k$, мы просто определим $x_t = x_{j'} \pm x_{k'}$ и $i' \leftarrow t$. Случаи $r_i \leftarrow r_j \circ r_k$ и $r_i \leftarrow c$ просты.

Если текущая команда выполняет ветвление при $r_i \le r_j$, определим $x_t = x_{t-1}$ и положим $V_1 = \{x \in U_{t-1} \mid x_{i'} \le x_{j'}\}$, $V_0 = U_{t-1} \setminus V_1$. Пусть U_t — большее из множеств V_0 и V_1; ветвление выполняется при $U_t = V_1$. Заметим, что в этом случае $|U_t| \ge |U_{t-1}|/2$.

Если текущей командой является $r_i \leftarrow r_j \gg r_k$, положим $W = \{x \in U_{t-1} \mid x \,\&\, \lfloor 2^{\lg n+s} - 2^s \rfloor \ne 0$ для некоторого $s \in S_{k'}\}$ и заметим, что $|W| \le |S_{k'}| \lg n \le 2^{t-1+e+f}$. Пусть $V_c = \{x \in U_{t-1} \setminus W \mid x_{k'} = c\}$ для $|c| < n$ и $V_n = U_{t-1} \setminus W \setminus \bigcup_{|c|<n} V_c$. Лемма B гласит, что не более $B_{k'} + 1 \le 2^{2^{t-1}-1} + 1$ из множеств V_c являются непустыми. Пусть наибольшим является множество U_t; и если это V_c, определим $x_t = x_{j'} \gg c$, $i' \leftarrow t$. В этом случае $|U_t| \ge (|U_{t-1}| - 2^{t-1+e+f})/(2^{2^{t-1}-1} + 1)$.

Аналогично для $r_i \leftarrow M[r_j \bmod 2^m]$ или $M[r_j \bmod 2^m] \leftarrow r_i$ положим $W = \{x \in U_{t-1} \mid x \,\&\, \lfloor 2^{m+s} - 2^s \rfloor \ne 0$ для некоторого $s \in S_{j'}\}$ и $V_z = \{x \in U_{t-1} \setminus W \mid x_{j'} \bmod 2^m = z\}$ для $0 \le z < 2^m$. Согласно лемме B не более $B_{j'} \le 2^{2^{t-1}-1}$ множеств V_z являются непустыми; пусть наибольшим оказывается $U_t = V_z$. Чтобы записать r_i в $M[z]$, определим $x_t = x_{t-1}$, $z'' \leftarrow i'$; для чтения r_i из $M[z]$ установим $i' \leftarrow t$ и положим $x_t = x_{z''}$, если z'' определено; в противном случае положим x_t равным предвычисленной константе $M[z]$. В обоих случаях $|U_t| \ge (|U_{t-1}| - 2^{t-1}m)/2^{2^{t-1}-1}$ достаточно велико.

Если $t < f$, мы не можем быть уверены, что $r_1 = \rho x$. Причина заключается в том, что множество $W = \{x \in U_t \mid x \,\&\, \lfloor 2^{\lg n+s} - 2^s \rfloor \ne 0$ для некоторого $s \in S_{1'}\}$ имеет размер $|W| \le |S_{1'}| \lg n \le 2^{t+e+f}$ и $|U_t \setminus W| \ge 2^{2^{e+f}-2^t+1} - 2^{t+e+f} > 2^{2^t-1} \ge |\{x_{1'} \,\&\, \lfloor 2^{\lg n} - 1 \rfloor \mid x_0 \in U_t \setminus W\}|$. Два элемента $U_t \setminus W$ не могут иметь одно и то же значение $\rho x = x_{1'} \,\&\, \lfloor 2^{\lg n} - 1 \rfloor$.

[Та же нижняя граница применима, даже если мы допустим произвольные $2^{2^{t-1}}$-вариантные ветвления, основанные на содержимом $(r_1, \ldots, r_l)$ в момент t.]

125. Начнем, как в ответе к упр. 124, но с $U_0 = [0\,..\,2^g)$. Упрощение рассуждений путем удаления множеств W приведет к множествам, таким, что $|U_t| \ge 2^g/\max(2^m, 2n)^t$; например, может иметься не более $2n$ различных команд сдвига.

Предположим, мы можем остановиться в момент t, такой, что $t < \lfloor \lg(h+1) \rfloor$. Доказательство теоремы P дает p и q, обладающие тем свойством, что $x^R \,\&\, \lfloor 2^p - 2^q \rfloor$ не зависит от $x \,\&\, \lfloor 2^{p+s} - 2^{q+s} \rfloor$. Следовательно, расширение леммы B из указания показывает, что x^R принимает не более $2^{2^t-1} \le 2^{(h-1)/2}$ различных значений для каждого набора прочих битов $\{x \,\&\, \lfloor 2^{p+s} - 2^{q+s} \rfloor \mid s \in S_t\}$. Следовательно, $r_1 = x_{1'}$ сможет быть корректным значением x^R для не более чем $2^{(h-1)/2+g-h}$ значений x. Но $2^{(h-1)/2+g-h}$ меньше, чем $|U_t|$, согласно (106).

126. М. С. Патерсон (M. S. Paterson) выдвинул связанную (но иную) гипотезу: для каждой 2-адической цепочки с k операциями сложения-вычитания имеется (возможно, огромное) целое число x с $\nu x = k+1$, такое, что цепочка не вычисляет $2^{\lambda x}$.

127. Йохан Хастад (Johan Hastad) [*Advances in Computing Research* **5** (1989), 143–170] показал, что любая схема полиномиального размера, вычисляющая функцию четности для входных данных $\{x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n\}$ с использованием логических элементов И и ИЛИ с бесконечным коэффициентом объединения по входу, должна иметь глубину $\Omega(\log n/\log\log n)$.

128. (Заметим также, что суффиксная функция четности $x^{\oplus}$ рассматривалась в упр. 36 и 66.)

130. Если ответ "нет", сам собой встает аналогичный вопрос о *переменной* a.

131. Эта программа выполняет типичный "поиск в ширину", сохраняя `LINK(q) = r`. Регистр `u` представляет собой вершину, исследуемую в настоящий момент; `v` является одним из ее преемников.

```
0H LDOU r,q,link   1   r ← LINK(q).      STOU v,q,link   |R|−|Q|   LINK(q) ← v.
   SET  u,r        1   u ← r.            STOU r,v,link   |R|−|Q|   LINK(v) ← r.
1H LDOU a,u,arcs   |R|  a ← ARCS(u).     SET  q,v        |R|−|Q|   q ← v.
   BZ   a,4F       |R|  S[u] = ∅?     3H PBNZ a,2B        S       Цикл по a.
2H LDOU v,a,tip    S   v ← TIP(a).    4H LDOU u,u,link     |R|     u ← LINK(u).
   LDOU a,a,next   S   a ← NEXT(a).      CMPU t,u,r        |R|     u ≠ r?
   LDOU t,v,link   S   t ← LINK(v).      PBNZ t,1B         |R|     Да — продолжить.
   PBNZ t,3F       S   Is v ∈ R?                                    ▮
```

132. (а) Мы всегда имеем $\tau(U) \subseteq \&_{u \notin U}\, \delta_u = \sigma(U)$. Равенство выполняется тогда и только тогда, когда $2^u \subseteq \rho(u')$ для всех $u \in U$ и $u' \in U$.

(б) Мы доказали, что $\tau(U) \subseteq \sigma(U)$; следовательно, $T \subseteq U$. А если $t \in T$, мы имеем $2^t \subseteq \rho_u$ для всех $u \in U$. Следовательно, $\sigma(T) \subseteq \tau(T)$.

(в) В пп. (а) и (б) доказано, что элементы C_n представляют клики.

(г) Если $u \subseteq v$, то $u \,\&\, \rho_k \subseteq v \,\&\, \rho_k$ и $u \,\&\, \delta_k \subseteq v \,\&\, \delta_k$; так что мы можем работать только с максимальными записями. Приведенный далее алгоритм использует кэширующее последовательное (а не связанное) распределение, некоторым образом аналогичное обменной поразрядной сортировке (алгоритм 5.2.2R).

Мы считаем, что $w_1 \ldots w_s$ представляет собой рабочее пространство из s беззнаковых слов, ограниченное $w_0 = 0$ и $w_{s+1} = 2^n - 1$. Элементы C^+_{k-1} изначально находятся в позициях $w_1 \ldots w_m$, и наша цель состоит в том, чтобы земенить их элементами C^+_k.

M1. [Инициализация.] Завершить работу алгоритма, если $\rho_k = 2^n - 1$. В противном случае установить $v \leftarrow 2^k$, $i \leftarrow 1$, $j \leftarrow m$.

M2. [Разбиение на v.] Пока $w_i \,\&\, v = 0$, устанавливать $i \leftarrow i + 1$. Пока $w_j \,\&\, v \neq 0$, устанавливать $j \leftarrow j - 1$. Затем, если $i > j$, перейти к шагу M3; в противном случае обменять $w_i \leftrightarrow w_j$, установить $i \leftarrow i + 1$, $j \leftarrow j - 1$ и повторить этот шаг.

M3. [Разделение $w_i \ldots w_m$.] Установить $l \leftarrow j$, $p \leftarrow s + 1$. Пока $i \leq m$, выполнять подпрограмму Q с $u = w_i$ и устанавливать $i \leftarrow i + 1$.

M4. [Комбинация максимальных элементов.] Установить $m \leftarrow l$. Пока $p \leq s$, устанавливать $m \leftarrow m + 1$, $w_m \leftarrow w_p$ и $p \leftarrow p + 1$. ▮

Подпрограмма Q использует глобальные переменные j, k, l, p и v. Она, по сути, заменяет слово u на $u' = u \,\&\, \rho_k$ и $u'' = u \,\&\, \delta_k$, оставляя их, если они максимальны. Если это так, u' переходит в верхнее рабочее пространство $w_p \ldots w_s$, но u'' остается внизу.

Q1. [Исследование u'.] Установить $w \leftarrow u \,\&\, \rho_k$ и $q \leftarrow s$. Если $w = u$, перейти к шагу Q4.

Q2. [Сравнимо?] Если $q < p$, перейти к шагу Q3. В противном случае если $w \& w_q = w$, перейти к шагу Q7. В противном случае если $w \,\&\, w_q = w_q$, перейти к шагу Q4. В противном случае установить $q \leftarrow q - 1$ и повторить Q2.

Q3. [Предположительное принятие u'.] Установить $p \leftarrow p - 1$ и $w_p \leftarrow w$. Если $p \leq m+1$, происходит переполнение памяти. В противном случае перейти к шагу Q7.

Q4. [Подготовка к циклу.] Установить $r \leftarrow p$ и $w_{p-1} \leftarrow 0$.

Q5. [Удаление не максимальных элементов.] Пока $w \,|\, w_q \neq w$, устанавливать $q \leftarrow q-1$. Пока $w \mid w_r = w$, устанавливать $r \leftarrow r + 1$. Затем, если $q < r$, перейти к шагу Q6;

в противном случае установить $w_q \leftarrow w_r$, $w_r \leftarrow 0$, $q \leftarrow q-1$, $r \leftarrow r+1$ и повторить этот шаг.

Q6. [Сброс p.] Установить $w_q \leftarrow w$ и $p \leftarrow q$. Завершить работу подпрограммы, если $w = u$.

Q7. [Исследование u''.] Установить $w \leftarrow u \mathbin{\&} \bar{v}$. Если $w = w_q$ для некоторого q в диапазоне $1 \le q \le j$, не делать ничего. В противном случае установить $l \leftarrow l+1$ и $w_l \leftarrow w$. ▌

На практике этот алгоритм работает достаточно эффективно; например, будучи примененным к графу ферзя размером 8×8 (упр. 7–129), он находит 310 максимальных клик после 306 513 обращений к памяти с использованием 397 слов рабочего пространства. Он находит 10 188 максимальных независимых множеств того же графа после примерно 310 миллионов обращений к памяти с использованием 15 090 слов; имеется соответственно $(728, 6912, 2456, 92)$ таких множеств размером $(5, 6, 7, 8)$, включая 92 знаменитых решения задачи о восьми ферзях.

Источник. N. Jardine and R. Sibson, *Mathematical Taxonomy* (Wiley, 1971), Appendix 5. Были опубликованы и многие другие алгоритмы для перечисления максимальных клик. См., например, W. Knödel, *Computing* **3** (1968), 239–240, **4** (1969), 75; C. Bron and J. Kerbosch, *CACM* **16** (1973), 575–577; S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, *SICOMP* **6** (1977), 505–517; E. Loukakis, *Computers and Math. with Appl.* **9** (1983), 583–589; D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou, *Inf. Proc. Letters* **27** (1988), 119–123. См. также упр. 5–23.

133. (а) Независимое множество представляет собой клику $\overline{G}$; так что мы дополняем G. (б) Вершинное покрытие является дополнением независимого множества; так что мы дополняем G, а затем дополняем полученный результат.

134. $a \mapsto 00$, $b \mapsto 01$, $c \mapsto 11$ представляет собой первое отображение класса II.

135. Унарные операторы просты: $\neg(x_l x_r) = \bar{x}_r \bar{x}_l$; $\diamond(x_l x_r) = x_r x_r$; $\square(x_l x_r) = x_l x_l$. А $x_l x_r \Leftrightarrow y_l y_r = (z_l \wedge z_r)(z_l \vee z_r)$, где $z_l = (x_l \equiv y_l)$ и $z_r = (x_r \equiv y_r)$.

136. (а) Классы II, III, IV_a и IV_c имеют оптимальную стоимость 4. Любопытно, что функции $z_l = x_l \vee y_l \vee (x_r \wedge y_r)$, $z_r = x_r \vee y_r$ одинаково подходят как для отображения $(a, b, c) \mapsto (00, 01, 11)$ из класса II, так и для отображения $(a, b, c) \mapsto (00, 01, 1*)$ из класса IV_c. [Эта операция эквивалентна насыщающему сложению, когда $a = 0$, $b = 1$, а c означает "больше 1."]

(б) Из симметрии между a, b и c вытекает, что мы должны проверить только классы I, IV_a и V_a; и оказывается, что эти классы имеют стоимости 6, 7 и 8. Победителем в классе I, с $(a, b, c) \mapsto (00, 01, 10)$, является $z_l = v_r \wedge \bar{u}_l$, $z_r = v_l \wedge \bar{u}_r$, где $u_l = x_l \oplus y_l$, $u_r = x_r \oplus y_r$, $v_l = y_r \oplus u_l$ и $v_r = y_l \oplus u_r$. [См. упр. 7.1.2–60, которое дает тот же ответ, но с $z_l \leftrightarrow z_r$. Причина этого в том, что в данной задаче мы имеем $(x + y + z) \bmod 3 = 0$, но в указанной задаче $(x + y - z) \bmod 3 = 0$; а $z_l \leftrightarrow z_r$ эквивалентно смене знака. Бинарная операция $z = x \circ y$ в этом случае также может быть охарактеризована тем фактом, что все элементы (x, y, z) одинаковы или все различны; таким образом, это знакомо игрокам в игру SET. Это единственная бинарная операция над n-элементными множествами, которая имеет $n!$ автоморфизмов и отличается от тривиальных примеров $x \circ y = x$ или $x \circ y = y$.]

(в) Стоимость 3 достигается только с классом I: положим $(a, b, c) \mapsto (00, 01, 10)$ и $z_l = (x_l \vee x_r) \wedge y_l$, $z_r = \bar{x}_r \wedge y_r$.

137. Действительно, $z = (x+1) \mathbin{\&} y$ при $(a, b, c) \mapsto (00, 01, 10)$. [Это искусственный пример.]

138. Простейший случай, известный автору, требует вычисления *двух* бинарных операций, таких как

$$\begin{pmatrix} a & b & b \\ a & b & b \\ c & a & a \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} a & b & a \\ a & b & a \\ c & a & c \end{pmatrix};$$

каждая имеет стоимость 2 в классе V_a, но при этом в классах I и II их стоимость составляет $(3,2)$ и $(2,3)$.

139. Вычисление z_2, по сути, эквивалентно упр. 136, (б); так что победителем оказывается естественное представление (111). К счастью, это представление также хорошо подходит для z_1, с $z_{1l} = x_l \wedge y_l$, $z_{1r} = x_r \wedge y_r$.

140. С представлением (111) сначала используйте полные бинарные сумматоры для вычисления $(a_1a_0)_2 = x_l + y_l + z_l$ и $(b_1b_0)_2 = x_r + y_r + z_r$ за $5+5=10$ шагов. После этого метод "жадных отпечатков" показывает, как вычислить четыре требуемые функции от (a_1, a_0, b_1, b_0) за еще восемь шагов: $u_l = a_1 \wedge \bar{b}_0$, $u_r = a_0 \wedge \bar{b}_1$; $t_1 = a_1 \oplus b_0$, $t_2 = a_0 \oplus b_1$, $t_3 = a_1 \oplus t_2$, $t_4 = a_0 \oplus t_1$, $v_l = t_3 \wedge \bar{t}_1$, $v_r = t_4 \wedge \bar{t}_2$. [Является ли этот метод наилучшим?]

141. Предположим, что мы вычисляем биты $a = a_0a_1\ldots a_{2m-1}$ и $b = b_0b_1\ldots b_{2m-1}$, такие, что

$$\begin{aligned} a_s &= [s=1 \text{ или } s=2, \text{ или } s \text{ представляет собой сумму различных чисел Улама} \le m \\ &\qquad \text{ровно одним способом}], \\ b_s &= [s \text{ представляет собой сумму различных чисел Улама} \le m \\ &\qquad \text{более чем одним способом}], \end{aligned}$$

для некоторого целого числа $m = U_n \ge 2$. Например, когда $m = n = 2$, мы имеем $a = 0111$ и $b = 0000$. Тогда $\{s \mid s \le m \text{ и } a_s = 1\} = \{U_1, \ldots, U_n\}$; а $U_{n+1} = \min\{s \mid s > m \text{ и } a_s = 1\}$. (Заметим, что $a_s = 1$ при $s = U_{n-1}+U_n$.) Следующие простые битовые операции сохраняют указанные условия: $n \leftarrow n+1$, $m \leftarrow U_n$ и

$$\begin{aligned} (a_m\ldots a_{2m-1}, b_m\ldots b_{2m-1}) \leftarrow ((a_m\ldots a_{2m-1} \oplus a_0\ldots a_{m-1}) \mathbin{\&} \overline{b_m\ldots b_{2m-1}}, \\ (a_m\ldots a_{2m-1} \mathbin{\&} a_0\ldots a_{m-1}) \mid b_m\ldots b_{2m-1}), \end{aligned}$$

где $a_s = b_s = 0$ для $2U_{n-1} \le s < 2U_n$ в правой части этого присваивания.

[См. M. C. Wunderlich, *BIT* **11** (1971), 217–224; *Computers in Number Theory* (1971), 249–257. Эти загадочные числа, впервые определенные С. Уламом (S. Ulam) в *SIAM Review* **6** (1964), 348, озадачили специалистов в области теории чисел на многие годы. Отношение U_n/n выглядит сходящимся к константе, ≈ 13.52; например, $U_{20000000} = 270\,371\,127$ и $U_{40000000} = 540\,752\,349$. Кроме того, Д. У. Вильсон (D. W. Wilson) эмпирически обнаружил, что эти числа образуют квазипериодические "кластеры", центры которых отличаются на кратное другой константе, ≈ 21.6016. Вычисления Джада Мак-Крейни (Jud McCranie) и автора для $U_n < 640\,000\,000$ указывают, что наибольший промежуток $U_n - U_{n-1}$ наблюдался между $U_{24576523} = 332\,250\,401$ и $U_{24576524} = 332\,251\,032$; а наименьший промежуток $U_n - U_{n-1} = 1$, вероятно, имеется только при $U_n \in \{2, 3, 4, 48\}$. Некоторые небольшие промежутки наподобие 6, 11, 14 и 16 не наблюдались ни разу.]

142. Алгоритм E в этом упражнении выполняет следующие операции над подкубами: (i) подсчитывает $*$ в данном подкубе c; (ii) для заданных c и c' проверяет выполнение условия $c \subseteq c'$; (iii) для заданных c и c' вычисляет $c \sqcup c'$ (если таковой существует). Операция (i) с помощью контрольного сложения выполняется просто; давайте посмотрим, какие из девяти классов двухбитных кодировок (119), (123), (124) лучше всего подходят для (ii) и (iii). Предположим, что $a = 0$, $b = 1$, $c = *$; симметрия между 0 и 1 означает, что нужно проверить только классы I, III, IV_a, IV_c, V_a и V_c.

Для отображения $(0,1,*) \mapsto (00,01,10)$, принадлежащего классу I, таблица истинности для $c \not\subseteq c'$ представляет собой 010*100*110***** в каждом компоненте. (Например, $0 \subseteq *$ и $* \not\subseteq 1$. В этой таблице истинности звездочки $*$ представляют собой безразличные значения для неиспользуемого кода 11.) Методы из раздела 7.1.2 указывают, что самые дешевые такие функции имеют стоимость 3; например, $c \subseteq c'$ тогда и только тогда, когда $((b \oplus b') \mid a) \,\&\, \bar{a}' = 0$. Кроме того консенсус $c \sqcup c' = c''$ существует тогда и только тогда, когда $\nu z = 1$, где $z = (b \oplus b') \,\&\, {\sim}(a \oplus a')$. И в этом случае $a'' = (a \oplus b \oplus b') \,\&\, {\sim}(a \oplus a')$, $b'' = (b \mid b') \,\&\, \bar{z}$. [Звездочка и битовые коды для этой цели были использованы M. A. Бройером (M. A. Breuer) в *Proc. ACM Nat. Conf.* **23** (1968), 241–250.]

Но класс III работает лучше при $(0,1,*) \mapsto (01,10,00)$. В этом случае $c \subseteq c'$ тогда и только тогда, когда $(\bar{c}_l \,\&\, c'_l) \mid (\bar{c}_r \,\&\, c'_r) = 0$; $c \sqcup c' = c''$ существует тогда и только тогда, когда $\nu z = 1$, где $z = x \,\&\, y$, $x = c_l \mid c'_l$, $y = c_r \mid c'_r$; а $c''_l = x \oplus z$, $c''_r = y \oplus z$. Мы экономим две операции для каждого консенсуса по сравнению с классом I, компенсируя лишний шаг при подсчете звездочек.

Классы IV_a, V_a и V_c оказываются далеко позади. Класс IV_c имеет некоторые достоинства, но класс III наилучший.

143. $f(x) = ((x \,\&\, m_1) \ll 17) \mid ((x \gg 17) \,\&\, m_1) \mid ((x \,\&\, m_2) \ll 15) \mid ((x \gg 15) \,\&\, m_2) \mid ((x \,\&\, m_3) \ll 10) \mid ((x \gg 10) \,\&\, m_3) \mid ((x \,\&\, m_4) \ll 6) \mid ((x \gg 6) \,\&\, m_4)$, где $m_1 = {}^{\#}\texttt{7f7f7f7f7f7f}$, $m_2 = {}^{\#}\texttt{fefefefefefe}$, $m_3 = {}^{\#}\texttt{3f3f3f3f3f3f}$, $m_4 = {}^{\#}\texttt{fcfcfcfcfcfcfc}$. [См., например, *Chess Skill in Man and Machine*, ed. by Peter W. Frey (1977), с. 59. Для вычисления $f(x)$ на машине MMIX достаточно пяти шагов (четырех операций MOR и одной OR), поскольку $f(x) = q \cdot x \cdot q' \mid q' \cdot x \cdot q$ с $q = {}^{\#}\texttt{40a05028140a0502}$ и $q' = {}^{\#}\texttt{2010884422110804}$.]

144. Узел $j \oplus (k \ll 1)$, где $k = j \,\&\, {-j}$.

145. Это предок листа $j \mid 1$ на высоте h.

146. Согласно (136) нам надо показать, что $\lambda(j \,\&\, {-i}) = \rho l$ при $l - 2^{\rho l} < i \le l \le j < l + 2^{\rho l}$. Искомый результат следует из (35), поскольку $-l \le -i < -l + 2^{\rho l}$.

147. (а) $\pi v_j = \beta v_j = j$, $\alpha v_j = 1 \ll \rho j$ и $\tau j = \Lambda$ для $1 \le j \le n$.

(б) Предположим, что $n = 2^{e_1} + \cdots + 2^{e_t}$, где $e_1 > \cdots > e_t \ge 0$, и пусть $n_k = 2^{e_1} + \cdots + 2^{e_k}$ для $0 \le k \le t$. Тогда $\pi v_j = j$ и $\beta v_j = \alpha v_j = n_k$ для $n_{k-1} < j \le n_k$. Кроме того, $\tau n_k = v_{n_{k-1}}$ для $1 \le k \le t$, где $v_0 = \Lambda$; все прочие $\tau j = \Lambda$.

148. Да, если $\pi y_1 = 010000$, $\pi y_2 = 010100$, $\pi x_1 = 010101$, $\pi x_2 = 010110$, $\pi x_3 = 010111$, $\beta x_3 = 010111$, $\beta y_2 = 010100$, $\beta x_2 = 011000$, $\beta y_1 = 010000$ и $\beta x_1 = 100000$.

149. Мы считаем, что изначально $\texttt{CHILD}(v) = \texttt{SIB}(v) = \texttt{PARENT}(v) = \Lambda$ для всех вершин v (включая $v = \Lambda$) и что имеется как минимум одна ненулевая вершина.

S1. [Создание трижды связанного дерева.] Для каждой из n дуг $u \longrightarrow v$ (возможно, $v = \Lambda$) установить $\texttt{SIB}(u) \leftarrow \texttt{CHILD}(v)$, $\texttt{CHILD}(v) \leftarrow u$, $\texttt{PARENT}(u) \leftarrow v$. (См. упр. 2.3.3–6.)

S2. [Начало первого обхода.] Установить $p \leftarrow \texttt{CHILD}(\Lambda)$, $n \leftarrow 0$ и $\lambda 0 \leftarrow -1$.

S3. [Вычисление β в простом случае.] Установить $n \leftarrow n + 1$, $\pi p \leftarrow n$, $\tau n \leftarrow \Lambda$ и $\lambda n \leftarrow 1 + \lambda(n \gg 1)$. Если $\texttt{CHILD}(p) \ne \Lambda$, установить $p \leftarrow \texttt{CHILD}(p)$ и повторить данный шаг; в противном случае установить $\beta p \leftarrow n$.

S4. [Восходящее вычисление τ.] Установить $\tau\beta p \leftarrow \texttt{PARENT}(p)$. Затем, если $\texttt{SIB}(p) \ne \Lambda$, установить $p \leftarrow \texttt{SIB}(p)$ и вернуться к шагу S3; в противном случае установить $p \leftarrow \texttt{PARENT}(p)$.

S5. [Вычисление β в трудном случае.] Если $p \ne \Lambda$, установить $h \leftarrow \lambda(n \,\&\, {-\pi p})$, затем $\beta p \leftarrow ((n \gg h) \mid 1) \ll h$ и вернуться к шагу S4.

S6. [Начало второго обхода.] Установить $p \leftarrow \texttt{CHILD}(\Lambda), \lambda 0 \leftarrow \lambda n$, $\pi\Lambda \leftarrow \beta\Lambda \leftarrow \alpha\Lambda \leftarrow 0$.

S7. [Нисходящее вычисление α.] Установить $\alpha p \leftarrow \alpha(\texttt{PARENT}(p)) \mid (\beta p \,\&\, -\beta p)$. Затем, если $\texttt{CHILD}(p) \neq \Lambda$, установить $p \leftarrow \texttt{CHILD}(p)$ и повторить этот шаг.

S8. [Продолжение обхода.] Если $\texttt{SIB}(p) \neq \Lambda$, установить $p \leftarrow \texttt{SIB}(p)$ и перейти к шагу S7. В противном случае установить $p \leftarrow \texttt{PARENT}(p)$ и повторить шаг S8, если $p \neq \Lambda$. ▮

150. Можно считать, что элементы A_j различны, рассматривая их как упорядоченные пары (A_j, j). Упомянутое в указании бинарное дерево поиска, являющееся частным случаем "декартовых деревьев", введенных Жаном Вюллемином (Jean Vuillemin) [*CACM* **23** (1980), 229–239], обладает тем свойством, что $k(i,j)$ является ближайшим общим предком i и j. Действительно, предками любого данного узла j являются те узлы k, для которых A_k является право-левым минимумом для $A_1 \ldots A_j$ либо A_k является лево-правым минимумом для $A_j \ldots A_n$.

Алгоритм из ответа к предыдущему упражнению выполняет требуемые предвычисления, за исключением того, что нам надо иначе настроить трижды связанное дерево на узлах $\{0, 1, \ldots, n\}$. Начнем, как и ранее, с $\texttt{CHILD}(v) = \texttt{SIB}(v) = \texttt{PARENT}(v) = 0$ для $0 \le v \le n$, и пусть $\Lambda = 0$. Будем считать, что $A_0 \le A_j$ для $1 \le j \le n$. Установим $t \leftarrow 0$ и выполним следующие шаги для $v = n$, $n-1$, $\ldots$, 1: установить $u \leftarrow 0$; затем, пока $A_v < A_t$, устанавливать $u \leftarrow t$ и $t \leftarrow \texttt{PARENT}(t)$. Если $u \neq 0$, установить $\texttt{SIB}(v) \leftarrow \texttt{SIB}(u)$, $\texttt{SIB}(u) \leftarrow 0$, $\texttt{PARENT}(u) \leftarrow v$, $\texttt{CHILD}(v) \leftarrow u$; в противном случае просто установить $\texttt{SIB}(v) \leftarrow \texttt{CHILD}(t)$. Установить также $\texttt{CHILD}(t) \leftarrow v$, $\texttt{PARENT}(v) \leftarrow t$, $t \leftarrow v$.

После того как дерево построено, продолжаем с шага S2. Время работы составляет $O(n)$, поскольку операция $t \leftarrow \texttt{PARENT}(t)$ выполняется не более одного раза для каждого узла t. [Этот красивый способ приведения задачи поиска минимума на отрезке к задаче поиска ближайшего общего предка был открыт Г. Н. Габовым (H. N. Gabow), Й. Л. Бентли (J. L. Bentley) и Р. Э. Таржаном (R. E. Tarjan), *STOC* **16** (1984), 137–138, которые также предложили следующее упражнение.]

151. Для узла v с k дочерними узлами u_1, $\ldots$, u_k определим последовательность узлов $S(v) = v$, если $k = 0$; $S(v) = vS(u_1)$, если $k = 1$; и $S(v) = S(u_1)v \ldots vS(u_k)$, если $k > 1$. (Следовательно, v появляется в $S(v)$ ровно $\max(k-1, 1)$ раз.) Если в лесу имеется k деревьев с корнями u_1, $\ldots$, u_k, запишем последовательность узлов $S(u_1)\Lambda \ldots \Lambda S(u_k) = V_1 \ldots V_N$. (Длина этой последовательности будет удовлетворять соотношению $n \le N < 2n$.) Пусть A_j является глубиной узла V_j для $1 \le j \le N$, где Λ имеет глубину 0. (Например, рассмотрим лес (141), но добавим к нему еще один дочерний узел $K \longrightarrow D$ и изолированный узел L. Тогда $V_1 \ldots V_{15} = CFAGJDHDK\Lambda BEI\Lambda L$ и $A_1 \ldots A_{15} = 231342323012301$.) Ближайшим общим предком u и v, при $u = V_i$ и $v = V_j$ является, таким образом, $V_{k(i,j)}$ из задачи о поиске минимума на отрезке. [См. J. Fischer and V. Heun, *Lecture Notes in Comp. Sci.* **4009** (2006), 36–48.]

152. На шаге V1 выполняется поиск уровня, выше которого αx и αy имеют биты, применимые к обоим их предкам. (См. упр. 148.) На шаге V2 при необходимости h увеличивается до уровня, где у них имеется общий предок, или до наивысшего уровня λn, если такового нет (а именно если $k = 0$). Если $\beta x \neq \beta z$, на шаге V4 выполняется поиск наивысшего уровня среди предков x, ведущих на уровень h; следовательно, становится известен наинизший предок $\hat{x}$, для которого $\beta\hat{x} = \beta z$ (или $\hat{x} = \Lambda$). Наконец на шаге V5 прямой порядок обхода говорит нам, кто из $\hat{x}$ и $\hat{y}$ является предком другого узла.

153. Этот указатель имеет ρj бит, так что он заканчивается после $\rho 1 + \rho 2 + \cdots + \rho j = j - \nu j$ бит упакованной строки согласно (61). [Здесь j четно. Навигационные стопки впервые появились в *Nordic Journal of Computing* **10** (2003), 238–262.]

154. Серые линии определяют треугольники с углами 36°, 36° и 90°, десять таких треугольников образуют пятиугольник с углами 72° в каждой вершине. Эти пятиугольники заполняют гиперболическую плоскость таким образом, что в каждой вершине их встречается ровно *пять*.

155. Сначала заметим, что $0 \le (\alpha 0)_{1/\phi} < \phi^{-1} + \phi^{-3} + \phi^{-5} + \cdots = 1$, поскольку отсутствуют следующие друг за другом единицы. Далее заметим, что $F_{-n}\phi \equiv \phi^{-n}$ (по модулю 1) согласно упр. 1.2.8–11. Теперь добавим $F_{k_1}\phi + \cdots + F_{k_r}\phi$. Например, $(4\phi) \bmod 1 = \phi^{-5} + \phi^{-2}$; $(-2\phi) \bmod 1 = \phi^{-4} + \phi^{-1}$.

Эти рассуждения доказывают также интересную формулу $\lfloor N(\alpha)\phi \rfloor = -N(\alpha 0)$.

156. (а) Начнем с $y \leftarrow 0$ и с k достаточно большого, чтобы $|x| < F_{k+1}$. Если $x < 0$, установить $k \leftarrow (k-1) \mid 1$, и, пока $x + F_k > 0$, устанавливать $k \leftarrow k-2$; затем установить $y \leftarrow y + (1 \ll k)$, $x \leftarrow x + F_{k+1}$; повторить. В противном случае, если $x > 1$, установить $k \leftarrow k \,\&\, -2$ и, пока $x - F_k \le 0$, устанавливать $k \leftarrow k-2$; затем установить $y \leftarrow y + (1 \ll k)$, $x \leftarrow x - F_{k+1}$; повторить. В противном случае установить $y \leftarrow y + x$ и завершить работу алгоритма с $y = (\alpha)_2$.

(б) Операции $x_1 \leftarrow a_1$, $y_1 \leftarrow -a_1$, $x_k \leftarrow y_{k-1} + a_k$, $y_k \leftarrow x_{k-1} - x_k$ вычисляют $x_k = N(a_1 \ldots a_k)$ и $y_k = N(a_1 \ldots a_k 0)$. [Требует ли $\Omega(n)$ шагов *каждая* широкословная цепочка для $N(a_1 \ldots a_n)$?]

157. Эти правила очевидны, за исключением двух случаев, включающих $(\alpha-)$. Для них мы имеем $N((\alpha-)0^k) = N(\alpha 0^k) + F_{-k-2}$ для всех $k \ge 0$, поскольку уменьшение никогда не выполняет "заем" справа. (Но аналогичная формула $N((\alpha+)0^k) = N(\alpha 0^k) + F_{-k-1}$ *не* выполняется.)

158. Увеличение удовлетворяет правилам $(\alpha 00)+ = \alpha 01$, $(\alpha 10)+ = (\alpha+)00$, $(\alpha 1)+ = (\alpha+)0$. Его можно осуществить с помощью шести 2-адических операций над целым числом $x = (\alpha)_2$, устанавливая $y \leftarrow x \mid (x \gg 1)$, $z \leftarrow y \,\&\, {\sim}(y+1)$, $x \leftarrow (x \mid z) + 1$.

Уменьшение ненулевого кодового слова более сложное. Оно удовлетворяет соотношениям $(\alpha 10^{2k})- = \alpha 0(10)^k$, $(\alpha 10^{2k+1})- = \alpha(01)^{k+1}$; следовательно, согласно следствию I его нельзя вычислить с помощью 2-адической цепочки. Если допустить использование монуса, достаточно семи операций: $y \leftarrow x - 1$, $z \leftarrow y \,\&\, \bar{x}$, $w \leftarrow z \,\&\, \mu_0$, $x \leftarrow y - w + (w \dot{-} (z - w))$.

159. Помимо фибоначчиевой системы счисления (146) и негафибоначчиевой системы счисления (147), имеется также *нечетная фибоначчиева система счисления*: *каждое положительное целое число x можно единственным образом записать в виде*

$$x = F_{l_1} + F_{l_2} + \cdots + F_{l_s}, \qquad \text{где } l_1 \gg l_2 \gg \cdots \gg l_s > 0 \text{ и } l_s \text{ нечетно.}$$

Следующая 19-шаговая 2-адическая цепочка для заданного негафибоначчиева кода α конвертирует $x = (\alpha)_2$ в $y = (\beta)_2$ в $z = (\gamma)_2$, где β — нечетное кодовое слово с $N(\alpha) = F(\beta)$, а γ — стандартное кодовое слово с $F(\beta) = F(\gamma 0)$: $x^+ \leftarrow x \,\&\, \mu_0$, $x^- \leftarrow x \oplus x^+$; $d \leftarrow x^+ - x^-$; $t \leftarrow d \mid x^-$, $t \leftarrow t \,\&\, {\sim}(t \ll 1)$; $y \leftarrow (d \,\&\, \bar{\mu}_0) \oplus t \oplus ((t \,\&\, x^-) \gg 1)$; $z \leftarrow (y+1) \gg 1$; $w \leftarrow z \oplus (4\mu_0)$; $t \leftarrow w \,\&\, {\sim}(w+1)$; $z \leftarrow (z \mid t) - (t \gg 1)$.

Соответствующие негафибоначчиево и нечетное представления удовлетворяют замечательному закону

$$F_{k_1+m} + \cdots + F_{k_r+m} = (-1)^m (F_{l_1-m} + \cdots + F_{l_s-m}) \qquad \text{для всех целых чисел } m.$$

Например, если $N(\alpha) < 0$, приведенные выше шаги будут конвертировать $x = (\alpha 0)_2$ в $y = (\beta)_2$, где $F((\beta \gg 2)0) = -N(\alpha)$. Кроме того, β представляет собой нечетный код для негафибоначчиева α тогда и только тогда, когда α^R является нечетным кодом для негафибоначчиева β^R при нечетном $|\alpha| = |\beta|$ и $N(\alpha) > 0$.

В соответствии со следствием I конечной 2-адической цепочки, работающей иным способом, не существует, поскольку фибоначчиев код 10^k соответствует негафибоначчиевому 10^{k+1} при нечетном k и $(10)^{k/2}1$ при k четном. Но если γ представляет собой стандартное фибоначчиево кодовое слово, мы можем вычислить $y = (\beta)_2$ из $z = (\gamma)_2$ путем установки $y \leftarrow z \ll 1$, $t \leftarrow y \,\&\, -y \,\&\, \bar{\mu}_0$, $y \leftarrow (t{=}0?\ y : y - 1 - ((t{-}1) \,\&\, \bar{\mu}_0))$. А тогда приведенный выше метод будет вычислять α^R из β^R. Общее время работы по преобразованию к негафибоначчиеву виду будет иметь порядок $\log|\gamma|$, для двух обращений строк.

160. Правила в разделе на самом деле неполные: они должны также определять ориентацию каждого соседа. Давайте потребуем, чтобы $\alpha_{sn} = \alpha$; $\alpha_{en} = \alpha$; $(\alpha 0)_{wn} = \alpha 0$, $(\alpha 1)_{wo} = \alpha 1$; $(\alpha 00)_{ns} = \alpha 00$, $(\alpha 10)_{nw} = \alpha 10$, $(\alpha 1)_{ne} = \alpha 1$; $(\alpha 0)_{oo} = \alpha 0$, $(\alpha 101)_{oo} = \alpha 101$, $(\alpha 1001)_{oo} = \alpha 1001$, $(\alpha 0001)_{ow} = \alpha 0001$. Тогда рассмотрение вариантов по индукции по расстоянию в графе d доказывает, что все ячейки в пределах d шагов от начальной ячейки обладают согласованными метками и ориентациями. (Обратите внимание на тождество $\alpha{+} = ((\alpha 0){-}) \gg 1$.) Кроме того, маркировка остается согласованной при добавлении координат y и перемещении при необходимости с одной полосы на другую с помощью δ-правил (153).

161. Да, соответствующий граф двудолен, поскольку все его ребра определяются множеством линий границ. (Гиперболический *цилиндр* раскрасить двумя цветами нельзя; но две смежные полосы, $y \bmod 2$, — можно.)

162. Удобно рассматривать гиперболическую плоскость "через другие очки", отображая ее точки на верхнюю полуплоскость $\Im z > 0$. Тогда "прямые линии" станут полуокружностями с центрами на оси x, с вертикальными линиями в качестве предельного случая. При таком представлении ребра $|z-1| = \sqrt{2}$, $|z| = r$ и $\Re z = 0$ определяют треугольник $36°$–$45°$–$90°$, если $r^2 = \phi + \sqrt{\phi}$. Каждый треугольник ABC имеет три соседа, CBA', ACB' и BAC', получающихся "отражением" двух его сторон относительно третьей, где отражение $|z - c'| = r'$ относительно $|z - c| = r$ представляет собой $|z - c - \frac{1}{2}(x_1 + x_2)| = \frac{1}{2}|x_1 - x_2|$, $x_j = r^2/(c' \pm r' - c)$.

Отображение $z \mapsto (z - z_0)/(z - \bar{z}_0)$ отображает верхнюю полуплоскость в единичный круг; когда $z_0 = \frac{1}{2}(\sqrt{\phi} - 1/\phi)(1 + 5^{1/4}i)$, центральный пятиугольник будет симметричен. Повторение отражений исходного треугольника с использованием поиска в ширину до достижения невидимых треугольников приведет к рис. 14. Для получения только лишь пятиугольников (без серых линий) можно начать с центральной ячейки и выполнять отражения относительно *ее* сторон, и т. д.

163. (Эту фигуру можно начертить так, как в упр. 162, начиная с вершин, которые проецируются на три точки, ir, $ir\omega$ и $ir\omega^2$, где $r^2 = \frac{1}{2}(1 + \sqrt{2})(4 - \sqrt{2} - \sqrt{6})$ и $\omega = e^{2\pi i/3}$. Используя обозначения, разработанные в 1852 году Л. Шляфли (L. Schläfli), ее можно описать как бесконечное замощение с параметрами $\{3, 8\}$, означающими, что в каждой вершине встречается восемь треугольников; см. работу Шляфли *Gesammelte Mathematische Abhandlungen* **1** (1950), 212. Аналогично пентагрид и замощение из упр. 154 имеют параметры Шляфли $\{5, 4\}$ и $\{5, 5\}$ соответственно.)

164. Оригинальное определение требует большего количества вычислений, даже несмотря на то, что оно может быть разложено:

$$\mathrm{custer}'(X) = X \,\&\, {\sim}(Y_{\mathrm{N}} \,\&\, Y \,\&\, Y_{\mathrm{S}}), \qquad Y = X_{\mathrm{W}} \,\&\, X \,\&\, X_{\mathrm{E}}.$$

Но основная причина предпочтения (157) заключается в том, что оно дает более тонкую границу, связанную ходом короля. Граница, связанная ходом ладьи, получающаяся в результате определения 1957 года, менее привлекательна, поскольку заметно темнее в случае

диагонали, чем в случае горизонтали или вертикали. (Поэкспериментируйте, и вы увидите это сами.)

165. Сначала представим $X^{(1)}$ как "внешнюю" границу исходных черных пикселей. Затем образуются изгибы наподобие отпечатков пальцев. Например, начиная с рис. 15, (а), мы получаем в растре 120×120

в конечном итоге приходя к причудливым бесконечно чередующимся рисункам. (*Каждый* ли непустой растр $M \times N$ ведет к такому 2-циклу?)

166. Если $X = \mathrm{custer}(X)$, сумма элементов $X + (X \gg\!\!\!\gg 1) + (X \ll 1) + (X \gg 1) + (X \ll\!\!\!\ll 1)$ не превышает $4MN + 2M + 2N$, поскольку это значение не больше 4 в каждой клетке прямоугольника и не больше 1 в смежных клетках. Эта сумма также в пять раз превышает количество черных пикселей. Следовательно, $f(M,N) \le \frac{4}{5}MN + \frac{2}{5}M + \frac{2}{5}N$. И обратно, мы получаем $f(M,N) \ge \frac{4}{5}MN - \frac{2}{5}$, допуская в строке i и столбце j черные пиксели, кроме случаев $(i+2j) \bmod 5 = 2$. (Эта задача эквивалентна поиску наименьшего доминирующего множества сетки $M \times N$.)

167. (а) С помощью 17 шагов можно построить полусумматор и три полных сумматора (см. 7.1.2–(23)), так что $(z_1z_2)_2 = x_{\rm NW} + x_{\rm W} + x_{\rm SW}$, $(z_3z_4)_2 = x_{\rm N} + x_{\rm S}$, $(z_5z_6)_2 = x_{\rm NE} + x_{\rm E} + x_{\rm SE}$ и $(z_7z_8)_2 = z_2 + z_4 + z_6$. Тогда $f = S_1(z_1, z_3, z_5, z_7) \wedge (x \vee z_8)$, где симметричной функции S_1 требуется семь операций согласно рис. 9 в разделе 7.1.2. [Это решение основано на идеях В. Ф. Манна (W. F. Mann) и Д. Слитора (D. Sleator).]

(б) Для заданных $x^- = X^{(t)}_{j-1}$, $x = X^{(t)}_j$ и $x^+ = X^{(t)}_{j+1}$ вычислим $a \leftarrow x^- \,\&\, x^+$ $(= z_3)$, $b \leftarrow x^- \oplus x^+$ $(= z_4)$, $c \leftarrow x \oplus b$, $d \leftarrow c \gg 1$ $(= z_6)$, $c \leftarrow c \ll 1$ $(= z_2)$, $e \leftarrow c \oplus d$, $c \leftarrow c \,\&\, d$, $f \leftarrow b \,\&\, e$, $f \leftarrow f \mid c$ $(= z_7)$, $e \leftarrow b \oplus e$ $(= z_8)$, $c \leftarrow x \,\&\, b$, $c \leftarrow c \mid a$, $b \leftarrow c \ll 1$ $(= z_5)$, $c \leftarrow c \gg 1$ $(= z_1)$, $d \leftarrow b \,\&\, c$, $c \leftarrow b \mid c$, $b \leftarrow a \,\&\, f$, $f \leftarrow a \mid f$, $f \leftarrow d \mid f$, $c \leftarrow b \mid c$, $f \leftarrow f \oplus c$ $(= S_1(z_1, z_3, z_5, z_7))$, $e \leftarrow e \mid x$, $f \leftarrow f \,\&\, e$.

[Отличное изложение радостей и печалей Жизни, включая доказательство возможности моделирования любой машины Тьюринга, имеется в книге Мартина Гарднера (Martin Gardner) *Wheels, Life and Other Mathematical Amusements* (1983), Chapters 20–22; см. также E. R. Berlekamp, J. H. Conway and R. K. Guy, *Winning Ways* **4** (A. K. Peters, 2004), Chapter 25.]*

Наконец, я получил то, что хотел, — по-видимому, непредсказуемые законы генетики.
. . .Перенаселенность, как и недоуплотненность, ведет к гибели.
Здоровое сообщество не является ни слишком плотным, ни слишком разреженным.

— ДЖОН Х. КОНВЕЙ (JOHN H. CONWAY), письмо
Мартину Гарднеру (Martin Gardner) (март 1970)

168. Приведенный далее алгоритм, использующий четыре n-битовых регистра x^-, x, x^+ и y, корректно работает даже тогда, когда $M = 1$ или $N = 1$. Он требует только около двух чтений и двух записей на слово растра для выполнения преобразования $X^{(t)}$ в $X^{(t+1)}$ из (158).

---

* Отличная высокоскоростная компьютерная реализация "Жизни" (и ее разновидностей) имеется по адресу `http://golly.sourceforge.net/`. — *Примеч. пер.*

C1. [Цикл по k.] Установить $A_{j0} \leftarrow 0$ для $0 \le j < M$. Затем выполнить шаг C2 для $k = 1, 2, \ldots, N'$. Затем перейти к шагу C5.

C2. [Цикл по j.] Установить $x \leftarrow A_{(M-1)k}$, $x^+ \leftarrow A_{0k}$ и $A_{Mk} \leftarrow x^+$. Затем выполнить шаги C3 и C4 для $j = 0, 1, \ldots, M-1$.

C3. [Перемещение вниз.] Установить $x^- \leftarrow x$, $x \leftarrow x^+$ и $x^+ \leftarrow A_{(j+1)k}$. (Теперь $x = A_{jk}$, и x^- хранит предыдущее значение $A_{(j-1)k}$.) Вычислите значения битовой функции $y \leftarrow f(x^- \gg 1, x^-, x^- \ll 1, x \gg 1, x, x \ll 1, x^+ \gg 1, x^+, x^+ \ll 1)$.

C4. [Обновление A_{jk}.] Установить $x^- \leftarrow A_{j(k-1)} \mathbin{\&} -2$, $y \leftarrow y \mathbin{\&} (2^{n-1} - 1)$, $A_{j(k-1)} \leftarrow x^- + (y \gg (n-2))$, $A_{jk} \leftarrow y + (x^- \ll (n-2))$.

C5. [Оборот.] Для $0 \le j < M$ установить $x \leftarrow A_{jN'} \mathbin{\&} -2^{n-1-d}$, $A_{jN'} \leftarrow x + (A_{j1} \gg d)$ и $A_{j1} \leftarrow A_{j1} + (x \ll d)$, где $d = 1 + (N-1) \bmod (n-2)$. ▮

[Во многих случаях, таких как (157) и (159), и даже (161), тор $M \times N$ эквивалентен массиву $(M-1) \times (N-1)$, окруженному нулями. Для упр. 173 можно очистить массив $(M-2) \times (N-2)$, ограниченный двумя строками и столбцами нулей. Но изображения "Жизни" (см. упр. 167) могут расти безгранично; их нельзя безопасно ограничить тором.]

169. Он быстро превратится в кролика, который затем взорвется. Начиная с момента 278, ситуация стабилизируется, и у нас остаются несколько "светофоров", три дополнительные "мигалки", а также три стабильных объекта ("ванная", "лодка" и "улей").

170. Если $M \ge 2$ и $N \ge 2$, на первом шаге вырезается верхняя строка и крайний справа столбец. Затем, если $M \ge 3$ и $N \ge 3$, на очередном шаге вырезаются нижняя строка и крайний слева столбец. Так что в общем случае после $t = \min(M, N) - 1$ шагов мы останемся с одной строкой или столбцом черных пикселей: первые $\lceil t/2 \rceil$ строк, последние $\lceil t/2 \rceil$ столбцов, последние $\lfloor t/2 \rfloor$ строк и первые $\lfloor t/2 \rfloor$ столбцов устанавливаются равными нулю. Автомат остановится после того, как выполнит два (непроизводительных) цикла.

171. Без (160): $x_1 \leftarrow x_{\rm SE} \mathbin{\&} \bar{x}_{\rm N}$, $x_2 \leftarrow x_{\rm N} \mathbin{\&} \bar{x}_{\rm SE}$, $x_3 \leftarrow x_{\rm E} \mathbin{\&} \bar{x}_1$, $x_4 \leftarrow x_{\rm NE} \mathbin{\&} \bar{x}_2$, $x_5 \leftarrow x_3 \mid x_4$, $x_6 \leftarrow x_{\rm W} \mathbin{\&} \bar{x}_5$, $x_7 \leftarrow x_1 \mathbin{\&} \bar{x}_{\rm NE}$, $x_8 \leftarrow x_7 \mathbin{\&} \bar{x}_{\rm NW}$, $x_9 \leftarrow x_{\rm E} \mid x_{\rm SW}$, $x_{10} \leftarrow x_8 \mathbin{\&} x_9$, $x_{11} \leftarrow x_{10} \mid x_6$, $x_{12} \leftarrow x_{\rm S} \mathbin{\&} x_{11}$, $x_{13} \leftarrow x_2 \mathbin{\&} \bar{x}_{\rm E}$, $x_{14} \leftarrow x_{13} \mathbin{\&} x_{\rm W}$, $x_{15} \leftarrow x_{\rm N} \mathbin{\&} x_{\rm NE}$, $x_{16} \leftarrow x_{\rm SW} \mathbin{\&} x_{\rm W}$, $x_{17} \leftarrow x_{15} \mid x_{16}$, $x_{18} \leftarrow x_{\rm NE} \mathbin{\&} x_{\rm SW}$, $x_{19} \leftarrow x_{17} \mathbin{\&} \bar{x}_{18}$, $x_{20} \leftarrow x_{\rm E} \mid x_{\rm SE}$, $x_{21} \leftarrow x_{20} \mid x_{\rm S}$, $x_{22} \leftarrow x_{\rm NW} \mathbin{\&} \bar{x}_{21}$, $x_{23} \leftarrow x_{22} \mathbin{\&} x_{19}$, $x_{24} \leftarrow x_{12} \mid x_{14}$, $g \leftarrow x_{23} \mid x_{24}$. С (160) установите $x_4 \leftarrow x_{\rm NE} \mathbin{\&} \bar{x}_{\rm N}$, а все остальное оставьте таким же.

172. Это утверждение не вполне истинно; рассмотрим следующие примеры.

'I' и 'H' слева показывают, что пиксели иногда остаются нетронутыми там, где соединяются пути, и что поворот на 90° может изменить ситуацию. Два следующих примера иллюстрируют необычное влияние зеркального отражения слева направо. Пример с ромбом демонстрирует, что очень толстые изображения могут быть неутончаемыми; ни один из его черных пикселей не может быть удален без изменения количества дыр. Последние примеры, один из которых вызван к жизни ответом к упр. 166, были сначала обработаны без учета случаев (160) и при этом остались неизменными. Но стоит применить (160), и это приведет к разительному утончению.

173. (a) Указание легко проверяется. Обратите внимание, что если X и Y закрыты, $X \mathbin{\&} Y$ закрыт; если X и Y открыты, $X \mid Y$ открыт. Таким образом, X^D закрыт, а X^L открыт; $X^{DD} = X^D$ и $X^{LL} = X^L$. (Фактически мы имеем $X^L = {\sim}({\sim}X)^D$, поскольку определения дуальны и получаются путем обмена черного и белого.) Теперь $X^{DL} \subseteq X^D$, так что

$X^{DLD} \subseteq X^{DD} = X^D$. И, дуально, $X^L \subseteq X^{LDL}$. Мы заключаем, что нет причин для отмывания чистого рисунка: $X^{DLDL} = (X^{DLD})^L \subseteq X^{DL} \subseteq (X^D)^{LDL} = X^{DLDL}$.

(б) Мы имеем $X^D = (X \mid X_{\rm W} \mid X_{\rm NW} \mid X_{\rm N})\,\&\,(X \mid X_{\rm N} \mid X_{\rm NE} \mid X_{\rm E})\,\&\,(X \mid X_{\rm E} \mid X_{\rm SE} \mid X_{\rm S})\,\&\,(X \mid X_{\rm S} \mid X_{\rm SW} \mid X_{\rm W})$. Кроме того, по аналогии с ответом к упр. 167, (б), эта функция может быть вычислена из x^-, x и x^+ за десять широкословных шагов: $f \leftarrow x \mid (x \gg 1) \mid ((x^- \mid (x^- \gg 1))\,\&\,(x^+ \mid (x^+ \gg 1)))$, $f \leftarrow f\,\&\,(f \ll 1)$. [Этот ответ включает идеи Д. Р. Фачса (D. R. Fuchs).]

Чтобы получить X^L, просто обменяйте $\mid$ и $\&$. [Дополнительную информацию можно почерпнуть из C. Van Wyk and D. E. Knuth, Report STAN-CS-79-707 (Stanford Univ., 1979), 15–36.]

174. Трехмерная цифровая топология изучалась в R. Malgouyres, *Theoretical Computer Science* **186** (1997), 1–41.

175. Таковых 25 во внешнем контуре, $2+3$ в глазах, $1+1$ в ушах, 4 в носу и 1 в улыбке, т. е. всего 37. (Все белые пиксели фона связаны ходом короля.)

176. (а) Если v не является изолированной вершиной, то имеется восемь простых случаев, которые следует рассмотреть, зависящих от того, какого вида сосед имеется у v в G.

(б) Некоторая $w' \in G'$ смежна или эквивалентна каждой вершине $N_u \cup N_v$. (Четыре случая.)

(в) Да. Фактически, согласно определению (161), мы всегда имеем $|S'(v')| \geq 2$.

(г) Пусть $N'_{v'} = \{v \mid v' \in N_v\}$. Если v' является восточным соседом u', назовем его $u'_{\rm E}$; либо $u' \in G$, либо $u'_{\rm S} \in G$. Этот элемент совпадает или смежен с каждой вершиной в $N'_{u'} \cup N'_{v'}$. Аналогичные рассуждения применимы и тогда, когда $v' = u'_{\rm N}$. Если $v' = u'_{\rm NE}$, в случае, когда $u' \in G$, нет никаких проблем. В противном случае $u'_{\rm W} \in G$, $u'_{\rm S} \in G$, и либо $u'_{\rm N} \in G$, либо $u'_{\rm E} \in G$; следовательно, $N'_{u'} \cup N'_{v'}$ является связным в G. Наконец, если $v' = u'_{\rm SE}$, доказательство оказывается простым при $u'_{\rm S} \in G$; в противном случае $u' \in G$ и $v' \in G$.

(д) Пусть для данного нетривиального компонента C графа G, обладающего тем свойством, что $v \in C$ и $v' \in S(v)$, C' представляет собой компонент G', который содержит v'. Этот компонент C' вполне определен, согласно п. (а) и (б). Пусть для заданного компонента C' графа G', обладающего тем свойством, что $v' \in C'$ и $v \in S'(v')$, C представляет собой компонент графа G, который содержит v. Этот компонент C нетривиален и вполне определен согласно п. (в) и (г). Наконец соответствие $C \leftrightarrow C'$ является взаимно однозначным.

177. Теперь вершинами G являются *белые* пиксели, смежные, когда они достижимы ходом *ладьи*. Так что мы определим $N_{(i,j)} = \{(i,j),(i{-}1,j),(i,j{+}1)\}$. Доказательство, подобное приведенному в ответе к упр. 176, но более простое, устанавливает взаимно однозначное соответствие между нетривиальными компонентами G и компонентами G'.

178. Заметим, что в соседних строках X^* два пикселя с одним и тем же значением являются соединенными ходом коня, только если они соединены ходом ладьи.

179. Пиксели $x_1 \ldots x_N$ каждой строки могут быть закодированы с использованием метода кодирования длин серий как последовательность целых чисел $0 = c_0 < c_1 < \cdots < c_{2m+1} = N+2$, так что $x_j = 0$ для $j \in [c_0\,..\,c_1) \cup [c_2\,..\,c_3) \cup \cdots \cup [c_{2m}\,..\,c_{2m+1})$ и $x_j = 1$ для $j \in [c_1\,..\,c_2) \cup \cdots \cup [c_{2m-1}\,..\,c_{2m})$. (Количество серий в строке в большинстве изображений достаточно мало́. Заметим, что неявно предполагается условие $x_0 = x_{N+1} = 0$.)

В приведенном ниже алгоритме используется модифицированное кодирование с $a_j = 2c_j - (j \bmod 2)$ для $0 \leq j \leq 2m+1$. Например, вторая строка чеширского кота имеет $(c_1, c_2, c_3, c_4, c_5) = (5, 8, 23, 25, 32)$; вместо этого мы будем использовать $(a_1, a_2, a_3, a_4, a_5) = (9, 16, 45, 50, 63)$. Причина заключается в том, что белые серии смежных строк являются "ладейно-смежными" тогда и только тогда, когда соответствующие отрезки $[a_j\,..\,a_{j+1})$

и $[b_k \mathinner{..} b_{k+1})$ перекрываются, и точно такое же условие описывает "королевско-смежные" черные серии в смежных строках. Таким образом, модифицированное кодирование красиво объединяет оба случая (см. упр. 178).

Мы строим трижды связанное дерево текущих компонентов, где каждый узел имеет несколько полей: CHILD, SIB и PARENT (связи в дереве); DORMANT (циклический список, с помощью связей SIB, из всех первых дочерних узлов, которые не связаны с текущей строкой); HEIR (узел, который его поглощает); ROW и COL (положение первого пикселя); и AREA (общее количество пикселей в компоненте).

Алгоритм обходит дерево в *двойном порядке* (см. упр. 2.3.1–18), используя пары указателей $(\mathtt{P},\mathtt{P}')$, где $\mathtt{P}' = \mathtt{P}$ при первом обходе P, $\mathtt{P}' = \mathtt{PARENT(P)}$ при втором обходе P. Преемником $(\mathtt{P},\mathtt{P}')$ является $(\mathtt{Q},\mathtt{Q}') = \mathrm{next}(\mathtt{P},\mathtt{P}')$, определяемый следующим образом: если $\mathtt{P} = \mathtt{P}'$ и $\mathtt{CHILD(P)} \neq \Lambda$, то $\mathtt{Q} \leftarrow \mathtt{Q}' \leftarrow \mathtt{CHILD(P)}$; в противном случае $\mathtt{Q} \leftarrow \mathtt{P}$ и $\mathtt{Q}' \leftarrow \mathtt{PARENT(Q)}$. Если $\mathtt{P} \neq \mathtt{P}'$ и $\mathtt{SIB(P)} \neq \Lambda$, то $\mathtt{Q} \leftarrow \mathtt{Q}' \leftarrow \mathtt{SIB(P)}$; в противном случае $\mathtt{Q} \leftarrow \mathtt{PARENT(P)}$ и $\mathtt{Q}' \leftarrow \mathtt{PARENT(Q)}$.

Если имеется m черных серий, дерево будет иметь $m+1$ узлов, не считая "спящих" или поглощенных узлов. Более того, подготовленные указатели $\mathtt{P}'_1, \ldots, \mathtt{P}'_{2m+1}$ двойного обхода $(\mathtt{P}_1,\mathtt{P}'_1), \ldots, (\mathtt{P}_{2m+1},\mathtt{P}'_{2m+1})$ в точности являются компонентами текущей строки, в порядке слева направо. Например, в (163) мы имеем $m = 5$; и $(\mathtt{P}'_1,\ldots,\mathtt{P}'_{11})$ указывают соответственно на ⓪, Ⓑ, ①, Ⓑ, ⓪, Ⓒ, ⓪, Ⓐ, ②, Ⓐ, ⓪.

I1. [Инициализация.] Установить $t \leftarrow 1$, $\mathtt{ROOT} \leftarrow \mathtt{LOC(NODE(0))}$, $\mathtt{CHILD(ROOT)} \leftarrow \mathtt{SIB(ROOT)} \leftarrow \mathtt{PARENT(ROOT)} \leftarrow \mathtt{DORMANT(ROOT)} \leftarrow \mathtt{HEIR(ROOT)} \leftarrow \Lambda$; а также установить $\mathtt{ROW(ROOT)} \leftarrow \mathtt{COL(ROOT)} \leftarrow 0$, $\mathtt{AREA(ROOT)} \leftarrow N+2$, $s \leftarrow 0$, $a_0 \leftarrow b_0 \leftarrow 0$, $a_1 \leftarrow 2N+3$.

I2. [Ввод новой строки.] Завершить работу алгоритма, если $s > M$. В противном случае устанавливать $b_k \leftarrow a_k$ для $k = 1, 2, \ldots$, пока не будет выполнено условие $b_k = 2N+3$; затем установить $b_{k+1} \leftarrow b_k$ в качестве "стопора". Установить $s \leftarrow s+1$. Если $s > M$, установить $a_1 \leftarrow 2N+3$; в противном случае $a_1, \ldots, a_{2m+1}$ представляют собой результат модифицированного кодирования длин серий строки s, рассмотренного выше. (Это кодирование может быть достигнуто с помощью функции ρ; см. (43).) Установить $j \leftarrow k \leftarrow 1$ и $\mathtt{P} \leftarrow \mathtt{P}' \leftarrow \mathtt{ROOT}$.

I3. [Поглощение коротких b.] Если $b_{k+1} \geq a_j$, перейти к шагу I9. В противном случае установить $(\mathtt{Q},\mathtt{Q}') \leftarrow \mathrm{next}(\mathtt{P},\mathtt{P}')$, $(\mathtt{R},\mathtt{R}') \leftarrow \mathrm{next}(\mathtt{Q},\mathtt{Q}')$ и выполнить четырехвариантное ветвление к (I4, I5, I6, I7) в зависимости от значения $2[\mathtt{Q} \neq \mathtt{Q}'] + [\mathtt{R} \neq \mathtt{R}'] = (0,1,2,3)$.

I4. [Случай 0.] (Сейчас $\mathtt{Q} = \mathtt{Q}'$ является дочерним узлом по отношению к $\mathtt{P}'$, а $\mathtt{R} = \mathtt{R}'$ является первым дочерним узлом по отношению к $\mathtt{Q}'$. Узел Q останется дочерним узлом $\mathtt{P}'$, но он будет предшествовать любому другому дочернему узлу R.) Включить R в $\mathtt{P}'$ (см. ниже). Установить $\mathtt{CHILD(Q)} \leftarrow \mathtt{SIB(R)}$ и $\mathtt{Q}' \leftarrow \mathtt{CHILD(R)}$. Если $\mathtt{Q}' \neq \Lambda$, установить $\mathtt{R} \leftarrow \mathtt{Q}'$ и, пока $\mathtt{R} \neq \Lambda$, устанавливать $\mathtt{PARENT(R)} \leftarrow \mathtt{P}'$, $\mathtt{R} \leftarrow \mathtt{SIB(R)}$; затем $\mathtt{SIB(R)} \leftarrow \mathtt{Q}$, $\mathtt{Q} \leftarrow \mathtt{Q}'$. Установить $\mathtt{CHILD(P)} \leftarrow \mathtt{Q}$, если $\mathtt{P} = \mathtt{P}'$, $\mathtt{SIB(P)} \leftarrow \mathtt{Q}$, если $\mathtt{P} \neq \mathtt{P}'$. Перейти к шагу I8.

I5. [Случай 1.] (Сейчас компонент $\mathtt{Q} = \mathtt{R}$ окружен $\mathtt{P}' = \mathtt{R}'$.) Если $\mathtt{P} = \mathtt{P}'$, установить $\mathtt{CHILD(P)} \leftarrow \mathtt{SIB(Q)}$; в противном случае установить $\mathtt{SIB(P)} \leftarrow \mathtt{SIB(Q)}$. Установить $\mathtt{R} \leftarrow \mathtt{DORMANT(R')}$. Затем, если $\mathtt{R} = \Lambda$, установить $\mathtt{DORMANT(R')} \leftarrow \mathtt{SIB(Q)} \leftarrow \mathtt{Q}$; в противном случае $\mathtt{SIB(Q)} \leftarrow \mathtt{SIB(R)}$ и $\mathtt{SIB(R)} \leftarrow \mathtt{Q}$. Перейти к шагу I8.

I6. [Случай 2.] (Сейчас $\mathtt{Q}'$ является родителем как $\mathtt{P}'$, так и R. Либо у $\mathtt{P} = \mathtt{P}'$ нет дочерних узлов, либо P представляет собой последний дочерний узел $\mathtt{P}'$.) Включить R в $\mathtt{P}'$ (см. ниже). Установить $\mathtt{SIB(P')} \leftarrow \mathtt{SIB(R)}$ и $\mathtt{R} \leftarrow \mathtt{CHILD(R)}$. Если

$\mathtt{P} = \mathtt{P}'$, установить $\mathtt{CHILD(P)} \leftarrow \mathtt{R}$; в противном случае $\mathtt{SIB(P)} \leftarrow \mathtt{R}$. Пока $\mathtt{R} \neq \Lambda$, устанавливать $\mathtt{PARENT(R)} \leftarrow \mathtt{P}'$ и $\mathtt{R} \leftarrow \mathtt{SIB(R)}$. Перейти к шагу I8.

I7. [Случай 3.] (Узел $\mathtt{P}' = \mathtt{Q}$ является последним дочерним узлом $\mathtt{Q}' = \mathtt{R}$, который является дочерним узлом $\mathtt{R}'$.) Включить $\mathtt{P}'$ в $\mathtt{R}'$ (см. ниже). Если $\mathtt{P} = \mathtt{P}'$, установить $\mathtt{P} \leftarrow \mathtt{R}$. В противном случае установить $\mathtt{P}' \leftarrow \mathtt{CHILD(P')}$ и, пока $\mathtt{P}' \neq \Lambda$, устанавливать $\mathtt{PARENT(P')} \leftarrow \mathtt{R}'$, $\mathtt{P}' \leftarrow \mathtt{SIB(P')}$; установить также $\mathtt{SIB(P)} \leftarrow \mathtt{SIB(Q')}$ и $\mathtt{SIB(Q')} \leftarrow \mathtt{CHILD(Q)}$. Если $\mathtt{Q} = \mathtt{CHILD(R)}$, установить $\mathtt{CHILD(R)} \leftarrow \Lambda$. В противном случае устанавливать $\mathtt{R} \leftarrow \mathtt{CHILD(R)}$, затем $\mathtt{R} \leftarrow \mathtt{SIB(R)}$, пока не будет выполнено условие $\mathtt{SIB(R)} = \mathtt{Q}$, затем $\mathtt{SIB(R)} \leftarrow \Lambda$. Наконец установить $\mathtt{P}' \leftarrow \mathtt{R}'$.

I8. [Продвижение k.] Установить $k \leftarrow k + 2$ и вернуться к шагу I3.

I9. [Обновление площади.] Установить $\mathtt{AREA(P')} \leftarrow \mathtt{AREA(P')} + \lceil a_j/2 \rceil - \lceil a_{j-1}/2 \rceil$. Затем вернуться к шагу I2, если $a_j = 2N + 3$.

I10. [Поглощение короткого a.] Если $a_{j+1} \geq b_k$, перейти к шагу I11. В противном случае установить $\mathtt{Q} \leftarrow \mathtt{LOC(NODE}(t)\mathtt{)}$ и $t \leftarrow t + 1$. Установить $\mathtt{PARENT(Q)} \leftarrow \mathtt{P}'$, $\mathtt{DORMANT(Q)} \leftarrow \mathtt{HEIR(Q)} \leftarrow \Lambda$; а также $\mathtt{ROW(Q)} \leftarrow s$, $\mathtt{COL(Q)} \leftarrow \lceil a_j/2 \rceil$, $\mathtt{AREA(Q)} \leftarrow \lceil a_{j+1}/2 \rceil - \lceil a_j/2 \rceil$. Если $\mathtt{P} = \mathtt{P}'$, установить $\mathtt{SIB(Q)} \leftarrow \mathtt{CHILD(P)}$ и $\mathtt{CHILD(P)} \leftarrow \mathtt{Q}$; в противном случае установить $\mathtt{SIB(Q)} \leftarrow \mathtt{SIB(P)}$ и $\mathtt{SIB(P)} \leftarrow \mathtt{Q}$. Наконец установить $\mathtt{P} \leftarrow \mathtt{Q}$, $j \leftarrow j + 2$ и вернуться к шагу I3.

I11. [Продвижение.] Установить $j \leftarrow j + 1$, $k \leftarrow k + 1$, $(\mathtt{P}, \mathtt{P}') \leftarrow \mathrm{next}(\mathtt{P}, \mathtt{P}')$ и перейти к шагу I3. ▮

"Включить P в Q" означает выполнение следующих действий: если $(\mathtt{ROW(P)}, \mathtt{COL(P)})$ меньше $(\mathtt{ROW(Q)}, \mathtt{COL(Q)})$, установить $(\mathtt{ROW(Q)}, \mathtt{COL(Q)}) \leftarrow (\mathtt{ROW(P)}, \mathtt{COL(P)})$. Установить $\mathtt{AREA(Q)} \leftarrow \mathtt{AREA(P)} + \mathtt{AREA(Q)}$. Если $\mathtt{DORMANT(Q)} = \Lambda$, установить $\mathtt{DORMANT(Q)} \leftarrow \mathtt{DORMANT(P)}$; в противном случае, если $\mathtt{DORMANT(P)} \neq \Lambda$, обменять $\mathtt{SIB(DORMANT(P))} \leftrightarrow \mathtt{SIB(DORMANT(Q))}$. Наконец установить $\mathtt{HEIR(P)} \leftarrow \mathtt{Q}$. (Связи HEIR могут использоваться при втором проходе для идентификации последнего компонента каждого пикселя. Заметим, что связи PARENT "спящих" узлов не поддерживаются в актуальном состоянии.)

[Аналогичный алгоритм был предложен Р. К. Луцем (R. K. Lutz) в *Comp. J.* **23** (1980), 262–269.]

180. Пусть $F(x, y) = x^2 - y^2 + 13$ и $Q(x, y) = F(x - \frac{1}{2}, y - \frac{1}{2}) = x^2 - y^2 - x + y + 13$. Применим алгоритм T для оцифровки гиперболы от $(\xi, \eta) = (-6, 7)$ до $(\xi', \eta') = (0, \sqrt{13})$; следовательно, $x = -6$, $y = 7$, $x' = 0$, $y' = 4$. Получающиеся в результате ребра представляют собой $(-6, 7)$ — $(-5, 7)$ — $(-5, 6)$ — $(-4, 6)$ — $(-4, 5)$ — $(-3, 5)$ — $(-3, 4)$ — $\cdots$ — $(0, 4)$. Затем применим его опять с $\xi = 0$, $\eta = \sqrt{13}$, $\xi' = 6$, $\eta' = 7$, $x = 0$, $y = 4$, $x' = 6$, $y' = 7$; будут найдены те же ребра (в обратном порядке), но с обратным знаком координаты x.

181. Выполните подразделение в точках (ξ, η), где $F_x(\xi, \eta) = 0$ или $F_y(\xi, \eta) = 0$, а именно в действительных корнях $\{Q(-(b(\eta + \frac{1}{2}) + d)/(2a), \eta + \frac{1}{2}) = 0,\ \xi = -(b(\eta + \frac{1}{2}) + d)/(2a) - \frac{1}{2}\}$ или в действительных корнях $\{Q(\xi + \frac{1}{2}, -(b(\xi + \frac{1}{2}) + e)/(2c)) = 0,\ \eta = -(b(\xi + \frac{1}{2}) + e)/(2c) - \frac{1}{2}\}$, если таковые существуют.

182. По индукции по $|x' - x| + |y' - y|$. Рассмотрим, например, случай $x > x'$ и $y < y'$. Из (iii) мы знаем, что (ξ, η) лежит в прямоугольнике $x - \frac{1}{2} \leq \xi < x + \frac{1}{2}$ и $y - \frac{1}{2} \leq \eta < y + \frac{1}{2}$, а из (ii), что кривая при движении из точки (ξ, η) в (ξ', η') монотонна. Следовательно, она должна выходить из прямоугольника через сторону $(x - \frac{1}{2}, y - \frac{1}{2})$ — $(x - \frac{1}{2}, y + \frac{1}{2})$ или $(x - \frac{1}{2}, y + \frac{1}{2})$ — $(x + \frac{1}{2}, y + \frac{1}{2})$. Последнее выполняется тогда и только тогда, когда $F(x - \frac{1}{2}, y + \frac{1}{2}) < 0$, поскольку кривая не может пересекать это ребро дважды при $x' < x$. А $F(x - \frac{1}{2}, y + \frac{1}{2})$ представляет собой значение $Q(x, y + 1)$, проверяемое на шаге T4, в соответствии

с инициализацией на шаге T1. (Мы считаем, что кривая не проходит *в точности* через $(x-\frac{1}{2}, y+\frac{1}{2})$, путем неявного добавления за сценой к функции F малой положительной величины.)

183. Рассмотрим, например, эллипс, определяемый как $F(x-\frac{1}{2}, y-\frac{1}{2}) = Q(x,y) = 13x^2 + 7xy + y^2 - 2 = 0$; этот эллипс представляет собой сигарообразную кривую, которая располагается примерно между $(-2,5)$ и $(1,-6)$. Предположим, что мы хотим оцифровать ее верхнюю правую границу. Гипотезы (i)–(iv) алгоритма T выполняются с

$$\xi = \sqrt{\frac{8}{3}} - \frac{1}{2}, \quad \eta = -\sqrt{\frac{98}{3}} - \frac{1}{2}, \quad \xi' = -\sqrt{\frac{98}{39}} - \frac{1}{2}, \quad \eta' = \sqrt{\frac{104}{3}} - \frac{1}{2},$$

$x=1$, $y=-6$, $x'=-2$, $y'=5$. На шаге T1 устанавливается $Q \leftarrow Q(1,-5) = 1$, что заставляет шаг T4 двигаться влево (L); фактически получающийся в результате путь представляет собой $\mathrm{L}^3\mathrm{U}^{11}$, в то время как корректная оцифровка согласно (164) имеет вид $\mathrm{U}^3\mathrm{LU}^4\mathrm{LU}^3\mathrm{LU}$. Сбой в работе алгоритма происходит потому, что $Q(x,y)=0$ имеет два корня на границе $(1,-5)$ — $(2,-5)$, а именно $((35 \pm -\sqrt{29})/26, -5)$, приводя к тому, что $Q(1,-5)$ имеет тот же знак, что и $Q(2,-5)$. (Один из этих корней располагается на границе, которую мы *не* пытаемся чертить.) Аналогичный сбой происходит и в случае параболы, определяемой уравнением $Q(x,y) = 9x^2 + 6xy + y^2 - y = 0$, $\xi = -5/12$, $\eta = -1/4$, $\xi' = -5/2$, $\eta' = -19/2$, $x=0$, $y=0$, $x'=-2$, $y'=9$. Гиперболы также не лишены этой неприятности (рассмотрите $6x^2 + 5xy + y^2 = 1$).

Алгоритмы дискретной геометрии печально известны своей чувствительностью; необычные ситуации приводят их в недоумение. Алгоритм T корректно работает для частей любого эллипса (или параболы), максимальная кривизна которого меньше 2. Максимальная кривизна эллипса с полуосями $\alpha \ge \beta$ равна α/β^2; сигарообразный пример имеет максимальную кривизну ≈ 42.5. Максимальная кривизна параболы $y = \alpha x^2$ равна $\alpha/2$; описанная выше аномальная парабола имеет максимальную кривизну ≈ 5.27. "Разумные" конические сечения не делают таких крутых поворотов.

Чтобы заставить алгоритм T корректно работать *без* гипотезы (v), нам придется немного его замедлить, заменив проверку '$Q<0$' на '$Q<0$ или X', где X представляет собой проверку знака производной, а именно X представляет собой соответственно проверку '$S>c$', '$R>a$', '$R<-a$', '$S<-c$' на шагах T2, T3, T4 и T5.

184. Положим $Q'(x,y) = -1 - Q(x,y)$. Ключевой момент заключается в том, что $Q(x,y) < 0$ тогда и только тогда, когда $Q'(x,y) \ge 0$. (Любопытно, что алгоритм принимает те же самые решения в обратном направлении, хотя проверяет значения Q' и Q в разных местах.)

185. Найдем положительное целое число h, такое, чтобы $d = (\eta - \eta')h$ и $e = (\xi' - \xi)h$ были целыми и $d+e$ было четным. Затем выполним алгоритм T с $x = \lfloor \xi + \frac{1}{2} \rfloor$, $y = \lfloor \eta + \frac{1}{2} \rfloor$, $x' = \lfloor \xi' + \frac{1}{2} \rfloor$, $y' = \lfloor \eta' + \frac{1}{2} \rfloor$ и $Q(x,y) = d(x-\frac{1}{2}) + e(y-\frac{1}{2}) + f$, где

$$f = \lfloor (\eta'\xi - \xi'\eta)h \rfloor - [d>0 \text{ и } (\eta'\xi - \xi'\eta)h \text{ целое}].$$

(Член '$d>0$' гарантирует, что противоположная прямая линия, от (ξ', η') до (ξ, η), будет иметь точно тот же вид; см. упр. 184.) Шаги T1 и T6–T9 становятся гораздо проще, чем в общем случае, поскольку $R=d$ и $S=e$ являются константами.

(Ф. Г. Стоктон (F. G. Stockton) [*CACM* **6** (1963), 161, 450] и Д. Э. Брезенхам (J. E. Bresenham) [*IBM Systems Journal* **4** (1965), 25–30] разработали подобные алгоритмы, но с разрешенными диагональными границами.)

186. (а) $B(\epsilon) = z_0 + 2\epsilon(z_1 - z_0) + O(\epsilon^2)$; $B(1-\epsilon) = z_2 - 2\epsilon(z_2 - z_1) + O(\epsilon^2)$.

(б) Каждая точка $S(z_0, z_1, z_2)$ представляет собой выпуклую комбинацию z_0, z_1 и z_2.

(в) Очевидно, истинно, поскольку $(1-t)^2 + 2(1-t)t + t^2 = 1$.

(г) Условие коллинеарности следует из (б). В противном случае, согласно (в), нам нужно рассмотреть только случай $z_0 = 0$ и $z_2 - 2z_1 = 1$, где $z_1 = x_1 + iy_1$ и $y_1 \neq 0$. В этом случае все точки лежат на параболе $4x = (y/y_1)^2 + 4yx_1/y_1$.

(д) Заметим, что $B(u\theta) = (1-u)^2 z_0 + 2u(1-u)((1-\theta)z_0 + \theta z_1) + u^2 B(\theta)$ для $0 \le u \le 1$.

[С. Н. Бернштейн ввел $B_n(z_0, z_1, \ldots, z_n; t) = \sum_k \binom{n}{k}(1-t)^{n-k} t^k z_k$ в *Сообщениях Харьковского математического общества* (2) **13** (1912), 1–2.]

187. Можно считать, что $z_0 = (x_0, y_0)$, $z_1 = (x_1, y_1)$ и $z_2 = (x_2, y_2)$, где координаты являются, скажем, числами с фиксированной точкой, представленными в виде 16-битовых целых чисел, деленных на 32.

Если z_0, z_1 и z_2 коллинеарны, используем метод из упр. 185 для черчения прямой линии от z_0 до z_2. (Если z_1 не лежит между z_0 и z_2, другие границы взаимно уничтожаются, поскольку алгоритмом заполнения к ним неявно применяется операция ЛИБО.) Это осуществляется тогда и только тогда, когда $D = x_0y_1 + x_1y_2 + x_2y_0 - x_1y_0 - x_2y_1 - x_0y_2 = 0$.

В противном случае точки (x, y) кривой $S(z_0, z_1, z_2)$ удовлетворяют уравнению $F(x, y) = 0$, где

$$F(x,y) = \big((x-x_0)(y_2-2y_1+y_0) - (y-y_0)(x_2-2x_1+x_0)\big)^2 \\ - 4D\big((x_1-x_0)(y-y_0) - (y_1-y_0)(x-x_0)\big),$$

а D определено выше. Для получения целочисленных коэффициентов мы выполняем умножение на 32^4; затем меняем знак формулы и вычитаем 1, если $D < 0$, чтобы удовлетворить условие (iv) алгоритма T и условие обратного порядка (см. упр. 184.)

Условие монотонности (ii) выполняется тогда и только тогда, когда $(x_1 - x_0)(x_2 - x_1) \ge 0$ и $(y_1 - y_0)(y_2 - y_1) \ge 0$. При необходимости мы можем использовать рекуррентное соотношение из упр. 186, (д), чтобы разбить $S(z_0, z_1, z_2)$ на максимум три монотонных подсквайна; например, выбор $\theta = (x_0 - x_1)/(x_0 - 2x_1 + x_2)$ дает монотонность по x. (При выполнении арифметических действий с числами с фиксированной точкой может появиться небольшая ошибка округления, но рекуррентность можно выполнить таким образом, чтобы сквайны были точно монотонными.)

Примечания. Когда z_0, z_1 и z_2 близки друг к другу, для большинства практических целей достаточно более простого и быстрого метода, основанного на упр. 186, (д) с $\theta = \frac{1}{2}$, если не беспокоиться о точном выборе между локальными последовательностями границ наподобие "вверх–и–влево" и "влево–и–вверх". В конце 1980-х годов Сампо Каасила (Sampo Kaasila) выбрал сквайны в качестве базового метода спецификации для шрифтов в формате TrueType, что и позволяет оцифровывать их так быстро. Система METAFONT достигает большей гибкости с помощью кубических сплайнов Безье [см. D. E. Knuth, *METAFONT: The Program* (Addison–Wesley, 1986)], но ценой большего времени работы. Однако впоследствии Джоном Хобби (John Hobby) для результирующих кубических кривых был разработан "шестирегистровый алгоритм" [*ACM Trans. on Graphics* **9** (1990), 262–277]. Воган Пратт (Vaughan Pratt) разработал *конические сплайны*, нечто среднее между сквайнами и кубическими сплайнами Безье, в *Computer Graphics* **19**, 3 (July 1985), 151–159. Сегменты конического сплайна могут быть эллиптическими, гиперболическими или параболическими, следовательно, они требуют меньшего количества промежуточных и управляющих точек, чем сквайны; кроме того, с ними можно работать с помощью алгоритма T.

188. В следующей программе, работающей с обратным порядком байтов, предполагается, что $n \le 74880$.

```
         LOC  Data_Segment              LDO  k,Initk
BITMAP   LOC  @+M*N/8              0H  SET  s,N/64
base     GREG @                    1H  SET  a,h        Трюк (см. ниже).
GRAYMAP  LOC  @+M*N/64                 SET  r,8
GTAB     BYTE 255,252,249,246,243  2H  LDOU t,base,k
         BYTE 240,236,233,230,227      MOR  u,c1,t
         BYTE 224,221,217,214,211      SUBU t,t,u      (По-четвертьбайтовая сумма).
         BYTE 208,204,201,198,194      MOR  u,c2,t
         BYTE 191,188,184,181,178      AND  t,t,mu1
         BYTE 174,171,167,164,160      ADDU t,t,u      (По-полубайтовая сумма).
         BYTE 157,153,150,146,142      MOR  u,c3,t
         BYTE 139,135,131,128,124      AND  t,t,mu2
         BYTE 120,116,112,108,104      ADDU t,t,u      (Побайтовая сумма).
         BYTE 100,96,92,88,84          ADDU a,a,t
         BYTE 79,75,70,66,61           INCL k,N/8      Переход к следующей строке.
         BYTE 56,52,46,41,36           SUB  r,r,1
         BYTE 30,24,18,10,0            PBNZ r,2B       Повторить 8 раз.
Initk    OCTA BITMAP-GRAYMAP       3H  SRU  t,a,56
corr     GREG N-8                      LDBU t,gtab,t
c1       GREG #4000100004000100        SLU  a,a,8
c2       GREG #2010000002010000        STBU t,z,0
c3       GREG #0804020100000000        INCL z,1
mu1      GREG #3333333333333333        PBN  a,3B       (Трюк).
mu2      GREG #0f0f0f0f0f0f0f0f        SUB  k,k,corr
h        GREG #8080808080808080        SUB  s,s,1
gtab     GREG GTAB-#80                 PBNZ s,1B       Цикл по столбцам.
         LOC  #100                     INCL k,7*N/8    Цикл по группам
MakeGray LDA  z,GRAYMAP                PBN  k,0B        из 8 строк. ▌
```

[Вдохновленный DVIPAGE Нейла Ханта (Neil Hunt), автор широко использовал такие изображения при подготовке нового издания *Искусства программирования* в 1992–1998 годах.]

189. Если строками растра являются $(X_0, X_1, \ldots, X_{63})$, выполните следующие операции для $k = 0, 1, \ldots, 5$: для всех i, таких, что $0 \le i < 64$ и $i \& 2^k = 0$, положить $j = i + 2^k$ и либо (а) установить $t \leftarrow (X_i \oplus (X_j \gg 2^k)) \& \mu_{6,k}$, $X_i \leftarrow X_i \oplus t$, $X_j \leftarrow X_j \oplus (t \ll 2^k)$; либо (б) установить $t \leftarrow X_i \,\&\, \bar{\mu}_{6,k}$, $u \leftarrow X_j \,\&\, \mu_{6,k}$, $X_i \leftarrow ((X_i \ll 2^k) \,\&\, \bar{\mu}_{6,k}) \mid u$, $X_j \leftarrow ((X_j \gg 2^k) \,\&\, \mu_{6,k}) \mid t$.

[Основная идея заключается в преобразовании $2^k \times 2^k$ подматриц для увеличения k, как в упр. 5–12. На машине MMIX можно добиться ускорения за счет применения инструкций MOR и MUX, как в упр. 208, и использования LDTU/STTU при $k = 5$. См. L. J. Guibas and J. Stolfi, *ACM Transactions on Graphics* **1** (1982), 204–207; M. Thorup, *J. Algorithms* **42** (2002), 217. Кстати, теорема P и ответ к упр. 54 показывают, что для транспонирования битовой матрицы $n \times n$ необходимы $\Omega(n \log n)$ операций с n-битовыми числами. Приложение, которому часто требуется выполнять транспонирование, может воспользоваться избыточным представлением, одновременно поддерживая матрицу в прямом и транспонированном виде.]

190. (а) Мы должны иметь $\alpha_{j+1} = f(\alpha_j) \oplus \alpha_{j-1}$ для $j \ge 1$, где $\alpha_0 = 0\ldots0$ и $f(\alpha) = ((\alpha \ll 1) \,\&\, 1\ldots1) \oplus \alpha \oplus (\alpha \gg 1)$. Элементы нижней строки α_m удовлетворяют условию четности тогда и только тогда, когда это правило делает α_{m+1} полностью нулевым.

(б) Истинно. Условие четности, накладываемое на элементы матрицы a_{ij}, имеет вид $a_{ij} = a_{(i-1)j} \oplus a_{i(j-1)} \oplus a_{i(j+1)} \oplus a_{(i+1)j}$, где $a_{ij} = 0$, если $i = 0$ или $i = m+1$, или $j = 0$, или

$j = n + 1$. Если матрицы (a_{ij}) и (b_{ij}) удовлетворяют этому условию, то ему удовлетворяет и (c_{ij}) при $c_{ij} = a_{ij} \oplus b_{ij}$.

(в) Верхняя левая подматрица, состоящая из всех строк, предшествующих первой полностью нулевой строке (если таковая имеется) и всех столбцов, предшествующих первому полностью нулевому столбцу (если таковой имеется), является идеальной. А эта подматрица определяет всю матрицу, поскольку шаблон с другой стороны от строки или столбца нулей представляет собой горизонтальное или вертикальное отражение ее соседей. Например, если $\alpha_{m'+1}$ равно нулю, то $\alpha_{m'+1+j} = \alpha_{m'+1-j}$ для $1 \le j \le m'$.

(г) Начиная с заданного вектора α_1 и используя правило из п. (а), мы всегда придем к строке с $\alpha_{m+1} = 0\ldots0$. Доказательство: мы должны иметь $(\alpha_j, \alpha_{j+1}) = (\alpha_k, \alpha_{k+1})$ для некоторого $0 \le j < k \le 2^{2n}$ согласно принципу голубиного гнезда. Если $j > 0$, мы также имеем $(\alpha_{j-1}, \alpha_j) = (\alpha_{k-1}, \alpha_k)$, поскольку $\alpha_{j-1} = f(\alpha_j) \oplus \alpha_{j+1} = f(\alpha_k) \oplus \alpha_{k+1} = \alpha_{k-1}$. Следовательно, первая повторяющаяся пара начинается со строки нулей α_k. Кроме того, мы имеем $\alpha_i = \alpha_{k-i}$ для $0 \le i \le k$; следовательно, первая полностью нулевая строка α_{m+1} встречается, когда m равно $k-1$ или $k/2-1$.

Строки $\alpha_1, \ldots, \alpha_m$ образуют идеальный шаблон, если только не имеется столбца, состоящего из нулей. Имеется $t > 0$ таких столбцов тогда и только тогда, когда $t+1$ является делителем $n+1$, а α_1 имеет вид $\alpha 0 \alpha^R 0 \ldots 0 \alpha$ (t четно) или $\alpha 0 \alpha^R 0 \ldots 0 \alpha^R$ (t нечетно), где $|\alpha| + 1 = (n+1)/(t+1)$.

(д) Этот стартовый вектор имеет вид, не запрещенный в п. (г).

191. (а) Первый представляет собой $\alpha_1, \alpha_2, \ldots$ тогда и только тогда, когда последний имеет вид $0\alpha_1 0 \alpha_1^R, 0\alpha_2 0 \alpha_2^R, \ldots$.

(б) Пусть бинарная строка $a_0 a_1 \ldots a_{N-1}$ соответствует полиному $a_0 + a_1 x + \cdots + a_{N-1} x^{N-1}$ и пусть $y = x^{-1} + 1 + x$. Тогда $\alpha_0 = 0\ldots0$ соответствует $F_0(y)$; $\alpha_1 = 10\ldots0$ соответствует $F_1(y)$; и по индукции α_j соответствует $F_j(y)$ по модулю $x^N + 1$ и по модулю 2. Например, когда $N = 6$, мы имеем $\alpha_2 = 110001 \leftrightarrow 1 + x + x^5$, поскольку $x^{-1} \bmod (x^6 + 1) = x^5$, и т. д.

(в) И вновь применяется индукция по j.

(г) Тождество в указании выполняется по индукции по m, поскольку оно очевидно истинно при $m = 1$ и $m = 2$. При работе по модулю 2 это тождество дает простые уравнения

$$F_{2k}(y) = yF_k(y)^2; \qquad F_{2k-1}(y) = (F_{k-1}(y) + F_k(y))^2.$$

Так что мы можем перейти от пары $P_k = (F_{k-1}(y) \bmod (x^N+1), F_k(y) \bmod (x^N+1))$ к паре P_{k+1} за $O(n)$ шагов и к паре P_{2k} за $O(n^2)$ шагов. Таким образом, мы можем вычислить $F_j(y) \bmod (x^N + 1)$ после $O(\log j)$ итераций. Умножение на $f_\alpha(x) + f_\alpha(x^{-1})$ и последующее уменьшение по модулю $x^N + 1$ позволяет нам вывести значение α_j.

Кстати, $F_{n+1}(x)$ является частным случаем $K_n(x, x, \ldots, x)$ континуант; см. 4.5.3–(4). Мы имеем $F_{n+1}(x) = \sum_{k=0}^n \binom{n-k}{k} x^{n-2k} = i^{-n} U_n(ix/2)$, где U_n — классический полином Чебышева, определяемый как $U_n(\cos\theta) = \sin((n+1)\theta)/\sin\theta$.

192. (а) Согласно упр. 191, (в), $c(q)$ является наименьшим $j > 0$, таким, что $(x+x^{-1})F_j(x^{-1}+1+x) \equiv 0$ (по модулю $x^{2q}+1$), с использованием полиномиальной арифметики по модулю 2. Или, что то же самое, это наименьшее положительное j, для которого $F_j(y)$ кратно $(x^{2q}+1)/(x^2+1) = (1 + x + \cdots + x^{q-1})^2$, когда $y = x^{-1} + 1 + x$.

(б) Используйте метод из упр. 191, (г) для вычисления $((x + x^{-1})F_j(y)) \bmod (x^{2q} + 1)$ при $j = M/p$ для всех простых делителей p числа M. Если результат окажется нулевым, установите $M \leftarrow M/p$ и повторите процесс. Если ни один из таких результатов не равен нулю, $c(q) = M$.

(в) Мы хотим показать, что $c(2^e)$ является делителем $3 \cdot 2^{e-1}$, но не $3 \cdot 2^{e-2}$ или 2^{e-1}. Последнее утверждение выполняется, поскольку $F_{2^{e-1}}(y) = y^{2^{e-1}-1}$ является взаимно

простым с $x^{2^{e+1}}+1$. Первое утверждение справедливо потому, что

$$F_{3\cdot 2^{e-1}}(y) = y^{2^{e-1}-1}F_3(y)^{2^{e-1}} = y^{2^{e-1}-1}(1+y)^{2^e} = y^{2^{e-1}-1}(x^{-1}+x)^{2^e},$$

что $\equiv 0$ по модулю $x^{2^{e+1}}+1$, но не по модулю $x^{2^{e+2}}+1$.

(г) $F_{2^e-1}(y) = \sum_{k=1}^{e} y^{2^e-2^k}$. Поскольку $y = x^{-1}(1+x+x^2)$ взаимно простое с x^q+1, мы имеем $y^{-1} \equiv a_0+a_1x+\cdots+a_{q-1}x^{q-1}$ (по модулю x^q+1) для некоторых коэффициентов a_i; следовательно,

$$y^{-2^k} \equiv a_0 + a_1x^{2^k} + \cdots + a_{q-1}x^{2^k(q-1)} \equiv a_0 + a_1x^{2^{k+e}} + \cdots + a_{q-1}x^{2^{k+e}(q-1)} \equiv y^{-2^{k+e}}$$

по модулю x^q+1 для $0 \le k < e$, и отсюда следует, что $F_{2^{2e}-1}(y)$ кратно $x^{2q}+1$.

(д) В этом случае $c(q)$ делит $4(2^{2e}-1)$. Доказательство: пусть $x^q+1 = f_1(x)f_2(x)\ldots\times f_r(x)$, где $f_1(x) = x+1$, $f_2(x) = x^2+x+1$, и каждое $f_i(x)$ неприводимо по модулю 2. Поскольку q нечетно, эти множители различны. Следовательно, в конечном поле полиномов по модулю $f_j(x)$ для $j \ge 3$ мы имеем $y^{-2^k} = y^{-2^{k+e}}$, как в п. (г). Следовательно, $F_{2^{2e}-1}(y)$ кратно $f_3(x)\ldots f_r(x) = (x^q+1)/(x^3+1)$. Так что $F_{4(2^{2e}-1)}(y) = y^3F_{2^{2e}-1}(y)^4$ кратно $(x^{2q}+1)/(x^2+1) = f_2(x)^2f_3(x)^2\ldots f_r(x)^2$, что и требовалось доказать.

(е) Если $F_{c(q)}(y)$ кратно $x^{2q}+1$, легко видеть, что $c(2q) = 2c(q)$. В противном случае $F_{3c(q)}(y)$ кратно $F_3(y) = (1+y)^2 = x^{-2}(1+x)^4$; следовательно, $F_{6c(q)}(y)$ кратно $x^{4q}+1$ и $c(2q)$ делит $6c(q)$. Последнее происходит только тогда, когда q нечетно.

Примечания. Шаблоны четности имеют отношение к популярной головоломке "Lights Out", изобретенной в начале 1980-х годов Дарио Ури (Dario Uri) и независимо от него почти в то же время Меро Ласло (László Mérő) и названной XLIGS. [См. David Singmaster, *Cubic Circular*, issues 7&8 (Summer 1985), 39–42; Dieter Gebhardt, *Cubism For Fun* **69** (March 2006), 23–25.] Клаус Сатнер (Klaus Sutner) разработал прочие аспекты этой теории в *Theoretical Computer Science* **230** (2000), 49–73.

193. Пусть $b_{(2i)(2j)} = a_{ij}$, $b_{(2i+1)(2j)} = a_{ij} \oplus a_{(i+1)j}$, $b_{(2i)(2j+1)} = a_{ij} \oplus a_{i(j+1)}$ и $b_{(2i+1)(2j+1)} = 0$ для $0 \le i \le m$ и $0 \le j \le n$, где мы считаем $a_{ij} = 0$ при $i = 0$ или $i = m+1$, или $j = 0$, или $j = n+1$. $(b_{(2i)1}, b_{(2i)2}, \ldots, b_{(2i)(2n+1)}) = (0, 0, \ldots, 0)$ не выполняется, поскольку $(a_{i1}, \ldots, a_{in}) \ne (0, \ldots, 0)$ для $1 \le i \le m$. Не выполняется также и $(b_{(2i+1)1}, b_{(2i+1)2}, \ldots, b_{(2i+1)(2n+1)}) = (0, 0, \ldots, 0)$, поскольку соседние строки $(a_{i1}, \ldots, a_{in})$ и $(a_{(i+1)1}, \ldots, a_{(i+1)n})$ всегда различны для $0 \le i \le m$, когда m нечетно.

194. Установим $\beta_i \leftarrow (1 \ll (n-i)) \mid (1 \ll (i-1))$ для $1 \le i \le m$, где $m = \lceil n/2 \rceil$. Установим также $\gamma_i \leftarrow (\beta_1 \mathbin{\&} \alpha_{i1}) + (\beta_2 \mathbin{\&} \alpha_{i2}) + \cdots + (\beta_m \mathbin{\&} \alpha_{im})$, где α_{ij} представляет собой j-ю строку шаблона четности, который начинается с β_i; вектор γ_i записывает диагональные элементы такой матрицы. Затем установим $r \leftarrow 0$ и применим подпрограмму N из ответа к упр. 195 к $i \leftarrow 1, 2, \ldots, m$. Получающиеся в результате векторы $\theta_1, \ldots, \theta_r$ являются базисом для всех $n \times n$ шаблонов четности с восьмикратной симметрией.

Чтобы проверить, является ли любой такой шаблон совершенным, допустим, что шаблон, начинающийся с θ_i, содержит нуль в строке c_i. Если одно из $c_i = n+1$, то ответ — да. Если $\operatorname{lcm}(c_1, \ldots, c_r) \le n$, то ответ — нет. Если же ни одно из этих условий не дает окончательный ответ, то можно прибегнуть к грубому перебору $2^r - 1$ ненулевых линейных комбинаций векторов θ.

Например, при $n = 9$ мы находим $\gamma_1 = 111101111$, $\gamma_2 = \gamma_3 = 010101010$, $\gamma_4 = 000000000$, $\gamma_5 = 001010100$; тогда $r = 0$, $\theta_1 = 011000110$, $\theta_2 = 000101000$, $c_1 = c_2 = 5$. Так что идеального решения в этом случае не существует.

В экспериментах автора для $n \le 3000$ "грубая сила" потребовалась только при $n = 1709$. В этом случае $r = 21$ и значения всех c_i оказались равными 171 или 855, за исключением $c_{21} = 342$. Тут же оказалось найденным решение $\theta_1 \oplus \theta_{21}$.

Ответами для $1 \le n \le 383$ являются 4, 5, 11, 16, 23, 29, 30, 32, 47, 59, 62, 64, 65, 84, 95, 101, 119, 125, 126, 128, 131, 154, 164, 170, 185, 191, 203, 204, 239, 251, 254, 256, 257, 263, 314, 329, 340, 341, 371, 383.

[Фрактал, подобный показанному на рис. 20 и называющийся "узором микадо", появился в статье H. Eriksson, K. Eriksson and J. Sjöstrand, *Advances in Applied Math.* **27** (2001), 365. См. также S. Wolfram, *A New Kind of Science* (2002), правило 150R на с. 439.]

195. Установить $\beta_i \leftarrow 1 \ll (m-i)$ и $\gamma_i \leftarrow \alpha_i$ для $1 \le i \le m$; установить также $r \leftarrow 0$. Затем выполнить следующую подпрограмму для $i = 1, 2, \ldots, m$.

N1. [Выделение младшего бита.] Установить $x \leftarrow \gamma_i \mathbin{\&} -\gamma_i$. Если $x = 0$, перейти к шагу N4.

N2. [Поиск j.] Найти наименьшее $j \ge 1$, такое, что $\gamma_j \mathbin{\&} x \ne 0$ и $\gamma_j \mathbin{\&} (x-1) = 0$.

N3. [Зависимый?] Если $j < i$, установить $\gamma_i \leftarrow \gamma_i \oplus \gamma_j$, $\beta_i \leftarrow \beta_i \oplus \beta_j$ и вернуться к шагу N1. (Эти операции сохраняют матричное равенство $C = BA$.) В противном случае завершить работу подпрограммы (поскольку γ_i линено независимо от $\gamma_1, \ldots, \gamma_{i-1}$).

N4. [Запись решения.] Установить $r \leftarrow r + 1$ и $\theta_r \leftarrow \beta_i$. ▮

В заключение $m - r$ ненулевых векторов γ_i представляют собой базис векторного пространства всех линейных комбинаций $\alpha_1, \ldots, \alpha_m$; они характеризуются своими младшими битами.

196. (а) #0a; #cea3; #e7ae97; #f09d8581.

(б) Если $\lambda x = \lambda x'$, результат очевиден, так как $l = l'$. В противном случае мы имеем либо $\alpha_1 < \alpha_1'$, либо ($\alpha_1 = \alpha_1'$ и $\alpha_2 < \alpha_2'$); последний случай может осуществиться только при $x \ge 2^{16}$.

(в) Установим $j \leftarrow k$; пока $\alpha_j \oplus \#80 < \#40$, будем устанавливать $j \leftarrow j - 1$. Тогда $\alpha(x^{(i)})$ начинается с α_j.

197. (а) #000a; #03a3; #7b97; #d834dd41.

(б) Лексикографический порядок *не* сохраняется, когда, скажем, $x = \#\mathtt{ffff}$ и $x' = \#10000$.

(в) Чтобы правильно ответить на этот вопрос, нужно знать, что 2048 целых чисел в диапазоне $\#d800 \le x < \#e000$ не являются корректными кодовыми точками UCS; они называются *заменителями* (*surrogates*). С учетом сказанного $\beta(x^{(i)})$ начинается с β_k, если $\beta_k \oplus \#dc00 \ge \#0400$, в противном случае $\beta(x^{(i)})$ начинается с β_{k-1}.

198. $a = \#e50000$, $b = 3$, $c = \#16$. (Можно положить $b = 0$, но тогда a будет огромным. Этот трюк предложен П. Рэйнод-Ричардом (P. Raynaud-Richard) в 1997 году. Указанные константы, предложенные Р. Пурнадером (R. Pournader) в 2008 году, являются наименьшими возможными.)

199. Нам необходимо, чтобы выполнялись условия $\alpha_1 > \#c1$; $2^8\alpha_1 + \alpha_2 < \#f490$; и либо $(\alpha_1 \mathbin{\&} -\alpha_1) + \alpha_1 < \#100$, либо $\alpha_1 + \alpha_2 > \#17f$. Эти условия выполняются тогда и только тогда, когда

$$(\#c1 - \alpha_1) \mathbin{\&} (2^8\alpha_1 + \alpha_2 - \#f490) \mathbin{\&} (((\alpha_1 \mathbin{\&} -\alpha_1) + \alpha_1 - \#100) \mid (\#17f - \alpha_1 - \alpha_2)) \;<\; 0.$$

Маркус Кун (Markus Kuhn) предложил добавить оператор '$\mathbin{\&}(\#20 - ((2^8\alpha_1 + \alpha_2) \oplus \#eda0))$', чтобы гарантировать, что $\alpha_1\alpha_2$ не начинает код заменителя.

200. Если $\$0 = (x_7 \ldots x_1 x_0)_{256}$, то $\$3$ представляет собой значение симметричной функции $S_2(x_7, x_4, x_2)$.

201. MOR x,c,x, где c = #f0f0f0f00f0f0f0f.

202. MOR x,x,c, где c = #c0c030300c0c0303; затем MOR x,mone,x. (См. ответ к упр. 209.)

203. a = #0008000400020001, b = #0f0f0f0f0f0f0f0f, c = #0606060606060606, d = #0000002700000000, e = #2a2a2a2a2a2a2a2a. (ASCII-код 0 представляет собой $6 + {}^{\#}2a$; ASCII-код a представляет собой $6 + {}^{\#}2a + 10 + {}^{\#}27$.)

204. p = #8008400420021001, q = #8020080240100401 (транспонированное p), r = #4080102004080102 (симметричная матрица) и m = #aa55aa55aa55aa55.

205. Тасовать, но с $p \leftrightarrow q$, r = #0804020180402010, m = #f0f0f0f00f0f0f0f.

206. Просто замените p на #0880044002200110. (Кстати, эти тасования можно также определить как перестановки в $z = (z_{63} \ldots z_1 z_0)_2$ другим способом: внешнее тасование отображает $z_j \mapsto z_{(2j) \bmod 63}$ для $0 \le j < 63$, в то время как внутреннее отображает $z_j \mapsto z_{(2j+1) \bmod 65}$.)

207. Выполните MOR y,p,x; MOR y,y,p; MOR t,y,q; PUT rM,m1; MUX y,y,t; MOR t,t,q; PUT rM,m2; MUX y,y,t. В обоих случаях p = #2004801002400801; в случае тройного тасования q = #4020100804020180, m_1 = #4949494949494949, m_2 = #dbdbdbdbdbdbdbdb; для обратного q = #0402018040201008, m_1 = #0707070707070707, m_2 = #3f3f3f3f3f3f3f3f.

208. (Решение Г. С. Уоррена-мл. (H. S. Warren, Jr.).) Описанный в разделе метод, использующий 7-, 14- и 28-обмен, можно реализовать с помощью всего лишь 12 команд:

```
MOR t,x,c1;  MOR t,c1,t;  PUT rM,m1;  MUX y,x,t;
MOR t,y,c2;  MOR t,c2,t;  PUT rM,m2;  MUX y,y,t;
MOR t,y,c3;  MOR t,c3,t;  PUT rM,m3;  MUX y,y,t;
```

здесь c_1 = #4080102004080102, c_2 = #2010804002010804, c_3 = #0804020180402010, m_1 = #aa55aa55aa55aa55, m_2 = #cccc3333cccc3333, m_3 = #f0f0f0f00f0f0f0f.

209. Достаточно всего лишь четырех команд: MXOR y,p,x; MXOR x,mone,x; MXOR x,x,q; XOR x,x,y; здесь p = #80c0e0f0f8fcfeff $= \bar{q}$, а регистр mone $= -1$.

210. SLU x,one,x; MOR x,b,x; AND x,x,a; MOR x,x,#ff; здесь регистр one $= 1$.

211. В общем случае элемент ij произведения булевых матриц AXB представляет собой $\bigvee\{x_{kl} \mid a_{ik}b_{lj} = 1\}$. В данной задаче мы выбираем $a_{ik} = [i \supseteq k]$ и $b_{lj} = [l \subseteq j]$; ответ — 'MOR t,f,a; MOR t,b,t', где a = #80c0a0f088ccaaff и b = #ff5533110f050301 $= a^T$.

(Обратите внимание, что этот трюк дает простую проверку монотонности $[f = \hat{f}]$. Кроме того, 64-битовый результат $(t_{63} \ldots t_1 t_0)_2$ дает коэффициенты мультилинейного представления

$$f(x_1, \ldots, x_6) = (t_{63} + t_{62}x_6 + \cdots + t_1x_1x_2x_3x_4x_5 + t_0x_1x_2x_3x_4x_5x_6) \bmod 2,$$

если заменить MXOR на MOR, согласно результату упр. 7.1.1–11.)

212. Если $\cdot$ означает MXOR, как в (183), а $b = (\beta_7 \ldots \beta_1\beta_0)_{256}$ имеет байты β_j, то мы можем вычислить

$$c = (a \cdot B_0^L) \oplus ((a \ll 8) \cdot (B_1^L + B_0^U)) \oplus ((a \ll 16) \cdot (B_2^L + B_1^U)) \oplus \cdots \oplus ((a \ll 56) \cdot (B_7^L + B_6^U)),$$

где $B_j^U = (q\beta_j)\ \&\ m$, $B_j^L = (((q\beta_j) \ll 8) + \beta_j)\ \&\ \overline{m}$, q = #0080402010080402 и m = #7f3f1f0f07030100. (Здесь $q\beta_j$ означает *обычное* умножение целых чисел.)

213. В данном вычислении с обратным порядком байтов регистр nn хранит $-n$, а регистр data указывает на октет, следующий в памяти за данными байтами $\alpha_{n-1} \ldots \alpha_1\alpha_0$ (первым идет байт α_{n-1}). Константы aa = #8381808080402010 и bb = #339bcf6530180c06 соответствуют матрицам A и B, найденным путем вычисления остатков $x^k \bmod p(x)$ для $72 \le k < 80$.

```
   SET  c,0          c ← 0.                      LDOU t,data,nn   t ← следующий октет.
   LDOU t,data,nn    t ← следующий октет.        XOR  u,u,c       u ← u ⊕ c.
   ADD  nn,nn,8      n ← n − 8.                  SLU  c,v,56      c ← v ≪ 56.
   BZ   nn,2F        Выполнено, если n = 0.      SRU  v,v,8       v ← v ≫ 8.
1H MXOR u,aa,t       u ← t · A.                  XOR  u,u,v       u ← u ⊕ v.
   MXOR v,bb,t       v ← t · B.                  XOR  t,t,u       t ← t ⊕ u.
   ADD  nn,nn,8      n ← n − 8.                  PBN  nn,1B       Повтор, если n > 0.  ▮
```

Аналогичный метод выполняет работу, не требуя при этом вспомогательной таблицы.

```
2H SET  nn,8         n ← 8.                      SRU  v,v,8       v ← v ≫ 8.
3H AND  x,t,ffooo    x ← старший байт.           XOR  t,t,v       t ← t ⊕ v.
   MXOR u,aaa,x      u ← x · A′.                 SUB  nn,nn,1     n ← n − 1.
   MXOR v,bbb,x      v ← x · B′.                 PBP  nn,3B       Повтор, если n > 0.
   SLU  t,t,8        t ← t ≪ 8.                  XOR  t,t,c       t ← t ⊕ c.
   XOR  t,t,u        t ← t ⊕ u.                  SRU  crc,t,48    Вернуть t ≫ 48.  ▮
```

Здесь `aaa` = #`8381808080808080`, `bbb` = #`0383c363331b0f05` и `ffooo` = #`ff00...00`.

...но книги Тупоконечников *давно запрещены...*

— ДЖОНАТАН СВИФТ (JONATHAN SWIFT)
Путешествия Гулливера (1726)

214. Рассматривая неразложимые делители характеристического полинома X, мы должны иметь $X^n = I$, где $n = 2^3 \cdot 3^2 \cdot 5 \cdot 7 \cdot 17 \cdot 31 \cdot 127 = 168\,661\,080$. Нейлл Клифт (Neill Clift) показал, что $l(n-1) = 33$, и нашел такую последовательность из 33 команд `MXOR` для вычисления $Y = X^{-1} = X^{n-1}$: `MXOR t,x,x; MXOR $1,t,x; MXOR $2,t,$1; MXOR $3,$2,$2; MXOR t,$3,$3;` S^6; `MXOR t,t,$2;` S^3; `MXOR $1,t,$1; MXOR t,$1,$3;` S^{13}; `MXOR t,t,$1;` S; `MXOR y,t,x;` здесь S обозначает '`MXOR t,t,t`'. Чтобы проверить, что X не является вырожденной матрицей, выполните `MXOR t,y,x` и сравните `t` с единичной матрицей #`8040201008040201`.

215. `SADD $0,x,0; SADD $1,x,a; NEG $0,32,$0; 2ADDU $1,$1,$0; SLU $0,b,$1;` затем `BN $0, Yes;` здесь `a` = #`aaaaaaaaaaaaaaaa` и `b` = #`2492492492492492`.

216. Начнем с $s_k \leftarrow 0$ и $t_k \leftarrow -1$ для $0 \le k < m$. Затем для $1 \le k \le m$ выполним следующее: если $x_k \ne 0$ и $x_k < 2^m$, установим $l \leftarrow \lambda x_k$ и $s_l \leftarrow s_l + x_k$; если $t_l < 0$ или $t_l > x_k$, установим также $t_l \leftarrow x_k$. Наконец установим $y \leftarrow 1$ и $k \leftarrow 0$; пока $y \ge t_k$ и $k < m$, будем устанавливать $y \leftarrow y + s_k$ и $k \leftarrow k + 1$. Для y и s_k достаточно n-битовой арифметики с двойной точностью. [Этот красивый алгоритм появился 22 марта 2008 года в блоге Д. Эппштейна (D. Eppstein).]

217. См. R. D. Cameron, *U.S. Patent 7400271* (15 July 2008); *Proc. ACM Symp. Principles and Practice of Parallel Programming* **13** (2008), 91–98.

218. Пусть b — любое целое число, обладающее тем свойством, что $b \bmod 8 = 5$. Тогда $x = b^{L(x)} \bmod 2^d$ для некоторого целого $L(x)$, зависящего от b, при $0 < x < 2^d$ и $x \bmod 4 = 1$ (см. раздел 3.2.1.2). Приведенный далее алгоритм вычисляет $s = 4L(x)$ для заданной таблицы чисел $t_k = -4L(2^k + 1)$ для $1 < k < d$, в предположении, что $t_k = 2^k$ для $k \ge d/2$. Установить $s \leftarrow 0$, $j \leftarrow 1$; затем, пока $j < d/2 - 1$, устанавливать $j \leftarrow j + 1$ и, если $x \,\&\, (1 \ll j) \ne 0$, устанавливать также $x \leftarrow (x + (x \ll j)) \bmod 2^d$, $s \leftarrow (s + t_j) \bmod 2^d$. Наконец установить $s \leftarrow (s + 1 - x) \bmod 2^d$.

Теперь, чтобы вычислить $a \cdot x^y$, можно действовать следующим образом (вся арифметика выполняется по модулю 2^d): если $x \,\&\, 2 \ne 0$, установить $x \leftarrow -x$ и $a \leftarrow (-1)^{y \& 1} a$. (Теперь $x \bmod 4 = 1$.) Установить $s \leftarrow 4L(x) \cdot y$, используя приведенный выше алгоритм, и $j \leftarrow 1$; затем, пока $s \ne 0$, устанавливать $j \leftarrow j + 1$ и, если $s \,\&\, (1 \ll j) \ne 0$, устанавливать

также $s \leftarrow s + t_j$, $a \leftarrow a + (a \ll j)$. Тогда требуемый ответ равен a. (При другом умножении мы можем вернуть $(1 - s)a$, если только $j \geq d/2$.)

Подходящие числа t_k можно вычислить путем установки $t_k \leftarrow 1 \ll k$ для $d - 1 \geq k \geq d/2$ и следующих действий для остальных k в порядке уменьшения: установить $x \leftarrow 1 + (1 \ll k)$, $x \leftarrow x + (x \ll k)$, $s \leftarrow 0$, $j \leftarrow k$; затем, пока $j < d/2 - 1$, устанавливать $j \leftarrow j + 1$ и, если $x \,\&\, (1 \ll j) \neq 0$, устанавливать также $x \leftarrow x + (x \ll j)$, $s \leftarrow s - t_j$; наконец установить $t_k \leftarrow (s + x - 1) \gg 1$. Например, при $d = 32$ мы получаем $t_{15} = {}^{\#}\texttt{20008000}$, $t_{14} = {}^{\#}\texttt{18004000}$, $t_{13} = {}^{\#}\texttt{0e002000}$, $t_{12} = {}^{\#}\texttt{07801000}$, $t_{11} = {}^{\#}\texttt{03e00800}$, $t_{10} = {}^{\#}\texttt{41f80400}$, $t_9 = {}^{\#}\texttt{18fe0200}$, $t_8 = {}^{\#}\texttt{0b7f8100}$, $t_7 = {}^{\#}\texttt{319fe080}$, $t_6 = {}^{\#}\texttt{5e8bf840}$, $t_5 = {}^{\#}\texttt{4a617e20}$, $t_4 = {}^{\#}\texttt{17c26f90}$, $t_3 = {}^{\#}\texttt{6119d1e8}$, $t_2 = {}^{\#}\texttt{2c30267c}$. (Эта процедура находит значения L для *некоторого* целого числа b без вычисления самого́ значения b!)

[Методы из данного упражнения имеют интересную связь с алгоритмами Бриггса (Briggs) и Фейнмана (Feynman) для *действительных* логарифмов и экспонент в упр. 1.2.2–25 и 1.2.2–28. Наша широкословная процедура для x^y работает также для вычисления обратного значения x по модулю 2^d, когда $y = -1$; но для этого есть и прямой алгоритм: установить $z \leftarrow 1$, $j \leftarrow 0$; пока $x \neq 1$, устанавливать $j \leftarrow j + 1$ и, если $x \,\&\, (1 \ll j) \neq 0$, устанавливать также $z \leftarrow (z + (z \ll j)) \bmod 2^d$, $x \leftarrow (x + (x \ll j)) \bmod 2^d$. Конечное значение z представляет собой обратное исходного нечетного числа x.]

РАЗДЕЛ 7.1.4

1. Вот как выглядят БДР для таблиц истинности 0000, 0001, ..., 1111 с указанием в нижней строке их размеров.

2. (Свойство упорядочения определяет направление каждой дуги.)

3. Имеются две функции с БДР размером 1 (а именно, две константные функции); ни одной размером 2 (поскольку два стока не могут быть оба достижимы при отсутствии узла ветвления); и $2n$ размером 3 (а именно, x_j и $\bar{x}_j$ для $1 \leq j \leq n$).

4. Установить $y \leftarrow {}^{\#}\texttt{0fffffeffffffe} \,\&\, \bar{x} + {}^{\#}\texttt{20000002}$, $y \leftarrow (y \gg 28) \,\&\, {}^{\#}\texttt{10000001}$, $x' \leftarrow x \oplus y$. (См. 7.1.3–(93).)

5. Вы получите $\overline{f(\bar{x}_1, \ldots, \bar{x}_n)} = f^D(x_1, \ldots, x_n)$, функцию, *дуальную* функции f (см. упр. 7.1.1–2).

6. Все наибольшие подтаблицы 1011000110010011, а именно 10110001, 10010011, 1011, 0001, 1001, 0011, являются различными бусинами; квадраты и дубликаты не будут наблюдаться, пока мы не доберемся до подтаблиц $\{10, 11, 00, 01\}$ длиной 2. Так что g имеет размер 11.

7. (а) Если таблица истинности f имеет вид $\alpha_0\alpha_1\ldots\alpha_{2^k-1}$, где каждое α_j представляет собой бинарную строку длиной 2^{n-k}, то таблица истинности g_k имеет вид $\beta_0\beta_2\ldots\beta_{2^k-2}$, где $\beta_{2j}=\alpha_{2j}\alpha_{2j+1}\alpha_{2j+1}\alpha_{2j+1}$.

(б) Таким образом, бусины f и g_k тесно связаны. Мы получаем БДР для g_k из БДР для f путем замены j на $j-1$ для $1\le j<k$ и замены .

8. (а) Теперь $\beta_{2j}=\alpha_{2j}\alpha_{2j+1}\alpha_{2j+1}\alpha_{2j}$. (б) Вновь заменим j на $j-1$ для $1\le j<k$. Если в f присутствует , но не , заменим на ; в противном случае заменим на . [E. Dubrova and L. Macchiarulo, *IEEE Trans.* **C-49** (2000), 1290–1292.]

9. Если $s=1$, решения нет. В противном случае установить $k\leftarrow s-1$, $j\leftarrow 1$ и многократно выполнять следующие шаги: (i) пока $j<v_k$, устанавливать $x_j\leftarrow 1$ и $j\leftarrow j+1$; (ii) остановиться, если $k=0$; (iii) если $h_k\ne 1$, установить $x_j\leftarrow 1$ и $k\leftarrow h_k$, в противном случае установить $x_j\leftarrow 0$ и $k\leftarrow l_k$; (iv) установить $j\leftarrow j+1$.

10. Пусть $I_k=(\bar v_k?\,l_k\colon h_k)$ для $0\le k<s$ и $I'_k=(\bar v'_k?\,l'_k\colon h'_k)$ для $0\le k<s'$. Можно считать, что $s=s'$; в противном случае $f\ne f'$. Приведенный далее алгоритм либо находит индексы $(t_0,\ldots,t_{s-1})$, такие, что I_k соответствует I'_{t_k}, либо заключает, что $f\ne f'$.

I1. [Инициализация и цикл.] Установить $t_{s-1}\leftarrow s-1$, $t_1\leftarrow 1$, $t_0\leftarrow 0$ и $t_k\leftarrow -1$ для $2\le k\le s-2$. Выполнить шаги I2–I4 для $k=s-1$, $s-2$, $\ldots$, 2 (в указанном порядке). Если эти шаги "выходят" в любой точке, мы имеем $f\ne f'$; в противном случае $f=f'$.

I2. [Проверка v_k.] Установить $t\leftarrow t_k$. (Теперь $t\ge 0$; в противном случае I_k не должно иметь предшественника.) Выйти, если $v'_t\ne v_k$.

I3. [Проверка l_k.] Установить $l\leftarrow l_k$. Если $t_l<0$, установить $t_l\leftarrow l'_t$; в противном случае выйти, если $l'_t\ne t_l$.

I4. [Проверка h_k.] Установить $h\leftarrow h_k$. Если $t_h<0$, установить $t_h\leftarrow h'_t$; в противном случае выйти, если $h'_t\ne t_h$. ▮

11. (а) Да, поскольку c_k корректно подсчитывает количество установок $x_{v_k}\ldots x_n$, ведущих из узла k в узел 1. (Фактически многие БДР-алгоритмы будут работать корректно (хотя и более медленно) при наличии эквивалентных узлов или избыточных ветвей. Однако приведенность важна, скажем, когда мы хотим быстро проверить равенство $f=f'$, как в упр. 10.)

(б) Нет. Предположим, например, что $I_3=(\bar 1?\,2\colon 1)$, $I_2=(\bar 1?\,0\colon 1)$, $I_1=(\bar 2?\,1\colon 1)$, $I_0=(\bar 2?\,0\colon 0)$; тогда алгоритм устанавливает $c_2\leftarrow 1$, $c_3\leftarrow\frac32$. (Однако см. упр. 35, (б).)

12. (а) Первое условие делает K независимым; второе делает его также максимальным.

(б) Нет при нечетном n; в противном случае имеется два множества чередующихся вершин.

(в) Вершина находится в ядре тогда и только тогда, когда это сток или когда она находится в ядре графа, полученного путем удаления всех стоковых вершин и их непосредственных предшественников.

[Ядра представляют выигрышные позиции в играх типа “Ним”, а также возникают в играх с n участниками. См. J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior* (1944), §30.1; C. Berge, *Graphs and Hypergraphs* (1973), Chapter 14.]

13. (а) Максимальная клика G представляет собой ядро $\overline{G}$, и наоборот. (б) Минимальное вершинное покрытие U представляет собой дополнение $V \setminus W$ ядра W, и наоборот (см. 7–(61)).

14. (а) Размер равен $4(n-2)+2[n=3]$. Когда $n \geq 6$, эти БДР образуют шаблон, в котором имеются четыре узла ветвления для переменных 4, 5, ..., $n-2$, а также фиксированный шаблон в верхней и нижней частях. Четыре ветви, по сути, представляют собой следующее.

(б) В этом случае ответами для $3 \leq n \leq 10$ являются $(7, 9, 14, 17, 22, 30, 37, 45)$; затем, как в (а), наблюдается фиксированный шаблон вверху и внизу с девятью узлами ветвления для каждой переменной в средине и общим размером $9(n-5)$. Девять узлов на каждом среднем уровне разделяются на три группы по три,

,

с одной группой для $x_1x_2=00$, другой — для $x_1x_2=01$ и третьей — для $x_1=1$.

15. Оба случая по индукции ведут к хорошо известным последовательностям чисел. (а) Числа Люка $L_n = F_{n+1} + F_{n-1} = \phi^n + \hat{\phi}^n$ [см. E. Lucas, *Théorie des Nombres* (1891), Chapter 18]. (б) Числа Перрина, определяемые следующим образом: $P_3 = 3$, $P_4 = 2$, $P_5 = 5$, $P_n = P_{n-2} + P_{n-3} = \chi^n + \hat{\chi}^n + \overline{\hat{\chi}}^n$. [См. E. Lucas, *Association Francaise pour l'Avancement des Sciences*, Compte-rendu **5** (1876), 62; R. Perrin, L'*Intermédiaire des Mathématiciens* **6** (1899), 76–77; Z. Füredi, *Journal of Graph Theory* **11** (1987), 463.]

16. Если БДР не является $\boxed{\bot}$, все решения генерируются вызовом $List(1, \mathrm{root})$, где $List(j, p)$ представляет собой следующую рекурсивную процедуру: если $v(p) > j$, установить $x_j \leftarrow 0$, вызвать $List(j+1, p)$, установить $x_j \leftarrow 1$ и вызвать $List(j+1, p)$. В противном случае посетить решение $x_1 \ldots x_n$, если p — стоковый узел $\boxed{\top}$. (Идея “посещения” комбинаторных объектов в процессе их генерации рассматривается в начале раздела 7.2.1.) Иначе установить $x_j \leftarrow 0$; вызвать $List(j+1, \mathrm{LO}(p))$, если $\mathrm{LO}(p) \neq \boxed{\bot}$; установить $x_j \leftarrow 1$; и вызвать $List(j+1, \mathrm{HI}(p))$, если $\mathrm{HI}(p) \neq \boxed{\bot}$.

Решения генерируются в лексикографическом порядке. Предположим, что всего их имеется N. Если k-е решение согласуется с $(k-1)$-м решением в позициях $x_1 \ldots x_{j-1}$, но не в x_j, то пусть $c(k) = n - j$; и пусть $c(1) = n$. Тогда время работы пропорционально $\sum_{k=1}^{N} c(k)$, которая в общем случае равна $O(nN)$. (Эта граница справедлива в силу того, что каждый узел ветвления БДР ведет как минимум к одному решению. Фактически на практике время работы обычно составляет $O(N)$.)

17. Это невозможно, поскольку имеется функция с $N = 2^{2k}$ и $B(f) = O(2^{2k})$, для которой каждые два решения отличаются более чем в 2^{k-1} битовых позициях. Время работы любого алгоритма, генерирующего все решения для такой функции, должно составлять $\Omega(2^{3k})$, поскольку между решениями требуется $\Omega(2^k)$ операций. Чтобы построить f, сначала положим

$$g(x_1, \ldots, x_k, y_0, \ldots, y_{2^k-1}) = \bigl[y_{(t_1 \ldots t_k)_2} = x_1t_1 \oplus \cdots \oplus x_kt_k \text{ для } 0 \leq t_1, \ldots, t_k \leq 1\bigr].$$

(Другими словами, g утверждает, что $y_0 \ldots y_{2^k-1}$ представляет собой строку $(x_1 \ldots x_k)_2$ матрицы Адамара; см. 4.6.4–(38).) Теперь положим $f(x_1, \ldots, x_k, y_0, \ldots, y_{2^k-1}, x'_1, \ldots, x'_k, y'_0, \ldots, y'_{2^k-1}) = g(x_1, \ldots, x_k, y_0, \ldots, y_{2^k-1}) \wedge g(x'_1, \ldots, x'_k, y'_0, \ldots, y'_{2^k-1})$. Ясно, что при упорядочении переменных таким способом $B(f) = O(2^{2k})$. Действительно, Т. Далхаймер (T. Dahlheimer) заметил, что $B(f) = 2B(g) - 2$, где $B(g) = 2^k + 1 + \sum_{j=1}^{2^k} 2^{\min(k, 1+\lceil \lg j \rceil)} = \frac{5}{3}2^{2k-1} + 2^k + \frac{5}{3}$.

18. Сначала $(W_1, \ldots, W_5) = (5, 4, 4, 4, 0)$. Затем $m_2 = w_4 = 4$ и $t_2 = 1$; $m_3 = t_3 = 0$; $m_4 = \max(m_3, m_2 + w_3) = 1$, $t_4 = 1$; $m_5 = W_4 - W_5 = 4$, $t_5 = 0$; $m_6 = w_2 + W_3 - W_5 = 2$, $t_6 = 1$; $m_7 = \max(m_5, m_4 + w_2) = 4$, $t_7 = 0$; $m_8 = \max(m_7, m_6 + w_1) = 4$, $t_8 = 0$. Окончательным решением является $x_1x_2x_3x_4 = 0001$.

19. $\sum_{j=1}^n \min(w_j, 0) \le \sum_{j=v_k}^n \min(w_j, 0) \le m_k \le \sum_{j=v_k}^n \max(w_j, 0) = W_{v_k} \le W_1$.

20. Установить $w_1 \leftarrow -1$, затем $w_{2j} \leftarrow w_j$ и $w_{2j+1} \leftarrow -w_j$ для $1 \le j \le n/2$. [Этот метод может также вычислить w_{n+1}. Последовательность названа так из-за работ А. Тью (A. Thue) *Skrifter udgivne af Videnskabs-Selskabet i Christiania*, Mathematisk-Naturvidenskabelig Klasse (1912), No. 1, §7, и Г. М. Морзе (H. M. Morse), *Trans. Amer. Math. Soc.* **22** (1921), 84–100, §14.]

21. Да; мы просто должны изменить знак каждого веса w_j. (Можно также изменить роли LO и HI в каждой вершине.)

22. Если $f(x) = f(x') = 1$, когда f представляет ядро графа, расстояние Хэмминга $\nu(x \oplus x')$ не может быть равно 1. В таких случаях БДР для f имеет $v_l = v + 1$ при $l \ne 0$ и $v_h = v + 1$ при $h \ne 0$.

23. БДР для функции связности любого связного графа будет иметь ровно $n-1$ сплошных дуг на каждом пути от корня к $\boxed{\top}$, поскольку такое количество ребер необходимо для связи n вершин и потому что БДР не имеет избыточных ветвей. (См. также теорему S.)

24. Применение алгоритма B с весами $(w'_{12}, \ldots, w'_{89}) = (-w_{12} - x, \ldots, -w_{89} - x)$, где x достаточно велико, чтобы сделать все новые веса w'_{uv} отрицательными. Максимум суммы $\sum w'_{uv} x_{uv}$ будет достигнут с $\sum x_{uv} = 8$, и эти ребра будут образовывать остовное дерево с наименьшей $\sum w_{uv} x_{uv}$. (Мы видели лучший алгоритм для поиска наименьших остовных деревьев в упр. 2.3.4.1–11, а другие методы решения этой задачи будут рассматриваться в разделе 7.5.4. Однако это упражнение указывает, что БДР может компактно представлять множество *всех* остовных деревьев.)

25. Ответом на шаге C1 следует сделать $(1+z)^{v_{s-1}-1} c_{s-1}$; значение c_k на шаге C2 становится равным $(1+z)^{v_l - v_k - 1} c_l + (1+z)^{v_h - v_k - 1} z c_h$.

26. В этом случае ответом на шаге C1 становится просто c_{s-1}; а значением c_k на шаге C2 является просто $(1 - p_{v_k}) c_l + p_{v_k} c_h$.

27. Мультилинейный полином $H(x_1, \ldots, x_n) = F(x_1, \ldots, x_n) - G(x_1, \ldots, x_n)$ является ненулевым по модулю q, поскольку он представляет собой ± 1 для некоторого выбора целых чисел, при котором каждое $x_k \in \{0, 1\}$. Если он имеет степень d (по модулю q), можно доказать, что имеется как минимум $(q-1)^d q^{n-d}$ множеств значений $(q_1, \ldots, q_n)$ с $0 \le q_k < q$, таких, что $H(q_1, \ldots, q_n) \bmod q \ne 0$. Это утверждение очевидно при $d = 0$. А если x_k является переменной, появляющейся в члене степени $d > 0$, коэффициент x_k представляет собой полином степени $d - 1$, который по индукции по d является ненулевым для как минимум $(q-1)^{d-1} q^{n-d}$ выборок $(q_1, \ldots, q_{k-1}, q_{k+1}, \ldots, q_n)$; для каждой из этих выборок имеется $q - 1$ значений q_k, таких, что $H(q_1, \ldots, q_n) \bmod q \ne 0$.

Следовательно, указанная вероятность равна $\ge (1 - 1/q)^d \ge (1 - 1/q)^n$. [См. M. Blum, A. K. Chandra and M. N. Wegman, *Information Processing Letters* **10** (1980), 80–82.]

28. $F(p) = (1-p)^n G(p/(1-p))$. Аналогично $G(z) = (1+z)^n F(z/(1+z))$.

29. На шаге C1 установить также $c'_0 \leftarrow 0$, $c'_1 \leftarrow 0$; вернуть c_{s-1} и c'_{s-1}. На шаге C2 установить $c_k \leftarrow (1-p)c_l + pc_h$ и $c'_k \leftarrow (1-p)c'_l - c_l + pc'_h + c_h$.

30. Эту работу выполняет следующий аналог алгоритма B (в предположении точной арифметики).

A1. [Инициализация.] Установить $P_{n+1} \leftarrow 1$ и $P_j \leftarrow P_{j+1}\max(1-p_j, p_j)$ для $n \ge j \ge 1$.

A2. [Цикл по k.] Установить $m_1 \leftarrow 1$ и выполнить шаг A3 для $2 \le k < s$. Затем выполнить шаг A4.

A3. [Обработка I_k.] Установить $v \leftarrow v_k$, $l \leftarrow l_k$, $h \leftarrow h_k$, $t_k \leftarrow 0$. Если $l \ne 0$, установить $m_k \leftarrow m_l(1-p_v)P_{v+1}/P_{v_l}$. Затем, если $h \ne 0$, вычислить $m \leftarrow m_h p_v P_{v+1}/P_{v_h}$; и если $l = 0$ или $m > m_k$, установить $m_k \leftarrow m$ и $t_k \leftarrow 1$.

A4. [Вычисление x.] Установить $j \leftarrow 0$, $k \leftarrow s-1$ и выполнять следующие операции, пока не будет выполнено условие $j = n$: пока $j < v_k - 1$, устанавливать $j \leftarrow j+1$ и $x_j \leftarrow [p_j > \frac{1}{2}]$; если $k > 1$, установить $j \leftarrow j+1$ и $x_j \leftarrow t_k$ и $k \leftarrow (t_k{=}0?\ l_k{:}\ h_k)$. ▌

31. C1′. [Цикл по k.] Установить $\alpha_0 \leftarrow \bot$, $\alpha_1 \leftarrow \top$ и выполнить шаг C2′ для $k = 2, 3, \ldots, s-1$. Затем перейти к шагу C3′.

C2′. [Вычисление α_k.] Установить $v \leftarrow v_k$, $l \leftarrow l_k$ и $h \leftarrow h_k$. Установить $\beta \leftarrow \alpha_l$ и $j \leftarrow v_l - 1$; затем, пока $j > v$, устанавливать $\beta \leftarrow (\bar{x}_j \circ x_j) \bullet \beta$ и $j \leftarrow j-1$. Установить $\gamma \leftarrow \alpha_h$ и $j \leftarrow v_h - 1$; затем, пока $j > v$, устанавливать $\gamma \leftarrow (\bar{x}_j \circ x_j) \bullet \gamma$ и $j \leftarrow j-1$. Наконец установить $\alpha_k \leftarrow (\bar{x}_v \bullet \beta) \circ (x_v \bullet \gamma)$.

C3′. [Завершение.] Установить $\alpha \leftarrow \alpha_{s-1}$ и $j \leftarrow v_{s-1} - 1$; затем, пока $j > 0$, устанавливать $\alpha \leftarrow (\bar{x}_j \circ x_j) \bullet \alpha$ и $j \leftarrow j-1$. Вернуть ответ α. ▌

Этот алгоритм выполняет операции $\circ$ и $\bullet$ не более $O(nB(f))$ раз. Верхнюю границу часто можно понизить до $O(n) + O(B(f))$; но сокращения наподобие вычисления W_k на шаге B1 не всегда доступны. [См. O. Coudert and J. C. Madre, *Proc. Reliability and Maint. Conf.* (IEEE, 1993), 240–245, §4; O. Coudert, *Integration* **17** (1994), 126–127.]

32. В упр. 25 '$\circ$' представляет собой сложение, '$\bullet$' — умножение, '$\bot$' равно 0, '$\top$' равно 1, '$\bar{x}_j$' равно 1, 'x_j' — z. Упражнение 26 аналогично, но '$\bar{x}_j$' равно $1-p_j$, а 'x_j' — p_j.

В упр. 29 объектами алгебры являются пары (c, c'), и мы имеем $(a, a') \circ (b, b') = (a + b, a' + b')$, $(a, a') \bullet (b, b') = (ab, ab' + a'b)$. Кроме того, '$\bot$' представляет собой $(0, 0)$, '$\top$' — $(1, 0)$, '$\bar{x}_j$' — $(1-p, -1)$, а 'x_j' — $(p, 1)$.

В упр. 30 '$\circ$' представляет собой операцию max, '$\bullet$' — умножение, '$\bot$' — $-\infty$, '$\top$' — 1, '$\bar{x}_j$' равно $1-p_j$, а 'x_j' — p_j. Операция умножения является дистрибутивной относительно операции max, поскольку величины являются либо неотрицательными, либо $-\infty$; для удовлетворения (22) мы должны определить $0 \bullet (-\infty) = -\infty$.

(Имеется много других возможностей в связи с повсеместной распространенностью в математике ассоциативных и дистрибутивных операторов. Алгебраические объекты не обязаны быть числами, полиномами или парами; это могут быть строки, матрицы, функции, множества чисел, множества строк, множества или мультимножества матриц пар функций от строк и т. д., и т. п. Множество других примеров мы увидим в разделе 7.3. Особенно важной оказывается min-плюс-алгебра, в которой $\circ = \min$ и $\bullet = +$, и мы могли бы воспользоваться ею в упр. 21 или 24. Ее часто называют *тропической*, неявно подчеркивая роль бразильского математика Имре Саймона (Imre Simon.)

33. Следует работать с тройками (c, c', c''), для которых $(a, a', a'') \circ (b, b', b'') = (a + b, a' + b', a'' + b'')$ и $(a, a', a'') \bullet (b, b', b'') = (ab, a'b + b'a, a''b + 2a'b' + ab'')$. Интерпретируем '$\bot$' как $(0, 0, 0)$, '$\top$' — как $(1, 0, 0)$, '$\bar{x}_j$' — как $(1, 0, 0)$ и 'x_j' — как $(1, w_j, w_j^2)$.

34. Пусть $x \vee y = \max(x, y)$. Работаем с парами (c, c'), для которых $(a, a') \circ (b, b') = (a \vee b, a' \vee b')$ и $(a, a') \bullet (b, b') = (a + b, (a' + b) \vee (a + b'))$. Интерпретируем '$\bot$' как $(-\infty, -\infty)$, '$\top$' — как $(0, -\infty)$, '$\bar{x}_j$' — как $(0, w''_j)$ и 'x_j' — как $(w_j, w'_j + w''_j)$. Первый компонент результата будет соответствовать результату алгоритма B; второй компонент — искомый максимум.

35. (а) Предложенная FBDD может быть представлена инструкциями $I_{s-1}, \ldots, I_0$, как в алгоритма C. Начнем с $R_0 \leftarrow R_1 \leftarrow \emptyset$, затем выполним следующие действия для $k = 2, \ldots, s - 1$: сообщить об ошибке, если $v_k \in R_{l_k} \cup R_{h_k}$; в противном случае установить $R_k \leftarrow \{v_k\} \cup R_{l_k} \cup R_{h_k}$. (Множество R_k идентифицирует все переменные, достижимые из I_k.)

(б) Полином надежности может быть вычислен так же, как и в ответе к упр. 26. Для подсчета решений мы по сути устанавливаем $p_1 = \cdots = p_n = \frac{1}{2}$ и выполняем умножение на 2^n: начнем с $c_0 \leftarrow 0$ и $c_1 \leftarrow 2^n$, затем установим $c_k \leftarrow (c_{l_k} + c_{h_k})/2$ для $1 < k < s$. Искомый ответ — c_{s-1}.

36. Вычислите множества R_k, как в ответе к упр. 35, (а). Вместо цикла по j, как указано на шаге C2′ в ответе к упр. 31, установите $\beta \leftarrow \alpha_l$, а затем $\beta \leftarrow (\bar{x}_j \circ x_j) \bullet \beta$ для всех $j \in R_k \setminus R_l \setminus \{v\}$; γ обрабатывается точно так же. Аналогично на шаге C3′ установите $\alpha \leftarrow (\bar{x}_j \circ x_j) \bullet \alpha$ для всех $j \notin R_{s-1}$.

37. Для любой заданной FBDD для f функция $G(z)$ представляет собой сумму $(1 + z)^{n - \text{длина} P} z^{\text{сплошные дуги в } P}$ по всем путям P от корня к $\boxed{\top}$. [См. *Theoretical Comp. Sci.* **3** (1976), 371–384.]

38. Ключевой факт состоит в том, что $x_j = 1$ обеспечивает $f = 1$ тогда и только тогда, когда мы имеем (i) $h_k = 1$ при $v_k = j$; (ii) $v_k = j$ как минимум на одном шаге k; (iii) не имеется шагов с ($v_k < j < v_{l_k}$ и $l_k \neq 1$) или ($v_k < j < v_{h_k}$ и $h_k \neq 1$).

K1. [Инициализация.] Установить $t_j \leftarrow 2$ и $p_j \leftarrow 0$ для $1 \le j \le n$.

K2. [Проверка всех ветвлений.] Выполнить следующие операции для $2 \le k < s$. Установить $j \leftarrow v_k$ и $q \leftarrow 0$. Если $l_k = 1$, установить $q \leftarrow -1$; в противном случае установить $p_j \leftarrow \max(p_j, v_{l_k})$. Если $h_k = 1$, установить $q \leftarrow +1$; в противном случае установить $p_j \leftarrow \max(p_j, v_{h_k})$. Если $t_j = 2$, установить $t_j \leftarrow q$; в противном случае при $t_j \neq q$ установить $t_j \leftarrow 0$.

K3. [Завершение.] Установить $m \leftarrow v_{s-1}$ и выполнить следующие действия для $j = 1, 2, \ldots, n$. Если $j < m$, установить $t_j \leftarrow 0$; затем, если $p_j > m$, установить $m \leftarrow p_j$. ▮

[См. S.-W. Jeong and F. Somenzi, *Logic Synthesis and Optimization* (1993), 154–156.]

39. $k(n + 1 - k) + 2$ для $1 \le k \le n$. (См. (26).)

40. (а) Предположим, что бинарные диаграммы решений для f и g имеют соответственно a_j и b_j узлов ветвления $\textcircled{j}$ для $1 \le j \le n$. Каждая подтаблица f порядка $n + 1 - k$ имеет вид $\alpha\beta\gamma\delta$, где α, β, γ и δ представляют собой подтаблицы порядка $n - 1 - k$. Соответствующими подтаблицами g являются $\alpha\alpha\delta\delta$; следовательно, они представляют собой бусины тогда и только тогда, когда $\alpha \neq \delta$, и в этом случае бусиной является либо $\alpha\beta\gamma\delta$, либо $\alpha\beta = \gamma\delta$. Следовательно, $b_k \le a_k + a_{k+1}$ и $b_{k+1} = 0$. Мы также имеем $b_j \le a_j$ для $1 \le j < k$, поскольку каждая бусина g порядка $> n + 1 - k$ "сжата" как минимум из одной такой бусины f. А $b_j \le a_j$ для $j > k + 1$, поскольку подтаблицы $(x_{k+2}, \ldots, x_n)$ идентичны, хотя могут и отсутствовать в g.

(б) Не всегда. Простейшим контрпримером является $f(x_1, x_2, x_3, x_4) = x_2 \wedge (x_3 \vee x_4)$, $h(x_1, x_2, x_1, x_4) = x_2 \wedge (x_1 \vee x_4)$, когда $B(f) = 5$ и $B(h) = 6$. (Однако мы всегда имеем $B(h) < 2B(f)$.)

41. (a) $3n-3$; (b) $2n$. (Общий шаблон проиллюстрирован здесь для $n=6$. Можно также показать, что "порядок органных труб" $\langle\, x_n^{F_1} x_1^{F_2} x_{n-1}^{F_3} x_2^{F_4} \ldots x_{\lfloor n/2\rfloor+[n\text{ четно}]}^{F_{n-1}}\; x_{\lceil n/2\rceil}^{F_{n-2}} \,\rangle$ дает профиль 1, 2, 4, ..., $2\lceil n/2\rceil-2$, $2\lfloor n/2\rfloor-1$, ..., 5, 3, 1, 2, приводя к общему размеру БДР $\binom{n}{2}+3$; похоже, что этот порядок является наихудшим для весов Фибоначчи.)

Функции $[F_n x_1 + \cdots + F_1 x_n \ge t]$ были изучены Й. Т. Батлером (J. T. Butler) и Ц. Сасао (T. Sasao), *Fibonacci Quart.* **34** (1996), 413–422.

42. (Ср. с упр. 2.) Шестнадцатью корнями являются узлы (1) и два стока.

43. (a) Поскольку $f(x_1,\ldots,x_{2n})$ — симметричная функция $S_n(x_1,\ldots,x_n,\bar x_{n+1},\ldots,\bar x_{2n})$, мы имеем $B(f)=1+2+\cdots+(n{+}1)+\cdots+3+2+2=n^2+2n+2$.

(б) Из соображений симметрии размер является тем же для $[\sum\{x_i \mid i\in I\} = \sum\{x_i \mid i\notin I\}]$, $|I|=n$.

44. Имеется не более $\min(k, 2^{n+2-k}-2)$ узлов, помеченных (k), для $1\le k\le n$, поскольку имеется $2^{n+2-k}-2$ симметричных функций от $(x_k,\ldots,x_n)$, не являющихся константами. Таким образом, Σ_n не превышает $2+\sum_{k=1}^n \min(k, 2^{n+2-k}-2)$, что можно выразить в аналитическом виде как $(n+2-b_n)(n+1-b_n)/2+2(2^{b_n}-b_n)$, где $b_n=\lambda(n+4-\lambda(n+4))$ и $\lambda n=\lfloor \lg n\rfloor$.

Симметричная функция, достигающая границы для наихудшего случая, может быть построена следующим образом (связанным с циклами де Брейна, строившимися в упр. 3.2.2–7). Пусть $p(x)=x^d+a_1x^{d-1}+\cdots+a_d$ — примитивный полином по модулю 2. Установим $t_k\leftarrow 1$ для $0\le k<d$; $t_k\leftarrow(a_1t_{k-1}+\cdots+a_dt_{k-d}) \bmod 2$ для $d\le k<2^d+d-2$; $t_k\leftarrow(1+a_1t_{k-1}+\cdots+a_dt_{k-d}) \bmod 2$ для $2^d+d-2\le k<2^{d+1}+d-3$; и $t_{2^{d+1}+d-3}\leftarrow 1$. Например, когда $p(x)=x^3+x+1$, мы получаем $t_0\ldots t_{16}=11100101101000111$.

Тогда (i) последовательность $t_1\ldots t_{2^d+d-3}$ содержит все d-кортежи за исключением 0^d и 1^d в качестве подстрок; (ii) последовательность $t_{2^d+d-2}\ldots t_{2^{d+1}+d-4}$ представляет собой циклический сдвиг $\bar t_0\ldots\bar t_{2^d-2}$; и (iii) $t_k=1$ для $2^d-1\le k\le 2^d+d-3$ и $2^{d+1}-2\le k\le 2^{d+1}+d-3$. Значит, последовательность $t_0\ldots t_{2^{d+1}+d-3}$ содержит все $(d+1)$-кортежи за исключением 0^{d+1} и 1^{d+1} в качестве подстрок. Установим $f(x)=t_{\nu x}$, чтобы максимизировать $B(f)$ при $2^d+d-4<n\le 2^{d+1}+d-3$.

Асимптотически $\Sigma_n=\frac12 n^2-n\lg n+O(n)$. [См. I. Wegener, *Information and Control* **62** (1984), 129–143; M. Heap, *J. Electronic Testing* **4** (1993), 191–195.]

45. Модуль M_1 имеет только три входа, (x_1,y_1,z_1), и только три выхода, $u_2=x_1$, $v_2=y_1x_1$, $w_2=z_1x_1$. Модуль M_{n-1} почти нормальный, но не имеет входного порта для z_{n-1} и не выводит u_n; он устанавливает $z_{n-2}=x_{n-1}y_{n-1}$. Модуль M_n имеет только три входа, (v_n,w_n,x_n), и один выход, $y_{n-1}=x_n$, вместе с основным выходом $w_n\vee v_nx_n$. При таких определениях зависимости между портами образуют ациклический ориентированный граф.

(Модули могут быть построены со всеми $b_k = 0$ и $a_k \le 5$ или даже с $a_k \le 4$, как мы увидим в упр. 47. Но (33) и (34) предназначены для иллюстрации обратных сигналов на простом примере, а не для демонстрации наиболее плотно сжатой конструкции.)

46. Для $6 \le k \le n-3$ имеется девять ветвей в k, соответствующих трем случаям, $(\bar{x}_1, x_1\bar{x}_2, x_1x_2)$, умноженным на три случая $(\bar{x}_{k-1}, \bar{x}_{k-2}x_{k-1}, \bar{x}_{k-3}x_{k-2}x_{k-1})$. Общий размер БДР оказывается равным $9n-38$, если $n \ge 6$.

47. Предположим, что f имеет q_k подтаблиц порядка $n-k$, так что ее КДР имеет q_k узлов с ветвлениями в x_{k+1}. Их можно закодировать с помощью $a_k = \lceil \lg q_k \rceil$ бит и построить модуль M_{k+1}, такой, что $b_k = b_{k+1} = 0$, который имитирует поведение этих q_k узлов ветвления. Таким образом, согласно (86)

$$\sum_{k=0}^{n} 2^{a_k 2^{b_k}} = \sum_{k=0}^{n} 2^{\lceil \lg q_k \rceil} \le \sum_{k=0}^{n} (2q_k - 1) = 2Q(f) - (n+1) \le (n+1)B(f).$$

(2^m-канальный мультиплексор показывает, что необходим дополнительный множитель $(n+1)$; в действительности теорема M дает верхнюю границу $Q(f)$.)

48. Суммы $u_k = x_1 + \cdots + x_k$ и $v_k = x_{k+1} + \cdots + x_n$ могут быть представлены $1 + \lambda k$ и $1 + \lambda(n-k)$ проводами соответственно. Пусть $t_k = x_k \wedge [u_k + v_k = k]$ и $w_k = t_1 \vee \cdots \vee t_k$. Можно построить модули M_k, имеющие входы u_{k-1} и w_{k-1} из M_{k-1} вместе со входами v_k из M_{k+1}; модуль M_k выводит $u_k = u_{k-1} + x_k$ и $w_k = w_{k-1} \vee t_k$ для модуля M_{k+1}, а также $v_{k-1} = v_k + x_k$ для M_{k-1}.

Если p представляет собой полином, $\sum_{k=0}^{n} 2^{p(a_k, b_k)} = 2^{(\log n)^{O(1)}}$ асимптотически меньше $2^{\Omega(n)}$. [См. K. L. McMillan, *Symbolic Model Checking* (1993), §3.5, где была опубликована теорема M с расширениями для нелинейных схем. Частный случай $b_1 = \cdots = b_n = 0$ до того был упомянут в работе C. L. Berman, *IEEE Trans.* **CAD-10** (1991), 1059–1066.]

49.

[См. I. Semba and S. Yajima, *Trans. Inf. Proc. Soc. Japan* **35** (1994), 1663–1665.]

50.

51. В этом случае $B(f_j) = 3j + 2$ для $1 \le j \le n$ и $B(f_{n+1}) = 3n + 1$; так что отдельные БДР оказываются только около 1/3 по размеру по сравнению с (36). Однако при этом почти нет совместно используемых узлов — только стоки и одно ветвление. Так что общий размер БДР составляет $(3n^2 + 9n)/2$.

52. Если база БДР для $\{f_1, \ldots, f_m\}$ имеет s узлов, то $B(f) = s + m + 1 + [s = 1]$.

53. Назовем узлы ветвления a, b, c, d, e, f, g, при этом $\texttt{ROOT} = a$. После выполнения шага R1 мы имеем $\texttt{HEAD[1]} = {\sim}a$, $\texttt{AUX}(a) = {\sim}0$; $\texttt{HEAD[2]} = {\sim}b$, $\texttt{AUX}(b) = {\sim}c$, $\texttt{AUX}(c) = {\sim}0$; $\texttt{HEAD[3]} = {\sim}d$, $\texttt{AUX}(d) = {\sim}e$, $\texttt{AUX}(e) = {\sim}f$, $\texttt{AUX}(f) = {\sim}g$, $\texttt{AUX}(g) = {\sim}0$.

После шага R3 с $v = 3$ мы имеем $s = {\sim}0$, $\texttt{AUX}(0) = {\sim}e$, $\texttt{AUX}(e) = f$, $\texttt{AUX}(f) = 0$; также $\texttt{AVAIL} = g$, $\texttt{LO}(g) = {\sim}1$, $\texttt{HI}(g) = d$, $\texttt{LO}(d) = {\sim}0$ и $\texttt{HI}(d) = \alpha$, где α — начальное значение AVAIL. (Узлы g и d были переработаны в пользу 1 и 0.) Затем шаг R4 устанавливает $s \leftarrow e$ и $\texttt{AUX}(0) \leftarrow 0$. (Оставшиеся узлы с $\texttt{V} = v$, начиная с s, связываются посредством AUX.)

Теперь R7, начиная с $p = q = e$ и $s = 0$, устанавливает $\texttt{AUX}(1) \leftarrow {\sim}e$, $\texttt{LO}(f) \leftarrow {\sim}e$, $\texttt{HI}(f) \leftarrow g$, $\texttt{AVAIL} \leftarrow f$; а R8 выполняет сброс $\texttt{AUX}(1) \leftarrow 0$.

Затем шаг R3 с $v = 2$ устанавливает $\texttt{LO}(b) \leftarrow 0$, $\texttt{LO}(c) \leftarrow e$ и $\texttt{HI}(c) \leftarrow 1$. Никаких других важных изменений нет, хотя некоторые поля AUX временно становятся отрицательными. Заканчиваются вычисления рис. 21.

54. Создаем узлы j для $1 < j \le 2^{n-1}$ путем установок $\texttt{V}(j) \leftarrow \lceil \lg j \rceil$, $\texttt{LO}(j) \leftarrow 2j - 1$ и $\texttt{HI}(j) \leftarrow 2j$; а также для $2^{n-1} < j \le 2^n$ путем установок $\texttt{V}(j) \leftarrow n$, $\texttt{LO}(j) \leftarrow f(x_1, \ldots, x_{n-1}, 0)$ и $\texttt{HI}(j) \leftarrow f(x_1, \ldots, x_{n-1}, 1)$ при $j = (1x_1 \ldots x_{n-1})_2 + 1$. Затем применим алгоритм R с $\texttt{ROOT} = 2$. (Можно опустить шаг R1 с помощью первоначальной установки $\texttt{AUX}(j) \leftarrow -j$ для $4 \le j \le 2^n$, а затем $\texttt{HEAD}[k] \leftarrow {\sim}(2^k)$ и $\texttt{AUX}(2^{k-1} + 1) \leftarrow -1$ для $1 \le k \le n$.)

55. Достаточно построить неприведенную диаграмму, поскольку после этого алгоритм R завершит работу. Пронумеруем вершины $1, \ldots, n$ таким образом, что никакая вершина, за исключением 1, не появляется до всех ее соседних вершин. Представим ребра дугами $a_1, \ldots, a_e$, где a_k представляет собой $u_k \longrightarrow v_k$ для некоторого $u_k < v_k$ и где дуги, имеющие $u_k = j$, являются последовательными, с $s_j \le k < s_{j+1}$ и $1 = s_1 \le \cdots \le s_n = s_{n+1} = e+1$. Определим "границу" $V_k = \{1, v_1, \ldots, v_k\} \cap \{u_k, \ldots, n\}$ для $1 \le k \le e$ и положим $V_0 = \{1\}$. Неприведенная диаграмма решений будет иметь ветви, связанные с дугой a_k, для всех разбиений V_{k-1}, соответствующих отношениям связности, возникающим из-за предыдущих ветвей.

Рассмотрим, например, $P_3 \,\square\, P_3$, где $(s_1, \ldots, s_{10}) = (1, 3, 5, 7, 8, 10, 11, 12, 13, 13)$ и $V_0 = \{1\}$, $V_1 = \{1, 2\}$, $V_2 = \{1, 2, 3\}$, $V_3 = \{2, 3, 4\}$, $\ldots$, $V_{12} = \{8, 9\}$. Ветвь, связанная с дугой a_1, идет от тривиального разбиения 1 множества V_0 к разбиению $1|2$ множества V_1, если $1 \not\!\!-\!\!\!- 2$, или к разбиению 12, если $1 \!-\!\!\!- 2$. (Запись '$1|2$' обозначает разбиение множества $\{1\} \cup \{2\}$, как в разделе 7.2.1.5.) Из $1|2$ ветвь, связанная с a_2, идет к разбиению $1|2|3$ множества V_2, если $1 \not\!\!-\!\!\!- 3$, в противном случае — к $13|2$; из 12 ветви идут соответственно к разбиениям $12|3$ и 123. Из $1|2|3$ обе ветви, связанные с a_3, идут к $\boxed{\perp}$, поскольку вершина 1 больше не может быть связана с другими. И т. д. В конечном итоге разбиения множества $V_e = V_{12}$ идентифицируются с $\boxed{\perp}$, за исключением тривиального разбиения, состоящего из одного множества, которое соответствует $\boxed{\top}$.

56. Начнем с $m \leftarrow 2$ на шаге R1 и $v_0 \leftarrow v_1 \leftarrow v_{\max} + 1$, $l_0 \leftarrow h_0 \leftarrow 0$, $l_1 \leftarrow h_1 \leftarrow 1$, как в (8). Будем считать, что $\texttt{HI}(0) = 0$ и $\texttt{HI}(1) = 1$. Опустим присваивания, включающие AVAIL, на шагах R3 и R7. После установки $\texttt{AUX}(\texttt{HI}(p)) \leftarrow 0$ на шаге R8 следует также установить $v_m \leftarrow v$, $l_m \leftarrow \texttt{HI}(\texttt{LO}(p))$, $h_m \leftarrow \texttt{HI}(\texttt{HI}(p))$, $\texttt{HI}(p) \leftarrow m$ и $m \leftarrow m + 1$. В конце шага R9 нужно установить $s \leftarrow m - [\texttt{ROOT} = 0]$.

57. Установить $\texttt{LO}(\texttt{ROOT}) \leftarrow {\sim}\texttt{LO}(\texttt{ROOT})$. (Мы кратко дополняем поле LO узлов, которые остаются доступными после ограничения.) Затем для $v = \texttt{V}(\texttt{ROOT})$, $\ldots$, $v_{\max}$ установить $p \leftarrow {\sim}\texttt{HEAD}[v]$, $\texttt{HEAD}[v] \leftarrow {\sim}0$ и выполнять следующие действия до тех пор, пока $p \ne 0$. (i) Установить $p' \leftarrow {\sim}\texttt{AUX}(p)$. (ii) Если $\texttt{LO}(p) \ge 0$, установить $\texttt{HI}(p) \leftarrow \texttt{AVAIL}$, $\texttt{AUX}(p) \leftarrow 0$ и $\texttt{AVAIL} \leftarrow p$ (узел p больше не может быть достигнут). В противном случае установить $\texttt{LO}(p) \leftarrow {\sim}\texttt{LO}(p)$; если $\texttt{FIX}[v] = 0$, установить $\texttt{HI}(p) \leftarrow \texttt{LO}(p)$; если $\texttt{FIX}[v] = 1$, установить $\texttt{LO}(p) \leftarrow \texttt{HI}(p)$; если $\texttt{LO}(\texttt{LO}(p)) \ge 0$, установить $\texttt{LO}(\texttt{LO}(p)) \leftarrow {\sim}\texttt{LO}(\texttt{LO}(p))$; если $\texttt{LO}(\texttt{HI}(p)) \ge 0$, установить $\texttt{LO}(\texttt{HI}(p)) \leftarrow {\sim}\texttt{LO}(\texttt{HI}(p))$; и установить $\texttt{AUX}(p) \leftarrow \texttt{HEAD}[v]$,

$\texttt{HEAD}[v] \leftarrow \sim p$. (iii) Установить $p \leftarrow p'$. Наконец, по завершении цикла по v, восстановить $\texttt{LO}(0) \leftarrow 0$, $\texttt{LO}(1) \leftarrow 1$.

58. Поскольку $l \neq h$ и $l' \neq h'$, мы имеем $l \diamond l' \neq h \diamond h'$, $l \diamond \alpha' \neq h \diamond \alpha'$ и $\alpha \diamond l' \neq \alpha \diamond h'$.

Предположим, что $\alpha \diamond \alpha' = \beta \diamond \beta'$, где $\beta = (v'', l'', h'')$ и $\beta' = (v''', l''', h''')$. Если $v'' = v'''$, мы имеем $v = v''$, $l \diamond l' = l'' \diamond l'''$ и $h \diamond h' = h'' \diamond h'''$. Если $v'' < v'''$, мы имеем $v = v''$, $l \diamond \alpha' = l'' \diamond \beta'$ и $h \diamond \alpha' = h'' \diamond \beta'$. В противном случае мы имеем $v' = v'''$, $\alpha \diamond l' = \beta \diamond l'''$ и $\alpha \diamond h' = \beta \diamond h'''$. Следовательно, по индукции $\alpha = \beta$ и $\alpha' = \beta'$ во всех случаях.

59. (а) Если h не является константой, мы имеем $B(f \diamond g) = 3B(h) - 2$, по сути, полученное взятием копии БДР для h и заменой ее стоковых узлов двумя другими копиями.

(б) Предположим, что профиль и квазипрофиль h представляют собой $(b_0, \ldots, b_n)$ и $(q_0, \ldots, q_n)$, где $b_n = q_n = 2$. Тогда в $f \diamond g$ имеются $b_k q_k$ ветвей в x_{2k+1} и $q_k b_{k-1}$ ветвей в x_{2k}, соответствующих упорядоченным парам бусин и подтаблиц h. Когда БДР для h содержит ветвление от α к β и от α' к β', где $\texttt{V}(\alpha) = j$, $\texttt{V}(\beta) = k$, $\texttt{V}(\alpha') = j'$ и $\texttt{V}(\beta') = k'$, БДР для $f \diamond g$ содержит соответствующую ветвь с $\texttt{V}(\alpha \diamond \alpha') = 2j - 1$ из $\alpha \diamond \alpha'$ в $\beta \diamond \alpha'$ при $j \le j' < k$ и с $\texttt{V}(\alpha \diamond \alpha') = 2j'$ из $\alpha \diamond \alpha'$ в $\alpha \diamond \beta'$ при $j' < j \le k'$.

60. Каждая бусина порядка $n - j$ упорядоченной пары (f, g) является либо одной из $b_j b'_j$ упорядоченных пар бусин f и g, либо одной из $b_j(q'_j - b'_j) + (q_j - b_j)b'_j$ упорядоченных пар, имеющих вид (бусина, не бусина) или (не бусина, бусина). Эта верхняя граница достигалась в примерах в упр. 59, (б) и 63.]

61. Считаем, что $v = V(\alpha) \le V(\beta)$. Пусть $\alpha_1, \ldots, \alpha_k$ являются узлами, указывающими на α, и пусть $\beta_1, \ldots, \beta_l$ являются узлами с $V(\beta_j) < v$, указывающими на β; на каждый корень указывает воображаемый узел. (Таким образом, $k =$ входящая степень(α) и $l \le$ входящая степень(β).) Тогда смешанные узлы, указывающие на $\alpha \diamond \beta$, могут быть трех типов: (i) $\alpha_i \diamond \beta_j$, где $V(\alpha_i) = V(\beta_j)$, и либо ($\mathrm{LO}(\alpha_i) = \alpha$ и $\mathrm{LO}(\beta_j) = \beta$), либо ($\mathrm{HI}(\alpha_i) = \alpha$ и $\mathrm{HI}(\beta_j) = \beta$); (ii) $\alpha \diamond \beta_j$, где $V(\alpha_i) < V(\beta_j)$ для некоторого i; либо (iii) $\alpha_i \diamond \beta$, где $V(\alpha_i) > V(\beta_j)$ для некоторого j.

62. БДР для f имеет по одному узлу на каждом уровне, а БДР для g имеет два, за исключением верхнего и нижнего уровней. БДР для $f \vee g$ имеет по четыре узла почти на каждом уровне согласно упр. 14, (а). БДР для $f \diamond g$ имеет семь узлов $\textcircled{j}$ при $5 \le j \le n - 3$, соответствующих упорядоченным парам подтаблиц (f, g), которые зависят от x_j, когда $(x_1, \ldots, x_{j-1})$ имеют фиксированные значения. Таким образом, $B(f) = n + O(1)$, $B(g) = 2n + O(1)$, $B(f \diamond g) = 7n + O(1)$ и $B(f \vee g) = 4n + O(1)$. (Также $B(f \wedge g) = 7n + O(1)$, $B(f \oplus g) = 7n + O(1)$, $B(f \wedge \bar{g}) = 6n + O(1)$.)

63. Профилями f и g являются соответственно $(1, 2, 2, \ldots, 2^{m-1}, 2^{m-1}, 2^m, 1, 1, \ldots, 1, 2)$ и $(0, 1, 2, 2, \ldots, 2^{m-1}, 2^{m-1}, 1, 1, \ldots, 1, 2)$; так что $B(f) = 2^{m+2} - 1 \approx 4n$ и $B(g) = 2^{m+1} + 2^m - 1 \approx 3n$. Профиль $f \wedge g$ начинается с $(1, 2, 4, \ldots, 2^{2m-2}, 2^{2m-1} - 2^{m-1})$, поскольку имеется единственное решение $x_1 \ldots x_{2m}$ уравнений

$$((x_1 \oplus x_2)(x_3 \oplus x_4) \ldots (x_{2m-1} \oplus x_{2m}))_2 = p, ((x_2 \oplus x_3) \ldots (x_{2m-2} \oplus x_{2m-1})x_{2m})_2 = q$$

для $0 \le p, q < 2^m$ и $p = q$ тогда и только тогда, когда $x_1 = x_3 = \cdots = x_{2m-1} = 0$. После этого профиль продолжается — $(2^{m+1} - 2, 2^{m+1} - 2, 2^{m+1} - 4, 2^{m+1} - 6, \ldots, 4, 2, 2)$; подфункциями являются $x_{2m+j} \wedge \bar{x}_{2m+k}$ или $\bar{x}_{2m+j} \wedge x_{2m+k}$ для $1 \le j < k \le 2^m$ вместе с x_{2m+j} и $\bar{x}_{2m+j}$ для $2 \le j \le 2^m$. В общем, мы имеем $B(f \wedge g) = 2^{2m+1} + 2^{m-1} - 1 \approx 2n^2$.

64. БДР для *любой* булевой комбинации f_1, f_2 и f_3 содержится в смеси $f_1 \diamond f_2 \diamond f_3$, размер которой не превышает $B(f_1)B(f_2)B(f_3)$.

65. $h = g ?\, f_1 : f_0$, где f_c представляет собой ограничение f, получающееся путем установки $x_j \leftarrow c$. Первая верхняя граница следует, как в ответе к упр. 64, поскольку $B(f_c) \le B(f)$. Вторая граница оказывается неверной, например $n = 2^m + 3m$ и $h = M_m(x; y) ?\, M_m(x'; y)$:

$M_m(x''; y)$, где $x = (x_1, \ldots, x_m)$, $x' = (x'_1, \ldots, x'_m)$, $x'' = (x''_1, \ldots, x''_m)$ и $y = (y_0, \ldots, y_{2^m-1})$; но такие неудачи редки. [См. R. E. Bryant, *IEEE Trans.* **C-35** (1986), 685; J. Jain, K. Mohanram, D. Moundanos, I. Wegener and Y. Lu, *ACM/IEEE Design Automation Conf.* **37** (2000), 681–686.]

66. Установить NTOP $\leftarrow f_0 + 1 - l$ и завершить работу алгоритма.

67. Пусть t_k обозначает положение шаблона POOLSIZE $- 2k$. На шаге S1 устанавливаются LEFT(t_1) $\leftarrow 5$, RIGHT(t_1) $\leftarrow 7$, $l \leftarrow 1$. На шаге S2 для $l = 1$ t_1 помещается как в LLIST[2], так и в HLIST[2]. На шаге S5 для $l = 2$ устанавливаются LEFT(t_2) $\leftarrow 4$, RIGHT(t_2) $\leftarrow 5$, L(t_1) $\leftarrow t_2$; LEFT(t_3) $\leftarrow 3$, RIGHT(t_3) $\leftarrow 6$, H(t_1) $\leftarrow t_3$. На шаге S2 для $l = 2$ устанавливается L(t_2) $\leftarrow 0$ и t_2 помещается в HLIST[3]; затем t_3 помещается в LLIST[3] и HLIST[3]. И т. д. Первый этап завершается с (LSTART[0], ..., LSTART[4]) $= (t_0, t_1, t_3, t_5, t_8)$ и

k	LEFT(t_k)	RIGHT(t_k)	L(t_k)	H(t_k)	k	LEFT(t_k)	RIGHT(t_k)	L(t_k)	H(t_k)
1	5 [α]	7 [ω]	t_2	t_3	5	3 [γ]	4 [φ]	t_6	t_8
2	4 [β]	5 [χ]	0	t_4	6	2 [δ]	2 [τ]	0	1
3	3 [γ]	6 [ψ]	t_4	t_5	7	2 [δ]	1 [$\top$]	0	1
4	3 [γ]	1 [$\top$]	t_7	1	8	1 [$\top$]	3 [υ]	1	0

что представляет смесь $\alpha \diamond \omega$ на рис. 24, но с $\bot \diamond x = x \diamond \bot = \bot$ и $\top \diamond \top = \top$.

Пусть $f_k = f_0 + k$. На втором этапе шаг S7 для $l = 4$ устанавливает LEFT(t_6) $\leftarrow \sim 0$, LEFT(t_7) $\leftarrow t_6$, LEFT(t_8) $\leftarrow \sim 1$ и RIGHT(t_6) $\leftarrow$ RIGHT(t_7) $\leftarrow$ RIGHT(t_8) $\leftarrow -1$. На шаге S8 отменяются изменения, внесенные в LEFT(0) и LEFT(1). На шаге S11 с $s = t_8$ устанавливаются LEFT(t_8) $\leftarrow \sim 2$, RIGHT(t_8) $\leftarrow t_8$, V(f_2) $\leftarrow 4$, LO(f_2) $\leftarrow 1$, HI(f_2) $\leftarrow 0$. С $s = t_7$ на этом шаге устанавливаются LEFT(t_7) $\leftarrow \sim 3$, RIGHT(t_7) $\leftarrow t_7$, V(f_3) $\leftarrow 4$, LO(f_3) $\leftarrow 0$, HI(f_3) $\leftarrow 1$; в то же время на шаге S10 должно быть установлено RIGHT(t_6) $\leftarrow t_7$. В конечном итоге шаблоны должны быть преобразованы в

k	LEFT(t_k)	RIGHT(t_k)	L(t_k)	H(t_k)	k	LEFT(t_k)	RIGHT(t_k)	L(t_k)	H(t_k)
1	~ 8	t_1	t_2	t_3	5	~ 4	t_5	t_7	t_8
2	~ 7	t_2	0	t_4	6	~ 0	t_7	0	1
3	~ 6	t_3	t_4	t_5	7	~ 3	t_7	0	1
4	~ 5	t_4	t_7	1	8	~ 2	t_8	1	0

(но затем они могут быть отброшены). Полученная в результате БДР для $f \wedge g$ имеет вид

k	V(f_k)	LO(f_k)	HI(f_k)	k	V(f_k)	LO(f_k)	HI(f_k)
2	4	1	0	6	2	5	4
3	4	0	1	7	2	0	5
4	3	3	2	8	1	7	6
5	3	3	1				

.

68. Если LEFT(t) < 0 в начале шага S10, установить RIGHT(t) $\leftarrow t$, $q \leftarrow$ NTOP, NTOP $\leftarrow q + 1$, LEFT(t) $\leftarrow \sim(q - f_0)$, LO(q) $\leftarrow \sim$LEFT(L(t)), HI(q) $\leftarrow \sim$LEFT(H(t)), V(q) $\leftarrow l$ и вернуться к шагу S9.

69. Убедитесь, что NTOP $\leq$ TBOT в конце шага S1 и при переходе от шага S11 к шагу S9. (*Не* вносите этот тест внутрь цикла в S11.) Убедитесь также, что NTOP $\leq$ HBASE непосредственно после установки HBASE на шаге S4.

70. Этот выбор делает хеш-таблицу немного меньшей; таким образом, переполнение памяти становится менее вероятным ценой немного большего количества коллизий. Но это также замедлит работу, поскольку теперь *make_template* потребуется проверять условие NTOP $\leq$ TBOT при каждом уменьшении TBOT.

71. Добавить новое поле, EXTRA(t) $= \alpha''$, к каждому шаблону t (см. (43)).

72. Вместо шагов S4 и S5 следует воспользоваться подходом из алгоритма R для карманной сортировки элементов связанных списков, начинающихся с LLIST[l] и HLIST[l]. Это

возможно, если использовать в указателях однобитовую подсказку для того, чтобы отличать связи в полях L от связей в полях H, поскольку тогда можно определять параметры LO и HI потомков t как функцию от t и ее "четности".

73. Если профилем БДР является $(b_0, \ldots, b_n)$, мы можем назначить $p_j = \lceil b_{j-1}/2^e \rceil$ страниц ветвям x_j. Вспомогательные таблицы из $p_1 + \cdots + p_{n+1} \le \lceil B(f)/2^e \rceil + n$ коротких целых чисел позволяют нам вычислить $V(p) = T[\pi(p)]$, $\mathrm{LO}(p) = \mathtt{LO}(M[\pi(p)] + \sigma(p))$, $\mathrm{HI}(p) = \mathtt{HI}(M[\pi(p)] + \sigma(p))$.

Например, если $e = 12$ и $n < 2^{16}$, мы можем представить произвольные БДР размером до $2^{32} - 2^{28} + 2^{16} + 2^{12}$ узлов с помощью 32-битовых виртуальных указателей LO и HI. Каждая БДР требует соответствующих вспомогательных таблиц T и M размером $\le 2^{20}$, сконструированных из ее профиля.

[Этот метод может значительно улучшить кэширующее поведение. Он основан на статье П. Ашара (P. Ashar) и М. Ченга (M. Cheong), *IEEE/ACM Internat. Conf. Computer-Aided Design* **CAD-94** (1994), 622–627, которые также разработали алгоритмы, подобные алгоритму S.]

74. Требуемое условие представляет собой $\mu_n(x_1, \ldots, x_{2^n}) \wedge [\bar{x}_1 = x_{2^n}] \wedge [\bar{x}_2 = x_{2^n-1}] \wedge \cdots \wedge [\bar{x}_{2^{n-1}} = x_{2^{n-1}+1}]$. Если установить $y_1 = x_1$, $y_2 = x_3$, $\ldots$, $y_{2^{n-2}} = x_{2^{n-1}-1}$, $y_{2^{n-2}+1} = \bar{x}_{2^{n-1}}$, $y_{2^{n-2}+2} = \bar{x}_{2^{n-1}-2}$, $\ldots$, $y_{2^{n-1}} = \bar{x}_2$, (49) приведет к эквивалентному условию $\mu_{n-1}(y_1, \ldots, y_{2^{n-1}}) \wedge [y_{2^{n-2}} \le \bar{y}_{2^{n-2}+1}] \wedge [y_{2^{n-2}-1} \le \bar{y}_{2^{n-2}+2}] \wedge \cdots \wedge [y_1 \le \bar{y}_{2^{n-1}}]$, которое замечательно подходит для вычисления алгоритмом S. (Вычисление должно выполняться слева направо; вычисление справа налево будет генерировать чудовищные промежуточные результаты.)

При таком подходе мы найдем, что имеется соответственно 1, 2, 4, 12, 81, 2 646, 1 422 564, 229 809 982 112 монотонных самодуальных функций от 1, 2, $\ldots$, 8 переменных. (См. табл. 7.1.1–3 и ответ к упр. 7.1.2–88.) Функции от 8 переменных характеризуются БДР с 130 305 082 узлами; алгоритму S для ее вычисления требуется около 204 миллиардов обращений к памяти.

75. Начнем с $\rho_1(x_1, x_2) = [x_1 \le x_2]$ и заменим $G_{2^n}(x_1, \ldots, x_{2^n})$ в (49) функцией $H_{2^n}(x_1, \ldots, x_{2^n}) = [x_1 \le x_2 \le x_3 \le x_4] \wedge \cdots \wedge [x_{2^n-3} \le x_{2^n-2} \le x_{2^n-1} \le x_{2^n}]$.

(Оказывается, что $B(\rho_9) = 3\,683\,424$; для вычисления этой БДР достаточно около 170 миллионов обращений к памяти; ρ_{10} находится почти на пределе досягаемости. Алгоритм C быстро дает точное количество регулярных булевых функций от n переменных для $1 \le n \le 9$, а именно 3, 5, 10, 27, 119, 1 173, 44 315, 16 175 190, 284 432 730 176. Аналогично можно подсчитать и самодуальные функции, как в упр. 74; эти числа, ранняя история которых рассматривалась в ответе к упр. 7.1.1–123, представляют собой 1, 1, 2, 3, 7, 21, 135, 2 470, 319 124, 1 214 554 343 для $1 \le n \le 10$.)

76. Будем говорить, что $x_0 \ldots x_{j-1}$ *обеспечивает* x_j, если $x_i = 1$ для некоторого $i \subseteq j$ с $0 \le i < j$. В таком случае $x_0 x_1 \ldots x_{2^n-1}$ соответствует клаттеру тогда и только тогда, когда $x_j = 0$, если $x_0 \ldots x_{j-1}$ обеспечивает x_j, для $0 \le j < 2^n$. А $\mu_n(x_0, \ldots, x_{2^n-1}) = 1$ тогда и только тогда, когда $x_j = 1$, если $x_0 \ldots x_{j-1}$ обеспечивает x_j. Так что мы получим искомую БДР из таковой для $\mu_n(x_1, \ldots, x_{2^n})$ путем (i) изменения каждой ветви $\textcircled{j}$ на $\boxed{j-1}$ и (ii) взаимообмена ветвей LO и HI в каждом узле ветвления, который имеет $\mathrm{LO} = \boxed{\perp}$. (Заметим, что согласно следствию 7.1.1Q простые импликанты каждой монотонной булевой функции соответствуют клаттерам.)

77. Продолжая ответ к предыдущему упражнению, будем говорить, что битовый вектор $x_0 \ldots x_{k-1}$ является *согласованным*, если мы имеем $x_j = 1$, когда $x_0 \ldots x_{j-1}$ обеспечивает x_j для $0 < j < k$. Пусть b_k представляет собой количество согласованных векторов длиной k. Например, $b_4 = 6$, так как согласованными являются четырехбитовые векторы $\{0000, 0001, 0011, 0101, 0111, 1111\}$. Заметим, что ровно $c_k = b_{k+1} - b_k$ клаттеров $\mathcal{S}$ обладают

тем свойством, что k представляет их "наибольшее" множество, $\max\{s \mid s$ представляет множество $\mathcal{S}\}$.

БДР для $\mu_n(x_1, \ldots, x_{2^n})$ имеет b_{k-1} узлов ветвления $\textcircled{k}$ при $1 \le k \le 2^{n-1}$. Доказательство: каждая подфункция, определяемая $x_1, \ldots, x_{k-1}$, либо тождественно ложна, либо определяет согласованный вектор $x_1 \ldots x_{k-1}$. В последнем случае подфункция является бусиной, поскольку она принимает различные значения при определенных установках $x_{k+1}, \ldots, x_{2^n}$. Действительно, если $x_1 \ldots x_{k-1}$ обеспечивает x_k, мы устанавливаем $x_{k+1} \leftarrow \cdots \leftarrow x_{2^n} \leftarrow 1$; в противном случае мы устанавливаем $x_j \leftarrow y_j$ для $k < j \le 2^n$, где $y_{j+1} = [x_{i+1} = 1$ для некоторого $i \subseteq j$ с $i + 1 < k]$; заметим, что $y_{2^n+k} = 0$.

С другой стороны, когда $k = 2^n - k'$ и $0 \le k' < 2^{n-1}$, имеется $b_{k'}$ ветвей $\textcircled{k}$. В этом случае неконстантные подфункции, возникающие из $x_1, \ldots, x_{k-1}$, приводят ко значениям y_j, показанным выше, где вектор $\bar{y}_{0'}\bar{y}_{1'} \ldots \bar{y}_{k'}$ является согласованным. (Здесь $0' = 2^n$, $1' = 2^n - 1$, и т. д.) И обратно, каждый такой согласованный вектор описывает такую подфункцию; можно, например, установить $x_j \leftarrow 0$, когда $j < k - 2^{n-1}$ или $2^{n-1} \le j < k$, в противном случае $x_j \leftarrow y_{2^{n-1}+j}$. Эта подфункция является бусиной тогда и только тогда, когда $y_{k'} = 1$ или $\bar{y}_{0'} \ldots \bar{y}_{(k-1)'}$ обеспечивает $\bar{y}_{k'}$. Таким образом, эти бусины соответствуют согласованным векторам длиной k'; а различные векторы определяют различные бусины.

Эти рассуждения показывают, что имеется $b_{k-1} - c_{k-1}$ ветвлений $\textcircled{k}$ с LO $= \boxed{\perp}$, когда $1 \le k \le 2^{n-1}$, и c_{2^n-k} таких ветвлений, когда $2^{n-1} < k \le 2^n$. Следовательно, ровно половина $B(\mu_n) - 2$ узлов ветвления имеет LO $= \boxed{\perp}$.

78. Для подсчета графов на n помеченных вершинах с максимальной степенью $\le d$ построим булеву функцию от $\binom{n}{2}$ переменных из его матрицы смежности, а именно функцию $\bigwedge_{k=1}^{n} S_{\le d}(X_k)$, где X_k представляет собой множество переменных в строке k матрицы. Например, когда $n = 5$, имеется 10 переменных, а функция имеет вид $S_{\le d}(x_1, x_2, x_3, x_4) \wedge S_{\le d}(x_1, x_5, x_6, x_7) \wedge S_{\le d}(x_2, x_5, x_8, x_9) \wedge S_{\le d}(x_3, x_6, x_8, x_{10}) \wedge S_{\le d}(x_4, x_7, x_9, x_{10})$. При $n = 12$ БДР для $d = (1, 2, \ldots, 10)$ имеют соответственно (5960, 137477, 1255813, 5295204, 10159484, 11885884, 9190884, 4117151, 771673, 28666) узлов, так что они быстро вычисляются с помощью алгоритма S. Для подсчета решений с максимальной степенью d вычтем количество решений для степени $\le d-1$ из количества для степени $\le d$; ответами для $0 \le d \le 11$ являются:

1	3038643940889754	29271277569846191555
140151	211677202624318662	17880057008325613629
3568119351	3617003021179405538	4489497643961740521
8616774658305	17884378201906645374	430038382710483623

[В общем случае имеется $t_n - 1$ графов на n помеченных вершинах с максимальной степенью 1, где t_n — количество инволюций 5.1.4–(40).]

Методы из раздела 7.2.3 превосходят БДР для таких перечислений, как здесь, при больших n, поскольку помеченные графы имеют $n!$ симметрий. Но когда n имеет умеренный размер, бинарные диаграммы решений быстро дают ответ и красиво характеризуют все решения.

79. В следующих результатах подсчетов, сделанных на основе бинарных диаграмм решений из ответа к предыдущему упражнению, каждый граф с k ребрами имел вес 2^{66-k}. Для получения вероятностей следует выполнить деление на 3^{66}.

73786976294838206464	116467254834302955464842637475 84
55315674993080529007 4112	77677416878709243055475188039 68
59853550286831523654847 6928	25144575345589759186086686883 84
68379835220584550117167595520	45273361563608993921819340390 4
138035892756457768347923329843 2	45968637738881805341545676736
702409637629839707696908153651 2	20931955804803138182929249 85

80. Если исходные функции f и g не имеют общих узлов БДР, оба алгоритма сталкиваются почти с одними и теми же подзадачами: алгоритм S работает со всеми узлами $f \diamond g$, которые не происходят от узлов вида $\alpha \diamond \boxed{\bot}$ или $\boxed{\bot} \diamond \beta$, в то время как (55) также избегает узлов, происходящих от $\alpha \diamond \boxed{\top}$ или $\boxed{\top} \diamond \beta$. Кроме того, (55) использует сокращения, встречаясь с нетривиальными подзадачами И(f', g') с $f' = g'$; алгоритм S не в состоянии распознать простоту таких случаев. Еще (55) может также выиграть, если столкнется с соответствующими записями, оставшимися от предыдущего вычисления.

81. Просто везде замените 'И' на 'ЛИБО' и '$\wedge$' на '$\oplus$'. Простыми случаями теперь являются $f \oplus 0 = f$, $0 \oplus g = g$ и $f \oplus g = 0$ при $f = g$. Мы должны также выполнить обмен $f \leftrightarrow g$, если $f > g \neq 0$.

Примечания. Автор в качестве эксперимента добавлял в кэш дополнительные записи '$f \oplus r = g$' и '$g \oplus r = f$'; но, похоже, они приносили больше вреда, чем пользы. Рассматривая другие бинарные операторы, заметим, что нет необходимости реализовывать оба оператора НО-НЕ$(f, g) = f \wedge \bar{g}$ и НЕ-НО$(f, g) = \bar{f} \wedge g$, поскольку последний представляет собой НО-НЕ(g, f). Кроме того, реализация ЛИБО$(1,$ ИЛИ$(f, g))$ может оказаться лучше реализации НЕ-ИЛИ$(f, g) = \neg(f \vee g)$.

82. Вычисление верхнего уровня $F \leftarrow$ И(f, g) начинается с f и g в регистрах компьютера, но $\texttt{REF}(f)$ и $\texttt{REF}(g)$ не включают "ссылки" как таковые. (Однако мы можем считать, что и f, и g "живы".)

Если алгоритм (55) обнаруживает, что $f \wedge g$ очевидным образом является r, он увеличивает $\texttt{REF}(r)$ на 1.

Если (55) находит $f \wedge g = r$ в кэше записей, он увеличивает $\texttt{REF}(r)$ и рекурсивно увеличивает $\texttt{REF}(\texttt{LO}(r))$ и $\texttt{REF}(\texttt{HI}(r))$ тем же способом, если r было "мертво".

Если на шаге U1 обнаруживается $p = q$, $\texttt{REF}(p)$ *уменьшается* на 1 (поверите вы в это или нет); это не приводит к уничтожению p.

Если на шаге U2 обнаруживается r, возможны два варианта: если r было "живым", устанавливаются $\texttt{REF}(r) \leftarrow \texttt{REF}(r) + 1$, $\texttt{REF}(p) \leftarrow \texttt{REF}(p) - 1$, $\texttt{REF}(q) \leftarrow \texttt{REF}(q) - 1$. В противном случае просто устанавливается $\texttt{REF}(r) \leftarrow 1$.

Когда на шаге U3 создается новый узел r, устанавливается $\texttt{REF}(r) \leftarrow 1$.

Наконец, после того как И на верхнем уровне возвращает значение r, которое мы хотим назначить F, необходимо сначала *разыменовать* F, если $F \neq \Lambda$; в данном случае это означает установку $\texttt{REF}(F) \leftarrow \texttt{REF}(F) - 1$ с рекурсивным разыменованием $\texttt{LO}(F)$ и $\texttt{HI}(F)$, если $\texttt{REF}(F)$ становится равным 0. Затем мы устанавливаем $F \leftarrow r$ (без изменения $\texttt{REF}(r)$).

[Кроме того, в подпрограмме квантификации, такой как (65), или в подпрограмме композиции (72) и r_l, и r_h должны быть разыменованы после того, как ИЛИ или MUX вычислит r.]

83. В упр. 61 показано, что подзадача $f \wedge g$ встречается не более одного раза на один вызов верхнего уровня, когда $\texttt{REF}(f) = \texttt{REF}(g) = 1$. [Эта идея принадлежит Ф. Соменци (F. Somenzi); см. статью, упомянутую в ответе к упр. 84. Многие узлы имеют количество ссылок, равное 1, поскольку среднее значение приблизительно равно 2 и поскольку стоки обычно имеют большие значения. Однако такое избегание обращения к кэшу в экспериментах автора не повышало общую производительность, возможно, из-за исследуемых примеров или из-за "случайных" попаданий в кэш в других операциях верхнего уровня.]

84. Имеются различные возможности, но явного победителя при этом нет. Размеры кэша и таблиц должны представлять собой степени 2 для облегчения вычислений хеш-функций. Размер таблицы уникальности для x_v должен быть грубо пропорционален числу (живых и мертвых) узлов, в настоящий момент выполняющих переходы по переменной x_v. Нет необходимости выполнять хеширование заново при уменьшении или увеличении таблицы.

В экспериментах автора во время написания данного раздела размер кэша удваивался, когда количество вставок с начала последней команды верхнего уровня превосходило текущий размер кэша в $\ln 2$ раз. (В этот момент случайная хеш-функция должна заполнять около половины слотов.) После сборки мусора кэш при необходимости уменьшался, так что он либо имел 256 слотов, либо был заполнен как минимум на четверть.

Отслеживать текущее количество мертвых узлов достаточно легко; следовательно, в любой момент мы знаем, какое количество памяти нам вернет сборка мусора. Автор получил удовлетворительные результаты путем вставки нового шага U2$\frac{1}{2}$ между U2 и U3: "увеличить C на 1, где C — глобальный счетчик. Если $C \bmod 1024 = 0$ и если как минимум 1/8 всех текущих узлов мертвы, выполнить сборку мусора".

[См. различные другие предложения, основанные на богатом экспериментальном опыте, в работе F. Somenzi, *Software Tools for Technology Transfer* **3** (2001), 171–181.]

85. Полная таблица должна хранить 2^{32} записей по 32 бита, т. е. 2^{34} байт (≈ 17.2 Гбайт). База БДР, рассматривавшаяся после (58), с около 136 миллионами узлов, с битами в чередующемся порядке "застежки-молнии", требует для хранения около 1.1 Гбайт памяти; база БДР, рассматривавшаяся в следствии Y, с последовательным порядком битов множителей, требует для хранения только около 400 Мбайт памяти.

86. Если $f = 0$ или $g = h$, вернуть g. Если $f = 1$, вернуть h. Если $g = 0$ или $f = g$, вернуть И(f, h). Если $h = 1$ или $f = h$, вернуть ИЛИ(f, g). Если $g = 1$, вернуть СЛЕДУЕТ(f, h); если $h = 0$, вернуть НО-НЕ(g, f). (Если бинарные СЛЕДУЕТ и/или НО-НЕ не реализованы непосредственно, соответствующие случаи могут принимать тернарный облик.)

87. Отсортировать заданные значения указателей f, g, h так, что $f \le g \le h$. Если $f = 0$, вернуть И(g, h). Если $f = 1$, вернуть ИЛИ(g, h). Если $f = g$ или $g = h$, вернуть g.

88. Тройка функций $(f, g, h) = (R_0, R_1, R_2)$ образует забавный пример при

$$R_a(x_1, \ldots, x_n) = \bigl[(x_n \ldots x_1)_2 \bmod 3 \ne a\bigr] = R_{(2a+x_1) \bmod 3}(x_2, \ldots, x_n).$$

Благодаря записям тернарная рекурсия находит $f \wedge g \wedge h = 0$ путем исследования на каждом уровне только одного случая; бинарное вычисление, скажем, $f \wedge g = \bar{h}$, определенно, оказывается более длинным.

Еще более эффектно положить $f = x_1 \wedge (x_2?\ F\colon G)$, $g = x_2 \wedge (x_1?\ G\colon F)$ и $h = x_1?\ \bar{x}_2 \wedge F\colon x_2 \wedge G$, где F и G являются функциями от $(x_3, \ldots, x_n)$, такими, что $B(F \wedge G) = \Theta(B(F)B(G))$, как в упр. 63. Тогда каждая из функций $f \wedge g$, $g \wedge h$ и $h \wedge f$ имеет большу́ю БДР, но тернарная рекурсия немедленно выясняет, что $f \wedge g \wedge h = 0$.

89. (а) Истинна; левая часть представляет собой $(f_{00} \vee f_{01}) \vee (f_{10} \vee f_{11})$, правая — $(f_{00} \vee f_{10}) \vee (f_{01} \vee f_{11})$.

(б) Аналогично истинна. (Квантор $\Box$ также является коммутативным.)

(в) Обычно ложна; см. п. (г).

(г) $\forall x_1 \exists x_2 f = (f_{00} \vee f_{01}) \wedge (f_{10} \vee f_{11}) = (\exists x_2 \forall x_1 f) \vee (f_{00} \wedge f_{11}) \vee (f_{01} \wedge f_{10})$.

90. Замените $\exists j_1 \ldots \exists j_m$ на $\Box j_1 \ldots \Box j_m$.

91. (а) $f \downarrow 1 = f$, $f \downarrow x_j = f_1$ и $f \downarrow \bar{x}_j = f_0$ в обозначениях (63).

(б) Этот закон дистрибутивности очевиден по определению $\downarrow$. (Он также истинен для $\vee$, $\oplus$ и т. д.)

(в) Истинно тогда и только тогда, когда g не тождественна нулю. (Следовательно, значение $f(x_1, \ldots, x_n) \downarrow g$ для $g \ne 0$ определяется только значениями $x_j \downarrow g$ для $1 \le j \le n$.)

(г) $f(x_1, 1, 0, x_4, 0, 1, x_7, \ldots, x_n)$. Это ограничение f по отношению к $x_2 = 1$, $x_3 = 0$, $x_5 = 0$, $x_6 = 1$ (см. упр. 57), также называется *кофактором* f по отношению к подкубу g. (Аналогичный результат выполняется, когда g представляет собой любое произведение литералов.)

(д) $f(x_1, \ldots, x_{n-1}, x_1 \oplus \cdots \oplus x_{n-1} \oplus 1)$. (Рассмотрите случай $f = x_j$ для $1 \le j \le n$.)

(е) $x_1?\ f(1, \ldots, 1)\colon f(0, \ldots, 0)$.

(ж) $f(1, x_2, \ldots, x_n) \downarrow g(x_2, \ldots, x_n)$.

(з) Если $f = x_2$ и $g = x_1 \vee x_2$, мы имеем $f \downarrow g = \bar{x}_1 \vee x_2$.

(и) CONSTRAIN(f, g) = "Если $f \downarrow g$ имеет очевидное значение, вернуть его. В противном случае, если $f \downarrow g = r$ находится в кэше записей, вернуть r. В противном случае представить f и g, как в (52); установить $r \leftarrow$ CONSTRAIN(f_h, g_h), если $g_l = 0$, $r \leftarrow$ CONSTRAIN(f_l, g_l), если $g_h = 0$, в противном случае $r \leftarrow$ UNIQUE$(v,$ CONSTRAIN(f_l, g_l), CONSTRAIN$(f_h, g_h))$; поместить '$f \downarrow g = r$' в кэш записей, и вернуть r". Здесь очевидными значениями являются $f \downarrow 0 = 0 \downarrow g = 0$; $f \downarrow 1 = f$; $1 \downarrow g = g \downarrow g = [g \neq 0]$.

[Оператор $f \downarrow g$ был введен в 1989 году О. Кудером (O. Coudert), К. Берте (C. Berthet) и Ж. К. Мадре (J. C. Madre). Примеры, такие как в п. (з), привели их также к предложению модифицированного оператора ограниченности $f \Downarrow g$, который имеет аналогичную рекурсию, за исключением того, что он использует $f \Downarrow (\exists x_v g)$ вместо $(\bar{x}_v?\ f_l \Downarrow g_l\colon f_h \Downarrow g_h)$ при $f_l = f_h$. См. *Lecture Notes in Computer Science* **407** (1989), 365–373.]

92. Для части "тогда" см. ответ к упр. 91, (г). Заметим также, что (i) $x_1 \downarrow g = x_1$ тогда и только тогда, когда $g_0 \neq 0$ и $g_1 \neq 0$, где $g_c = g(c, x_2, \ldots, x_n)$; (ii) $x_n \downarrow g = x_n$ тогда и только тогда, когда $\partial x_n g = 0$ и $g \neq 0$.

Предположим, что $f^\pi \downarrow g^\pi = (f \downarrow g)^\pi$ для всех f и π. Если $g \neq 0$ не является подкубом, существует индекс j, такой, что $g_0 \neq 0$, $g_1 \neq 0$ и $\partial x_j g \neq 0$, где $g_c = g(x_1, \ldots, x_{j-1}, c, x_{j+1}, \ldots, x_n)$. Согласно предыдущему абзацу мы имеем (i) $x_j \downarrow g = x_j$ и (ii) $x_j \downarrow g \neq x_j$, т. е. получаем противоречие.

93. Пусть $f = J(x_1, \ldots, x_n; f_1, \ldots, f_n)$ и $g = J(x_1, \ldots, x_n; g_1, \ldots, g_n)$, где

$$f_v = x_{n+1} \vee \cdots \vee x_{5n} \vee J(x_{5n+1}, \ldots, x_{6n}; [v{-}1], \ldots, [v{-}n]),$$
$$g_v = x_{n+1} \vee \cdots \vee x_{5n} \vee J(x_{5n+1}, \ldots, x_{6n}; [v{=}1]{+}[v{-}1], \ldots, [v{=}n]{+}[v{-}n]),$$

а J представляет собой функцию соединения из упр. 52.

Если G может быть раскрашен тремя красками, пусть $\hat{f} = J(x_1, \ldots, x_n; \hat{f}_1, \ldots, \hat{f}_n)$, где

$$\hat{f}_v = x_{n+1} \vee \cdots \vee x_{5n} \vee J(x_{5n+1}, \ldots, x_{6n}; \hat{f}_{v1}, \ldots, \hat{f}_{vn}),$$

а $\hat{f}_{vw} = [v$ и w имеют разные цвета$]$. Тогда $B(\hat{f}) < n + 3(5n) + 2$.

И обратно, предположим, что существует приближенная функция $\hat{f}$, такая, что $B(\hat{f}) < 16n + 2$, и пусть $\hat{f}_v$ является подфункцией с $x_1 = [v{=}1]$, ..., $x_n = [v{=}n]$. Различны не более трех из этих подфункций, поскольку каждая отличная $\hat{f}_v$ должна иметь ветвление по каждой из переменных x_{n+1}, ..., x_{5n}. Раскрасим вершины таким образом, чтобы u и v имели одинаковый цвет тогда и только тогда, когда $\hat{f}_u = \hat{f}_v$; это может случиться, только когда $u \not\sim v$, так что раскраска корректна.

[M. Sauerhoff and I. Wegener, *IEEE Transactions* **CAD-15** (1996), 1435–1437.]

94. *Случай 1.* $v \neq g_v$. Тогда мы не квантифицируем x_v; следовательно, $g = g_h$ и $f \mathrm{E}\, g = \bar{x}_v?\ f_l \mathrm{E}\, g\colon f_h \mathrm{E}\, g$.

Случай 2. $v = g_v$. Тогда $g = x_v \wedge g_h$ и $f \mathrm{E}\, g = (f_l \mathrm{E}\, g_h) \vee (f_h \mathrm{E}\, g_h) = r_l \vee r_h$. В подслучае $v \neq f_v$ мы имеем $f_l = f_h = f$; следовательно, $r_l = r_h$ и мы можем непосредственно привести $f \mathrm{E}\, g$ к $f \mathrm{E}\, g_h$ (пример "концевой рекурсии").

[Руделл (Rudell) заметил, что порядок квантификации в (65) соответствует восходящему порядку переменных. Этот порядок удобный, но не всегда наилучший; иногда лучше удалять кванторы $\exists$ по одному в ином порядке, основываясь на знаниях об используемых функциях.]

95. Если $r_l = 1$ и $v = g_v$, можно установить $r \leftarrow 1$ и забыть об r_h. (Это изменение приводило в некоторых экспериментах автора к стократному ускорению.)

96. Для $\forall$ просто заменим E на A и ИЛИ на И. Для $\complement$ заменим E на D и ИЛИ на ЛИБО; также, если $v \neq f_v$, возвращаем 0. [Подпрограммы для кванторов да/нет $\curlywedge$ и $\mathbb{N}$ аналогичны случаю $\complement$. Кванторы да/нет должны использоваться только при $m = 1$; в противном случае их применение имеет мало смысла.]

97. В восходящем направлении количество работы на каждом уровне в худшем случае пропорционально количеству узлов на этом уровне.

98. Функция $\mathrm{NOTEND}(x) = \exists y \exists z(\mathrm{ADJ}(x,y) \wedge \mathrm{ADJ}(x,z) \wedge [y \neq z])$ идентифицирует все вершины степени ≥ 2. Следовательно, $\mathrm{ENDPT}(x) = \mathrm{KER}(x) \wedge \neg\mathrm{NOTEND}(x)$. А $\mathrm{PAIR}(x,y) = \mathrm{ENDPT}(x) \wedge \mathrm{ENDPT}(y) \wedge \mathrm{ADJ}(x,y)$.

[Например, когда G представляет собой граф граничащих штатов США, со штатами, упорядоченными, как в (104), мы имеем $B(\mathrm{NOTEND}) = 992$, $B(\mathrm{ENDPT}) = 264$ и $B(\mathrm{PAIR}) = 203$. Перед применением $\exists y \exists z$ размер БДР равен 50 511. Имеется ровно 49 ядер степени 1. Девять компонентов размером 2 получаются путем объединения следующих трех решений.

Общая стоимость этих вычислений с применением упомянутых алгоритмов составляет около 14 миллионов обращений к памяти и требует около 6.3 Мбайт памяти — всего лишь 52 обращений к памяти на ядро.]

99. Найдите треугольник взаимно смежных штатов и зафиксируйте их цвета. Размер БДР также существенно уменьшится, если выбрать штаты с высокими степенями на "средних" уровнях. Например, путем установки $a_{\mathtt{MO}} = b_{\mathtt{MO}} = a_{\mathtt{TN}} = \bar{b}_{\mathtt{TN}} = \bar{a}_{\mathtt{AR}} = b_{\mathtt{AR}} = 1$ мы превращаем 25 579 узлов во всего лишь 4642 (а общее время работы падает до менее 2 миллионов обращений к памяти).

[В рукописи Брайанта о БДР была детально рассмотрена раскраска графов, но при публикации в 1986 году он решил заменить этот материал другим.]

100. Заменим $\mathrm{IND}(x_{\mathtt{ME}}, \ldots, x_{\mathtt{CA}})$ на $\mathrm{IND}(x_{\mathtt{ME}}, \ldots, x_{\mathtt{CA}}) \wedge S_{12}(x_{\mathtt{ME}}, \ldots, x_{\mathtt{CA}})$, чтобы получить независимые множества, состоящие из 12 узлов; эта БДР имеет размер 1964. Затем используем (73) как и ранее, и прибегнем к трюку из упр. 99, получая функцию COLOR с 184 260 узлами и 12 554 677 864 решениями. (Время работы составляет около 26 миллионов обращений к памяти.)

101. Если вес штата равен w, назначим $2w$ и w в качестве соответствующих весов его переменным a и b и применим алгоритм B. (Например, переменная $a_{\mathtt{WY}}$ получает вес $2(23+25) = 96$.) Показанное здесь решение с кодами цветов ① ② ③ ④, является единственным.

102. Основная идея заключается в том, что при изменении g_j все результаты в кэше для функций с $f_v > j$ остаются действительными. Чтобы воспользоваться этим принципом, мы можем поддерживать массив "временны́х меток" $G_1 \geq G_2 \geq \cdots \geq G_n \geq 0$, по одной для каждой переменной. Имеется время "тактового генератора" $G \geq G_1$, представляющее количество различных выполненных или подготовленных композиций; еще одна переменная G' записывает, было ли значение G изменено после последнего вызова COMPOSE. Изначально $G = G' = G_1 = \cdots = G_n = 0$. Подпрограмма $\mathrm{NEWG}(j, g)$ реализована следующим образом.

N1. [Простой случай?] Если $g_j = g$, выйти из подпрограммы. В противном случае установить $g_j \leftarrow g$.

N2. [Можно выполнить сброс?] Если $g \neq x_j$ или если $j < n$ и $G_{j+1} > 0$, перейти к шагу N4.

N3. [Сброс меток.] Пока $j > 0$ и $g_j = x_j$, устанавливать $G_j \leftarrow 0$ и $j \leftarrow j - 1$. Затем, если $j = 0$, установить $G \leftarrow G - G'$, $G' \leftarrow 0$ и выйти из подпрограммы.

N4. [Обновление G?] Если $G' = 0$, установить $G \leftarrow G + 1$ и $G' \leftarrow 1$.

N5. [Новые метки.] Пока $j > 0$ и $G_j \neq G$, устанавливать $G_j \leftarrow G$ и $j \leftarrow j - 1$. Выйти из подпрограммы. ▌

(Счетчики ссылок также должны поддерживаться соответствующим образом.) Перед вызовом верхнего уровня подпрограммы COMPOSE установите $G' \leftarrow 0$. Измените подпрограмму COMPOSE в (72) так, чтобы она использовала при обращении к кэшу $f[G_v]$, где $v = f_v$; проверка '$v > m$' становится проверкой '$G_v = 0$'.

103. Эквивалентная формула $g(f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n))$ может быть реализована с помощью операции COMPOSE (72). (Однако преподаватель был оправдан, когда оказалось, что его формула может быть вычислена более чем в сто раз быстрее формулы Шустрого, несмотря на тот факт, что она использует в два раза больше переменных! В его приложении вычисление $(y_1 = f_1(x_1, \ldots, x_n)) \wedge \cdots \wedge (y_m = f_m(x_1, \ldots, x_n)) \wedge g(y_1, \ldots, y_m)$ оказывается гораздо проще, чем вычисления операций COMPOSE от $g_j(f_1, \ldots, f_m)$ для каждой подфункции g_j функции g; см., например, упр. 162.)

104. Следующий рекурсивный алгоритм COMPARE(f, g) требует не более $O(B(f)B(g))$ шагов при использовании совместно с кэшем записей. Если $f = g$, вернуть '='. В противном случае при $f = 0$ или $g = 1$ вернуть '<'; при $f = 1$ или $g = 0$ вернуть '>'. В противном случае представить f и g, как в (52); вычислить $r_l \leftarrow$ COMPARE(f_l, g_l). Если r_l представляет собой '‖', вернуть '‖'; в противном случае вычислить $r_h \leftarrow$ COMPARE(f_h, g_h). Если r_h представляет собой '‖', вернуть '‖'. В противном случае: если r_l представляет собой '=', вернуть r_h; если r_h представляет собой '=', вернуть r_l; если $r_l = r_h$, вернуть r_l. Иначе вернуть '‖'.

105. (а) Унатная функция с полярностями $(y_1, \ldots, y_n)$ имеет $\curlywedge x_j f = 0$ при $y_j = 1$ и $\mathbb{N} x_j f = 0$ при $y_j = 0$ для $1 \le j \le n$. И наоборот, функция f унатна, если эти условия выполняются для всех j. (Заметим, что $\curlywedge x_j f = \mathbb{N} x_j f = 0$ тогда и только тогда, когда $\mathbb{C} x_j f = 0$, тогда и только тогда, когда f не зависит от x_j. В таких случаях y_j не имеет значения; в противном случае y_j определяется однозначно.)

(б) Приведенный далее алгоритм поддерживает глобальные переменные $(p_1, \ldots, p_n)$, изначально равные нулю и обладающие тем свойством, что $p_j = +1$, если y_j должно быть равно 0, и $p_j = -1$, если y_j должно быть равно 1; p_j будет оставаться нулевым, если f не зависит от x_j. С пониманием этого UNATE(f) определяется следующим образом. Если f представляет собой константу, вернуть *true*. В противном случае представить f, как в (50). Вернуть *false*, если либо UNATE(f_l), либо UNATE(f_h) равно *false*; в противном случае установить $r \leftarrow$ COMPARE(f_l, f_h) с использованием упр. 104. Если r представляет собой '‖', вернуть *false*. Если r представляет собой '<', вернуть *false*, если $p_v < 0$, в противном случае установить $p_v \leftarrow +1$ и вернуть *true*. Если r представляет собой '>', вернуть *false*, если $p_v > 0$, в противном случае установить $p_v \leftarrow -1$ и вернуть *true*.

Этот алгоритм весьма часто быстро завершается. Он опирается на тот факт, что $f(x) \le g(x)$ для всех x тогда и только тогда, когда $f(x \oplus y) \le g(x \oplus y)$ для всех x при фиксированном y. Если мы просто хотим протестировать монотонность функции f, то вместо нулевого значения переменные p должны быть инициализированы значением $+1$.

106. Определим $\text{HORN}(f,g,h)$ следующим образом. Если $f > g$, обменяем $f \leftrightarrow g$. Затем, если $f = 0$ или $h = 1$, вернем *true*. В противном случае если $g = 1$ или $h = 0$, вернем *false*. В противном случае представим f, g и h как в (59). Вернем *true*, если $\text{HORN}(f_l, g_l, h_l)$, $\text{HORN}(f_l, g_h, h_l)$, $\text{HORN}(f_h, g_l, h_l)$ и $\text{HORN}(f_h, g_h, h_h)$ все равны *true*; в противном случае вернем *false*. [Этот алгоритм опубликован в T. Horiyama and T. Ibaraki, *Artificial Intelligence* **136** (2002), 189–213; его авторам также принадлежит алгоритм, аналогичный алгоритму из ответа к упр. 105, (б).]

107. Пусть $e\$f\$g\$h$ означает, что из $e(x) = f(y) = g(z) = 1$ вытекает $h(\langle xyz\rangle) = 1$. Тогда f является функцией Крома тогда и только тогда, когда $f\$f\$f\$f$, и мы можем использовать следующий рекурсивный алгоритм $\text{KROM}(e,f,g,h)$. Переупорядочить $\{e,f,g\}$ так, чтобы $e \le f \le g$. Затем, если $e = 0$ или $h = 1$, вернуть *true*. В противном случае если $f = 1$ или $h = 0$, вернуть *false*. В противном случае представить e, f, g, h с помощью кватернарного аналога (59). Вернуть *true*, если все $\text{KROM}(e_l, f_l, g_l, h_l)$, $\text{KROM}(e_l, f_l, g_h, h_l)$, $\text{KROM}(e_l, f_h, g_l, h_l)$, $\text{KROM}(e_l, f_h, g_h, h_h)$, $\text{KROM}(e_h, f_l, g_l, h_l)$, $\text{KROM}(e_h, f_l, g_h, h_h)$, $\text{KROM}(e_h, f_h, g_l, h_h)$ и $\text{KROM}(e_h, f_h, g_h, h_h)$ равны *true*; в противном случае вернуть *false*.

108. Пометим узлы как $\{1,\ldots,s\}$, причем корень получает метку 1, а стоки получают метки $\{s-1, s\}$; тогда $(s-3)!$ перестановок прочих меток дают различные ориентированные ациклические графы для одной и той же функции. Указанное неравенство следует из того, что каждая инструкция ($\bar v_k$? l_k: h_k) имеет не более $n(s-1)^2$ возможных вариантов для $1 \le k \le s-2$. (В действительности оно выполняется также для произвольных *ветвящихся программ*, а именно для бинарных диаграмм решений в общем случае, независимо от того, являются ли они упорядоченными и/или приведенными.)

Поскольку $1/(s-3)! < (s-1)^3/s!$ и $s! > (s/e)^s$, мы имеем $b(n,s) < (nse)^s$. Пусть $s_n = 2^n/(n+\theta)$, где $\theta = \lg e = 1/\ln 2$; тогда $\lg b(n,s_n) < s_n \lg(ns_ne) = 2^n(1 - (\lg(1+\theta/n))/(n+\theta)) = 2^n - \Omega(2^n/n^2)$. Так что вероятность того, что случайная булева функция от n переменных имеет $B(f) \le s_n$, не превышает $1/2^{\Omega(2^n/n^2)}$. А это очень малая величина.

109. $1/2^{\Omega(2^n/n^2)}$ очень мало́ даже при умножении на $n!$.

110. Пусть $f_n = M_m(x_{n-m+1},\ldots,x_n; 0,\ldots,0,x_1,\ldots,x_{n-m}) \vee (\bar x_{n-m+1} \wedge \cdots \wedge \bar x_n \wedge [0\ldots0x_1\ldots x_{n-m}$ является квадратом$]$), когда $2^{m-1}+m-1 < n < 2^m+m$. Каждый член этой формулы имеет 2^m+m-n нулей; второй член уничтожает все 2^m-битовые квадраты. [См. H.-T. Liaw and C.-S. Lin, *IEEE Transactions* **C-41** (1992), 661–664; Y. Breitbart, H. Hunt III, and D. Rosenkrantz, *Theoretical Comp. Sci.* **145** (1995), 45–69.]

111. Пусть $\mu n = \lambda(n - \lambda n)$, и заметим, что $\mu n = m$ тогда и только тогда, когда $2^m + m \le n < 2^{m+1} + m + 1$. Сумма для $0 \le k < n - \mu n$ равна $2^{n-\mu n} - 1$; прочие члены суммы дают $2^{2^{\mu n}}$.

112. Предположим, что $k = n - \lg n + \lg\alpha$. Тогда

$$\frac{(2^{2^{n-k}}-1)^{2^k}}{2^{2^n}} = \exp\Bigl(\frac{2^n\alpha}{n}\ln\Bigl(1-\frac{1}{2^{n/\alpha}}\Bigr)\Bigr) = \exp\Bigl(-\frac{2^{n-n/\alpha}\alpha}{n}\Bigl(1+O\Bigl(\frac{1}{2^{n/\alpha}}\Bigr)\Bigr)\Bigr).$$

Если $\alpha \le \frac12$, мы имеем $2^{n-n/\alpha}\alpha/n \le 1/(n2^{n+1})$; следовательно, $\hat b_k = (2^{n/\alpha} - 2^{n/(2\alpha)}) \times (2^{n-n/\alpha}\alpha/n)(1+O(2^{-n/\alpha})) = 2^k(1-O(2^{-n/(2\alpha)}))$. А если $\alpha \ge 2$, мы имеем $2^{n-n/\alpha}\alpha/n \ge 2^{n/2+1}/n$; таким образом $\hat b_k = (2^{2^{n-k}} - 2^{2^{n-k-1}})(1+O(\exp(-2^{n/2}/n)))$.

[Дисперсия b_k найдена в работе I. Wegener, *IEEE Trans.* **C-43** (1994), 1262–1269.]

113. Эта идея, на первый взгляд, довольно привлекательна, но при более близком знакомстве теряет свой блеск. Согласно теореме U на нижних уровнях базы БДР находится сравнительно немного узлов; и алгоритмы наподобие алгоритма S затрачивают сравнительно небольшое время на работу с этими уровнями. Кроме того, неконстантные

стоковые узлы могут усложнить ряд других алгоритмов, в первую очередь, алгоритмов переупорядочения.

114. Например, таблица истинности может иметь вид 01010101 00110011 00001111 00001111.

115. Пусть $N_k = b_0 + \cdots + b_{k-1}$ представляет собой количество узлов $\textcircled{j}$ БДР, для которых $j \le k$. Сумма входящих степеней этих узлов составляет как минимум N_k; сумма исходящих степеней равна $2N_k$; и имеется внешний указатель на корень. Таким образом, от k верхних уровней к нижним могут идти не более $N_k + 1$ ветвей. Каждая подтаблица порядка $n - k$ соответствует некоторой такой ветви. Следовательно, $q_k \le N_k + 1$.

Более того, мы должны иметь $q_k \le b_k + \cdots + b_n$, поскольку каждая подтаблица порядка $n - k$ соответствует единственной бусине порядка $\le n - k$.

В случае (124) заменим 'БДР' на 'НДР', 'b_k' на 'z_k', 'бусина' на 'z-бусина', а 'q_k' на 'q'_k'.

116. (а) Пусть $v_k = 2^{2^k} + 2^{2^{k-1}} + \cdots + 2^{2^0}$. Тогда $Q(f) \le \sum_{k=1}^{n+1} \min(2^{k-1}, 2^{2^{n+1-k}}) = U_n + v_{\lambda(n-\lambda n)-1}$. Примеры наподобие (78) показывают, что эта верхняя граница не может быть улучшена.

(б) $\hat{q}_k/\hat{b}_k = 2^{2^{n-k}}/(2^{2^{n-k}} - 2^{2^{n-k-1}})$ для $0 \le k < n$; $\hat{q}_n = \hat{b}_n$.

117. $q_k = 2^k$ для $0 \le k \le m$, а $q_{m+k} = 2^m + 2 - k$ для $1 \le k \le 2^m$. Следовательно, $Q(f) = 2^{2m-1} + 7 \cdot 2^{m-1} - 1 \approx B(f)^2/8$. (Такие f делают КДР непривлекательными с практической точки зрения.)

118. Если $n = 2^m - 1$, мы имеем $h_n(x_1, \ldots, x_n) = M_m(z_{m-1}, \ldots, z_0; 0, x_1, \ldots, x_n)$, где $(z_{m-1} \ldots z_0)_2 = x_1 + \cdots + x_n$ вычислимо за $5n - 5m$ шагов согласно упр. 7.1.2–30, а M_m требует других $2n + O(\sqrt{n})$ шагов согласно упр. 7.1.2–39. Поскольку $h_n(x_1, \ldots, x_n) = h_{n+k}(x_1, \ldots, x_n, 0, \ldots, 0)$, мы имеем $C(h_n) \le 14n + O(\sqrt{n})$ для всех n. (Поработав немного больше, можно уменьшить эту величину до $7n + O(\sqrt{n} \log n)$; не смогут ли читатели достичь большего?)

Стоимость h_4 составляет $6 = L(h_4)$, а $x_2 \oplus ((x_1 \oplus (x_2 \wedge \bar{x}_4)) \wedge (\bar{x}_3 \oplus (\bar{x}_2 \wedge x_4)))$ представляет собой формулу наименьшей длины. (Кроме того, $C(h_5) = 10$ и $L(h_5) = 11$.)

119. Истинно. Например, $S_{2,3,5}(x_1, \ldots, x_6) = h_{13}(x_1, x_2, 0, 0, 1, 1, 0, 1, 0, x_3, x_4, x_5, x_6)$.

120. Мы имеем $h_n^\pi(x_1, \ldots, x_n) = h_n(y_1, \ldots, y_n)$, где $y_j = x_{j\pi}$ для $1 \le j \le n$. А $h_n(y_1, \ldots, y_n) = y_{y_1 + \cdots + y_n} = y_{x_1 + \cdots + x_n} = x_{(x_1 + \cdots + x_n)\pi}$.

121. (а) Если $y_k = \bar{x}_{n+1-k}$, мы имеем $h_n(y_1, \ldots, y_n) = y_{\nu y} = y_{n-\nu x} = \bar{x}_{n+1-(n-\nu x)} = \bar{x}_{\nu x+1}$.

(б) Если $x = (x_1, \ldots, x_n)$ и $t \in \{0, 1\}$, мы имеем $h_{n+1}(x, t) = (t?\ x_{\nu x+1} \colon x_{\nu x})$.

(в) Нет. Например, ψ отображает $0^k 11 \mapsto 0^{k-1} 101 \mapsto 0^{k-2} 10^2 1 \mapsto \cdots \mapsto 10^k 1 \mapsto 0^k 11$. (Несмотря на простое определение, перестановка ψ обладает замечательными свойствами, включая наличие фиксированных точек, таких как 10011010000101011000111001011 и 11101111011001011101111101111.)

(г) На самом деле $\hat{h}_n(x_1 \ldots x_n) = x_1$(!) по индукции с использованием рекуррентного соотношения из п. (б).

(Если $f(x_1, \ldots, x_n)$ является *произвольной* булевой функцией, а τ представляет собой *произвольную* перестановку бинарных векторов $x_1 \ldots x_n$, можно записать $f(x) = \hat{f}(x\tau)$ и может оказаться, что работать с преобразованной функцией $\hat{f}$ гораздо проще. Поскольку $f(x) \wedge g(x) = \hat{f}(x\tau) \wedge \hat{g}(x\tau)$, преобразование И от двух функций представляет собой И от их преобразований, и т. д. Рассмотренные в разделе перестановки вектора $(x_1 \ldots x_n)\pi = x_{1\pi} \ldots x_{n\pi}$, которые просто преобразуют индексы, представляет собой простой частный случай этого общего принципа. Но этот принцип в определенном смысле оказывается *слишком* общим, поскольку каждая функция f тривиально имеет как минимум одну перестановку τ, для которой $\hat{f}$ является худой в смысле упр. 170; вся сложность f может быть перенесена в τ. Даже простые преобразования наподобие ψ имеют ограниченную

полезность, поскольку не образуют хорошие композиции; например, $\psi\psi$ не является преобразованием того же самого типа. Но линейные преобразования, выполняющие отображения $x \mapsto xT$ для некоторых несингулярных бинарных матриц T, доказанно полезны для упрощения БДР. [См. S. Aborhey, *IEEE Trans.* **C-37** (1988), 1461–1465; J. Bern, C. Meinel, and A. Slobodová, *ACM/IEEE Design Automation Conf.* **32** (1995), 408–413; C. Meinel, F. Somenzi, and T. Theobald, *IEEE Trans.* **CAD-19** (2000), 521–533.])

122. Например, при $n = 7$ рекуррентное соотношение из ответа к упр. 121, (б) дает

,

где затененные узлы вычисляют подфункцию h^{DR} от переменных, которые еще не были протестированы. Упрощения находятся в нижней части, поскольку $h_2(x_1, x_2) = x_1$ и $h_2^{DR}(x_1, x_2) = x_2$. [См. D. Sieling and I. Wegener, *Theoretical Comp. Sci.* **141** (1995), 283–310.]

123. Пусть $t = k - s = \bar{x}_1 + \cdots + \bar{x}_k$. Имеется реестр для каждой комбинации s' единиц и t' нулей, такой, что $s' + t' = w$, $s' \le s$ и $t' \le t$. Сумма $\binom{w}{s'} = \binom{w}{t'}$ по всем таким (s', t') равна (97). (Заметим также, что она равна 2^w тогда и только тогда, когда $w \le \min(s, t)$.)

124. Пусть $m = n - k$. Каждый реестр $[r_0, \ldots, r_m]$ соответствует функции от $(x_{k+1}, \ldots, x_n)$, таблица истинности которой является бусиной, за исключением четырех случаев: (i) $[0, \ldots, 0] = 0$; (ii) $[1, \ldots, 1] = 1$; (iii) $[0, x_n, 1] = x_n$ (который не зависит от x_{n-1}); (iv) $[1, \ldots, 1, x_{k+1}, 0, \ldots, 0]$ (где имеется p единиц, так что $x_{k+1} = r_p$), которая представляет собой $S_{<p}(x_{k+2}, \ldots, x_n)$.

Приведенный далее алгоритм за полиномиальное время вычисляет $q_k = q$ и $b_k = q - q'$ путем подсчета всех реестров. Когда все записи $[r_0, \ldots, r_m]$ представляют собой нули или единицы, имеется тонкость, связанная с тем, что такие реестры могут существовать для разных значений s; не хотелось бы подсчитывать их дважды. Решение заключается в поддержке четырех множеств

$$C_{ab} = \{r_1 + \cdots + r_{m-1} \mid r_0 = a \text{ и } r_m = b \text{ в некотором реестре}\}.$$

Значение 0π должно быть искусственно установлено равным $n + 1$, а не 0. Мы считаем, что $0 \le k < n$.

H1. [Инициализация.] Установить $m \leftarrow n - k$, $q \leftarrow q' \leftarrow s \leftarrow 0$, $C_{00} \leftarrow C_{01} \leftarrow C_{10} \leftarrow C_{11} \leftarrow \emptyset$.

H2. [Поиск v и w.] Установить $v \leftarrow \sum_{j=1}^{m-1} [(s + j)\pi \le k]$, $w \leftarrow v + [s\pi \le k] + [(s + m)\pi \le k]$. Если $v = m - 1$, перейти к шагу H5.

H3. [Проверка небусин.] Установить $p \leftarrow -1$. Если $v \ne m - 2$, перейти к шагу H4. В противном случае, если $m = 2$ и $(s + 1)\pi = n$, установить $p \leftarrow [(s + 2)\pi \le k]$. В противном случае, если $w = m$ и $(s + j)\pi = k + 1$ для некоторого $j \in [1 \,..\, m-1]$, установить $p \leftarrow j$.

H4. [Добавление биномов.] Для всех s' и t', таких, что $s'+t'=w$, $0\le s'\le s$ и $0\le t'\le k-s$, установить $q\leftarrow q+\binom{w}{s'}$ и $q'\leftarrow q'+[s'=p]$. Затем перейти к шагу H6.

H5. [Запоминание 0–1 реестров.] Для всех s' и t', таких, как на шаге H4, выполнить следующие действия. Если $(s+m)\pi\le k$, установить $C_{00}\leftarrow C_{00}\cup s'$ и $C_{01}\leftarrow C_{01}\cup(s'-1)$; в противном случае установить $C_{01}\leftarrow C_{01}\cup s'$. Если $s\pi\le k$ и $(s+m)\pi\le k$, установить $C_{10}\leftarrow C_{10}\cup(s'-1)$ и $C_{11}\leftarrow C_{11}\cup(s'-2)$. Если $s\pi\le k$ и $(s+m)\pi>k$, установить $C_{11}\leftarrow C_{11}\cup(s'-1)$.

H6. [Цикл по s.] Если $s<k$, установить $s\leftarrow s+1$ и вернуться к шагу H2.

H7. [Завершение.] Для $ab=00$, 01, 10 и 11 установить $q\leftarrow q+\binom{m-1}{r}$ для всех $r\in C_{ab}$. Установить также $q'\leftarrow q'+[0\in C_{00}]+[m-1\in C_{11}]$. ▮

125. Пусть $S(n,m)=\binom{n}{0}+\cdots+\binom{n}{m}$. При $0<s\le k$ и $s\ge 2k-n+2$ имеется $S(k+1-s,s)-1$ неконстантных реестров. Единственные другие неконстантные реестры возникают по одному при $s=0$ и $k<(n-1)/2$. Константные реестры учесть сложнее, но обычно их $S(n+1-k,2k+1-n)$ штук, появляющихся при $s=2k-n$ или $s=2k+1-n$. Принимая во внимание граничные условия и небусины, мы находим

$$b_k = S(n-k,2k-n)+\sum_{s=0}^{n-k}S(n-k-s,\,2k+1-n+s) \\ -\min(k,n-k)-[n=2k]-[3k\ge 2n-1]-1$$

для $0\le k<n$. Хотя $S(n,m)$ не имеет простого вида, можно выразить $\sum_{k=0}^{n-1}b_k$ как $B_{n/2}+\sum_{0\le m\le n-2k\le n}(n+3-m-2k)\binom{k}{m}+$(мелочь) при четном n, и то же самое выражение работает при нечетном n, если заменить $B_{n/2}$ на $A_{(n+1)/2}$. С двойной суммой можно справиться, выполнив сначала суммирование по k, поскольку $(k+1)\binom{k}{m}=(m+1)\binom{k+1}{m+1}$:

$$\sum_{m=0}^{n}\Bigl((n+5-m)\binom{\lfloor(n-m+2)/2\rfloor}{m+1}-(2m+2)\binom{\lfloor(n-m+4)/2\rfloor}{m+2}\Bigr).$$

Оставшуюся сумму можно разбить на четыре части, в зависимости от четности m и/или n. Здесь оказываются полезными производящие функции. Пусть $A(z)=\sum_{k\le n}\binom{n-k}{2k}z^n$ и $B(z)=\sum_{k\le n}\binom{n-k}{2k+1}z^n$. Тогда $A(z)=1+\sum_{k<n}\binom{n-k-1}{2k}z^n+\sum_{k<n}\binom{n-k-1}{2k-1}z^n=1+\sum_{k\le n}\binom{n-k}{2k}z^{n+1}+\sum_{k\le n}\binom{n-k}{2k+1}z^{n+2}=1+zA(z)+z^2B(z)$. Аналогичный вывод доказывает, что $B(z)=zB(z)+zA(z)$. Следовательно,

$$A(z)=\frac{1-z}{1-2z+z^2-z^3}=\frac{1-z^2}{1-z-z^2-z^4},\quad B(z)=\frac{z}{1-2z+z^2-z^3}=\frac{z+z^2}{1-z-z^2-z^4}.$$

Таким образом, $A_n=2A_{n-1}-A_{n-2}+A_{n-3}=A_{n-1}+A_{n-2}+A_{n-4}$ для $n\ge 4$ и B_n удовлетворяет тем же рекуррентным соотношениям. Фактически, мы имеем $A_n=(3P_{2n+1}+7P_{2n}-2P_{2n-1})/23$ и $B_n=(3P_{2n+2}+7P_{2n+1}-2P_{2n})/23$ с использованием чисел Перрина из упр. 15.

Кроме того, устанавливая $A^*(z)=\sum_{k\le n}k\binom{n-k}{2k}z^n$ и $B^*(z)=\sum_{k\le n}k\binom{n-k}{2k+1}z^n$, мы находим $A^*(z)=z^2A(z)B(z)$ и $B^*(z)=z^2B(z)^2$. Если собрать все это вместе, можно получить замечательную точную формулу

$$B(h_n)=\frac{56P_{n+2}+77P_{n+1}+47P_n}{23}-\left\lfloor\frac{n^2}{4}\right\rfloor-\left\lfloor\frac{7n+1}{3}\right\rfloor+(n\bmod 2)-10.$$

Исторические справки. Последовательность $\langle A_n\rangle$, по всей видимости, впервые была изучена Р. Остином (R. Austin) и Р. К. Гаем (R. K. Guy), *Fibonacci Quarterly* **16** (1978),

84–86; она подсчитывает бинарные $x_1 \ldots x_{n-1}$ со всеми единицами одно за другим. Константа χ, как было показано К. Л. Сигелем (C. L. Siegel), представляет собой наименьшее "число Писо", а именно наименьшее алгебраическое число > 1, все сопряженные к которому находятся внутри единичной окружности; см. *Duke Math. J.* **11** (1944), 597–602.

126. Когда $n \geq 6$, мы имеем $b_k = F_{\lfloor (k+7)/2 \rfloor} + F_{\lceil (k+7)/2 \rceil} - 4$ для $1 \leq k < 2n/3$ и $b_k = 2^{n-k+2} - 6 - [k = n-2]$ для $4n/5 \leq k < n$. Но основные ограничения на $B(h_n^\pi)$ исходят от $2n/15$ элементов профиля между этими двумя областями, и методы из ответа к упр. 125 можно расширить для работы с ними. Интересные последовательности

$$A_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-2k}{3k}, \qquad B_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-2k}{3k+1}, \qquad C_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n-2k}{3k+2}$$

имеют соответствующие производящие функции $(1-z)^2/p(z)$, $(1-z)z/p(z)$, $z^2/p(z)$, где $p(z) = (1-z)^3 - z^5$. Эти последовательности возникают в данной задаче из-за того, что $\sum_{k=0}^{n} \binom{\lfloor n-2k/3 \rfloor}{k} = A_n + B_{n-1} + C_{n-2}$. Они растут как α^n, где $\alpha \approx 1.7016$ представляет собой действительный корень уравнения $(\alpha-1)^3\alpha^2 = 1$.

Размер БДР не может быть выражен в аналитическом виде, но существует аналитическая запись с использованием членов последовательности от $A_{\lfloor n/3 \rfloor}$ до $A_{\lfloor n/3 \rfloor + 4}$ с точностью $O(2^{n/4}/\sqrt{n})$. Таким образом, $B(h_n^\pi) = \Theta(\alpha^{n/3})$.

127. (Перестановка $\pi = (3,\ 5,\ 7,\ \ldots,\ 2n'-1,\ n,\ n-1,\ n-2,\ \ldots,\ 2n',\ 2n'-2,\ \ldots,\ 4,\ 2,\ 1)$, $n' = \lfloor 2n/5 \rfloor$, оказывается оптимальной для h_n при $12 < n \leq 24$; но она дает $B(h_{100}^\pi) = 1\,366\,282\,025$. Как показано в упр. 152, просеивание работает гораздо лучше; но практически гарантированно существуют еще лучшие перестановки.)

128. Рассмотрим, например, $M_3(x_4, x_2, x_7; x_6, x_1, x_8, x_3, x_9, x_{11}, x_5, x_{10})$. Первые m переменных $\{x_4, x_2, x_7\}$ называются "адресными битами"; прочие 2^m называются "целевыми". Подфункции, соответствующие $x_1 = c_1, \ldots, x_k = c_k$, могут быть описаны реестрами вариантов, аналогичными (96). Например, когда $k = 2$, имеется три реестра, $[x_6, 0, x_9, x_{11}]$, $[x_6, 1, x_9, x_{11}]$, $[x_8, x_3, x_5, x_{10}]$; результат в этом случае получается путем использования $(x_4x_7)_2$ для выбора соответствующего компонента. Только третий из них зависит от x_3; следовательно, $q_2 = 3$ и $b_2 = 1$. Когда $k = 6$, реестрами являются $[0,0]$, $[0,1]$, $[1,0]$, $[1,1]$, $[x_8,0]$, $[x_8,1]$, $[x_9,x_{11}]$, $[0,x_{10}]$ и $[1,x_{10}]$ с компонентами, выбираемыми переменной x_7; следовательно, $q_6 = 9$ и $b_6 = 7$.

В общем случае, если переменные $\{x_1, \ldots, x_k\}$ включают a адресных битов и t целевых, реестры будут иметь $A = 2^{m-a}$ записей. Разделим множество всех 2^m целевых битов на 2^a подмножества, зависящих от адресных битов, и предположим, что s_j из этих подмножеств содержат j известных целевых битов. (Таким образом, $s_0 + s_1 + \cdots + s_A = 2^a$ и $s_1 + 2s_2 + \cdots + As_A = t$. Мы имеем $(s_0, \ldots, s_4) = (1,1,0,0,0)$ при $k = 2$ и $a = t = 1$ в примере выше; и $(s_0, s_1, s_2) = (1,2,1)$ при $k = 6$, $a = 2$, $t = 4$.) Тогда общее количество реестров q_k равно $2^0 s_0 + 2^1 s_1 + \cdots + 2^{A-1} s_{A-1} + 2^A [s_A > 0]$. Если x_{k+1} является адресным битом, количество реестров b_k, зависящих от x_{k+1}, равно $q_k - 2^{A/2}[s_A > 0]$. В противном случае $b_k = 2^c$, где c — количество констант, находящихся в реестрах, содержащих целевой бит x_{k+1}.

129. (Решение М. Соергофа (M. Sauerhoff); см. I. Wegener, *Branching Programs* (2000), Theorem 6.2.13.) Поскольку $P_m(x_1, \ldots, x_{m^2}) = Q_m(x_1, \ldots, x_{m^2}) \wedge S_m(x_1, \ldots, x_{m^2})$ и $B(S_m) = m^3 + 2$, мы имеем $B(P_m^\pi) \leq (m^3+2)B(Q_m^\pi)$. Применяем теорему К.

(Должна существовать более строгая нижняя граница, поскольку, как представляется, Q_m имеет *бóльшие* БДР, чем P_m. Например, при $m = 5$ перестановка $(1\pi, \ldots, 25\pi) = (3,\ 1,\ 5,\ 7,\ 9,\ 2,\ 4,\ 6,\ 8,\ 10,\ 11,\ 12,\ 13,\ 14,\ 15,\ 16,\ 20,\ 23,\ 17,\ 21,\ 19,\ 18,\ 22,\ 24,\ 25)$ является наилучшей для Q_5; но $B(Q_5^\pi) = 535$ в то время как $B(P_5) = 229$.)

130. (а) Каждый путь, который начинается в корне БДР и содержит s ветвей HI и t ветвей LO, определяет подфункцию, которая соответствует графам, в которых обязательны s смежностей, а t — запрещены. Мы покажем, что эти $\binom{s+t}{s}$ подфункций различны.

Если подфункции g и h соответствуют различным путям, можно найти k вершин W, обладающих следующими свойствами: (i) W содержит вершины w и w' с w — w', обязательным в g и запрещенным в h. (ii) Не имеется смежностей между вершинами W, обязательными в h или запрещенными в g. (iii) Если $u \in W$ и $v \notin W$, и u — v обязательно в h, то $u = w$ или $u = w'$. (Эти условия делают не более $2s+t = m-k$ вершин непригодными для помещения в W.)

Можно установить оставшиеся переменные таким образом, что u — v тогда и только тогда, когда $\{u, v\} \subseteq W$, если смежность не является ни обязательной, ни запрещенной. Такое присваивание делает $g = 1$, $h = 0$.

(б) Рассмотрим подфункцию от $C_{m,\lceil m/2\rceil}$, в которой вершины $\{1, \ldots, k\}$ обязаны быть изолированными, но u — v при $k < u \le \lceil m/2\rceil < v \le m$. Тогда k-клика на $\lfloor m/2\rfloor$ вершинах $\{\lceil m/2\rceil + 1, \ldots, m\}$ эквивалентна $\lceil m/2\rceil$-клике на $\{1, \ldots, m\}$. Другими словами, эта подфункция от $C_{m,\lceil m/2\rceil}$ представляет собой $C_{\lfloor m/2\rfloor,k}$.

Теперь выберем $k \approx \sqrt{m/3}$ и применим (а). [I. Wegener, *JACM* **35** (1988), 461–471.]

131. (а) Можно показать, что профиль имеет вид $(1, 1, 2, 4, \ldots, 2^{q-1}, (p-2) \times (2^q - 1, q \times 2^{q-1}), 2^q - 1, 2^{q-1}, \ldots, 4, 2, 1, 2)$, где $r \times b$ означает r-кратное повторение b. Следовательно, общий размер равен $(pq + 2p - 2q + 2)2^{q-1} - p + 2$.

(б) В случае упорядочения $x_1, x_2, \ldots, x_p, y_{11}, y_{21}, \ldots, y_{p1}, \ldots, y_{1q}, y_{2q}, \ldots, y_{pq}$ профиль принимает вид $(1, 2, 4, \ldots, 2^{p-1}, (q-1)p \times (2^{p-1}), 2^{p-1}, \ldots, 4, 2, 1, 2)$, делая общий размер равным $(pq - p + 4)2^{p-1}$.

(в) Предположим, что среди первых k переменных в некотором упорядочении находится ровно $m = \lfloor \min(p, q)/2\rfloor$ переменных x; можно считать, что это $\{x_1, \ldots, x_m\}$. Рассмотрим 2^m путей в КДР для C, таких, что $x_j = \bar{x}_{m+j}$ для $1 \le j \le p - m$ и $y_{ij} = [i = j$ или $i = j+m$, или $j > m]$. Эти пути должны проходить через различные узлы на уровне k. Следовательно, $q_k \ge 2^m$; воспользуйтесь (85). [См. M. Nikolskaia and L. Nikolskaia, *Theor. Comp. Sci.* **255** (2001), 615–625.]

Оптимальными упорядочениями для $(p, q) = (4, 4)$, $(4, 5)$ и $(5, 4)$ с использованием упр. 138 являются:

$$x_1 y_{11} x_2 y_{21} x_3 y_{31} y_{41} y_{12} y_{22} y_{32} y_{42} y_{13} y_{23} y_{33} y_{43} y_{14} y_{24} y_{34} y_{44} x_4 \text{ (размер 108);}$$
$$x_1 y_{11} x_2 y_{21} x_3 y_{31} y_{41} y_{12} y_{22} y_{32} y_{42} y_{13} y_{23} y_{33} y_{43} y_{14} y_{24} y_{34} y_{44} y_{15} y_{25} y_{35} y_{45} x_4 \text{ (размер 140);}$$
$$x_1 y_{11} x_2 y_{21} y_{12} y_{22} y_{13} y_{23} y_{14} y_{24} x_3 y_{31} y_{32} y_{33} y_{34} x_4 y_{41} y_{42} y_{51} y_{52} y_{43} y_{53} y_{44} y_{54} x_5 \text{ (размер 167).}$$

132. Согласно табл. 7.1.1–5 имеется 616 126 существенно различных классов функций от 5 переменных. Максимальное значение $B_{\min}(f)$, равное 17, достигается в 38 из этих классов. Три класса обладают тем свойством, что $B(f^\pi) = 17$ для *всех* перестановок π; один такой пример, $((x_2 \oplus x_4 \oplus (x_1 \wedge (x_3 \vee \bar{x}_4))) \wedge ((x_2 \oplus x_5) \vee (x_3 \oplus x_4))) \oplus (x_5 \wedge (x_3 \oplus (x_1 \vee \bar{x}_2)))$, обладает интересными симметриями $f(x_1, x_2, x_3, x_4, x_5) = f(\bar{x}_2, \bar{x}_3, \bar{x}_4, \bar{x}_1, \bar{x}_5) = f(x_2, \bar{x}_5, x_1, x_3, \bar{x}_4)$.

Кстати, максимальная разность $B_{\max}(f) - B_{\min}(f) = 10$ достигается только в классе "функции соединения" x_1? x_2: x_3? x_4: x_5 при $B_{\min} = 7$ и $B_{\max} = 17$.

(При $n = 4$ имеется 222 класса; и $B_{\min}(f) = 10$ в 25 из них, включая S_2 и $S_{2,4}$. Класс, проиллюстрированный таблицей истинности `16ad`, уникально труден, в том смысле что $B_{\min}(f) = 10$ и большинство из 24 перестановок дают $B(f^\pi) = 11$.)

133. Представим каждое подмножество $X \subseteq \{1, \ldots, n\}$ n-битовым целым числом $i(X) = \sum_{x \in X} 2^{x-1}$, и пусть $b_{i(X),x}$ обозначает вес ребра между X и $X \cup x$. Установим $c_0 \leftarrow 0$ и для $1 \le i < 2^n$ установим $c_i \leftarrow \min\{c_{i \oplus j} + b_{i \oplus j,x} \mid j = 2^{x-1} \text{ и } i \,\&\, j \ne 0\}$. Тогда $B_{\min}(f) = c_{2^n - 1} + 2$, и оптимальное упорядочение может быть найдено, если вспомнить,

какие $x = x(i)$ минимизируют каждое c_i. Для поиска $B_{\max}$ в этом рецепте следует заменить 'min' на 'max'.

134.

Максимальный профиль, $(1,2,4,2,2)$, встречается на путях, таких как $\emptyset \to \{2\} \to \{2,3\} \to \{2,3,4\} \to \{1,2,3,4\}$. Минимальный профиль, $(1,2,2,1,2)$, встречается только на путях $\emptyset \to (\{3\}$ или $\{4\}) \to \{3,4\} \to \{1,3,4\} \to \{1,2,3,4\}$. (Пять из 24 возможных путей имеют профиль $(1,2,3,2,2)$ и не улучшаются просеиванием ни по одной из переменных.)

135. Пусть $\theta_0 = 1$, $\theta_1 = x_1$, $\theta_2 = x_1 \wedge x_2$ и $\theta_n = x_n ?\ \theta_{n-1} : \theta_{n-3}$ для $n \geq 3$. Можно доказать, что когда $n \geq 4$, $B(\theta_n^\pi) = n+2$ тогда и только тогда, когда $(n\pi, \ldots, 1\pi) = (1, \ldots, n)$. Ключевым фактом является то, что, если $k < n$ и $n \geq 5$, подфункции, получаемые путем установки $x_k \leftarrow 0$ или $x_k \leftarrow 1$, различны, и обе они зависят от переменных $\{x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n\}$, за исключением того, что подфункция для $x_{n-1} \leftarrow 0$ не зависит от x_{n-2}. Таким образом веса $\{x_k\} \to \{x_k, x_l\}$ в главной профилограмме равны 2, за исключением случаев $k = n$ или $(k,l) = (n-1, n-2)$. Ниже $\{x_{n-1}, x_{n-2}\}$ имеются три подфункции, а именно $x_n ?\ \theta_{n-4} : \theta_{n-3}$, $x_n ?\ \theta_{n-5} : \theta_{n-3}$ и θ_{n-3}; все они зависят от $\{x_1, \ldots, x_{n-3}\}$, а две из них — от x_n.

136. Пусть $n = 2n' - 1$ и $m = 2m' - 1$. Входные данные образуют матрицу размером $m \times n$, и мы вычисляем медиану m медиан строк. Пусть V_i обозначает переменные в строке i. Если X представляет собой подмножество mn переменных, обозначим $X_i = X \cap V_i$ и $r_i = |X_i|$. Подфункции типа $(s_1, \ldots, s_m)$ возникают, когда ровно s_i элементов X_i установлены равными 1; этими подфункциями являются

$$\langle S_1 S_2 \ldots S_m \rangle, \qquad \text{где } S_i = S_{\geq n' - s_i}(V_i \backslash X_i) \text{ и } 0 \leq s_i \leq r_i \text{ для } 1 \leq i \leq m.$$

Когда $x \notin X$, мы хотим подсчитать, сколько из этих подфункций зависят от x. Из соображений симметрии можно считать, что $x = x_{mn}$. Заметим, что симметричная пороговая функция $S_{\geq t}(x_1, \ldots, x_n)$ равна 0, если $t > n$, или 1, если $t \leq 0$; она зависит от всех n переменных, если $1 \leq t \leq n$. В частности, S_m зависит от x ровно для $r_m \$ n = \min(r_m + 1, n - r_m)$ выборов s_m.

Пусть $a_j = \sum_{i=1}^{m-1} [r_i = j]$ для $0 \leq j \leq n$. Тогда a_n из функций $\{S_1, \ldots, S_{m-1}\}$ являются константными, а $a_{n-1} + \cdots + a_{n'}$ от них могут как быть, так и не быть константными. Выбор неконстантных c_i дает нам количество различных подфункций, зависящих от x, равное $(r_m \$ n)((a_n + a_{n-1} + \cdots + a_{n'} - c_{n-1} - \cdots - c_{n'}) \$ m)$, умноженному на

$$\binom{a_{n-1}}{c_{n-1}} \cdots \binom{a_{n'}}{c_{n'}} 1^{a_0} 2^{a_1} \ldots (n')^{a_{n'-1}} (n'-1)^{c_{n'}} (n'-2)^{c_{n'+1}} \ldots 1^{c_{n-1}}.$$

Суммирование по $\{c_{n-1}, \ldots, c_{n'}\}$ дает нам ответ.

Когда переменные естественным образом построчно упорядочены, эти формулы применимы с $r_m = k \bmod n$, $a_n = \lfloor k/n \rfloor$, $a_0 = m - 1 - a_n$. Элемент профиля b_k для $0 \leq k < mn$,

таким образом, равен $(\lfloor k/n\rfloor\$m)((k \bmod n)\$n)$, и мы имеем $\sum_{k=0}^{mn} b_k = (m'n')^2 + 2$. Это упорядочение оптимально, хотя, похоже, не имеется простого доказательства этого факта; например, некоторые упорядочения могут уменьшать b_{n+2} или b_{2n-2} от 4 до 3, увеличивая при этом b_k для других k.

Каждый путь от вершины до низа главной профилограммы можно представить как $\alpha_0 \to \alpha_1 \to \cdots \to \alpha_{mn}$, где каждое α_j представляет собой строку $r_{j1}\ldots r_{jm}$ с $0 \le r_{j1} \le \cdots \le r_{jm} \le n$, $r_{j1} + \cdots + r_{jm} = j$ с увеличением на каждом шаге одной координаты. Например, одним из путей при $m = 5$ и $n = 3$ является 00000 → 00001 → 00011 → 00111 → 00112 → 00122 → 00123 → 01123 → 11123 → 11223 → 12223 → 12233 → 12333 → 22333 → 23333 → 33333. Этот путь можно конвертировать в "естественный" путь путем ряда шагов, которые не увеличивают общий вес ребер, следующим образом. В начальном отрезке, до первого раза, когда $r_{jm} = n$, первыми выполняем все переходы по крайней справа координате. (Таким образом, первыми шагами пути в примере станут 00000 → 00001 → 00002 → 00003 → 00013 → 00113 → 00123.) Затем в последнем отрезке после последнего $r_{j1} = 0$ последними выполняем все переходы по крайней слева координате. (Последними шагами, таким образом, станут 01123 → 01223 → 02223 → 02233 → 02333 → 03333 → 13333 → 23333 → 33333.) Затем, после первых n шагов аналогично нормализуем вторые с конца координаты (00003 → 00013 → 00023 → 00033 → 00133 → 01133 → 01233 → 02233); а перед последними n шагами — вторые координаты (00133 → 00233 → 00333 → 01333 → 02333 → 03333). И т. д.

[Этот метод доказательства "назад и вперед" был вдохновлен цитируемой ниже статьей Боллиг (Bollig) и Вегенера (Wegener). Можно ли улучшить каждое неоптимальное упорядочение с помощью простого просеивания?]

137. Если добавить клику из c новых вершин и $\binom{c}{2}$ новых ребер, стоимость оптимального размещения увеличится на $\binom{c+1}{3}$. Так что можно считать, что заданный граф имеет m ребер и n вершин $\{1,\ldots,n\}$, где m и n нечетны и достаточно велики. Соответствующая функция f, которая зависит от $mn + m + 1$ переменных x_{ij} и s_k для $1 \le i \le m$, $1 \le j \le n$ и $0 \le k \le m$, представляет собой $J(s_0, s_1, \ldots, s_m; h, g_1, \ldots, g_m)$, где $g_i = (x_{iu_i} \oplus x_{iv_i}) \wedge \bigwedge\{x_{iw} \mid w \notin \{u_i, v_i\}\}$, когда i-м ребром является u_i — v_i и где $h = \langle\langle x_{11}\ldots x_{m1}\rangle\ldots\langle x_{1n}\ldots x_{mn}\rangle\rangle$ представляет собой транспозицию функции в упр. 136.

Можно показать, что $B_{\min}(f) = \min_\pi \sum_{u-v} |u\pi - v\pi| + (\frac{m+1}{2})^2(\frac{n+1}{2})^2 + mn + m + 2$; оптимальное упорядочение использует $(\frac{m+1}{2})^2(\frac{n+1}{2})^2$ узлов для h, $n + |u_i\pi - v_i\pi|$ узлов для g_i, по одному узлу для каждого s_k, и два стоковых узла, минус один узел, разделенный между h и некоторым g_i. [См. B. Bollig and I. Wegener, *IEEE Trans.* **C-45** (1996), 993–1002.]

138. (а) Пусть $X_k = \{x_1, \ldots, x_k\}$. Узлы КДР на глубине k представляют подфункции, которые могут получиться, когда константы замещают переменные X_k. Мы можем добавить к каждому узлу n-битовое поле DEP для определения, от каких именно переменных $X_n \setminus X_k$ он зависит. Например, КДР для f из (92) имеет следующие подфункции и поля DEP:

глубина 0: 0011001001110010 [1111];
глубина 1: 00110010 [0111], 01110010 [0111];
глубина 2: 0010 [0011], 0011 [0010], 0111 [0011];
глубина 3: 00 [0000], 01 [0001], 10 [0001], 11 [0000].

Изучение всех полей DEP на глубине k дает нам веса главной профилограммы между X_k и $X_k \cup x_l$ для $0 \le k < l \le n$.

(б) Представим узлы на глубине k в виде троек $N_{kp} = (l_{kp}, h_{kp}, d_{kp})$ для $0 \le p < q_k$, где (l_{kp}, h_{kp}) представляет собой указатели (LO, HI), а d_{kp} записывает биты DEP. Если $k < n$, эти узлы выполняют ветвление по x_{k+1}, так что мы имеем $0 \le l_{kp}, h_{kp} < q_{k+1}$; но если $k = n$, мы имеем $l_{n0} = h_{n0} = 0$ и $l_{n1} = h_{n1} = 1$ для представления $\boxed{\bot}$ и $\boxed{\top}$. Определим

$d_{kp} = \sum\{2^{t-k-1} \mid N_{kp}$ зависит от $x_t\}$; следовательно, $0 \le d_{kp} < 2^{n-k}$. Например, КДР (82) эквивалентна $N_{00} = (0,1,7)$; $N_{10} = (0,1,3)$, $N_{11} = (1,2,3)$; $N_{20} = (0,0,0)$, $N_{21} = (0,1,1)$, $N_{22} = (1,1,0)$; $N_{30} = (0,0,0)$, $N_{31} = (1,1,0)$.

Для подъема с глубины b до глубины a мы, по сути, делаем две копии узлов на глубинах $b-1$, $b-2$, $\ldots$, a, — по одной для случаев $x_{b+1} = 0$ и $x_{b+1} = 1$. Эти копии перемещаются вниз до глубин b, $b-1$, $\ldots$, $a+1$, и с ними выполняется приведение для удаления дубликатов. Затем каждый оригинальный узел на глубине a заменяется узлом с ветвлением по x_{b+1}; его поля LO и HI указывают соответственно на 0- и 1-копию оригинала.

Этот процесс включает некоторую достаточно простую обработку списков для обновления полей DEP в процессе карманной сортировки: узлы распаковываются в рабочую область, состоящую из вспомогательных массивов r, s, t, u и v, изначально нулевых. Вместо использования l_{kp} и h_{kp} для LO и HI мы сохраняем HI в ячейке u_p рабочей области и связываем v_p с предыдущим узлом (если таковой имеется) с таким же полем LO; кроме того, мы делаем s_l указывающим на последний узел, для которого $\mathrm{LO} = l$ (если таковой имеется). В приведенном ниже алгоритме UNPACK(p,l,h) используется как аббревиатура для "$u_p \leftarrow h$, $v_p \leftarrow s_l$, $s_l \leftarrow p+1$".

Когда узлы на глубине k распакованы таким образом в массивы s, u и v, следующая подпрограмма ELIM(k) пакует их обратно в структуру КДР с удаленными дубликатами. Она также устанавливает r_p равным новому адресу узла p.

E1. [Цикл по l.] Установить $q \leftarrow 0$ и $t_h \leftarrow 0$ для $0 \le h < q_{k+1}$. Выполнить шаг E2 для $0 \le l < q_{k+1}$. Затем установить $q_k \leftarrow q$ и завершить работу алгоритма.

E2. [Цикл по p.] Установить $p \leftarrow s_l$ и $s_l \leftarrow 0$. Пока $p > 0$, выполнять шаг E3 и устанавливать $p \leftarrow v_{p-1}$. Затем продолжить выполнение шага E1.

E3. [Упаковка узла $p-1$.] Установить $h \leftarrow u_{p-1}$. (Распакованный узел имеет $(\mathrm{LO}, \mathrm{HI}) = (l,h)$.) Если $t_h \ne 0$ и $l_{k(t_h-1)} = l$, установить $r_{p-1} \leftarrow t_h - 1$. В противном случае установить $l_{kq} \leftarrow l$, $h_{kq} \leftarrow h$, $d_{kq} \leftarrow ((d_{(k+1)l} \mid d_{(k+1)h}) \ll 1) + [l \ne h]$, $r_{p-1} \leftarrow q$, $q \leftarrow q+1$, $t_h \leftarrow q$. Продолжить выполнение шага E2. ▌

Теперь мы можем использовать ELIM для подъема от b до a. (i) Для $k = b-1$, $b-2$, $\ldots$, a выполнить следующие шаги. Для $0 \le p < q_k$ установить $l \leftarrow l_{kp}$, $h \leftarrow h_{kp}$; если $k = b-1$, UNPACK$(2p, l_{bl}, h_{bl})$ и UNPACK$(2p+1, l_{bh}, h_{bh})$, в противном случае UNPACK$(2p, r_{2l}, r_{2h})$ и UNPACK$(2p+1, r_{2l+1}, r_{2h+1})$ (тем самым создавая две копии N_{kp} в рабочей области). Затем выполнить ELIM$(k+1)$. (ii) Для $0 \le p < q_a$ выполнить UNPACK(p, r_{2p}, r_{2p+1}). Затем выполнить ELIM(a). (iii) Если $a > 0$, установить $l \leftarrow l_{(a-1)p}$, $h \leftarrow h_{(a-1)p}$, $l_{(a-1)p} \leftarrow r_l$, $h_{(a-1)p} \leftarrow r_h$ для $0 \le p < q_{a-1}$.

Эта процедура подъема искажает содержимое полей DEP выше глубины a, поскольку переменные становятся переупорядоченными. Но мы будем использовать ее только с теми полями, которые больше не нужны.

(в) По индукции на первых 2^{n-2} шагах вычисляются все подмножества, не содержащие n; затем выполняется подъем от $n-1$ до 0, а на остальных шагах вычисляются все подмножества, которые содержат n.

(г) Начнем с установок $y_k \leftarrow k$ и $w_k \leftarrow 2^k - 1$ для $0 \le k < n$. В приведенном далее алгоритме массив y представляет текущее упорядочение переменных, а битовая карта $w_k = \sum\{2^{y_j} \mid 0 \le j < k\}$ — множество переменных на k верхних уровнях.

Мы наращиваем подпрограмму ELIM(k), так что она также вычисляет искомые веса ребер главной профилограммы: счетчики c_j изначально равны 0 для $0 \le j < n-k$; после установки d_{kq} на шаге E3 мы устанавливаем $c_j \leftarrow c_j + 1$ для каждого j, такого, что $2^j \subseteq d_{kq}$; наконец мы устанавливаем $b_{w_k, y_{k+j}+1} \leftarrow c_j$ для $0 \le j < n-k$, используя обозначения из ответа к упр. 133. [Для ускорения мы можем подсчитывать байты, а не биты, увеличивая $c_{j,(d_{kq} \gg 8j)\&\#\mathtt{ff}}$ на 1 для $0 \le j < (n-k)/8$.]

Мы инициализируем поля DEP, выполняя следующие действия для $k = n-1$, $n-2$, ..., 0: UNPACK(p, l_{kp}, h_{kp}) для $0 \le p < q_k$; ELIM(k); если $k > 0$, установить $l \leftarrow l_{(k-1)p}$, $h \leftarrow h_{(k-1)p}$, $l_{(k-1)p} \leftarrow r_l$ и $h_{(k-1)p} \leftarrow r_h$ для $0 \le p < q_{k-1}$.

Основной цикл алгоритма теперь выполняет следующие действия для $1 \le i < 2^{n-1}$. Установить $a \leftarrow \nu i - 1$ и $b \leftarrow \nu i + \rho i$. Установить $(y_a, \ldots, y_b) \leftarrow (y_b, y_a, \ldots, y_{b-1})$ и $(w_{a+1}, \ldots, w_b) \leftarrow (2^{y_b} + w_a, \ldots, 2^{y_b} + w_{b-1})$. Выполнить подъем от b до a с помощью процедуры из п. (б); но использовать на шаге (ii) исходную (не дополненную) подпрограмму ELIM для вызова ELIM(a) на шаге (ii).

(д) Пространство, необходимое для узлов на глубине k, не превышает $Q_k = \min(2^k, 2^{2^{n-k}})$; нам также нужна память для $2\max(Q_1, \ldots, Q_n)$ элементов в массивах r, u, v плюс $\max(Q_1, \ldots, Q_n)$ элементов в массивах s и t. Так что в окончательном ответе доминирует $O(2^n n)$ для выходных данных $b_{w,x}$.

Подпрограмма ELIM(k) вызывается $\binom{n}{k}$ раз в нарощенном виде для $0 \le k < n$ и $\binom{n-1}{k+1}$ раз в недополненном. В обоих случаях время ее работы составляет $O(q_k(n-k))$. Таким образом, общее время работы составляет $O\bigl(\sum_k \binom{n}{k} 2^k (n-k)\bigr) = O(3^n n)$ и будет значительно меньшим, если КДР никогда не станет большой. (Например, оно равно $O((1+\sqrt{2})^n n)$ для функции h_n.)

[Первый точный алгоритм для определения оптимального упорядочения переменных в БДР был разработан С. Д. Фридманом (S. J. Friedman) и К. Д. Суповитом (K. J. Supowit), *IEEE Trans.* **C-39** (1990), 710–713. Они использовали расширенные таблицы истинности вместо КДР, получив для $m = 1$ метод, который требовал $\Theta(3^n/\sqrt{n})$ памяти и $\Theta(3^n n^2)$ времени и мог быть усовершенствован до $\Theta(3^n n)$.]

139. Применим тот же алгоритм почти без изменений: будем рассматривать все узлы КДР с ветвлением по x_a как находящиеся на уровне 0, а все узлы с ветвлением по x_{b+1} — как стоки. Таким образом, мы выполним 2^{b-a} подъемов, а не 2^{n-1}. (Этот алгоритм не полагается на предположение о том, что $q_0 = 1$ и $q_n = 2$, за исключением анализа требуемых памяти и времени в п. (д).)

140. Можно найти кратчайшие пути в сети без предварительного знания сети путем генерации вершин и дуг "на лету" при необходимости. В разделе 7.3 указывается, что расстояние $d(X, Y)$ каждой дуги $X \to Y$ может быть изменено на $d'(X, Y) = d(X, Y) - l(X) + l(Y)$ для любой функции $l(X)$ без изменения кратчайших путей. Если пересмотренные расстояния d' неотрицательны, $l(X)$ представляет собой нижнюю границу для расстояния от X к цели; трюк заключается в том, чтобы найти хорошую нижнюю границу, которую не слишком сложно вычислить.

Если $|X| = l$ и если КДР для f с X на ее верхних l уровнях имеет неконстантные узлы на следующем уровне, то $l(X) = \max(q, n-l)$ представляет собой подходящую нижнюю границу для задачи $B_{\min}$. [См. R. Drechsler, N. Drechsler and W. Günther, *ACM/IEEE Design Automation Conf.* **35** (1998), 200–205.] Тем не менее, чтобы этот подход мог конкурировать с алгоритмом из упр. 138, нужна более сильная нижняя граница, если только f не имеет относительно короткой БДР, которая не может быть достигнута очень многими способами.

141. Ложно. Рассмотрим $g(x_1 \vee \cdots \vee x_6,\ x_7 \vee \cdots \vee x_{12},\ (x_{13} \vee \cdots \vee x_{16}) \oplus x_{18},\ x_{17},\ x_{19} \vee \cdots \vee x_{22})$, где $g(y_1, \ldots, y_5) = ((((\bar{y}_1 \vee y_5) \wedge y_4) \oplus y_3) \wedge ((y_1 \wedge y_2) \oplus y_4 \oplus y_5)) \oplus y_5$. Тогда $B(g) = 40 = B_{\min}(g)$ не может быть достигнуто с последовательными $\{x_{13}, \ldots, x_{16}, x_{18}\}$. [M. Teslenko, A. Martinelli and E. Dubrova, *IEEE Trans.* **C-54** (2005), 236–237.]

142. (а) Предположим, что m нечетно. Подфункциями, которые возникают после того, как станут известны $(x_1, \ldots, x_{m+1})$, являются $[w_{m+2}x_{m+2} + \cdots + w_n x_n > 2^{m-1}m - 2^{m-2} - t]$, где $0 \le t \le 2^m$. Подслучаи $x_{m+2} + \cdots + x_n = (m-1)/2$ показывают, что как минимум $\binom{m-1}{(m-1)/2}$ из этих подфункций различны.

Но порядок органных труб $\langle x_1 x_2^{2^m-1} x_3^1 x_4^{2^m-2} x_5^2 \ldots x_{n-2}^{2^m-2^{m-2}} x_{n-1}^{2^{m-2}} x_n^{2^{m-1}} \rangle$ гораздо лучше: пусть $t_k = x_1 + (2^m-1)x_2 + x_3 + \cdots + (2^m - 2^{k-1})x_{2k} + 2^{k-1}x_{2k+1}$ для $1 \le k < m-1$. Оставшаяся подфункция зависит не более чем от $2k+2$ различных значений, $\lceil t_k/2^k \rceil$.

(б) Пусть $n = 1 + 4m^2$. Переменными являются x_0 и x_{ij} для $0 \le i, j < 2m$; весами являются $w_0 = 1$ и $w_{ij} = 2^i + 2^{2m+1+j}m$. Пусть X_l представляет собой первые l переменных в некотором порядке, и предположим, что X_l включает элементы в i_l строках и j_l столбцах матрицы (x_{ij}). Докажем, что если $\max(i_l, j_l) = m$, то $q_l \ge 2^m$; следовательно, $B(f) > 2^m$ согласно (85).

Пусть I и J являются подмножествами $\{1, \ldots, 2m\}$, такими, что $|I| = |J| = m$ и $X_l \subseteq x_0 \cup \{x_{ij} \mid i \in I,\ j \in J\}$; и пусть I' и J' представляют собой дополняющие подмножества. Выберем m элементов $X' \subseteq X_l \setminus x_0$ в разных строках (или, если $i_l < m$, в разных столбцах). Рассмотрим 2^m путей в КДР, определяемых следующим образом: $x_0 = 0$ и $x_{ij} = 0$, если $x_{ij} \in X_l \setminus X'$; также $x_{i'j} = x_{ij'} = \bar{x}_{i'j'} = \bar{x}_{ij}$ для $i \in I$, $j \in J$, где $i \leftrightarrow i'$ и $j \leftrightarrow j'$ являются паросочетаниями между $I \leftrightarrow I'$ и $J \leftrightarrow J'$. Тогда имеется 2^m различных значений $t = \sum_{i \in I, j \in J} w_{ij} x_{ij}$; но на каждом пути $\sum_{0 \le i,j < 2m} w_{ij} x_{ij} = (2^{2m}-1)(1+2^{2m+1}m)$. Эти пути должны проходить через различные узлы на уровне l. В противном случае, если $t \ne t'$, один из нижних подпутей будет вести к $\boxed{\bot}$, а другой к $\boxed{\top}$.

[Эти результаты получены в работе K. Hosaka, Y. Takenaga, T. Kaneda and S. Yajima, *Theoretical Comp. Sci.* **180** (1997), 47–60, авторы которой доказали также, что $|Q(f) - Q(f^R)| < n$. Всегда ли также самодуальная пороговая функция удовлетворяет условию $|B(f) - B(f^R)| < n$?]

143. Фактически в алгоритмах из упр. 133 и 138 доказывается, что для этих весов наилучшим является порядок органных труб: (1, 1023, 1, 1022, 2, 1020, 4, 1016, 8, 1008, 16, 992, 32, 960, 64, 896, 128, 768, 256, 512) дает профиль (1, 2, 2, 4, 3, 6, 4, 8, 5, 10, 4, 8, 3, 6, 2, 4, 1, 2, 2, 1, 2) и $B(f) = 80$. Наихудшее упорядочение (1022, 896, 512, 64, 8, 1, 4, 32, 1008, 1020, 768, 992, 1016, 1023, 960, 256, 128, 16, 2, 1) делает $B(f) = 1913$.

(Можно решить, что свойства бинарной записи являются ключевыми для этого примера. Однако $\langle x_1 x_2 x_3^2 x_4^4 x_5^8 x_6^{16} x_7^{31} x_8^{60} x_9^{116} x_{10}^{224} x_{11}^{224} x_{12}^{448} x_{13}^{564} x_{14}^{620} x_{15}^{649} x_{16}^{664} x_{17}^{672} x_{18}^{676} x_{19}^{678} x_{20}^{679} \rangle$ в действительности является той же функцией согласно упр. 7.1.1–103(!).)

144. (5, 7, 7, 10, 6, 9, 5, 4, 2); КДР-не-БДР-узлы соответствуют f_1, f_2, f_3, 0, 1.

145. $B_{\min} = 31$ достигается в (36). Наихудшее упорядочение для $(x_3x_2x_1x_0)_2 + (y_3y_2y_1y_0)_2$ представляет собой y_0, y_1, y_2, y_3, x_2, x_1, x_0, x_3 и делает $B_{\max} = 107$. Кстати, наихудшим упорядочением для 24 входов 12-битового сложения $(x_{11} \ldots x_0)_2 + (y_{11} \ldots y_0)_2$ оказывается y_0, y_1, $\ldots$, y_{11}, x_{10}, x_8, x_6, x_4, x_3, x_5, x_2, x_7, x_1, x_9, x_0, x_{11}, дающее $B_{\max} = 39\,111$.

[Б. Боллиг (B. Bollig), Н. Рэндж (N. Range) и И. Вегенер (I. Wegener), *Lecture Notes in Comp. Sci.* **4910** (2008), 174–185, доказали, что $B_{\min} = 9n-5$ для сложения двух n-битовых чисел при $n > 1$, а также что для 2^m-канального мультиплексора $B_{\min}(M_m) = 2n - 2m + 1$.]

146. (а) Очевидно, что $b_0 \le q_0$; а если $q_0 = b_0 + a_0$, то $b_1 \le 2b_0 + a_0 = b_0 + q_0$. Также $q_0 - b_0 = a_0 \le b_1 + q_2 \le q_2^2$, количества двухбуквенных строк q_2-буквенного алфавита; аналогично $b_0 + b_1 + q_2 \le (b_1 + q_2)^2$. (Те же соотношения выполняются между q_k, q_{k+2}, b_k и b_{k+1}.)

(б) Пусть подфункции на уровне 2 имеют таблицы истинности α_j для $1 \le j \le q_2$; воспользуемся ими для построения бусин β_1, $\ldots$, β_{b_1} на уровне 1. Пусть $(\gamma_1, \ldots, \gamma_{q_2+b_1})$ представляют собой таблицы истинности $(\alpha_1\alpha_1, \ldots, \alpha_{q_2}\alpha_{q_2}, \beta_1, \ldots, \beta_{b_1})$. Если $b_0 \le b_1/2$, пусть функции на уровне 0 имеют таблицы истинности $\{\beta_{2i-1}\beta_{2i} \mid 1 \le i \le b_0\} \cup \{\beta_j\beta_j \mid 2b_0 < j \le b_1\} \cup \{\gamma_j\gamma_j \mid 1 \le j \le b_0 + q_0 - b_1\}$. В противном случае нетрудно определить b_0 бусин, которые включают все β, и использовать их на уровне 0 вместе с небусинами $\{\gamma_j\gamma_j \mid 1 \le j \le q_0 - b_0\}$.

147. Прежде чем начать любое переупорядочение, мы очищаем кэш и выполняем сборку всего мусора. Приведенный далее алгоритм обменивает уровни $(u) \leftrightarrow (v)$ при $v = u + 1$. Он работает путем создания связных списков одиночных, связанных и скрытых узлов, на которые указывают переменные S, T и H (изначально Λ), с использованием вспомогательных полей LINK, которые могут быть временно заимствованы из алгоритма хеш-таблиц списков уникальности, после того как они будут перестроены.

T1. [Построение S и T.] Для каждого (u)-узла p установить $q \leftarrow \mathtt{LO}(p)$, $r \leftarrow \mathtt{HI}(p)$ и удалить p из его хеш-таблицы. Если $\mathtt{V}(q) \neq v$ и $\mathtt{V}(r) \neq v$ (p является одиночным), установить $\mathtt{LINK}(p) \leftarrow S$ и $S \leftarrow p$. В противном случае (p является связанным) установить $\mathtt{REF}(q) \leftarrow \mathtt{REF}(q) - 1$, $\mathtt{REF}(r) \leftarrow \mathtt{REF}(r) - 1$, $\mathtt{LINK}(p) \leftarrow T$ и $T \leftarrow p$.

T2. [Построение H и перемещение видимых узлов.] Для каждого (v)-узла p установить $q \leftarrow \mathtt{LO}(p)$, $r \leftarrow \mathtt{HI}(p)$ и удалить p из его хеш-таблицы. Если $\mathtt{REF}(p) = 0$ (p является скрытым), установить $\mathtt{REF}(q) \leftarrow \mathtt{REF}(q) - 1$, $\mathtt{REF}(r) \leftarrow \mathtt{REF}(r) - 1$, $\mathtt{LINK}(p) \leftarrow H$ и $H \leftarrow p$; в противном случае (p является видимым) установить $\mathtt{V}(p) \leftarrow u$ и $\mathrm{INSERT}(u, p)$.

T3. [Перемещение одиночных узлов.] Пока $S \neq \Lambda$, устанавливать $p \leftarrow S$, $S \leftarrow \mathtt{LINK}(p)$, $\mathtt{V}(p) \leftarrow v$ и $\mathrm{INSERT}(v, p)$.

T4. [Превращение связанных узлов.] Пока $T \neq \Lambda$, устанавливать $p \leftarrow T$, $T \leftarrow \mathtt{LINK}(p)$ и выполнять следующие действия. Установить $q \leftarrow \mathtt{LO}(p)$, $r \leftarrow \mathtt{HI}(p)$. Если $\mathtt{V}(q) > v$, установить $q_0 \leftarrow q_1 \leftarrow q$; в противном случае установить $q_0 \leftarrow \mathtt{LO}(q)$ и $q_1 \leftarrow \mathtt{HI}(q)$. Если $\mathtt{V}(r) > v$, установить $r_0 \leftarrow r_1 \leftarrow r$; в противном случае установить $r_0 \leftarrow \mathtt{LO}(r)$ и $r_1 \leftarrow \mathtt{HI}(r)$. Затем установить $\mathtt{LO}(p) \leftarrow \mathrm{UNIQUE}(v, q_0, r_0)$, $\mathtt{HI}(p) \leftarrow \mathrm{UNIQUE}(v, q_1, r_1)$ и $\mathrm{INSERT}(u, p)$.

T5. [Уничтожение скрытых узлов.] Пока $H \neq \Lambda$, устанавливать $p \leftarrow H$, $H \leftarrow \mathtt{LINK}(p)$ и удалять узел p. (Все оставшиеся узлы — живые.) ▌

Подпрограмма $\mathrm{INSERT}(v, p)$ просто помещает узел p в таблицу уникальности x_v, используя ключ $(\mathtt{LO}(p), \mathtt{HI}(p))$; этого ключа уже не будет в наличии. Подпрограмма UNIQUE на шаге T4 подобна алгоритму U, но вместо использования ответа к упр. 82 рассматривает счетчики ссылок совсем по-другому на шагах U1 и U2: если U1 обнаруживает $p = q$, он *увеличивает* $\mathtt{REF}(p)$ на 1; если U2 обнаруживает r, он просто устанавливает $\mathtt{REF}(r) \leftarrow \mathtt{REF}(r) + 1$.

Внутренне переменные ветвления остаются в своем естественном порядке $1, 2, \ldots, n$ сверху вниз. Таблицы отображений ρ и π представляют текущую перестановку с точки зрения внешнего пользователя, с $\rho = \pi^-$; таким образом, переменная пользователя x_v находится на уровне $v\pi - 1$, а узел $\mathrm{UNIQUE}(v, p, q)$ на уровне $v - 1$ представляет функцию пользователя $(\bar{x}_{v\rho}?\ p\colon q)$. Для поддержки этих отображений устанавливаем $j \leftarrow u\rho$, $k \leftarrow v\rho$, $u\rho \leftarrow k$, $v\rho \leftarrow j$, $j\pi \leftarrow v$, $k\pi \leftarrow u$.

148. Ложно. Например, рассмотрим шесть стоков и девять функций-источников с расширенными таблицами истинности 1156, 2256, 3356, 4456, 5611, 5622, 5633, 5644, 5656. Восемь из этих узлов связанные и один видимый, но нет ни одного скрытого или одиночного. Имеется 16 новичков: 15, 16, 25, 26, 35, 36, 45, 46, 51, 61, 52, 62, 53, 63, 54, 64. Так что обмен отображает 15 узлов в 31. (Мы можем использовать узлы $B(x_3 \oplus x_4, x_3 \oplus \bar{x}_4)$ в качестве стоков.)

149. Последовательные профили ограничены значениями $(b_0, b_1, \ldots, b_n)$, $(b_0 + b_1, 2b_0, b_2, \ldots, b_n)$, $(b_0 + b_1, 2b_0 + b_2, 4b_0, b_3, \ldots, b_n)$, $\ldots$, $(2^0 b_0 + b_1, \ldots, 2^{k-2} b_0 + b_{k-1}, 2^{k-1} b_0, b_k, \ldots, b_n)$. Аналогично в дополнение к теореме J^+ мы также имеем $B(f_1^\pi, \ldots, f_m^\pi) \le B(f_1, \ldots, f_m) + 2(b_0 + \cdots + b_{k-1})$, поскольку обмены вносят не более $2b_{k-1}$, $2b_{k-2}$, $\ldots$, $2b_0$ новых узлов.

150. Можно считать, что $m = 1$, как в упр. 52. Предположим, что мы хотим переместить x_k в позицию, являющуюся j-й в упорядочении, где $j \neq k$. Сначала вычислим ограничения

на f, когда $x_k = 0$ и $x_k = 1$ (см. упр. 57); назовем их g и h. Затем перенумеруем оставшиеся переменные: если $j < k$, заменим $(x_j, \ldots, x_{k-1})$ на $(x_{j+1}, \ldots, x_k)$; в противном случае заменим $(x_{k+1}, \ldots, x_j)$ на $(x_k, \ldots, x_{j-1})$. Затем вычислим $f \leftarrow (\bar{x}_j \wedge g) \vee (x_j \wedge h)$, используя вариант алгоритма S с линейным временем работы из упр. 72.

Чтобы показать, что этот метод имеет указанное в условии время работы, достаточно доказать следующее: *пусть $g(x_1, \ldots, x_n)$ и $h(x_1, \ldots, x_n)$ представляют собой функции, такие, что из $g(x) = 1$ вытекает $x_j = 0$, а из $h(x) = 1$ вытекает $x_j = 1$. Тогда комбинация $g \diamond h$ имеет не более чем в два раза больше узлов, чем $g \vee h$.* Но это почти очевидно, если рассматривать таблицы истинности: например, если $n = 3$ и $j = 2$, таблицы истинности для g и h имеют соответственно вид $ab00cd00$ и $00st00uv$. Бусины β у $g \vee h$ на уровнях $< j$ однозначно соответствуют бусинам $\beta' \diamond \beta''$ у $g \diamond h$ на этих уровнях, поскольку $\beta = \beta' \vee \beta''$ можно "разложить" только одним способом, помещая нули в соответствующие места. А бусины β у $g \vee h$ на уровнях $\geq j$ соответствуют не более двум бусинам у $g \diamond h$, а именно $\beta \diamond \boxed{\perp}$ и/или $\boxed{\perp} \diamond \beta$.

[См. P. Savický and I. Wegener, *Acta Informatica* **34** (1997), 245–256, Theorem 1.]

151. Установить $t_k \leftarrow 0$ для $1 \leq k \leq n$ и выполнить операцию обмена $x_{j-1} \leftrightarrow x_j$, а также $t_{j-1} \leftrightarrow t_j$. Затем установить $k \leftarrow 1$ и выполнять следующие действия до достижения $k > n$: если $t_k = 1$, установить $k \leftarrow k + 1$; в противном случае установить $t_k \leftarrow 1$ и просеять x_k.

(Этот метод многократно просеивает верхнюю среди еще не просеянных переменную. Исследователи испытывали более привлекательные стратегии, такие как просеивание первым наибольшего уровня; но ни одна из них не оказалась превосходящей предложенный здесь метод.)

152. Применение алгоритма J так, как в ответе к упр. 151 дает $B(h^{\pi}_{100}) = 1\,382\,685\,050$ после 17 179 обменов, что почти так же хорошо, как и результат "настроенной вручную" перестановки (95). Пара следующих просеиваний уменьшает размер до 300 451 396, а затем до 262 969 049; дальнейшие повторения сходятся к размеру 231 376 264 узлов после общего количества обменов, равного 232 951.

В случае обрыва циклов на шагах J2 и J5 при $S > 1.05s$ результаты оказываются даже лучшими (!), хотя и выполняется меньше обменов: 1 342 191 700 узлов после одного просеивания в конечном счете сокращаются до 208 478 228 после общего количества обменов, равного 139 245. Более того, Филип Стапперс (Filip Stappers) использовал просеивание вместе со случайным обменом в сентябре 2010 года и получил значение $B(h^{\pi}_{100})$, равное всего лишь 198 961 868, со следующей перестановкой π, являющейся "текущим чемпионом":

3	4	6	8	10	12	14	16	18	20	22	24	27	28	30	32	35	37	39	41
43	45	47	49	51	53	54	83	85	98	99	100	79	77	81	75	73	95	71	97
69	96	57	91	67	59	65	60	63	62	64	61	66	87	58	68	56	94	93	70
92	72	90	74	76	78	80	89	88	86	84	82	55	52	50	48	46	44	42	40
38	36	34	33	31	29	26	25	23	21	19	17	15	13	11	9	7	5	1	2

Кстати, если просеивать переменные h_{100} в порядке размера профиля, так что первой просеивается x_{60}, затем — x_{59}, x_{61}, x_{58}, x_{57}, x_{62}, x_{56} и т. д. (где бы они ни находились в данный момент), получающаяся в результате БДР имеет 2 196 768 534 узла.

Простой "нисходящий обмен" вместо полного просеивания неприменим для h_{100}: $\binom{100}{2}$ обменов $x_1 \leftrightarrow x_2$, $x_3 \leftrightarrow x_1$, $x_3 \leftrightarrow x_2$, $\ldots$, $x_{100} \leftrightarrow x_1$, $\ldots$, $x_{100} \leftrightarrow x_{99}$ полностью обращают порядок всех переменных, не влияя на размер БДР ни на одном шаге.

153. Каждый логический элемент легко синтезируется с использованием рекурсии наподобие (55). Около 1 Мбайт памяти и 3.5 миллионов обращений к памяти достаточно для построения всей базы БДР из 8242 узлов. Используя упр. 138, можно сделать вывод об оптимальности упорядочения x_7, x_3, x_9, x_1, o_9, o_1, o_3, o_7, x_4, x_6, o_6, o_4, o_2, o_8, x_2, x_8, o_5, x_5 и о том, что $B_{\min}(y_1, \ldots, y_9) = 5308$.

Переупорядочение переменных *не* рекомендуется для такого рода задач, поскольку имеется всего лишь 18 переменных. Например, автопросеивание при каждом удвоении размера потребует более 100 миллионов обращений к памяти, сокращая при этом 8242 узла до примерно 6400.

154. Да: при последнем просеивании `CA` переместилась между `ID` и `OR`, а выполняя всю работу в обратном направлении, можно вывести, что первое просеивание поместило `ME` между `MA` и `RI`.

155. Лучший результат автора в случае (а) имеет вид

```
ME NH VT MA CT RI NY DE NJ MD PA DC VA OH WV KY NC SC GA FL AL IN MI IA
IL MO TN AR MS TX LA CO WI KS SD ND NE OK WY MN ID MT NM AZ OR CA WA UT NV
```

и дает $B(f_1^\pi) = 403$, $B(f_2^\pi) = 677$, $B(f_1^\pi, f_2^\pi) = 1073$; в случае (б) упорядочение

```
NH ME MA VT CT RI NY DE NJ MD PA VA DC OH WV KY TN NC SC GA FL AL IN MI
IL IA AR MO MS TX LA CO KS OK WI SD NE ND MN WY ID MT AZ NM UT OR CA WA NV
```

дает $B(f_1^\pi) = 352$, $B(f_2^\pi) = 702$, $B(f_1^\pi, f_2^\pi) = 1046$.

156. Можно ожидать, что два "просеивания вверх" будут как минимум столь же хороши, как и одно двустороннее просеивание. В действительности же тесты Р. Руделла (R. Rudell) показали, что одного лишь просеивания вверх, определенно, недостаточно. Иногда спуски необходимы для того, чтобы компенсировать временные подъемы переменных, хотя их оптимальная окончательная позиция находится ниже.

157. Внимательное изучение ответа к упр. 128 показывает, что мы всегда улучшаем размер, когда первый адресный бит, за которым следует целевой, перемещается в положение после всех целевых битов. [Но простые обмены слишком слабы. Например, $M_2(x_1, x_6; x_2, x_3, x_4, x_5)$ и $M_3(x_1, x_{10}, x_{11}; x_2, x_3, \ldots, x_9)$ локально оптимальны при обменах $x_{j-1} \leftrightarrow x_j$ для любых j.]

158. Рассмотрим сначала случай $m = 1$ и $n = 3t - 1 \geq 5$. Тогда если $n\pi = k$, количество узлов с ветвлением по j равно a_j, если $j\pi < k$, b_j, если $j\pi = k$, и a_{n+2-j}, если $j\pi > k$, где

$$a_j = j - 3\max(j - 2t, 0), \qquad b_j = \min(j, t, n + 1 - j).$$

Случаями с последовательными $\{x_1, \ldots, x_{n-1}\}$ являются $k = 1$ и $B(f^\pi) = 3t^2 + 2$; $k = n$ и $B(f^\pi) = 3t^2 + 1$. Но когда $k = \lceil n/2 \rceil$, мы имеем $B(f^\pi) = \lfloor 3t/2 \rfloor (\lceil 3t/2 \rceil - 1) + n - \lfloor t/2 \rfloor + 2$.

Аналогичные вычисления применимы при $m > 1$: мы имеем $B(f^\pi) > 6\binom{p/3}{2} + B(g^\pi)$, когда π делает $\{x_1, \ldots, x_p\}$ последовательными, но $B(f^\pi) \approx 2\binom{p/2}{2} + \frac{p}{3}B(g^\pi)$, когда π помещает $\{x_{p+1}, \ldots, x_{p+m}\}$ в средину. Поскольку g фиксировано, $pB(g^\pi) = O(n)$ при $n \to \infty$.

[Если g является функцией того же вида, мы получаем примеры, когда симметричные переменные в g лучше разделить. Но булевы функции, для которых оптимальное значение $B(f^\pi)$ меньше, чем 3/4 наилучшего, при том ограничении, что никакие блоки симметричных переменных не являются разделенными, неизвестны. См. D. Sieling, *Random Structures & Algorithms* **13** (1998), 49–70.]

159. Это почти симметричная функция, так что имеется только девять вариантов. Когда центральный элемент x помещается в позицию $(1, 2, \ldots, 9)$ от вершины, размер БДР становится соответственно $(43, 43, 42, 39, 36, 33, 30, 28, 28)$.

160. (а) Вычислим $\bigwedge_{i=0}^{9} \bigwedge_{j=0}^{9} (\neg L_{ij}(X))$, булеву функцию от 64 переменных, например, путем 100-кратного применения COMPOSE к относительно простой функции L из упр. 159. В экспериментальной программе автора для поиска этой БДР (имеющей при обычном упорядочении 251 873 узла) потребовалось 35 Мбайт памяти и около 320 миллионов обращений к ней. Затем алгоритм С быстро находит интересующий нас ответ: 21 929 490 122. (Количество решений 11×11, 5 530 201 631 127 973 447, может быть найдено таким же образом.)

(б) Производящая функция представляет собой $1 + 64z + 2016z^2 + 39740z^3 + \cdots + 80z^{45} + 8z^{46}$, и алгоритм B быстро находит восемь решений с весом 46. Три из них различны по отношению к симметрии шахматной доски; наиболее симметричное решение показано ниже, на рис. (A0).

(в) БДР для $\bigwedge_{i=1}^{8}\bigwedge_{j=1}^{8}(\neg L_{ij}(X))$ имеет 305 507 узлов и 21 942 036 750 решений. Поэтому должно быть 12 546 628 диких решений.

(г) Теперь производящая функция представляет собой $40z^{14} + 936z^{15} + 10500z^{16} + \cdots + 16z^{55} + z^{56}$; примеры с весами 14 и 56 показаны ниже, на рис. (A1) и (A2).

(д) Ровно 28 с весом 27 и 54 с весом 28, все ручные; см. (A3).

(е) Имеется соответственно (26260, 5, 347, 0, 122216) решений, найденных с помощью около (228, 3, 32, 1, 283) миллионов обращений к памяти. Среди самых легких и самых тяжелых решений (1) — (A4) и (A5); самое красивое решение (2) — (A6); (A7) и (A9) решают (3) легко и (5) тяжело. Шаблон (4), основанный на бинарном представлении π, не имеет предшественников 8×8; но он имеет, например, предшественника 9×10, показанного на рис. (A8).

(A0) (A1) (A2) (A3) (A4) (A5) (A6) (A7) (A8) (A9)

161. (а) При нормальном построчном упорядочении $(x_{11}, x_{12}, \ldots, x_{n(n-1)}, x_{nn})$ БДР имеет 380 727 узлов и характеризует 4 782 725 решений. Вычисления требуют 100 Мбайт памяти и около 2 миллиардов обращений к ней. (Аналогично 29 305 144 137 натюрмортов размером 10×10 могут быть подсчитаны с помощью 14 492 923 узлов, полученных в результате менее чем 50 миллиардов обращений к памяти.)

(б) Это решение, по сути, единственное; см. (B1) ниже. Имеется также единственное (и очевидное) решение с весом 36.

(в) Теперь БДР имеет 128 переменных с упорядочением $(x_{11}, y_{11}, \ldots, x_{nn}, y_{nn})$. Можно сначала построить БДР для $[L(X) = Y]$ и $[L(Y) = X]$, а затем выполнить их пересечение; но это оказывается плохой идеей, требующей более 36 миллионов узлов даже в случае размера 7×7. Гораздо лучше применить ограничения $L_{ij}(X) = y_{ij}$ и $L_{ij}(Y) = x_{ij}$ построчно, а также добавить лексикографическое ограничение $X < Y$, так чтобы пораньше исключить натюрморты. Вычисление в этом случае может быть завершено с использованием 1.6 Гбайт памяти и около 20 миллиардов обращений к ней; в наличии имеется 978 563 узла и 582 769 решений.

(г) Решение вновь единственное с точностью до поворота; см. "свечу зажигания" (B2) $\leftrightarrow$ (B3). (А (B4) $\leftrightarrow$ (B5) представляет собой единственный перевертыш размером 7×7 с постоянным весом 26. Жизнь полна изумительных вещей!)

(B1) (B2) (B3) (B4) (B5) (B6)

162. Пусть $T(X) = [X$ ручная$]$ и $E_k(X) = [X$ сбегает после k шагов$]$. Мы можем вычислить БДР для каждого E_k, используя рекуррентное соотношение

$$E_1(X) = \neg T(X); \qquad E_{k+1}(X) = \exists Y(T(X) \wedge [L(X) = Y] \wedge E_k(Y)).$$

(Здесь $\exists Y$ обозначает $\exists y_{11}\, \exists y_{12} \cdots \exists y_{66}$. Как упоминалось в ответе к упр. 103, это рекуррентное соотношение оказывается гораздо эффективнее правила $E_{k+1} = T(X) \wedge E_k(L_{11}(X),$

$\ldots, L_{66}(X))$, хотя последнее выглядит более "элегантно".) Количество решений, $|E_k|$, оказывается равным ($806544 \cdot 2^{16}$, $657527179 \cdot 2^4$, 2105885159, 763710262, 331054880, 201618308, 126169394, 86820176, 63027572, 41338572, 30298840, 17474640, 9797472, 5258660, 3058696, 1416132, 523776, 204192, 176520, 62456, 13648, 2776, 2256, 440, 104, 0) для $k = (1, 2, \ldots, 26)$; таким образом, $\sum_{k=1}^{25} |E_k| = 67\,166\,017\,379$ из $2^{36} = 68\,719\,476\,736$ возможных конфигураций, которые в конечном итоге сбегают из клетки 6×6. (Один из 104 "тормозов" из E_{25} показан выше, на рис. (B6).)

Методы БДР отлично подходят для данной задачи при малых k; например, $B(E_1) = 101$ и $B(E_2) = 14\,441$. Но E_k в конечном итоге становится сложной "нелокальной" функцией: размер достигает пика $B(E_6) = 28\,696\,866$, после которого количество решений становится достаточно малым, чтобы размер уменьшился. До квантификации в формуле $T(X) \wedge [L(X) = Y] \wedge E_5(Y)$ присутствует более 80 миллионов узлов; это выходит за рамки разумного. Действительно, БДР для $\bigvee_{k=1}^{25} E_k(X)$ требует памяти больше, чем ее 2^{33}-байтовая таблица истинности. Следовательно, "прямой" метод для этого упражнения предпочтительнее применения БДР.

(Клетки, бóльшие, чем 6×6, выглядят невозможно сложными с точки зрения *любого* известного метода.)

163. Предположим сначала, что $\circ$ представляет собой $\wedge$. Мы получим БДР для $f = g \wedge h$, беря БДР для g и заменяя ее сток $\boxed{\top}$ корнем БДР для h. Чтобы представить также $\bar{f}$, сделаем отдельную копию БДР для g и используем базу БДР как для h, так и для $\bar{h}$; заменим в копии $\boxed{\bot}$ на $\boxed{\top}$ и заменим $\boxed{\top}$ в копии корнем БДР для $\bar{h}$. Эта диаграмма решений является приведенной, поскольку h — не константа.

Аналогично, если $\circ$ представляет собой $\oplus$, мы получаем БДР для $f = g \oplus h$ (и, возможно, $\bar{f}$) из БДР для g (и, возможно, $\bar{g}$) после замены $\boxed{\bot}$ и $\boxed{\top}$ корнями бинарных диаграмм решений для h и $\bar{h}$.

Другие бинарные операторы $\circ$, по сути, такие же, поскольку $B(f) = B(\bar{f})$. Например, если $f = g \supset h = \overline{g \wedge \bar{h}}$, мы имеем $B(f) = B(\bar{f}) = B(g) + B(\bar{h}) - 2 = B(g) + B(h) - 2$.

164. Пусть $U_1(x_1) = V_1(x_1) = x_1$, $U_{n+1}(x_1, \ldots, x_{n+1}) = x_1 \oplus V_n(x_2, \ldots, x_{n+1})$, и $V_{n+1}(x_1, \ldots, x_{n+1}) = U_n(x_1, \ldots, x_n) \wedge x_{n+1}$. Тогда по индукции можно показать, что $B(f) \le B(U_n) = 2^{\lceil (n+1)/2 \rceil} + 2^{\lfloor (n+1)/2 \rfloor} - 1$ для всех бесповторных f, а также что мы всегда имеем $B(f, \bar{f}) \le B(V_n, \overline{V_n}) = 2^{\lceil n/2 \rceil + 1} + 2^{\lfloor n/2 \rfloor + 1} - 2$. (Но оптимальное упорядочение сильно снижает эти размеры до $B(U_n^\pi) = \lfloor \frac{3}{2} n + 2 \rfloor$ и $B(V_n^\pi, \overline{V_n}^\pi) = 2n + 2$.)

165. По индукции также доказывается, что $B(u_{2m}, \bar{u}_{2m}) = 2^m F_{2m+3} + 2$, $B(u_{2m+1}, \bar{u}_{2m+1}) = 2^{m+1} F_{2m+3} + 2$, $B(v_{2m}, \bar{v}_{2m}) = 2^{m+1} F_{2m+1} + 2$, $B(v_{2m+1}, \bar{v}_{2m+1}) = 2^{m+1} F_{2m+3} + 2$.

166. Можно считать, как в ответе к упр. 163, что $\circ$ представляет собой либо $\wedge$, либо $\oplus$. Путем перенумерации можно также считать, что $j\sigma = j$ для $1 \le j \le n$, следовательно, $f^\sigma = f$. Пусть $(b_0, \ldots, b_n)$ представляет собой профиль f, а $(b'_0, \ldots, b'_n)$ — профиль $(f, \bar{f})$; и пусть $(c_{1\pi}, \ldots, c_{(n+1)\pi})$ и $(c'_{1\pi}, \ldots, c'_{(n+1)\pi})$ — профили f^π и $(f^\pi, \bar{f}^\pi)$, где $(n+1)\pi = n+1$. Тогда $c_{j\pi}$ представляет собой количество подфункций для $f^\pi = g^\pi \circ h^\pi$, которые зависят от $x_{j\pi}$ после установки переменных $\{x_{1\pi}, \ldots, x_{(j-1)\pi}\}$ равными фиксированным значениям. Аналогично $c'_{j\pi}$ является количеством таких подфункций для f^π или $\bar{f}^\pi$. Мы попытаемся доказать, что $b_{j\pi-1} \le c_{j\pi}$ и $b'_{j\pi-1} \le c'_{j\pi}$ для всех j.

Случай 1. $\circ$ представляет собой $\wedge$. Можно считать, что $n\pi = n$, поскольку оператор $\wedge$ коммутативен. *Случай 1, а.* $1 \le j\pi \le k$. Тогда $b_{j\pi-1}$ и $b'_{j\pi-1}$ подсчитывают подфункции, в которых определены только переменные $x_{i\pi}$ с $1 \le i < j$ и $1 \le i\pi \le k$. Эти подфункции для $g \wedge h$ или $\bar{g} \vee \bar{h}$ имеют аналоги, которые подсчитываются в $c_{j\pi}$ и $c'_{j\pi}$, поскольку h^π не является константой ни в какой подфункции при $n\pi = n$. *Случай 1, б.* $k < j\pi \le n$. Тогда $b_{j\pi-1}$ и $b'_{j\pi-1}$ подсчитывают подфункции для h или $\bar{h}$, которые имеют аналоги, подсчитываемые в $c_{j\pi}$ и $c'_{j\pi}$.

Случай 2. $\circ$ представляет собой $\oplus$. Можно считать, что $1\pi = 1$, поскольку оператор $\oplus$ коммутативен. Тогда применимы рассуждения, аналогичные рассуждениям из случая 1. [*Discrete Applied Math.* **103** (2000), 237–258.]

167. Пусть $f = f_{1n}$; будем рекурсивно вычислять $c_{ij} = B_{\min}(f_{ij})$, $c'_{ij} = B_{\min}(f_{ij}, \bar{f}_{ij})$ и перестановку π_{ij} множества $\{i, \ldots, j\}$ для каждой подфункции $f_{ij}(x_i, \ldots, x_j)$ следующим образом. Если $i = j$, мы имеем $f_{ij}(x_i) = x_i$; пусть $c_{ij} = 3$, $c'_{ij} = 4$, $\pi_{ij} = i$. В противном случае $i < j$, и мы имеем $f_{ij}(x_i, \ldots, x_j) = f_{ik}(x_i, \ldots, x_k) \circ f_{(k+1)j}(x_{k+1}, \ldots, x_j)$ для некоторого k и некоторого оператора $\circ$. Если $\circ$ наподобие $\wedge$, пусть $c_{ij} = c_{ik} + c_{(k+1)j} - 2$ и либо ($c'_{ij} = 2c_{ik} + c'_{(k+1)j} - 4$, $\pi_{ij} = \pi_{ik}\pi_{(k+1)j}$), либо ($c'_{ij} = 2c_{(k+1)j} + c'_{ik} - 4$, $\pi_{ij} = \pi_{(k+1)j}\pi_{ik}$) минимизирует c'_{ij}. Если $\circ$ наподобие $\oplus$, пусть $c'_{ij} = c'_{ik} + c'_{(k+1)j} - 2$ и либо ($c_{ij} = c_{ik} + c'_{(k+1)j} - 2$, $\pi_{ij} = \pi_{ik}\pi_{(k+1)j}$), либо ($c_{ij} = c_{(k+1)j} + c'_{ik} - 2$, $\pi_{ij} = \pi_{(k+1)j}\pi_{ik}$) минимизирует c_{ij}.

(Перестановки π_{ij}, в данном описании представленные в виде строк, в компьютере могут быть представлены как связанные списки. Можно также построить оптимальную БДР для f рекурсивно за $O(B_{\min}(f))$ шагов, воспользовавшись ответом к упр. 163.)

168. (а) Это утверждение преобразует и упрощает рекуррентные соотношения (112) и (113).

(б) Истинно по индукции; также $x \geq n$.

(в) Легко проверяемо. Заметим, что T является отражением относительно прямой $y = (\sqrt{2} - 1)x$ с наклоном $22\frac{1}{2}^\circ$.

(г) Если $z \in S_k$ и $z' \in S_{n-k}$, мы имеем $|z| = q^\beta$ и $|z'| = q'^\beta$, где $q \leq k$ и $q' \leq n - k$ по индукции. Из соображений симметрии можно положить $q = (1 - \delta)t$ и $q' = (1 + \delta)t$, где $t = \frac{1}{2}(q + q') \leq \frac{1}{2}n$. Тогда, если верно первое указание, мы имеем $|z \bullet z'| \leq (2t)^\beta \leq n^\beta$. Мы также будем иметь $|z \circ z'| \leq n^\beta$ согласно п. (в), поскольку $|z^T| = |z|$.

Чтобы доказать первое указание, заметим, что максимум $|z \bullet z'|$ достигается при $y = y'$. Когда $y \geq y'$, мы имеем $|z \bullet z'|^2 = (x + x' + y')^2 + y^2 = r^2 + 2(x' + y')x + (x' + y')^2$; наибольшее значение для заданного z' достигается при $y = y'$. Аналогичные рассуждения применимы при $y' \geq y$.

Теперь, когда $y = y'$, мы имеем $y = \sqrt{rr'}\sin\theta$ для некоторого θ; и можно показать, что $x + x' \leq (r + r')\cos\theta$. Таким образом, $z \bullet z' = (x + x' + y, y)$ лежит в эллипсе из второго указания. В этом эллипсе мы имеем $(a\cos\theta + b\sin\theta)^2 + (b\sin\theta)^2 = a^2/2 + b^2 + u\sin 2\theta + v\cos 2\theta = a^2/2 + b^2 + w\sin(2\theta + \tau)$, где $u = ab$, $v = \frac{1}{2}a^2 - b^2$, $w^2 = u^2 + v^2$ и $\cos\tau = u/w$. Следовательно, $|z \bullet z'|^2 \leq \frac{1}{2}a^2 + b^2 + w$. А $4w^2 = (r + r')^4 + 4(rr')^2 \leq (r^2 + (2\sqrt{5} - 2)rr' + r'^2)^2$, так что

$$|z \bullet z'|^2 \leq r^2 + (\sqrt{5} + 1)rr' + r'^2, \qquad r = (1 - \delta)^\beta,\ r' = (1 + \delta)^\beta.$$

Остается доказать, что эта величина не превышает $2^{2\beta} = 2\phi^2$; или, что то же самое, $f_t(2) \leq f_t(2\beta)$, где $f_t(\alpha) = (e^{t/\alpha} + e^{-t/\alpha})^\alpha - 2^\alpha$ и $t = \beta\ln((1 - \delta)/(1 + \delta))$. Фактически можно показать, что, когда $\alpha \geq 2$, функция f_t является возрастающей функцией от α. [См. G. Bennett, *AMM* **117** (2010), 334–351. Граница $O(n^\beta)$ для S_n, похоже, требует деликатного анализа; ранние попытки Соергофа (Sauerhoff), Вегенера (Wegener) и Верхнера (Werchner) оказались ошибочными. Приведенное здесь доказательство найдено А. К. Чангом (A. X. Chang) и В. И. Спитковски (V. I. Spitkovsky) в 2007 году.]

169. Эта гипотеза была проверена для $m \leq 7$. [Многие другие любопытные свойства также остаются необъясненными. В момент написания этого ответа "группой изучения любопытных явлений" готовится к публикации статья о том, что же известно к настоящему времени.]

170. (а) 2^{2n-1}. В $\textcircled{j}$ при $1 \leq j < n$ имеется четыре варианта, а именно LO $= \boxed{\perp}$ или LO $= \boxed{\top}$, или HI $= \boxed{\perp}$, или HI $= \boxed{\top}$; и два варианта для $\textcircled{n}$.

(б) 2^{n-1}, поскольку половина вариантов в каждом узле ветвления отбрасывается.

(в) Действительно, если $t = (t_1 \ldots t_n)_2$, мы имеем LO $= \boxed{\perp}$ в $\textcircled{j}$ при $t_j = 1$ и HI $= \boxed{\top}$ в $\textcircled{j}$ при $t_j = 0$. (Эта идея была применена к случайной генерации битов в упр. 3.4.1–25.

Поскольку имеется 2^{n-1} таких значений t, мы показали, что каждая монотонная худая функция является пороговой с весами $\{2^{n-1}, \ldots, 2, 1\}$. Другие худые функции получаются с помощью операции дополнения отдельных переменных.)

(г) $\bar{f}_t(\bar{x}) = [(\bar{x})_2 < t] = [(x)_2 > \bar{t}] = [(x)_2 > 2^n - 1 - t] = f_{2^n - t}(x)$.

(д) Согласно теореме 7.1.1Q кратчайшая DNF представляет собой ИЛИ простых импликант, и ее обобщенный вид демонстрируется случаем $n = 10$ и $t = (1100010111)_2$: $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge x_5) \vee (x_1 \wedge x_2 \wedge x_6 \wedge x_7) \vee (x_1 \wedge x_2 \wedge x_6 \wedge x_8 \wedge x_9 \wedge x_{10})$. (По одному члену для каждого нуля в t, и еще один дополнительно.) Кратчайшая CNF дуальна кратчайшей DNF дуальной функции, которая соответствует $2^n - t = (0011101001)_2$: $(x_1) \wedge (x_2) \wedge (x_3 \vee x_4 \vee x_5 \vee x_6) \wedge (x_3 \vee x_4 \vee x_5 \vee x_7 \vee x_8) \wedge (x_3 \vee x_4 \vee x_5 \vee x_7 \vee x_9) \wedge (x_3 \vee x_4 \vee x_5 \vee x_7 \vee x_{10})$.

171. Заметим, что классы бесповторных, регулярных, худых и монотонных функций являются замкнутыми по отношению к операции дуальности и ограничениям. Худая функция очевидным образом является бесповторной; монотонная пороговая функция с $w_1 \geq \cdots \geq w_n$ является регулярной; а регулярная функция является монотонной. Мы должны показать, что регулярная бесповторная функция является худой.

Предположим, что $f(x_1, \ldots, x_n) = g(x_{i_1}, \ldots, x_{i_k}) \circ h(x_{j_1}, \ldots, x_{j_l})$, где $\circ$ представляет собой нетривиальный бинарный оператор, и мы имеем $i_1 < \cdots < i_k$, $j_1 < \cdots < j_l$, $k + l = n$ и $\{i_1, \ldots, i_k, j_1, \ldots, j_l\} = \{1, \ldots, n\}$. (Это условие слабее условия "бесповторности".) Мы можем считать, что $i_1 = 1$. Используя ограничения и индукцию, получаем, что и g, и h являются худыми и монотонными; таким образом, их простые импликанты имеют специальный вид из упр. 170, (д). Оператор $\circ$ должен быть монотонным, так что это либо $\vee$, либо $\wedge$. В связи с дуальностью можно считать, что $\circ$ представляет собой $\vee$.

Случай 1. f имеет простую импликанту длиной 1. Тогда x_1 представляет собой простую импликанту f в силу регулярности. Следовательно, $f(x_1, \ldots, x_n) = x_1 \vee f(0, x_2, \ldots, x_n)$, и мы можем использовать индукцию.

Случай 2. Все простые импликанты g и h имеют длину > 1. Тогда $x_{j_1} \wedge \cdots \wedge x_{j_p}$ является простой импликантой для некоторого $p \geq 2$, но $x_{j_1 - 1} \wedge x_{j_2} \wedge \cdots \wedge x_{j_p}$ таковой не является, что противоречит регулярности. [См. T. Eiter, T. Ibaraki and K. Makino, *Theor. Comp. Sci.* **270** (2002), 493–524.]

172. Рассматривая CNF для f_t в упр. 170, (д), мы видим, что когда $t = (t_1 \ldots t_n)_2$, количество функций Хорна, получаемое путем дополнения переменных, на одну больше количества для $(t_2 \ldots t_n)_2$ при $t_1 = 0$, но в два раза превышает количество при $t_1 = 1$. Таким образом, пример $t = (1100010111)_2$ соответствует $2 \times (2 \times (1 + (1 + (1 + (2 \times (1 + (2 \times (2 \times 2))))))))$ функциям Хорна. Суммирование по всем t дает s_n, где $s_n = (2^{n-2} + s_{n-1}) + 2s_{n-1}$, и $s_1 = 2$; а решением этого рекуррентного соотношения является $3^n - 2^{n-1}$.

Чтобы сделать и f, и $\bar{f}$ функциями Хорна, будем считать (согласно дуальности), что $t \bmod 4 = 3$. Тогда мы должны дополнить x_j тогда и только тогда, когда $t_j = 0$, за исключением строки единиц справа от t. Например, когда $t = (1100010111)_2$, мы должны дополнить x_3, x_4, x_5, x_7, а затем не более одной переменной из $\{x_8, x_9, x_{10}\}$. Это дает $\rho(t + 1) + 1 \geq 3$ вариантов выбора, связанных с f_t. Суммирование по всем t с $t \bmod 4 = 3$ дает $2^n - 1$; так что окончательный ответ равен $2^{n+1} - 2$.

173. Рассмотрим сначала монотонные функции. Можно записать $t = (0^{a_1} 1^{a_2} \ldots 0^{a_{2k-1}} 1^{a_{2k}})_2$, где $a_1 + \cdots + a_{2k} = n$, $a_1 \geq 0$, $a_j \geq 1$ для $1 < j < 2k$ и $a_{2k} \geq 2$ при $t \bmod 4 = 3$. Когда $t \bmod 4 = 1$, $2^n - t$ имеет указанный вид. Тогда f_t имеет $a_1! \, a_2! \ldots a_{2k}!$ автоморфизмов, так что она эквивалентна $n!/(a_1! \, a_2! \ldots a_{2k}!) - 1$ другим, ни одна из которых не является худой. Суммирование по всем t дает $2(P_n - nP_{n-1})$ монотонных булевых функций, которые могут быть переведены в "худую" форму путем переупорядочения переменных при $n \geq 2$, где P_n равно количеству слабых упорядочений (упр. 5.3.1–3). [См. J. S. Beissinger and U. N. Peled, *Graphs and Combinatorics* **3** (1987), 213–219.]

Каждая такая монотонная функция соответствует 2^n различным унатным функциям, которые остаются равно худыми при дополнении переменных. (Это функции, обладающие тем свойством, что все их ограничения являются канализирующими, известны также как "унатные каскады" или "обобщенные пороговые бесповторные функции".)

174. (а) Назначим узлам $\textcircled{1}$, ..., $\textcircled{n}$, $\boxed{\top}$, $\boxed{\bot}$ номера 0, ..., $n{-}1$, n, $n{+}1$; и пусть из узла k к узлам (a_{2k+1}, a_{2k+2}) для $0 \le k < n$ ведут ветви (LO, HI). Затем определим p_k для $1 \le k \le 2n$ следующим образом: пусть $l = \lfloor (k-1)/2 \rfloor$ и $P_l = \{p_1, \ldots, p_{2l}\}$. Установим $p_k \leftarrow a_k$, если $a_k \notin P_l$; в противном случае если a_k представляет собой m-й наименьший элемент $P_l \cap \{l+1, \ldots, n+1\}$, установим p_k равным m-му наименьшему элементу $\{n+2, \ldots, n+l+1\} \setminus P_l$. (Это построение принадлежит Т. Далхаймеру (T. Dahlheimer).)

(б) Инверсия $p_1^{-1} \ldots p_{2n}^{-1}$ перестановки Деллака удовлетворяет условию $2(k-n)-1 \le p_k^{-1} \le 2k$. Она соответствует разупорядочению Дженоччи $q_1 \ldots q_{2n+2}$ при $q_2 = 1$, $q_{2n+1} = 2n+2$ и $q_{2k+2} = 1 + p_k^{-1}$, $q_{2k-1} = 1 + p_{k+n}^{-1}$ для $1 \le k \le n$.

(в) Пусть для заданной перестановки $q_1 \ldots q_{2n+2}$ первым элементом последовательности q_k^{-1}, $q_{q_k^{-1}}^{-1}$, ..., который $\ge k$, является r_k. Это преобразование отображает перестановки Дженоччи в пистолеты Дюмона и обладает тем свойством, что $q_k = k$ тогда и только тогда, когда $r_k = k \notin \{r_1, \ldots, r_{k-1}\}$.

(г) Каждый узел (j,k) представляет множество строк $r_1 \ldots r_j$, где $(1,0) = \{1\}$, а другие множества определяются следующими правилами перехода: предположим, что $r_1 \ldots r_j \in (j,k)$, и пусть $l = 2k$. Если $k = 0$, то $(j+1,k)$ содержит $1r_1^+ \ldots r_j^+$ при четном j и $2r_1^+ \ldots r_j^+$ при нечетном j, где r^+ означает $r+1$. Если $k > 0$, то $(j+1,k)$ содержит $r_1^+ \ldots r_l^+ (l{+}1) r_{l+1}^+ \ldots r_j^+$ при четном j и $r_1^\pm \ldots r_{l-1}^\pm (l) r_l^\pm \ldots r_j^\pm$ при нечетном j, где $r^\pm$ означает $r+1$ при $r \ge l$ и $r-1$ при $r < l$. Действуя по вертикали, если $l \le j-3$ и j нечетно, $(j,k+1)$ содержит $r_1 \ldots r_l r_{l+2} r_{l+3} (l{+}3) r_{l+4} \ldots r_j$. С другой стороны, если $k = 1$ и j четно, $(j,0)$ содержит $r_2 r_1 r_3 \ldots r_j$. Наконец, если $k > 1$ и j четно, $(j,k-1)$ содержит строку $r'_1 \ldots r'_{l-3} (l{-}2) r'_{l-2} r'_{l-1} r'_{l+1} \ldots r'_j$, где r' обозначает l при $r = l-2$, в противном случае $r' = r$. (Можно показать, что элементами $(2j,k)$ являются пистолеты Дюмона для перестановок Дженоччи порядка $2j$, наибольшей фиксированной точкой которых является $2k$.)

Все эти построения обратимы. Например, путь (1,0) → (2,0) → (3,0) → (3,1) → (4,1) → (5,1) → (6,1) → (7,1) → (7,2) → (7,3) → (8,3) → (8,2) → (8,1) → (8,0) соответствует пистолетам 1 → 22 → 133 → 333 → 4244 → 53355 → 624466 → 7335577 → 7355577 → 7355777 → 82448688 → 82646888 → 82466888 → 28466888. Последний пистолет, который может быть представлен диаграммой , соответствует разупорядочению Дженоччи $q_1 \ldots q_8 = 61537482$. А это разупорядочение соответствует $p_1^{-1} \ldots p_6^{-1} = 231546$ и перестановке Деллака $p_1 \ldots p_6 = 312546$. Эта перестановка, в свою очередь, соответствует $a_1 \ldots a_6 = 312343$, что означает тонкую БДР

.

Пусть d_{jk} — количество пистолетов в (j,k), которое также равно количеству ориентированных путей от $(1,0)$ до (j,k). Эти числа легко найти сложением, начиная с

												38227	38227	···	
									2073	2073	38227	76454	···		
								155	155	2073	4146	36154	112608	···	
						17	17	155	310	1918	6064	32008	144616	···	
				3	3	17	34	138	448	1608	7672	25944	170560	···	
		1	1	3	6	14	48	104	552	1160	8832	18272	188832	···	
1	1	1	2	2	8	8	56	56	608	608	9440	9440	198272	···	;

и суммы по столбцам $D_j = \sum_k d_{jk}$ равны $(D_1, D_2, \ldots) = (1, 1, 2, 3, 8, 17, 56, 155, 608, 2073, 9440, 38227, 198272, 929569, \ldots)$. Элементы этой последовательности с четными номерами, D_{2n}, известны как числа Дженоччи G_{2n+2}. Поэтому элементы с нечетными номерами, D_{2n+1}, названы "медианными числами Дженоччи". Количество S_n тонких БДР равно $d_{(2n+2)0} = D_{2n+1}$.

Ссылки. Л. Эйлер (L. Euler) рассматривал числа Дженоччи в томе 2 своего труда *Institutiones Calculi Differentialis* (1755), в главе 7, где он показал, что нечетные целые числа G_{2n} можно выразить через числа Бернулли: $G_{2n} = (2^{2n+1} - 2)|B_{2n}|$ и $z \tan \frac{z}{2} = \sum_{n=1}^{\infty} G_{2n} z^{2n}/(2n)!$. А. Дженоччи (A. Genocchi) исследовал эти числа в *Annali di Scienze Matematiche e Fisiche* **3** (1852), 395–405; а Л. Зейдель (L. Seidel) в *Sitzungsberichte math.-phys. Classe, Akademie Wissen. München* **7** (1877), 157–187, открыл, что они могут быть вычислены аддитивным путем из чисел d_{jk}. Их комбинаторная важность была открыта гораздо позже; см. D. Dumont, *Duke Math. J.* **41** (1974), 305–318; D. Dumont and A. Randrianarivony, *Discrete Math.* **132** (1994), 37–49. Тем временем И. Деллак (H. Dellac) предложил кажущуюся никак не связанной с указанными числами задачу, эквивалентную подсчету того, что мы именуем перестановками Деллака; см. L'*Intermédiaire des Math.* **7** (1900), 9–10, 328; *Annales de la Faculté sci. Marseille* **11** (1901), 141–164.

Существует также *непосредственная* связь между тонкими БДР и путями из п. (г), открытая в 2007 году Торстеном Далхаймером (Thorsten Dahlheimer). Заметим сначала, что неограниченные пистолеты Дюмона порядка $2n+2$ соответствуют тонким БДР, являющимся упорядоченными, но не обязательно приведенными, поскольку можно положить $r_1 \ldots r_{2n} r_{2n+1} r_{2n+2} = (2a_1) \ldots (2a_{2n})(2n{+}2)(2n{+}2)$. Количество таких пистолетов, в которых $\min\{i \mid r_{2i-1} = r_{2i}\} = l$, оказывается равным $d_{(2n+2)(n+1-l)}$.

Чтобы это доказать, можно использовать новые правила переходов вместо таковых из ответа к п. (г): предположим, что $r_1 \ldots r_j \in (j, k)$, и пусть $l = j - 2k$. Тогда $(j + 1, k)$ содержит $r_1^+ \ldots r_l^+ r_l^+ \ldots r_j^+$, когда j нечетно, и $r_1^\pm \ldots r_{l-1}^\pm (l{-}1) r_l^\pm \ldots r_j^\pm$, когда j четно. Если j нечетно, $(j, k+1)$ содержит $1 r_1 r_3 \ldots r_j$ при $l = 3$, а при $l > 3$ оно содержит $r_1' \ldots r_{l-4}' (l{-}4) r_{l-3}' r_{l-2}' r_l' \ldots r_j'$, где $r' = r + 2[r = l{-}4]$. Наконец, если j четно и $k > 0$, $(j, k-1)$ содержит $r_1 \ldots r_{l-1} q r_{l+2} r_{l+2} \ldots r_j$, где $q = l$, если $r_l = r_{l+1}$, в противном случае $q = r_{l+1}$.

При таких магических переходах приведенный выше путь соответствует $1 \to 22 \to 313 \to 133 \to 2244 \to 31355 \to 424466 \to 5153577 \to 5135577 \to 1535577 \to 22646688 \to 26446688 \to 26466688 \to 26466888$; так что $a_1 \ldots a_6 = 132334$.

175. Эта задача кажется требующей подхода, отличного от методов, работающих при $b_0 = \cdots = b_{n-1} = 1$. Предположим, что у нас имеется база БДР из N узлов, включая два стока — $\boxed{\bot}$ и $\boxed{\top}$, вместе с разными ветвлениями, помеченными как $\textcircled{2}, \ldots, \textcircled{n}$, и будем считать, что ровно s из этих узлов являются источниками (имеющими нулевую входящую степень). Пусть $c(b, s, t, N)$ — количество способов добавления b дополнительных узлов, помеченных как $\textcircled{1}$ таким образом, что источниками остаются ровно $s+b-t$ узлов. (Таким образом, $0 \le t \le 2b$; ровно t из старых узлов-источников теперь достижимы из ветвления $\textcircled{1}$.) Тогда количество неконстантных булевых функций $f(x_1, \ldots, x_n)$, имеющих профиль БДР $(b_0, \ldots, b_n)$, равно $T(b_0, \ldots, b_{n-1}; 1)$, где

$$T(b_0; s) = 2[s = b_0 = 1] + [s = 2][b_0 = 0] + [s = 2][b_0 = 2];$$

$$T(b_0, \ldots, b_{n-1}; s) = \sum_{t=\max(0, b_0 - s)}^{2b_0} c(b_0, s{+}t{-}b_0, t, b_1{+}\cdots{+}b_{n-1}{+}2)\, T(b_1, \ldots, b_{n-1}; s{+}t{-}b_0).$$

Можно показать, что $c(b,s,t,N) = \sum_{r=0}^{2b} a_{rb}p_{tr}(s,N)/b!$, где мы имеем $(N(N-1))^{\underline{b}} = \sum_{r=0}^{2b} a_{rb}N^r$ и $p_{tr}(s,N) = \sum_k \binom{r}{k}\left\{{k \atop t}\right\}s^{\underline{t}}(N-s)^{r-k} = \sum_k \left\{{r \atop k}\right\}\binom{k}{t}s^{\underline{t}}(N-s)^{\underline{k-t}} = r!\,[w^t z^r] \times e^{(N-s)z}(we^z - w + 1)^s$.

176. (а) Если $p \neq p'$, мы имеем $\sum_{a\in A, b\in B}[h_{a,b}(p) = h_{a,b}(p')] \le |A||B|/2^l$ согласно определению универсального хеширования. Пусть $r_i(a,b)$ является количеством $p \in P$, таких, что $h_{a,b}(p) = i$. Тогда

$$\sum_{a\in A, b\in B}\ \sum_{0\le i<2^l} r_i(a,b)^2 = \sum_{a\in A, b\in B}\ \sum_{p\in P}\ \sum_{p'\in P}[h_{a,b}(p) = h_{a,b}(p')]$$
$$\le |P||A||B| + \sum_{p\in P}\ \sum_{p'\in P}[p\neq p']\frac{|A||B|}{2^l} = 2^t|A||B|\Bigl(1 + \frac{2^t-1}{2^l}\Bigr).$$

С другой стороны, $\sum_{i=0}^{2^l-1} r_i(a,b)^2 = \sum_{i=0}^{2^l-1}(r_i(a,b) - 2^t/|I|)^2 + 2^{2t}/|I| \ge 2^{2t}/|I|$ для любых a и b. Подобные формулы применимы, когда имеется $s_j(a,b)$ решений уравнения $h_{a,b}(q) = j$. Так что должны быть $a \in A$ и $b \in B$, такие, что

$$\frac{2^{2t}}{|I|} + \frac{2^{2t}}{|J|} \;\le\; \sum_{i\in I} r_i(a,b)^2 + \sum_{j\in J} s_j(a,b)^2 \;\le\; 2^{t+1}\Bigl(1 + \frac{2^t-1}{2^l}\Bigr) \;\le\; \frac{2^{2t}}{2^l} + \frac{2^{2t}}{(1-\epsilon)2^t}.$$

(б) Средние l бит чисел $aq_k + b$ и $aq_{k+2} + b$ отличаются как минимум на 2, так что средние $l-1$ бит чисел aq_k и aq_{k+2} должны быть различными.

(в) Пусть q и q' являются различными элементами Q^*, такими, что $(g(q') - g(q)) \bmod 2^{l-1} \ge 2^{l-2}$. (В противном случае можно обменять $q \leftrightarrow q'$.) Если $l \ge 3$, из условия $g(p) + g(q) = 2^{l-1}$ вытекает, что $f_q(p) = 0$. Теперь мы имеем $(g(p) + g(q')) \bmod 2^{l-1} = (g(q') - g(q)) \bmod 2^{l-1}$; кроме того, и $g(q')$, и $g(p)$ четны. Следовательно, перенос не может распространиться так, чтобы изменить средний бит, и мы имеем $f_{q'}(p) = 1$.

(г) Множество Q'' имеет как минимум $(1-\epsilon)2^{l-1}$ элементов, и то же относится к аналогичному множеству P''. Не более 2^{l-2} элементов Q'' имеют нечетное $g(q)$; и не более $2^{l-1} + 1 - |P''|$ элементов с четным $g(q)$ не находятся в Q^*. Таким образом, $|Q^*| \ge (1-\epsilon)2^{l-1} - 2^{l-2} - 2^{l-1} - 1 + (1-\epsilon)2^{l-1} = (1-4\epsilon)2^{l-2} - 1$, и мы имеем $B_{\min}(Z_{n,a}) \ge (1-4\epsilon)2^{l-1} - 2$ согласно (85).

Наконец выберем $l = t-4$ и $\epsilon = 1/9$. При $n < 14$ теорема очевидна.

177. Предположим, что $k \ge n/2$ и $x = 2^{k+1}x_h + x_l$, $y = 2^k y_h + y_l$. Тогда $(xy \gg k) \bmod 2^{n-k}$ зависит от $2x_hy_l$, x_ly_h и $x_ly_l \gg k$ по модулю 2^{n-k}, так что $q_{2k+1} \le 2^{n-k-1+n-k+n-k}$.

Суммируя, мы получаем $\sum_{k=0}^{2n} q_k \le \sum_{0\le k\le 6n/5} 2^k + \sum_{6n/5<k\le 2n} 2^{3n-2\lfloor k/2\rfloor - \lceil k/2\rceil}$.Если $n = 5t + (0,1,2,3,4)$, сумма равна в точности $(2^{\lceil 6n/5\rceil}\cdot(19,10,12,13,17) - 12)/7$. [М. Зауэрхоф (M. Sauerhoff) в *Discrete Applied Math.* **158** (2010), 1195–1204, доказал для этого упорядочения нижнюю границу $\Omega(2^{6n/5})$.]

178. Можно записать $x = 2^k x_h + x_l$, как в доказательстве теоремы A; но сейчас $x_l = \hat{x}_l + (x \bmod 2)$, где $\hat{x}_l$ четно, а $x \bmod 2$ пока что неизвестно. Аналогично $y = 2^k y_h + y_l = 2^k y_h + \hat{y}_l + (y \bmod 2)$. Пусть $\hat{z}_l = \hat{x}_l\hat{y}_l \bmod 2^k$. На уровне $2k-2$ для $n/2 \le k < n$ нам нужно "запомнить" только три $(n-k)$-битовых числа, $\hat{x}_l \bmod 2^{n-k}$, $\hat{y}_l \bmod 2^{n-k}$, $(\hat{x}_l\hat{y}_l \gg k) \bmod 2^{n-k}$, и три "переноса", $c_1 = (\hat{x}_l + \hat{z}_l) \gg k$, $c_2 = (\hat{y}_l + \hat{z}_l) \gg k$, $c_3 = (\hat{x}_l + \hat{y}_l + \hat{z}_l) \gg k$. Эти шесть величин дают нам средний бит, поскольку известны x_h, y_h, $x \bmod 2$ и $y \bmod 2$.

Для переносов имеется только шесть возможных вариантов: $c_1c_2c_3 = 000$, 001, 011, 101, 111 и 112. Таким образом, $q_{2k-2} \le 6\cdot 2^{(n-k-1)+(n-k-1)+(n-k)}$. Аналогично, когда $n/2 \le k < n-1$, мы имеем $q_{2k-1} \le 6\cdot 2^{(n-k-2)+(n-k-1)+(n-k)}$. Эти оценки вместе с $q_k \le 2^k$ дают нам $\sum_{k=0}^{2n-4} q_k \le (2^{6t}\cdot(37,86,184,464,1024) - 268)/28$, когда $n = 5t + (0,1,2,3,4)$.

Фактические размеры БДР для функции f из теоремы А и функции g из данного упражнения равны $B(f) = (169, 381, 928, 2188, 5248, 12373, 29400, 68777, 162768, 377359, 879709)$ и $B(g) = (165, 352, 806, 1802, 4195, 9774, 22454, 52714, 121198, 278223, 650188)$ для $6 \le n \le 16$; так что этот вариант дает около 25% экономии. Немного лучшее упорядочение получается путем тестирования (младший-бит(x), старший-бит(y), старший-бит(x), младший-бит(y)) на четырех последних уровнях, дающего $B(h) = B(g) - 20$ для $n \ge 6$. Тогда $B(h)/B_{\min}(f) \approx (1.07, 1.05, 1.04, 1.04, 1.04, 1.01, 1.02)$ для $6 \le n \le 12$, так что это упорядочение может оказаться ближе к оптимальному при $n \to \infty$.

180. Полагая $a_{m+1} = a_{m+2} = \cdots = 0$, мы можем считать, что $m \ge p$. Пусть $a = (a_p \ldots a_1)_2$, и запишем $x = 2^k x_h + x_l$, как в доказательстве теоремы А. Если $p \le n$, мы имеем $q_k \le 2^{p-k}$ для $0 \le k < p$, поскольку заданная функция $f = Z^{(p)}_{m,n}(a; x)$ зависит только от a, x_h и $(ax_l \gg k) \bmod 2^{p-k}$. Таким образом, можно считать, что $p > n$.

Рассмотрим мультимножество $A = \{2^k x_h a \bmod 2^{p-1} \mid 0 \le x_h < 2^{n-k}\}$. Запишем $A = \{2^{p-1} - \alpha_1, \ldots, 2^{p-1} - \alpha_s\}$, где $s = 2^{n-k}$ и $0 < \alpha_1 \le \cdots \le \alpha_s = 2^{p-1}$, и пусть $\alpha_{s+i} = \alpha_i + 2^{p-1}$ для $0 \le i \le s$. Тогда $q_k \le 2s$, поскольку f зависит только от a, x_h и индекса $i \in [0\,..\,2s)$, такого, что $\alpha_i \le ax_l \bmod 2^p < \alpha_{i+1}$.

Следовательно, $\sum_{k=0}^{n} q_k \le \sum_{k=0}^{n} \min(2^k, 2^{n+1-k}) = 2^{\lfloor n/2 \rfloor + 1} + 2^{\lceil n/2 \rceil + 1} - 3$.

181. Согласно упр. 170 для каждого $(x_1, \ldots, x_m)$ требуется только $O(n)$ дополнительных узлов.

182. Да; Б. Боллиг (B. Bollig) [*Lecture Notes in Comp. Sci.* **4978** (2008), 306–317] показала, что этот рост равен $\Omega(2^{n/432})$. Кстати, $B_{\min}(L_{12,12}) = 1158$ получается при странном упорядочении $L_{12,12}(x_{18}, x_{17}, x_{16}, x_{15}, x_{14}, x_{12}, x_{10}, x_8, x_6, x_4, x_2, x_1; x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{13}, x_{11}, x_9, x_7, x_5, x_3, x_{24})$; а $B_{\max}(L_{12,12}) = 9302$ получается при $L_{12,12}(x_{24}, x_{23}, x_{20}, x_{19}, x_{22}, x_{11}, x_6, x_7, x_8, x_9, x_{10}, x_{13}; x_1, x_2, x_3, x_4, x_5, x_{21}, x_{18}, x_{17}, x_{16}, x_{15}, x_{14}, x_{12})$. Аналогично $B_{\min}(L_{8,16}) = 606$ и $B_{\max}(L_{8,16}) = 3415$ не так уж страшно далеки друг от друга. Могут ли как $B_{\min}(L_{m,n})$, так и $B_{\max}(L_{m,n})$ предположительно представлять собой $\Theta(2^{\min(m,n)})$?

183. Профиль $(b_0, b_1, \ldots)$ начинается с (1, 1, 1, 2, 3, 5, 7, 11, 15, 23, 31, 47, 63, 95, ...). При $k > 0$ для каждой пары целых чисел (a, b) существует узел на уровне $2k$, такой, что $2^{k-1} \le a, b < 2^k$ и $ab < 2^{2k-1} < (a+1)(b+1)$; этот узел представляет функцию $\big[((a+x)/2^k)((b+y)/2^k) \ge \frac{1}{2}\big]$. Когда b задано, в соответствующем диапазоне имеется $\lceil 2^{2k-1}/b \rceil - \lfloor 2^{2k-1}/(b+1) \rfloor$ вариантов выбора для a; следовательно, $b_{2k} = \sum_{2^{k-1} \le b < 2^k} \big(\lceil 2^{2k-1}/b \rceil - \lfloor 2^{2k-1}/(b+1) \rfloor\big)$, которая телескопируется к $2^k - 1$. Аналогичные рассуждения показывают, что $b_{2k+1} = 2^k + 2^{k-1} - 1$.

184. В $b_{m(i-1)+j-1}$ вносят вклад два вида бусин: один для каждого выбора из i столбцов, как минимум один из которых $< j$; и один для каждого выбора из $i-1$ столбцов, в котором отсутствует как минимум один элемент $\ge j$. Таким образом, $b_{m(i-1)+j-1} = \big(\binom{m}{i} - \binom{m+1-j}{i}\big) + \big(\binom{m}{i-1} - \binom{j-1}{m+1-i}\big)$. Суммирование по $1 \le i, j \le m$ дает $B(P_m) = (2m-3)2^m + 5$. (Кстати, $q_k = b_k + 1$ для $2 \le k < m^2$.)

НДР имеет просто $z_{m(i-1)+j-1} = \binom{m-1}{i-1}$ для $1 \le i, j \le m$, по одному для каждого выбора из $i-1$ столбцов $\ne j$; следовательно, $Z(P_m) = m2^{m-1} + 2 \approx \frac{1}{4}B(P_m)$. (Нижняя граница теоремы К применима также и к узлам НДР, поскольку только такие узлы получают этикетки; следовательно, естественное упорядочение переменных оптимально для НДР. Естественное упорядочение может быть оптимально и для БДР; известно, что эта гипотеза справедлива для $m \le 5$.)

185. Предположим, что $f(x) = t_{\nu x}$ для некоторого бинарного вектора $t_0 \ldots t_n$. Тогда подфункции порядка $d > 0$ соответствуют различным подстрокам $t_i \ldots t_{i+d}$. Такие подстроки τ соответствуют бусинам тогда и только тогда, когда $\tau \neq 0^{d+1}$ и $\tau \neq 1^{d+1}$; z-бусинам они соответствуют тогда и только тогда, когда $\tau \neq 0^{d+1}$ и $\tau \neq 10^d$.

Таким образом, максимум $Z(f)$ представляет собой функцию S_n из ответа к упр. 44. Чтобы достичь этого наихудшего случая, нам нужен бинарный вектор длиной $2^{d+1} + d - 2$, который содержит все $(d+1)$-кортежи, за исключением 0^{d+1} и 10^d, в качестве подстрок; такие векторы можно охарактеризовать как первые $2^{d+1} + d - 2$ элементов любого цикла де Брейна с периодом 2^{d+1}, начинающегося с $0^d 1$.

186. $\bar{x}_1 \wedge \bar{x}_2 \wedge x_3 \wedge \bar{x}_4 \wedge \bar{x}_5 \wedge \bar{x}_6$.

187. (Эти диаграммы следует сравнить с ответом к упр. 1.)

188. Чтобы избежать вложенных скобок, пусть ϵ, a, b и ab обозначают подмножества $\emptyset$, $\{1\}$, $\{2\}$ и $\{1,2\}$. Таким образом, семействами являются $\emptyset$, $\{ab\}$, $\{a\}$, $\{a, ab\}$, $\{b\}$, $\{b, ab\}$, $\{a, b\}$, $\{a, b, ab\}$, $\{\epsilon\}$, $\{\epsilon, ab\}$, $\{\epsilon, a\}$, $\{\epsilon, a, ab\}$, $\{\epsilon, b\}$, $\{\epsilon, b, ab\}$, $\{\epsilon, a, b\}$, $\{\epsilon, a, b, ab\}$ в порядке таблицы истинности.

189. При $n = 0$ это только константные функции; при $n > 0$ это только функции 0 и $x_1 \wedge \cdots \wedge x_n$. (Однако имеется много функций, таких как $x_2 \wedge (x_1 \vee \bar{x}_3)$, обладающих тем свойством, что $(b_0, \ldots, b_n) = (z_0, \ldots, z_n)$.)

190. (а) Только $x_1 \oplus \cdots \oplus x_n$ и $1 \oplus x_1 \oplus \cdots \oplus x_n$ для $n \geq 0$. (б) Это условие выполняется тогда и только тогда, когда все подтаблицы порядка 1 представляют собой либо 01, либо 11. Так что при $n > 0$ имеется $2^{2^{n-1}}$ решений, а именно все функции, такие, что $f(x_1, \ldots, x_{n-1}, 1) = 1$.

191. Язык L_n таблиц истинности для всех таких функций имеет контекстно-свободную грамматику $L_0 \to 1$; $L_{n+1} \to L_n L_n \mid L_n 0^{2^n}$. Таким образом, искомое количество $l_n = |L_n|$ удовлетворяет условиям $l_0 = 1$, $l_{n+1} = l_n(l_n + 1)$; так что $(l_0, l_1, l_2, \ldots)$ представляет собой последовательность (1, 2, 6, 42, 1806, 3263442, 10650056950806, ...). Асимптотически $l_n = \theta^{2^n} - \frac{1}{2} - \epsilon$, где $0 < \epsilon < \theta^{-2^n}/8$ и

$$\theta = 1.59791\,02180\,31873\,17833\,80701\,18157\,45531\,23622+.$$

[См. *CMath*, упр. 4.37 и 4.59, где $l_n + 1$ названо e_{n+1} ("число Евклида"), а θ названо E^2. Числа $l_n + 1$ введены Д. Д. Сильвестером (J. J. Sylvester) в связи с его изучением египетских дробей, *Amer. J. Math.* **3** (1880), 388. Заметим, что монотонно убывающая функция, подобно функции, представляющей независимые множества, всегда имеет $z_n = 1$.]

192. (а) 10101101000010110.

(б) Истинно по индукции по $|\tau|$, поскольку $\alpha \neq \beta \neq 0^n$ тогда и только тогда, когда $\alpha^Z \neq \beta^Z \neq 0^n$.

(в) Бусинами f порядка k являются z-бусины f^Z порядка k для $0 < k \leq n$. Следовательно, бусины f^Z являются также z-бусинами $(f^Z)^Z = f$. Следовательно, если $(b_0, \ldots, b_n)$ и $(z_0, \ldots, z_n)$ представляют собой профиль и z-профиль f, в то время как $(b'_0, \ldots, b'_n)$ и $(z'_0, \ldots, z'_n)$ являются профилем и z-профилем f^Z, мы имеем $b_k = z'_k$ и $z_k = b'_k$ для $0 \leq k < n$.

(Мы также имеем $z_n = z'_n$, но они оба могут быть равны 1 вместо 2. *Квазипрофили* f и f^Z могут отличаться, но, в связи с нулевыми подтаблицами, только не более чем на 1 на каждом уровне.)

193. $S_{\geq k}(x_1,\ldots,x_n)$ по индукции по n. (Следовательно, также $S^Z_{\geq k}(x_1,\ldots,x_n)=S_k(x_1,\ldots,x_n)$. В упр. 249 имеются аналогичные примеры.)

194. Определим $a_1\ldots a_{2n}$ как в ответе к упр. 174, но с использованием НДР вместо БДР. При этом $(1,\ldots,1)$ является z-профилем тогда и только тогда, когда $(2a_1)\ldots(2a_{2n})$ представляет собой неограниченный пистолет Дюмона порядка $2n$. Так что ответ представляет собой число Дженоччи G_{2n+2}.

195. z-профиль имеет вид $(1,2,4,4,3,2,2)$. Мы получаем оптимальный z-профиль $(1,2,3,2,3,2,2)$ из $M_2(x_4,x_2;x_5,x_6,x_3,x_1)$; а наихудший z-профиль $(1,2,4,8,12,2,2)$ — из $M_2(x_5,x_6;x_1,x_2,x_3,x_4)$, как в (78). (Кстати, алгоритм из упр. 197 можно использовать для того, чтобы показать, что $Z_{\min}(M_4)=116$ получен с поразительно своеобразным упорядочением $M_4(x_8,x_5,x_{17},x_2;x_{20},x_{19},x_{18},x_{16},x_{15},x_{13},x_{14},x_{12},x_{11},x_9,x_{10},x_4,x_7,x_6,x_3,x_1)$!)

196. Например, $M_m(x_1,\ldots,x_m;e_{m+1},\ldots,e_n)$, где $n=m+2^m$, а e_j представляет собой элементарную функцию из упр. 203. Тогда мы имеем $Z(f)=2(n-m)+1$ и $Z(\bar f)=(n-m+7)(n-m)/2-2$.

197. Ключевая идея заключается в изменении смысла полей DEP так, чтобы d_{kp} теперь было $\sum\{2^{t-k-1}\mid N_{kp}$ поддерживает $x_t\}$, где мы говорим, что $g(x_1,\ldots,x_m)$ *поддерживает* x_j, если имеется решение $g(x_1,\ldots,x_m)=1$ с $x_j=1$.

Для реализации этого изменения мы вводим вспомогательный массив $(\zeta_0,\ldots,\zeta_n)$, где $\zeta_k=q$, если N_{kq} обозначает подфункцию 0, и $\zeta_k=-1$, если такой подфункции на уровне k нет. Изначально $\zeta_n\leftarrow 0$, и мы устанавливаем $\zeta_k\leftarrow -1$ в начале шага E1. На шаге E3 операция установки d_{kq} должна стать следующей: "если $d_{(k+1)h}\neq\zeta_{k+1}$, установить $d_{kq}\leftarrow((d_{(k+1)l}\mid d_{(k+1)h})\ll 1)+1$; в противном случае установить $d_{kq}\leftarrow d_{(k+1)l}\ll 1$. Установить также $\zeta_k\leftarrow q$, если $d_{(k+1)l}=d_{(k+1)h}=\zeta_{k+1}$."

(Главная z-профилограмма может использоваться, как ранее, для минимизации $z_0+\cdots+z_{n-1}$; но если важен *абсолютный* минимум, требуется дополнительная работа по рассмотрению z_n.)

198. Реинтерпретируя (50), представим произвольное семейство множеств f в виде $(\bar x_v?\ f_l\colon f_h)$, где $v=f_v$ индексирует первую переменную, которую *поддерживает* f; см. ответ к упр. 197. Таким образом, f_l представляет собой подсемейство f, которое не поддерживает x_v, а f_h представляет собой подсемейство, которое это делает (но с удаленным x_v). Мы также положим $f_v=\infty$, если f не поддерживает ничего (т. е. если f представляет собой либо $\emptyset$, либо $\{\emptyset\}$, внутренне представленное как $\boxed{\bot}$ или $\boxed{\top}$; см. ответ к упр. 200). В (52) $v=\min(f_v,g_v)$ теперь индексирует первую переменную, *поддерживаемую* либо f, либо g; таким образом, $f_h=\emptyset$, если $f_v>g_v$, и $g_h=\emptyset$, если $f_v<g_v$.

Подпрограмма И(f,g), в стиле НДР, в настоящее время вместо (55) имеет следующий вид: "представить f и g, как в (52). Пока $f_v\neq g_v$, вернуть $\emptyset$, если либо $f=\emptyset$, либо $g=\emptyset$; в противном случае установить $f\leftarrow f_l$, если $f_v<g_v$, установить $g\leftarrow g_l$, если $f_v>g_v$. Выполнить обмен $f\leftrightarrow g$, если $f>g$. Вернуть f, если $f=g$ или $f=\emptyset$. В противном случае, если $f\wedge g=r$ находится в кэше записей, вернуть r. В противном случае вычислить $r_l\leftarrow$ И(f_l,g_l) и $r_h\leftarrow$ И(f_h,g_h); установить $r\leftarrow$ ZUNIQUE(v,r_l,r_h) с применением алгоритма наподобие алгоритма U, за исключением того, что первый шаг возвращает p при $q=\emptyset$, а не при $q=p$; поместить '$f\wedge g=r$' в кэш записей и вернуть r." (См. также предложение в ответе к упр. 200.)

Счетчик ссылок обновляется, как и в упр. 82, с небольшими изменениями; например, шаг U1 уменьшает значение счетчика ссылок $\boxed{\bot}$ (и только этого узла), когда $q=\emptyset$. Важно записать подпрограмму контроля корректности, которая дважды проверит все счетчики ссылок и прочую избыточность во всей базе БДР/НДР, так, чтобы в корне пресечь появление труднообнаруживаемых ошибок. Пока все подпрограммы не будут

тщательнейшим образом протестированы, подпрограмма проверки должна вызываться очень часто.

199. (а) Если $f = g$, вернуть f. Если $f > g$, обменять $f \leftrightarrow g$. Если $f = \emptyset$, вернуть g. Если $f \vee g = r$ находится в кэше записей, вернуть r. В противном случае

$$\begin{array}{ll}
\text{установить } v \leftarrow f_v,\ r_l \leftarrow \text{ИЛИ}(f_l, g_l),\ r_h \leftarrow \text{ИЛИ}(f_h, g_h), & \text{если } f_v = g_v; \\
\text{установить } v \leftarrow f_v,\ r_l \leftarrow \text{ИЛИ}(f_l, g),\ r_h \leftarrow f_h, \text{ увеличить } \texttt{REF}(f_h) \text{ на } 1, & \text{если } f_v < g_v; \\
\text{установить } v \leftarrow g_v,\ r_l \leftarrow \text{ИЛИ}(f, g_l),\ r_h \leftarrow g_h, \text{ увеличить } \texttt{REF}(g_h) \text{ на } 1, & \text{если } f_v > g_v.
\end{array}$$

Затем установить $r \leftarrow \text{ZUNIQUE}(v, r_l, r_h)$; кэшировать r и вернуть его, как в ответе к упр. 198.

(б) Если $f = g$, вернуть $\emptyset$. В противном случае действовать, как в п. (а), но использовать ($\oplus$, ЛИБО), а не ($\vee$, ИЛИ).

(в) Если $f = \emptyset$ или $f = g$, вернуть $\emptyset$. Если $g = \emptyset$, вернуть f. В противном случае, если $g_v < f_v$, установить $g \leftarrow g_l$ и начать заново. В противном случае

$$\begin{array}{ll}
\text{установить } r_l \leftarrow \text{НО-НЕ}(f_l, g_l),\ r_h \leftarrow \text{НО-НЕ}(f_h, g_h), & \text{если } f_v = g_v; \\
\text{установить } r_l \leftarrow \text{НО-НЕ}(f_l, g),\ r_h \leftarrow f_h, \text{ увеличить } \texttt{REF}(f_h) \text{ на } 1, & \text{если } f_v < g_v.
\end{array}$$

Затем установить $r \leftarrow \text{ZUNIQUE}(f_v, r_l, r_h)$ и закончить работу, как обычно.

200. Если $f = \emptyset$, вернуть g. Если $f = h$, вернуть ИЛИ(f, g). Если $g = h$, вернуть g. Если $g = \emptyset$ или $f = g$, вернуть И(f, h). Если $h = \emptyset$, вернуть НО-НЕ(g, f). Если $f_v < g_v$ и $f_v < h_v$, установить $f \leftarrow f_l$ и начать все сначала. Если $h_v < f_v$ и $h_v < g_v$, установить $h \leftarrow h_l$ и начать все сначала. В противном случае проверить кэш и работать рекурсивно, как обычно.

201. В приложениях НДР, в которых разрешены проецирующие функции и/или операция дополнения, лучше всего вначале, при общей инициализации, зафиксировать множество булевых переменных. В противном случае *каждая* внешняя функция в базе НДР должна изменяться, когда в игру вступает новая переменная.

Предположим поэтому, что мы решили работать с функциями от $(x_1, \ldots, x_N)$, где N заранее задано. В ответе к упр. 198 при $f = \emptyset$ или $f = \{\emptyset\}$ мы полагали $f_v = N+1$, а не ∞. Тогда функция тавтологии $1 = \wp$ имеет $(N+1)$-узловую НДР $\textcircled{1} \text{—} \textcircled{2} \text{—} \cdots \text{—} \textcircled{N} \text{—} \boxed{\top}$, которую мы строим, как только становится известно N. Пусть t_j представляет собой узел $\textcircled{j}$ этой структуры с $t_{N+1} = \boxed{\top}$. НДР для x_j теперь имеет вид $\textcircled{1} \text{—} \cdots \text{—} \textcircled{j} \text{—} t_{j+1} \boxed{\bot}$; таким образом, база НДР для множества всех x_j будет занимать $\binom{N+1}{2}$ узлов в дополнение к представлениям $\emptyset$ и $\wp$.

Если N мало́, все N проецирующих функций можно приготовить заранее. Но во многих приложениях НДР N велико; а проецирующие функции редко требуются при использовании "алгебры семейств" для построения структур, как в упр. 203–207. Так что в общем случае лучше дождаться, когда действительно требуется проецирующая функция, перед тем как ее создавать.

Кстати, для ускорения операций синтеза в упр. 198 и 199 можно использовать частично тавтологические функции t_j: если $v = f_v \le g_v$ и $f = t_v$, мы имеем И$(f, g) = g$, ИЛИ$(f, g) = f$, а также (если $v \le h_v$) MUX$(f, g, h) = h$, MUX$(g, h, f) =$ ИЛИ(g, h).

202. В шаге T4 замените '$q_0 \leftarrow q_1 \leftarrow q$' на '$q_0 \leftarrow q$, $q_1 \leftarrow \emptyset$' и '$r_0 \leftarrow r_1 \leftarrow r$' на '$r_0 \leftarrow r$, $r_1 \leftarrow \emptyset$'. Кроме того, используйте ZUNIQUE вместо UNIQUE; на шаге T4 эта подпрограмма увеличивает $\texttt{REF}(p)$ на 1, если шаг U1 обнаруживает $q = \emptyset$.

Более тонкое изменение требуется для поддержки частично тавтологических функций из ответа к упр. 201 в актуальном состоянии из-за их специального смысла. Корректное поведение заключается в поддержке t_u неизменным и в установке $t_v \leftarrow \texttt{LO}(t_u)$.

203. (а) $f \sqcup g = \{\{1,2\},\{1,3\},\{1,2,3\},\{3\}\} = (e_1 \sqcup ((e_2 \sqcup (e_3 \cup \epsilon)) \cup e_3)) \cup e_3$; вторым является $(e_1 \sqcup e_2) \cup \epsilon$, поскольку $f \sqcap g = (e_1 \sqcup (e_2 \cup \epsilon)) \cup e_3 \cup \epsilon$ и $f \boxplus e_1 = e_1 \cup e_2 \cup e_3$.

(б) $(f\sqcup g)(z) = \exists x\,\exists y\,(f(x)\wedge g(y)\wedge(z\equiv x\vee y))$; $(f\sqcap g)(z) = \exists x\,\exists y\,(f(x)\wedge g(y)\wedge(z\equiv x\wedge y))$; $(f\boxplus g)(z) = \exists x\,\exists y\,(f(x)\wedge g(y)\wedge(z\equiv x\oplus y))$. Другой формулой является $(f\boxplus g)(z) = \bigvee\{f(z\oplus y)\mid g(y)=1\} = \bigvee\{g(z\oplus x)\mid f(x)=1\}$.

(в) И (i), и (ii) истинны; кроме того, $f\boxplus(g\cup h) = (f\boxplus g)\cup(f\boxplus h)$. Формула (iii) ложна в общем случае, хотя мы имеем $f\sqcup(g\sqcap h)\subseteq(f\sqcup g)\sqcap(f\sqcup h)$. Формула (iv) имеет мало смысла; правая часть представляет собой $(f\sqcup f)\cup(f\sqcup h)\cup(g\sqcup f)\cup(g\sqcup h)$ согласно (i). Формула (v) истинна, поскольку все три части представляют собой $\emptyset$. А (vi) истинна тогда и только тогда, когда $f\neq\emptyset$.

(г) Только (ii) истинно всегда. Для (i) условием должно быть $f\sqcap g\subseteq\epsilon$, поскольку из $f\sqcap g=\emptyset$ вытекает $f\perp g$. В случае (iii) заметим, что $|f\sqcup g| = |f\sqcap g| = |f\boxplus g| = 1$ при $|f| = |g| = 1$. Наконец в утверждении (iv) у нас $f\perp g \implies f\sqcup g = f\boxplus g$; но обратное не выполняется, когда, скажем, $f = g = e_1\cup\epsilon$.

(д) $f=\emptyset$ в (i) и $f=\epsilon$ в (ii); кроме того, $\epsilon\boxplus g = g$ для всех g. Решений для (iii) не существует, поскольку f должно быть $\{\{1,2,3,\ldots\}\}$, а мы рассматриваем только конечные множества. Но в конечном универсуме из ответа к упр. 201 мы имеем $f=\{\{1,\ldots,N\}\}$. (Это семейство U обладает тем свойством, что $(f\boxplus U)\sqcup(g\boxplus U) = (f\sqcap g)\boxplus U$.) Общее решение (iv) имеет вид $f = e_1\sqcup e_2\sqcup f'$, где f' представляет собой произвольное семейство; аналогично общее решение (v) имеет вид $f = (e_1\sqcup f')\cup(e_2\sqcup f'')\cup(e_1\sqcup e_2\sqcup(f'\cup f''\cup f'''))$, где f', f'' и f''' — произвольны. В (vi) $f = ((((e_1\sqcup e_2)\cup\epsilon)\sqcup f')\cup((e_1\cup e_2)\sqcup f''))\sqcup(e_3\cup\epsilon)$, где $f'\cup f''\perp e_1\cup e_2\cup e_3$; это представление следует из упр. 204, (e). В (vii) $|f| = 1$. Наконец (viii) характеризует функции Хорна (теорема 7.1.1H).

204. (а) Это соотношение очевидно из определения. (Кроме того, $(f\cup g)/h\supseteq(f/h)\cup(g/h)$.)

(б) $f/e_2 = \{\{1\},\emptyset\} = e_1\cup\epsilon$; $f/e_1 = e_2\cup e_3$; $f/\epsilon = f$; следовательно, $f/(e_1\cup\epsilon) = e_2\cup e_3$.

(в) Деление на $\emptyset$ приводит к проблемам, поскольку *все* множества α принадлежат $f/\emptyset$. (Но если мы ограничим рассмотрение семействами подмножеств $\{1,\ldots,N\}$, как в упр. 201 и 207, то мы имеем $f/\emptyset = \wp$; кроме того, $\wp/\wp = \epsilon$ и $f/\wp = \emptyset$ при $f\neq\wp$.) Ясно, что $f/\epsilon = f$. А $f/f = \epsilon$, когда $f\neq\emptyset$. Наконец $(f \bmod g)/g = \emptyset$, когда $g\neq\emptyset$, поскольку из $\alpha\in(f\bmod g)/g$ и $\beta\in g$ вытекает, что $\alpha\cup\beta\in f$, $\alpha\in f/g$ и $\alpha\cup\beta\notin(f/g)\sqcup g$, так что мы получаем противоречие.

(г) Если $\beta\in g$, мы имеем $\beta\cup\alpha\in f$ и $\beta\cap\alpha=\emptyset$ для всех $\alpha\in f/g$; этим доказывается указание. Следовательно, $f/g\subseteq f/(f/(f/g))$. Кроме того, $f/h\subseteq f/g$ при $h\supseteq g$ согласно (а); положим $h = f/(f/g)$.

(д) Пусть $f//g$ представляет собой семейство в новом определении. Тогда $f/g\subseteq f//g$, поскольку $g\sqcup(f/g)\subseteq f$ и $g\perp(f/g)$. И обратно, если $\alpha\in f//g$ и $\beta\in g$, мы имеем $\alpha\in h$ для некоторого h с $g\sqcup h\subseteq f$ и $g\perp h$; так что $\alpha\cup\beta\in f$ и $\alpha\cap\beta=\emptyset$.

(е) Если f имеет такое представление, мы должны иметь $g = f/e_j$ и $h = f\bmod e_j$. И обратно, эти семейства удовлетворяют $e_j\perp g\cup h$. (Этот закон представляет собой фундаментальный рекурсивный принцип, лежащий в основе НДР, — так же как единственное представление $f = (x_j?\ g\colon h)$, с g и h, не зависящими от x_j, лежит в основе БДР.)

(ж) Оба истинны. (Для их доказательства представьте f и g, как в п. (е).)

[Р. К. Брейтон (R. K. Brayton) и К. Мак-Мюллен (C. McMullen) ввели частное и остаток в *Proc. Int. Symp. Circuits and Systems* (IEEE, 1982), 49–54, но немного в ином контексте: они работали с семействами несравнимых множеств подкубов.]

205. Во всех случаях мы строим рекурсию, основанную на упр. 204, (е). Например, если $f_v = g_v = v$, мы имеем $f\sqcup g = (\bar v?\ f_l\sqcup g_l\colon (f_l\sqcup g_h)\cup(f_h\sqcup g_l)\cup(f_h\sqcup g_h))$; $f\sqcap g = (\bar v?\ (f_l\sqcap g_l)\cup(f_l\sqcap g_h)\cup(f_h\sqcap g_l)\colon f_h\sqcap g_h)$; $f\boxplus g = (\bar v?\ (f_l\boxplus g_l)\cup(f_h\boxplus g_h)\colon (f_h\boxplus g_l)\cup(f_l\boxplus g_h))$.

(а) Если $f_v < g_v$ или ($f_v = g_v$ и $f > g$), обменять $f\leftrightarrow g$. Если $f=\emptyset$, вернуть f; если $f=\epsilon$, вернуть g. Если $f\sqcup g = r$ находится в кэше записей, вернуть r. Если $f_v > g_v$, установить $r_l\leftarrow\text{JOIN}(f,g_l)$ и $r_h\leftarrow\text{JOIN}(f,g_h)$; в противном случае установить $r_l\leftarrow\text{JOIN}(f_l,g_l)$, $r_{lh}\leftarrow\text{JOIN}(f_l,g_h)$, $r_{hl}\leftarrow\text{JOIN}(f_h,g_l)$, $r_{hh}\leftarrow\text{JOIN}(f_h,g_h)$, $r_h\leftarrow$

$\text{OROR}(r_{lh}, r_{hl}, r_{hh})$ и разыменовать r_{lh}, r_{hl}, r_{hh}. Закончить с $r \leftarrow \text{ZUNIQUE}(g_v, r_l, r_h)$; кэшировать эту запись и вернуть ее, как в упр. 198.

(Можно также вычислить r_h по формуле $\text{ИЛИ}(r_{lh}, \text{JOIN}(f_h, \text{ИЛИ}(g_l, g_h)))$, или по формуле $\text{ИЛИ}(r_{hl}, \text{JOIN}(\text{ИЛИ}(f_l, f_h), g_h))$. Иногда один путь оказывается существенно лучше двух других.)

Операция DISJOIN, которая производит семейство *непересекающихся* объединений $\{\alpha \cup \beta \mid \alpha \in f,\ \beta \in g,\ \alpha \cap \beta = \emptyset\}$, аналогична, но с опущенным r_{hh}.

(б) Если $f_v < g_v$ или ($f_v = g_v$ и $f > g$), обменять $f \leftrightarrow g$. Если $f \le \epsilon$, вернуть f. (Мы считаем $\emptyset < \epsilon$ и $\epsilon <$ всех прочих.) В противном случае, если запись $\text{MEET}(f, g)$ не была кэширована, имеются два случая. Если $f_v > g_v$, установить $r_h \leftarrow \text{ИЛИ}(g_l, g_h)$, $r \leftarrow \text{MEET}(f, r_h)$, и разыменовать r_h; в противном случае действовать, как в п. (а), но с $l \leftrightarrow h$. Кэшировать и вернуть r, как обычно.

(в) Эта операция подобна (а), но $r_l \leftarrow \text{ИЛИ}(r_{ll}, r_{hh})$ и $r_h \leftarrow \text{ИЛИ}(r_{lh}, r_{hl})$.

(г) Сначала мы реализуем важные простые случаи f/e_v и $f \bmod e_v$.

$$\text{EZDIV}(f, v) = \begin{cases} \text{Если } f_v = v, \text{ вернуть } f_h; \text{ если } f_v > v, \text{ вернуть } \emptyset. \text{ В противном} \\ \text{случае искать } f/e_v = r \text{ в кэше; если его там нет, вычислить как} \\ r \leftarrow \text{ZUNIQUE}(f_v, \text{EZDIV}(f_l, v), \text{EZDIV}(f_h, v)). \end{cases}$$

$$\text{EZMOD}(f, v) = \begin{cases} \text{Если } f_v = v, \text{ вернуть } f_l; \text{ если } f_v > v, \text{ вернуть } f. \text{ В противном} \\ \text{случае искать } f \bmod e_v = r \text{ в кэше; если его там нет, вычислить} \\ \text{как } r \leftarrow \text{ZUNIQUE}(f_v, \text{EZMOD}(f_l, v), \text{EZMOD}(f_h, v)). \end{cases}$$

Сейчас $\text{DIV}(f, g) =$ “Если $g = \emptyset$, см. ниже; если $g = \epsilon$, вернуть f. В противном случае, если $f \le \epsilon$, вернуть $\emptyset$; если $f = g$, вернуть ϵ. Если $g_l = \emptyset$ и $g_h = \epsilon$, вернуть $\text{EZDIV}(f, g_v)$. В противном случае, если $f/g = r$ имеется в кэше записей, вернуть r. В противном случае установить $r_l \leftarrow \text{EZDIV}(f, g_v)$, $r \leftarrow \text{DIV}(r_l, g_h)$ и разыменовать r_l. Если $r \ne \emptyset$ и $g_l \ne \emptyset$, установить $r_h \leftarrow \text{EZMOD}(f, g_v)$ и $r_l \leftarrow \text{DIV}(r_h, g_l)$, разыменовать r_h, установить $r_h \leftarrow r$ и $r \leftarrow \text{И}(r_l, r_h)$, разыменовать r_l и r_h. Вставить ‘$f/g = r$’ в кэш записей и вернуть r”. Деление на $\emptyset$ возвращает $\wp$, если существует фиксированный универсум $\{1, \ldots, N\}$, как в упр. 201. В противном случае получается ошибка (поскольку не существует универсального семейства $\wp$).

(д) Если $g = \emptyset$, вернуть f. Если $g = \epsilon$, вернуть $\emptyset$. Если $(g_l, g_h) = (\emptyset, \epsilon)$, вернуть $\text{EZMOD}(f, g_v)$. Если $f \bmod g = r$ кэшировано, вернуть его. В противном случае установить $r \leftarrow \text{DIV}(f, g)$ и $r_h \leftarrow \text{JOIN}(r, g)$, разыменовать r, установить $r \leftarrow \text{HO-HE}(f, r_h)$ и разыменовать r_h. Кэшировать и вернуть r.

[Ш. Минато (S. Minato) привел $\text{EZDIV}(f, v)$, $\text{EZREM}(f, v)$ и $\text{DELTA}(f, e_v)$ в своей оригинальной статье о НДР. Его алгоритмы для $\text{JOIN}(f, g)$ и $\text{DIV}(f, g)$ были приведены в продолжении, *ACM/IEEE Design Automation Conf.* **31** (1994), 420–424.]

206. Нетрудно доказать верхнюю границу $O(Z(f)^3 Z(g)^3)$ для случаев (а) и (б), как и $O(Z(f)^2 Z(g)^2)$ для случая (в). Однако существуют ли примеры, требующие такого большого времени работы? И может ли время работы для (г) быть экспоненциальным? На практике все пять программ выглядят достаточно быстрыми.

207. Если $f = e_{i_1} \cup \cdots \cup e_{i_l}$ и $k \ge 0$, пусть $\text{SYM}(f, v, k)$ представляет собой булеву функцию, истинную тогда и только тогда, когда ровно k из переменных $\{x_{i_1}, \ldots, x_{i_l}\} \cap \{x_v, x_{v+1}, \ldots\}$ равны 1 и $x_1 = \cdots = x_{v-1} = 0$. Мы вычисляем $(e_{i_1} \cup \cdots \cup e_{i_l}) \S k$ путем вызова $\text{SYM}(f, 1, k)$.

$\text{SYM}(f, v, k) =$ “Пока $f_v < v$, устанавливать $f \leftarrow f_l$. Если $f_v = N + 1$ и $k > 0$, вернуть $\emptyset$. Если $f_v = N + 1$ и $k = 0$, вернуть частично тавтологическую функцию t_v (см. ответ к упр. 201). Если $f \S v \S k = r$ имеется в кэше, вернуть r. В противном случае установить $r \leftarrow \text{SYM}(f, f_v + 1, k)$. Если $k > 0$, установить $q \leftarrow \text{SYM}(f_l, f_v + 1, k - 1)$ и $r \leftarrow \text{ZUNIQUE}(f_v, r, q)$. Пока $f_v > v$, устанавливать $f_v \leftarrow f_v - 1$, увеличивать $\texttt{REF}(r)$ на 1

и устанавливать $r \leftarrow \text{ZUNIQUE}(f_v, r, r)$. Поместить '$f \S v \S k = r$' в кэш и вернуть r". Время работы составляет $O((k+1)N)$. Заметим, что $\emptyset \S 0 = \wp$.

208. Необходимо просто опустить множители $2^{v_{s-1}-1}$, $2^{v_l - v_k - 1}$ и $2^{v_h - v_k - 1}$ в шагах C1 и C2. (Мы также получаем производящую функцию путем установки $c_k \leftarrow c_l + zc_h$ на шаге C2; см. упр. 25.) *Количество решений равно числу путей от корня НДР до* $\boxed{\top}$.

209. Сначала вычисляем $\delta_n \leftarrow \bot$ и $\delta_j \leftarrow (\bar{x}_{j+1} \circ x_{j+1}) \bullet \delta_{j+1}$ для $n > j \geq 1$. Затем, там, где в ответе к упр. 31 говорится '$\alpha \leftarrow (\bar{x}_j \circ x_j) \bullet \alpha$', меняем его на '$\alpha \leftarrow (\bar{x}_j \bullet \alpha) \circ (x_j \bullet \delta_j)$'. Делаем также аналогичные изменения с β и γ вместо α.

210. Фактически, когда $x = x_1 \ldots x_n$, можно заменить νx в определении g любой линейной функцией $c(x) = c_1 x_1 + \cdots + c_n x_n$, таким образом характеризуя все оптимальные решения обобщенной задачи булева программирования, рассматриваемой в алгоритме B.

Для каждого узла ветвления x в НДР с полями $\mathtt{V}(x)$, $\mathtt{LO}(x)$ и $\mathtt{HI}(x)$ можно вычислить его оптимальное значение $\mathtt{M}(x)$ и новые связи $\mathtt{L}(x)$, $\mathtt{H}(x)$ следующим образом: пусть $m_l = \mathtt{M}(\mathtt{LO}(x))$ и $m_h = c_{\mathtt{V}(x)} + \mathtt{M}(\mathtt{HI}(x))$, где $\mathtt{M}(\boxed{\bot}) = -\infty$ и $\mathtt{M}(\boxed{\top}) = 0$. Тогда $\mathtt{L}(x) \leftarrow \mathtt{LO}(x)$, если $m_l \geq m_h$, в противном случае $\mathtt{L}(x) \leftarrow \boxed{\bot}$; $\mathtt{H}(x) \leftarrow \mathtt{HI}(x)$, если $m_l \leq m_h$, в противном случае $\mathtt{H}(x) \leftarrow \boxed{\bot}$. НДР для g получается путем приведения связей $\mathtt{L}$ и $\mathtt{H}$, достижимых из корня. Заметим, что $Z(g) \leq Z(f)$, а все вычисления требуют $O(Z(f))$ шагов. (На это красивое свойство НДР указал О. Кудер (O. Coudert); см. ответ к упр. 237.)

211. Да, только если матрица не имеет нулевых строк. Фактически без таких строк профиль и z-профиль f удовлетворяют соотношению $b_k \geq q_k - 1 \geq z_k$ для $0 \leq k < n$, поскольку единственной подфункцией на уровне k, не зависящей от x_{k+1}, является константа 0.

212. Наилучшим вариантом в экспериментах автора было создание НДР для каждого терма $T_j = S_1(X_j)$ в (129) с использованием алгоритма из упр. 207 с последующей сборкой их вместе с помощью оператора И. Например, в задаче (128) мы имеем $X_1 = \{x_1, x_2\}$, $X_2 = \{x_1, x_3, x_4\}$, $\ldots$, $X_{64} = \{x_{105}, x_{112}\}$; чтобы создать терм $S_1(X_2) = S_1(x_1, x_3, x_4)$, НДР которого имеет 115 узлов, просто образуем 5-узловую НДР для $e_1 \cup (e_3 \cup e_4)$ и вычислим $T_2 \leftarrow (e_1 \cup e_3 \cup e_4) \S 1$.

Но в каком порядке следует выполнять операторы И после того как мы получим отдельные термы $T_1, \ldots, T_n$ из (129)? Рассмотрим задачу (128). *Метод 1*: $T_1 \leftarrow T_1 \wedge T_2$, $T_1 \leftarrow T_1 \wedge T_3$, $\ldots$, $T_1 \leftarrow T_1 \wedge T_{64}$. Этот "нисходящий" метод сначала заполняет верхние уровни и требует около 6.2 миллиона обращений к памяти. *Метод 2*: $T_{64} \leftarrow T_{64} \wedge T_{63}$, $T_{64} \leftarrow T_{64} \wedge T_{62}$, $\ldots$, $T_{64} \leftarrow T_{64} \wedge T_1$. С помощью заполнения первыми нижних уровней ("восходящий" метод) время работы сокращается до около 1.75 миллиона обращений к памяти. *Метод 3*: $T_2 \leftarrow T_2 \wedge T_1$, $T_4 \leftarrow T_4 \wedge T_3$, $\ldots$, $T_{64} \leftarrow T_{64} \wedge T_{63}$; $T_4 \leftarrow T_4 \wedge T_2$, $T_8 \leftarrow T_8 \wedge T_6$, $\ldots$, $T_{64} \leftarrow T_{64} \wedge T_{62}$; $T_8 \leftarrow T_8 \wedge T_4$, $T_{16} \leftarrow T_{16} \wedge T_{12}$, $\ldots$, $T_{64} \leftarrow T_{64} \wedge T_{60}$; $\ldots$; $T_{64} \leftarrow T_{64} \wedge T_{32}$. Этот "сбалансированный" подход также требует около 1.75 миллиона обращений к памяти. *Метод 4*: $T_{33} \leftarrow T_{33} \wedge T_1$, $T_{34} \leftarrow T_{34} \wedge T_2$, $\ldots$, $T_{64} \leftarrow T_{64} \wedge T_{32}$; $T_{49} \leftarrow T_{49} \wedge T_{33}$, $T_{50} \leftarrow T_{50} \wedge T_{34}$, $\ldots$, $T_{64} \leftarrow T_{64} \wedge T_{48}$; $T_{57} \leftarrow T_{57} \wedge T_{49}$, $T_{58} \leftarrow T_{58} \wedge T_{50}$, $\ldots$, $T_{64} \leftarrow T_{64} \wedge T_{56}$; $\ldots$; $T_{64} \leftarrow T_{64} \wedge T_{63}$. Это гораздо лучший способ балансировки работы, требующий всего около 850 тысяч обращений к памяти. *Метод 5*: Аналогичная стратегия балансировки, использующая тернарную операцию И-И, оказывается еще лучшей, сто́ящей только 675 тысяч обращений к памяти. (Во всех пяти случаях следует добавить около 190 тысяч обращений к памяти для образования 64 начальных термов T_j.)

Кстати, можно уменьшить размер НДР с 2300 до 1995 с помощью требования $x_1 = 0$ и $x_2 = 1$ в (128) и (129), поскольку "транспонирование" каждого покрытия является другим покрытием. Однако существенно эта идея время работы не снижает.

Строки (128) находятся в убывающем лексикографическом порядке, который может не быть идеальным. Но динамическое упорядочение переменных бесполезно при таком

большом количестве переменных. (Просеивание снижает размер от 2300 до 1887, но требует *длительного* времени работы.)

Очевидно, что желательно продолжить изучение этого вопроса для широкого диапазона задач точного покрытия.

213. Мы получаем двудольный граф с 30 вершинами в одной части и 32 в другой. (Каждая кость домино соединяет белый и черный квадраты, а мы удалили два черных квадрата.) Построчная сумма $(1,\ldots,1,1,*,*)$ имеет единицы как минимум в 31 "белой" позиции, так что последние ее две координаты должны быть либо $(2,1)$, либо $(3,2)$.*

214. Добавим дальнейшие ограничения к условию покрытия (128), а именно $\bigwedge_{j=1}^{14} S_{\geq 1}(Y_j)$, где Y_j представляет собой множество x_i, которые пересекают j-ю потенциальную линию ошибки. (Например, $Y_1 = \{x_2, x_4, x_6, x_8, x_{10}, x_{12}, x_{14}, x_{15}\}$ представляет собой множество способов вертикального размещения костей домино в двух верхних рядах доски; каждое $|Y_j| = 8$.) Получающаяся в результате НДР имеет 9812 узлов и характеризует 25 506 решений. Кстати, размер БДР — 26 622. [Безошибочное замощение костями домино доски размером $m \times n$ существует тогда и только тогда, когда mn четно, $m \geq 5$, $n \geq 5$ и $(m,n) \neq (6,6)$; см. R. L. Graham, *The Mathematical Gardner* (Wadsworth International, 1981), 120–126. Решение, показанное в (127), является единственным примером размером 8×8, симметричным как по горизонтали, так и по вертикали; на рис. 29, (б) можно встретить решение, симметричное относительно поворота на 90°.]

215. В этот раз мы добавляем ограничения $\bigwedge_{j=1}^{49} S_{\geq 1}(Z_j)$, где Z_j представляет собой множество четырех размещений x_i, окружающих точку внутреннего угла. (Например, $Z_1 = \{x_1, x_2, x_4, x_{16}\}$.) Эти ограничения уменьшают размер НДР до 66. Имеется только два таких решения, одно из которых получается транспонированием другого, и их легко найти вручную. [См. Y. Kotani, *Puzzlers' Tribute* (A. K. Peters, 2002), 413–420. Множество всех покрытий татами было охарактеризовано Дином Хикерсоном (Dean Hickerson); соответствующая производящая функция была получена Фрэнком Раски (Frank Ruskey) и Дженнифер Вудкок (Jennifer Woodcock), *Electronic J. Combinatorics* **16**, 1 (2009), #R126.]

216. (а) Назначим каждой строке (128) три переменные, (a_i, b_i, c_i), соответствующие цветам домино, если выбрана строка i. Каждый узел ветвления НДР для f в (129) теперь превращается в три узла ветвления. Мы можем воспользоваться симметрией относительно транспонирования, заменяя f на $f \wedge x_2$; это уменьшает с 2300 до 1995 размер НДР, который увеличивается до 5981 при утроении каждого узла ветвления.

Теперь выполним И в смежных ограничениях для всех 682 случаев $\{i, i'\}$, где строки i и i' представляют собой смежные позиции домино. Такие ограничения имеют вид $\neg\bigl((a_i \wedge a_{i'}) \vee (b_i \wedge b_{i'}) \vee (c_i \wedge c_{i'})\bigr)$, и мы применяем их в восходящем направлении, как в методе 2 ответа к упр. 212. Это вычисление раздувает НДР, пока она не достигает размера более чем в 800 тысяч узлов; но в конечном итоге он опускается до 584 205.

Искомый ответ оказывается равным 13 343 246 232 (что, конечно же, кратно $3! = 6$, поскольку каждая перестановка трех цветов дает другое решение).

(б) Этот вопрос отличен от вопроса в п. (а), поскольку многие покрытия (включая показанное на рис. 29, (б)) могут быть раскрашены в три цвета несколькими способами; мы же хотим учесть каждое из них только один раз.

Предположим, что $f(a_1, b_1, c_1, \ldots, a_m, b_m, c_m) = f(x_1, \ldots, x_{3m})$ является функцией с $a_i = x_{3i-2}$, $b_i = x_{3i-1}$ и $c_i = x_{3i}$, такой, что из $f(x_1, \ldots, x_{3m}) = 1$ вытекает $a_i + b_i + c_i \leq 1$

* В свое время эта задача предлагалась в журнале для школьников "*Квант*", где решение основывалось на том, что при покрытии шахматной доски костями домино количество покрытых черных квадратов должно быть равно количеству покрытых белых квадратов, а при вырезанных угловых клетках этого достичь невозможно. — *Примеч. пер.*

для $1 \le i \le m$. Давайте определим "*открашивание*" (*uncoloring*) $\$f$ для f как

$$\$f(x_1,\ldots,x_m) = \exists y_1 \cdots \exists y_{3m}\,(f(y_1,\ldots,y_{3m}) \\ \wedge (x_1 = y_1 + y_2 + y_3) \wedge \cdots \wedge (x_m = y_{3m-2} + y_{3m-1} + y_{3m})).$$

Непосредственная рекурсивная подпрограмма вычисляет НДР для $\$f$ из НДР для f. Этот процесс преобразует 584 205 узлов, полученных в п. (a), в НДР размером 33 731, из которой мы и получаем ответ: 3 272 232.

(Время работы для п. (a) составляет 1.2 миллиарда обращений к памяти плюс 1.3 миллиарда для открашивания; общее количество требующейся памяти — около 44 Мбайт. Аналогичные вычисления на основе БДР, а не НДР, стоят $13.6 + 1.5$ миллиарда обращений к памяти и требуют 185 Мбайт памяти.)

От переводчика: в комментарии (в TEX-файле) к данному упражнению автор просит при переводе на немецкий язык использовать в качестве цветов желтый, красный и черный. В связи с этим можно было бы добавить вопрос о раскраске в два цвета для стран с двухцветными флагами, но в этом случае ответ слишком прост и очевиден — имеется 4 покрытия, причем три из них получаются из четвертого путем транспонирования и обмена цветов.

217. Условие раздельности добавляет 4198 дополнительных ограничений вида $\neg(x_i \wedge x_{i'})$, где строки i и i' определяют смежное размещение конгруэнтных фигур. Применение этих ограничений при вычислении конъюнкции $\bigwedge_{j=1}^{468} S_1(X_j)$ в экспериментах автора оказалось плохой идеей; еще худшей идеей оказалась попытка построения отдельной НДР для новых ограничений. Гораздо лучшим оказалось построение 512 227-узловой НДР, как ранее, с последующим внедрением новых ограничений по одному, начиная с ограничений на нижних уровнях. Полученная в результате НДР размером 31 300 699 узлов оказывается полностью построенной после 286 миллиардов обращений к памяти и доказывает, что имеется ровно 7 099 053 234 102 раздельных решений.

Можно также задаться вопросом о *строго* раздельных решениях, в которых конгруэнтные фигуры не могут даже касаться одна другой углами; это требование добавляет еще 1948 ограничений. Всего имеется 554 626 216 строго раздельных покрытий, найденных ценой 245 миллиардов обращений к памяти с НДР размером 4 785 236 узлов. (Стандартный перебор с возвратом находит их быстрее и с очень небольшим количеством памяти.)

218. Это задача точного покрытия. Например, при $n = 3$ матрица имеет вид

001001010	(−−2−−2)
010001001	(−3−−−3)
010010010	(−2−−2−)
010100100	(−1−1−−)
100010001	(3−−−3−)
100100010	(2−−2−−)
101000100	(1−1−−−)

и в общем случае имеется $3n$ столбцов и $\binom{2n-1}{2} - \binom{n}{2}$ строк. Рассмотрим случай $n = 12$: НДР от 187 переменных имеет 192 636 узлов. Ее можно найти ценой 300 миллионов обращений к памяти с использованием метода 4 из ответа к упр. 212 (бинарное балансирование); в этом случае метод 5 оказывается на 25% медленнее метода 4. БДР получается гораздо большей (2 198 195 узлов), а стоимость оказывается большей 900 миллионов обращений к памяти.

Таким образом, для данной задачи НДР явно предпочтительнее БДР и идентифицирует $L_{12} = 108\,144$ решения с приемлемой эффективностью. (Однако метод "танцующих связей" из раздела 7.2.2 оказывается примерно в четыре раза быстрее и требует гораздо меньше памяти.)

219. (а) 1267; (б) 2174; (в) 2958; (г) 3721; (д) 4502. (Для формирования НДР для WORDS(n) требуется выполнить $n-1$ ИЛИ с 7-узловыми НДР для $\mathtt{w}_1 \sqcup \mathtt{h}_2 \sqcup \mathtt{i}_3 \sqcup \mathtt{c}_4 \sqcup \mathtt{h}_5$, $\mathtt{t}_1 \sqcup \mathtt{h}_2 \sqcup \mathtt{e}_3 \sqcup \mathtt{r}_4 \sqcup \mathtt{e}_5$, и т. д.)

220. (а) Существует один узел a_2 для потомков каждой начальной буквы, за которой во второй позиции может следовать a (aargh, babel, ..., zappy); таких букв 23 — все, за исключением q, u и x. Имеется также один узел b_2 для каждой начальной буквы, за которой может следовать буква b (abbey, ebony, oboes). Однако действующее правило не такое простое; например, имеются три, а не четыре узла z_2 из-за одинаковых окончаний czars и tzars.

(б) Не существует v_5, так как нет пятибуквенных слов, заканчивающихся на v. (Коллекция Stanford GraphBase не включает arxiv или webtv.) Три узла для w_5 возникают потому, что один означает случаи, в которых за буквами $< w_5$ должна следовать w (aglo и многие другие); другой узел означает случаи, в которых должны следовать либо w, либо y (stra, или resa, или когда мы видим allo, но не allot); имеется также узел w_5 для случая, когда за unse не следует ни e, ни t, поскольку за ним должно следовать либо w, либо x. Аналогично два узла для x_5 представляют случаи, в которых должна быть буква x или последней буквой должна быть либо x, либо y (после rela). Имеется только один узел y_5, поскольку нет таких четырех букв, за которыми могли бы следовать как y, так и z. Конечно же, имеются только один узел z_5, и два стока.

221. Для каждой возможной z-бусины ζ мы вычисляем вероятность встретиться с ней и выполняем суммирование по всем ζ. Для определенности рассмотрим z-бусину, которая соответствует ветвлению по r_3, и предположим, что она представляет подсемейство из 10 трехбуквенных суффиксов. Имеется ровно $\binom{6084}{10} - \binom{5408}{10} \approx 1.3 \times 10^{31}$ таких z-бусин, и согласно принципу включения и исключения каждая из них получается с вероятностью $\sum_{k\ge1} \binom{676}{k}(-1)^{k+1}\binom{11881376-6084k}{5757-10k}\Big/\binom{11881376}{5757} \approx 2.5 \times 10^{-32}$. [*Указание*: $|\{r,s,t,u,v,w,x,y,z\}| = 9$, $676 = 26^2$ и $6084 = 9 \times 26^2$.] Таким образом, такие z-бусины вносят в общее значение вклад, равный около 0.33. Согласно аналогичному анализу r_3-z-бусины для подсемейств размером 1, 2, 3, 4, 5, ... дают вклады, примерно равные 11.5, 32.3, 45.1, 41.9, 29.3, ...; так что можно ожидать около 188.8 ветвления по r_3 в среднем. Итоговая сумма

$$\sum_{l=1}^{5}\sum_{j=1}^{26}\sum_{s=1}^{5757}\left(\binom{26^{5-l}(27-j)}{s} - \binom{26^{5-l}(26-j)}{s}\right) \times \sum_{k=1}^{\infty}\binom{26^{l-1}}{k}(-1)^{k+1}\binom{26^5 - 26^{5-l}(27-j)k}{5757-sk}\Big/\binom{26^5}{5757},$$

плюс 2 для стоков, приближается к ≈ 7151.986. Средний z-профиль представляет собой $\approx$ (1.00, ..., 1.00; 25.99, ..., 25.99; 188.86, ..., 171.43; 86.31, ..., 27.32; 3.53, ..., 1.00; 2.00).

222. (а) Это множество всех подмножеств слов из F. (Имеется 50 569 таких подслов из $27^5 = 14\,348\,907$ возможных. Они описываются НДР размером 18 784, построенной из F и $\wp$ с помощью ответа к упр. 205, (б) ценой около 15 миллионов обращений к памяти.)

(б) Эта формула дает тот же результат, что и $F \sqcap \wp$, поскольку каждый член F содержит ровно один элемент каждого X_j. Но вычисление при этом оказывается гораздо медленнее — около 370 миллионов обращений к памяти — несмотря на тот факт, что $Z(X) = 132$ почти всегда столь же мало́, как и $Z(\wp) = 131$. (Заметим, что $|\wp| = 2^{130}$, в то время как $|X| = 26^5 \approx 2^{23.5}$.)

(в) $(F/P) \sqcup P$, где $P = \mathtt{t}_1 \sqcup \mathtt{u}_3 \sqcup \mathtt{h}_5$ представляет собой шаблон. (Искомые слова — touch, tough, truth. Это вычисление при использовании алгоритма из ответа к упр. 205 требует около 3000 обращений к памяти.) Другими претендентами на простые формулы

являются $F \cap Q$, где Q описывает допустимые слова. Если установить $Q = \mathtt{t}_1 \sqcup X_2 \sqcup \mathtt{u}_3 \sqcup X_4 \sqcup \mathtt{h}_5$, то мы имеем $Z(Q) = 57$, а стоимость опять оказывается равной ≈ 3000 обращений к памяти. С другой стороны, при $Q = (\mathtt{t}_1 \cup \mathtt{u}_3 \cup \mathtt{h}_5)\ \S\,3$ мы имеем $Z(Q) = 132$ и стоимость вырастает примерно до 9000 обращений к памяти. (Здесь $|Q|$ равно 26^2 в первом случае, но 2^{127} во втором — что *противоположно* накопленным в п. (а) и (б) интуитивно понятным представлениям! Поди разберись...)

(г) $F \cap ((V_1 \cup \cdots \cup V_5)\ \S\,k)$. Количество таких слов равно (24, 1974, 3307, 443, 9, 0) для $k = (0, \ldots, 5)$ соответственно, что получается из НДР размерами (70, 1888, 3048, 686, 34, 1).

(д) Искомые шаблоны удовлетворяют соотношению $P = (F \sqcap \wp) \cap Q$, где $Q = ((X_1 \cup \cdots \cup X_5)\ \S\,3)$. Мы имеем $Z(Q) = 386$, $Z(P) = 14221$ и $|P| = 19907$.

(е) Формула для этого случая хитрее. Сначала $P_2 = F \sqcap F$ дает F вместе со всеми шаблонами, удовлетворяемыми двумя различными словами; мы имеем $Z(P_2) = 11289$, $|P_2| = 21234$ и $|P_2 \cap Q| = 7753$. Но $P_2 \cap Q$ *не* является ответом; например, здесь пропущен шаблон `*atc*`, который встречается восемь раз, но только в контексте `*atch`. Правильный ответ — $P_2' \cap Q$, где $P_2' = (P_2 \setminus F) \sqcap \wp$. Тогда $Z(P_2') = 8947$, $Z(P_2' \cap Q) = 7525$, $|P_2' \cap Q| = 10472$.

(ж) $G_1 \cup \cdots \cup G_5$, где $G_j = (F/(\mathtt{b}_j \cup \mathtt{o}_j)) \sqcup \mathtt{b}_j$. Ответами являются `bared`, `bases`, `basis`, `baths`, `bobby`, `bring`, `busts`, `herbs`, `limbs`, `tribs`.

(з) Шаблоны, допускающие все гласные на втором месте: `b*lls`, `b*nds`, `m*tes`, `p*cks`.

(и) Первая формула дает все слова, средние три буквы которых являются гласными. Вторая дает все шаблоны с указанными первой и последней буквами, для которых имеется как минимум одно соответствующее слово с тремя вставленными гласными. Имеется 30 решений для первой формулы, но только 27 — для второй (поскольку, например, `louis` и `luaus` дают один и тот же шаблон). Кстати, комплементарное семейство $\wp \setminus F$ имеет $2^{130} - 5757$ членов и 46 316 узлов в своей НДР.

223. (а) $d(\alpha,\mu) + d(\beta,\mu) + d(\gamma,\mu) = 5$, поскольку $d(\alpha,\mu) = [\alpha_1 \neq \mu_1] + \cdots + [\alpha_5 \neq \mu_5]$.

(б) Для заданных семейств f, g и h, семейство $\{\mu \mid \mu = \langle\alpha\beta\gamma\rangle$ для некоторых $\alpha \in f$, $\beta \in g$, $\gamma \in h$ с $\alpha \neq \mu$, $\beta \neq \mu$, $\gamma \neq \mu$ и $\alpha \cap \beta \cap \gamma = \emptyset\}$ может быть определено рекурсивно для того, чтобы позволить вычисление НДР, если рассмотреть восемь вариантов, в которых ослаблены подмножества ограничений, являющихся неравенствами. В экспериментальной системе автора НДР для медиан `WORDS(`n`)` для $n = (100, 1000, 5757)$ имеют соответственно (595, 14389, 71261) узлов и характеризуют (47, 7310, 86153) пятибуквенных решений. Среди 86153 медиан при $n = 5757$ имеются `chads`, `stent`, `blogs`, `ditzy`, `phish`, `bling` и `tetch`; фактически `tetch` $= \langle$`fetch teach total`$\rangle$ возникает всегда при $n = 1000$. (Время работы, составляющее около (.01, 2, 700) миллиардов обращений к памяти соответственно, не было особо впечатляющим; вероятно, НДР — не лучший инструмент для данной задачи. Тем не менее программирование оказалось весьма поучительным.)

(в) Когда $n = 100$, ровно (1, 14, 47) медиан `WORDS(`n`)` принадлежат `WORDS(100)`, `WORDS(1000)`, `WORDS(5757)` соответственно; решение с наиболее распространенными словами представляет собой `while` $= \langle$`white whole still`$\rangle$. При $n = 1000$ соответствующие числа равны (38, 365, 1276); а при $n = 5757$ это (78, 655, 4480). Наиболее распространенные английские слова, *не* являющиеся медианами трех других английских слов, — `their`, `first` и `right`.

224. Каждая дуга $u \longrightarrow v$ ориентированного ациклического графа соответствует вершине v леса. НДР имеет ровно один узел ветвления для каждой дуги. Указатель LO этого узла ведет к правому брату соответствующей вершины v или к $\boxed{\perp}$, если v не имеет правых братьев. Указатель HI ведет к левому дочернему по отношению к v узлу или к $\boxed{\top}$, если v представляет собой лист. Эти дуги можно упорядочить многими способами (например, в прямом или обратном порядке обхода, по порядку уровней), не изменяя НДР.

225. Как в упр. 55, попытаемся пронумеровать вершины таким способом, чтобы "граница" между ранними и поздними вершинами оставалась достаточно небольшой; в таком случае нет необходимости помнить слишком много о решениях, которые были сделаны на ранних вершинах. В данном случае мы также хотим, чтобы вершина-источник s имела номер 1.

В ответе к упр. 55 соответствующее состояние из предыдущих ветвлений соответствует отношению эквивалентности (разбиению множеств); но сейчас мы выражаем его таблицей $mate[i]$ для $j \le i \le l$, где $j = u_k$ представляет собой меньшую вершину текущего ребра u_k — v_k и где $l = \max\{v_1, \ldots, v_{k-1}\}$. Пусть $mate[i] = i$, если вершина i до сих пор не была задействована; пусть $mate[i] = 0$, если вершина i уже была задействована дважды. В противном случае $mate[i] = r$ и $mate[r] = i$, если предыдущие ребра образуют простой путь с конечными точками $\{i, r\}$. Изначально мы устанавливаем $mate[i] \leftarrow i$ для $1 \le i \le n$, за исключением того, что $mate[1] \leftarrow t$ и $mate[t] \leftarrow 1$. (Если $t > l$, значение $mate[t]$ хранить не требуется, поскольку оно может быть определено из значений $mate[i]$ для $j \le i \le l$.)

Пусть $j' = u_{k+1}$ и $l' = \max\{v_1, \ldots, v_k\}$ представляют собой значения j и l после рассмотрения ребра k; предположим также, что $u_k = j$, $v_k = m$, $mate[j] = \hat{\jmath}$, $mate[m] = \hat{m}$. Мы не можем выбрать ребро j — m, если $\hat{\jmath} = 0$ или $\hat{m} = 0$. В противном случае, если $\hat{\jmath} \ne m$, после выбора ребра j — m путем присваиваний $mate[j] \leftarrow 0$, $mate[m] \leftarrow 0$, $mate[\hat{\jmath}] \leftarrow \hat{m}$, $mate[\hat{m}] \leftarrow \hat{\jmath}$ (в указанном порядке) может быть вычислена новая таблица $mate$.

В противном случае мы имеем $\hat{\jmath} = m$ и $\hat{m} = j$; мы должны видеть конец игры. Пусть i является наименьшим целым, таким, что $i > j$, $i \ne m$ и либо $i > l'$, либо $mate[i] \ne 0$ и $mate[i] \ne i$. Новое состояние после выбора ребра j — m равно $\emptyset$, если $i \le l'$, в противном случае оно равно ϵ.

Независимо от того, выбрано ребро или нет, новым состоянием будет $\emptyset$, если $mate[i] \ne 0$ и $mate[i] \ne i$ для некоторого i в диапазоне $j \le i < j'$.

Например, вот первые шаги для путей от 1 до 9 в решетке 3×3 (см. (132)).

k	j	l	m	$mate[1] \ldots mate[9]$	$\hat{\jmath}$	$\hat{m}$	$mate'[1] \ldots mate'[9]$
1	1	1	2	9 2 3 4 5 6 7 8 1	9	2	0 9 3 4 5 6 7 8 2
2	1	2	3	9 2 3 4 5 6 7 8 1	9	3	0 2 9 4 5 6 7 8 3
2	1	2	3	0 9 3 4 5 6 7 8 2	0	3	—
3	2	3	4	0 2 9 4 5 6 7 8 3	2	4	0 4 9 2 5 6 7 8 3
3	2	3	4	0 9 3 4 5 6 7 8 2	9	4	0 0 3 9 5 6 7 8 4

Здесь $mate'$ описывает следующее состояние при выбранном ребре j — m. Переходы между состояниями $mate_{j..l} \mapsto mate'_{j'..l'}$ представляют собой $9 \mapsto (\overline{12}?\ 92{:}\ 09)$; $92 \mapsto (\overline{13}?\ \emptyset{:}\ 29)$; $09 \mapsto (\overline{13}?\ 93{:}\ \emptyset)$; $29 \mapsto (\overline{24}?\ 294{:}\ 492)$; $93 \mapsto (\overline{24}?\ 934{:}\ 039)$.

После того как найдены все достижимые состояния, НДР можно получить путем приведения эквивалентных состояний с использованием процедуры наподобие алгоритма R. (В задаче с решеткой 3×3 57 узлов ветвления приводятся к 28 плюс два стока. Показанная в разделе НДР с 22 ветвлениями получена путем последовательной оптимизации с использованием упр. 197.)

226. Просто опустите начальные присваивания '$mate[1] \leftarrow t$, $mate[t] \leftarrow 1$'.

227. Замените в двух местах проверку '$mate[i] \ne 0$ и $mate[i] \ne i$' на '$mate[i] \ne 0$'. Замените также '$i \le l'$' на '$i \le n$'.

228. Используйте ответ к предыдущему упражнению со следующими дальнейшими изменениями. Добавьте фиктивную вершину $d = n + 1$ с новыми ребрами v — d для всех $v \ne s$; принятие этого нового ребра будет означать "закончить в v." Инициализируйте таблицу $mate$ значениями $mate[1] \leftarrow d$, $mate[d] \leftarrow 1$. При вычислении l и l' оставьте d за пределами максимизации. Когда начинается исследование сохраненной таблицы $mate$, начинайте с $mate[d] \leftarrow 0$, а затем, если встречаете $mate[i] = d$, устанавливайте $mate[d] \leftarrow i$.

229. 149 692 648 904 из последних путей идут от VA до MD; в графе (133) опущен DC. (Однако графы (18) имеют меньше чем (133) *гамильтоновых* путей, поскольку (133) имеет 1 782 199 гамильтоновых путей от CA до ME, которые не проходят от VA до MD.)

230. Единственные минимальный и максимальный маршруты идут из ME, и оба заканчиваются в WA.

11 698 миль

18 040 миль

Пусть $g(z) = \sum z^{\mathrm{miles}(r)}$, просуммированное по всем маршрутам r. Средняя стоимость $g'(1)/g(1) = 1022014257375/68656026 \approx 14886.01$ может быть быстро вычислена так, как в ответе к упр. 29.

(Аналогично $g''(1) = 15243164303013274$, так что стандартное отклонение составляет ≈ 666.2.)

Примечание переводчика. Воспользовавшись графом (133, y), можно найти соответствующие (также единственные) маршруты для Украины, приведенные ниже. В этом случае при одной и той же начальной точке конечные точки минимального и максимального маршрутов различны.

4309 км

5869 км

Усреднение по всем 3410 гамильтоновым путям дает среднюю длину, равную 5127.04 км (заметим, что ни один гамильтонов путь не заканчивается ни в Херсоне, ни во Львове).

231. Алгоритм из ответа к упр. 225 дает прото-НДР с 8 062 831 узлом ветвления; она приводится к НДР с 3 024 214 ветвлениями. Количество решений с применением ответа к упр. 208 составляет 50 819 542 770 311 581 606 906 543.

232. С помощью ответа к упр. 227 находим $h = 721\,613\,446\,615\,109\,970\,767$ гамильтоновых путей из угла к его горизонтальному соседу и $d = 480\,257\,285\,722\,344\,701\,834$ из них к его диагональному соседу; в обоих случаях соответствующая НДР содержит около 1.3 миллиона узлов. Количество ориентированных гамильтоновых циклов составляет $2h + d = 1\,923\,484\,178\,952\,564\,643\,368$. (Деление на 2 дает количество *неориентированных* гамильтоновых циклов.)

По сути, максимальной длины $8 + 56\sqrt{2}$ достигают только два обхода королем:

и

.

233. Может использоваться аналогичная процедура, но с $mate[i] = r$ и $mate[r] = -i$, когда предыдущие выборы определяют ориентированный путь от i до r. Последовательно обработаем все дуги $u_k \longrightarrow v_k$ и $u_k \longleftarrow v_k$ при $u_k = j < v_k = m$. Определим $\hat{\jmath} = -j$, если $mate[j] = j$, в противном случае $\hat{\jmath} = mate[j]$. Выбор $j \longrightarrow m$ некорректен, если $\hat{\jmath} \geq 0$ или $\hat{m} \leq 0$. Обновленное правило для этого выбора, когда он корректен, имеет вид $mate[j] \leftarrow 0$, $mate[m] \leftarrow 0$, $mate[-\hat{\jmath}] \leftarrow \hat{m}$, $mate[\hat{m}] \leftarrow \hat{\jmath}$.

234. 437 ориентированных циклов можно представить НДР с ≈ 800 узлами. Кратчайшими циклами, конечно, являются AL $\longrightarrow$ LA $\longrightarrow$ AL и MN $\longrightarrow$ NM $\longrightarrow$ MN. Имеется 37 циклов длиной 17 (максимальной), таких как (ALARINVTNMIDCOKSC), т. е. AL $\longrightarrow$ LA $\longrightarrow \cdots \longrightarrow$ SC $\longrightarrow$ CA $\longrightarrow$ AL.

Кстати, рассматриваемый ориентированный граф представляет собой ориентированный граф дуг D^* ориентированного графа D с 26 вершинами {A,B,...,Z}, 49 дугами которого являются A $\longrightarrow$ L, A $\longrightarrow$ R, ..., W $\longrightarrow$ Y. Каждый ориентированный путь в D^* представляет собой ориентированный путь в D и обратно (см. упр. 2.3.4.2–21); но ориентированные *циклы* в D^* не обязательно просты в D. Фактически D имеет только 37 ориентированных циклов с единственным самым длинным среди них циклом (ARINMOKSDC).

Если расширить наше рассмотрение, распространив его на 62 почтовых кода из упр. 7–54, (в), то количество ориентированных циклов вырастет до 38336, включая единственный 1-цикл (A), а также 192 с длиной 23, такие как (APRIALASCTNMNVINCOKSDCA). Для определения всего семейства ориентированных циклов в этом случае достаточно НДР с около 17 000 узлов.

235. Этот ориентированный граф имеет 7912 дуг; но его можно существенно обрезать, удаляя дуги с нулевой входящей степенью, а также дуги с нулевой исходящей степенью. Например, мы отбрасываем owner $\longrightarrow$ nerdy, поскольку nerdy оказывается тупиком; фактически все преемники owner оказываются удаленными, так что убирается и crown. В конечном счете мы остаемся со 112 дугами, соединяющими 85 слов, и задача в основном решается вручную.

Имеется только 74 ориентированных цикла. Наикратчайший цикл единственный, slant $\longrightarrow$ antes $\longrightarrow$ tesla $\longrightarrow$ slant, и может быть описан, как в предыдущем упражнении, аббревиатурой '(slante)'. Двумя длиннейшими циклами являются $(\alpha\omega)$ и $(\beta\omega)$, где $\alpha =$ picastepsomaso, $\beta =$ pointrotherema, а

$$\omega = \texttt{nicadrearedidoserumorelicite slabsitaresetuplenactoricedarerunichesto}.$$

236. (а) Предположим, что $\alpha \in f$ и $\beta \in g$. Если $\alpha \subseteq \beta$, то $\alpha \in f \sqcap g$. Если $\alpha \cap \beta \in f$, то $\alpha \cap \beta \notin f \nearrow g$. Аналогичные рассуждения, или использование п. (б), показывают, что $f \searrow g = f \setminus (f \sqcup g)$.

Примечания. Операции дополнения "$f \nwarrow g = f \setminus (f \searrow g) = \{\alpha \in f \mid \alpha \supseteq \beta$ для некоторого $\beta \in g\}$" для надмножеств и "$f \swarrow g = f \setminus (f \nearrow g) = \{\alpha \in f \mid \alpha \subseteq \beta$ для некоторого $\beta \in g\}$" для подмножеств также важны в приложениях. Они были опущены в этом упражнении только потому, что пять операций уже выглядят достаточно пугающе. Операция надмножества была введена О. Кудером (O. Coudert), Ж. К. Мадром (J. C. Madre) и Г. Фрейссом (H. Fraisse) [*ACM/IEEE Design Automation Conference* **30** (1993), 625–630]. Тождество $f \nwarrow g = f \cap (f \sqcup g)$ было отмечено Х. Окуно (H. Okuno), Ш. Минато (S. Minato) и Х. Исозаки (H. Isozaki) [*Information Processing Letters* **66** (1998), 195–199], которые также перечислили ряд законов из п. (г).

(б) Достаточно элементарной теории множеств. (Первые шесть тождеств появляются в парах, каждое из тождеств эквивалентно своему напарнику. Строго говоря, f^C включает бесконечные множества, и U является И бесконечного числа переменных; но формулы выполняются в любом конечном универсуме. Заметим, что при переводе на язык булевых функций $f^C(x) = f(\bar{x})$ является дополнением булево дуальной функции f^D; см. упр. 7.1.1–2. Имеется ли какое-то применение дуальности для $f^\sharp$, а именно того, что {Из $\alpha \mid \beta \in f$ вытекает $\alpha \cup \beta \neq U\}^\uparrow$? Если да, то мы можем обозначить ее как $f^\flat$.)

(в) Истинны все, за исключением (ii), которое должно гласить $x_1^\uparrow = x_1^{C\downarrow C} = \bar{x}_1^{\downarrow C} = \epsilon^C = U$.

(г) "Тождествами", которые следует вычеркнуть, в данном случае являются (ii), (viii), (ix), (xiv) и (xvi); прочие стóят того, чтобы их запомнить. Что касается (ii)–(vi), то заметим, что $f = f^\uparrow$ тогда и только тогда, когда $f = f^\downarrow$, тогда и только тогда, когда f является клат-

тером. Формула (xiv) должна иметь вид $f \searrow g^{\downarrow} = f \searrow g$, дуальный (xiii). Формула (xvi) почти верна; она неверна, только когда $f = \emptyset$ или $g = \emptyset$. Формула (ix), пожалуй, наиболее интересна: в действительности мы имеем $f^{\sharp\sharp} = f$ тогда и только тогда, когда f является клаттером.

(д) В предположении конечности универсума всех вершин мы имеем (i) $f = \wp \searrow g$ и (ii) $g = (\wp \setminus f)^{\downarrow}$, где $\wp$ представляет собой универсальное семейство из упр. 201 и 222, поскольку g является семейством минимальных зависимых множеств. (Пуристы должны заменить в этих формулах $\wp_V = \bigsqcup_{v \in V}(\epsilon \cup e_v)$ на $\wp$. Те же отношения справедливы в любом гиперграфе, в котором ни одно ребро не содержится в другом.)

237. MAXMAL(f) = "Если $f = \emptyset$ или $f = \epsilon$, вернуть f. Если $f^{\uparrow} = r$ кэшировано, вернуть r. В противном случае установить $r \leftarrow$ MAXMAL(f_l), $r_h \leftarrow$ MAXMAL(f_h), $r_l \leftarrow$ NONSUB(r, r_h), разыменовать r, установить $r \leftarrow$ ZUNIQUE(f_v, r_l, r_h); кэшировать и вернуть r".

MINMAL(f) = "Если $f = \emptyset$ или $f = \epsilon$, вернуть f. Если $f^{\downarrow} = r$ кэшировано, вернуть r. В противном случае установить $r_l \leftarrow$ MINMAL(f_l), $r \leftarrow$ MINMAL(f_h), $r_h \leftarrow$ NONSUP(r, r_l), разыменовать r, установить $r \leftarrow$ ZUNIQUE(f_v, r_l, r_h); кэшировать и вернуть r".

NONSUB(f, g) = "Если $g = \emptyset$, вернуть f. Если $f = \emptyset$ или $f = \epsilon$ или $f = g$, вернуть $\emptyset$. Если $f \nearrow g = r$ кэшировано, вернуть r. В противном случае представить f и g, как в (52). Если $v < g_v$, установить $r_l \leftarrow$ NONSUB(f_l, g), $r_h \leftarrow f_h$ и увеличить REF(f_h) на 1; в противном случае установить $r_h \leftarrow$ NONSUB(f_l, g_l), $r \leftarrow$ NONSUB(f_l, g_h), $r_l \leftarrow$ И(r, r_h), разыменовать r и r_h и установить $r_h \leftarrow$ NONSUB(f_h, g_h). Наконец $r \leftarrow$ ZUNIQUE(v, r_l, r_h); кэшировать и вернуть r".

NONSUP(f, g) = "Если $g = \emptyset$, вернуть f. Если $f = \emptyset$ или $g = \epsilon$, или $f = g$, вернуть $\emptyset$. Если $f_v > g_v$, вернуть NONSUP(f, g_l). Если $f \searrow g = r$ кэшировано, вернуть r. В противном случае установить $v = f_v$. Если $v < g_v$, установить $r_l \leftarrow$ NONSUP(f_l, g) и $r_h \leftarrow$ NONSUP(f_h, g); в противном случае установить $r_l \leftarrow$ NONSUP(f_h, g_h), $r \leftarrow$ NONSUP(f_h, g_l), $r_h \leftarrow$ И(r, r_l), разыменовать r и r_l и установить $r_l \leftarrow$ NONSUP(f_l, g_l). Наконец $r \leftarrow$ ZUNIQUE(v, r_l, r_h); кэшировать и вернуть r".

MINHIT(f) = "Если $f = \emptyset$, вернуть ϵ. Если $f = \epsilon$, вернуть $\emptyset$. Если $f^{\sharp} = r$ кэшировано, вернуть r. В противном случае установить $r \leftarrow$ ИЛИ(f_l, f_h), $r_l \leftarrow$ MINHIT(r), разыменовать r, $r \leftarrow$ MINHIT(f_l), $r_h \leftarrow$ NONSUP(r, r_l), разыменовать r, и установить $r \leftarrow$ ZUNIQUE(f_v, r_l, r_h); кэшировать и вернуть r".

Как в упр. 206, неизвестно время работы этих подпрограмм в наихудшем случае. Хотя NONSUB и NONSUP можно вычислить через JOIN или MEET и НО-НЕ, согласно упр. 236, (а), эта непосредственная реализация обычно быстрее. Может оказаться предпочтительной замена '$f = \epsilon$' на '$\epsilon \in f$' в MINMAL и MINHIT; а также '$g = \epsilon$' на '$\epsilon \in g$' в NONSUP.

[Оливье Кудер (Olivier Coudert) ввел и реализовал операторы $f^{\uparrow}$, $f \nearrow g$ и $f \searrow g$ в *Proc. Europ. Design and Test Conf.* (IEEE, 1997), 224–228. Он также разработал рекурсивную реализацию интересного оператора $f \odot g = (f \sqcup g)^{\uparrow}$; однако в экспериментах автора гораздо лучшие результаты были получены без нее. Например, если f представляет собой 177-узловую НДР для независимых множеств смежных штатов США, операция $g \leftarrow$ JOIN(f, f) стоит около 350 тысяч обращений к памяти, а $h \leftarrow$ MAXMAL(g) — около 3.6 миллиона обращений к памяти; но внезапно вычисление $h \leftarrow$ MAXJOIN(f, f) потребовало более 69 *миллиардов* обращений к памяти. Конечно, картину могут изменить усовершенствованные стратегии кэширования и сборки мусора.]

238. Можно вычислить 177-узловую НДР для семейства f независимых множеств, используя упорядочение (104), двумя способами. При использовании булевой алгебры (67) $f = \neg \bigvee_{u-v}(x_u \wedge x_v)$; стоимость составляет около 1.1 миллиона обращений к памяти при использовании алгоритмов из ответов к упр. 198–201. С другой стороны, при применении алгебры семейств мы имеем $f = \wp \searrow \bigsqcup_{u-v}(e_u \sqcup e_v)$ согласно упр. 236, (д); стоимость при использовании ответа к упр. 237 оказывается меньше 175 тысяч обращений к памяти.

Подмножествами, дающими подграфы, раскрашиваемые двумя и тремя цветами, являются соответственно $g = f \sqcup f$ и $h = g \sqcup f$; максимальными подмножествами являются $g^\uparrow$ и $h^\uparrow$. Мы имеем $Z(g) = 1009$, $Z(g^\uparrow) = 3040$, $Z(h) = 179$, $Z(h^\uparrow) = 183$, $|g| = 9\,028\,058\,789\,780$, $|g^\uparrow| = 2\,949\,441$, $|h| = 543\,871\,144\,820\,736$ и $|h^\uparrow| = 384$. Последующая стоимость вычислений g, $g^\uparrow$, h и $h^\uparrow$ примерно равна 350 Кμ, 3.6 Мμ, 1.1 Мμ и 230 Кμ. (Мы можем вычислить $h^\uparrow$ посредством, скажем, $(g^\uparrow \sqcup f)^\uparrow$; но это оказывается плохой идеей.)

Максимальные порожденные двух- и трехдольные подграфы имеют соответственно производящие функции $7654z^{25}+\cdots+9040z^{33}+689z^{34}$ и $128z^{43}+84z^{44}+112z^{45}+36z^{46}+24z^{47}$. Вот типичные примеры наименьшего и наибольшего из них.

(Сравните с наименьшим и наибольшим "однодольными" подграфами в 7–(61) и 7–(62).)

Заметим, что семейства g и h говорят нам о том, какие порожденные подграфы могут быть раскрашены в два и три цвета, но *не* говорят о том, как это сделать.

239. Поскольку $h = ((e_1 \cup \cdots \cup e_{49}) \S\, 2) \setminus g$ представляет собой множество не ребер G, кликами являются $f = \wp \searrow h$, а максимальная клика есть ни что иное как $f^\uparrow$. Например, мы имеем $Z(f) = 144$ для 214 клик графа США и $Z(f^\uparrow) = 130$ для 60 максимальных из них. В этом случае максимальные клики состоят из 57 треугольников (которые легко видимы в (18)) вместе с тремя ребрами, которые не являются частью никакого из треугольников: `AZ` — `NM`, `WI` — `MI`, `NH` — `ME`.

Пусть f_k описывает множества, покрываемые k кликами. Тогда $f_1 = f$, а $f_{k+1} = f_k \sqcup f$ для $k \ge 1$. (Вычислять f_{16} как $f_8 \sqcup f_8$ — идея не из хороших; гораздо быстрее выполнить каждое соединение отдельно, даже если промежуточные результаты нас не интересуют.)

Наибольшие элементы f_k в графе США имеют размеры 3, 6, 9, ..., 36, 39, 41, 43, 45, 47, 48, 49 для $1 \le k \le 19$; в таком небольшом графе, как наш, эти значения легко определить вручную. Однако вопрос о максимальных* элементах гораздо более тонкий, и, пожалуй, лучший инструмент для его исследования — бинарные диаграммы решений с подавлением нулей. НДР для $f_1, \ldots, f_{19}$ легко находится ценой 30 миллионов обращений к памяти вычислений, и они невелики: $\max Z(f_k) = Z(f_{11}) = 9547$. Другие 400 миллионов обращений к памяти дают НДР для $f_1^\uparrow, \ldots, f_{19}^\uparrow$, которые также невелики: $\max Z(f_k^\uparrow) = Z(f_{11}^\uparrow) = 9458$.

Мы находим, например, что производящая функция для $f_{18}^\uparrow$ имеет вид $12z^{47} + 13z^{48}$; восемнадцати клик достаточно для покрытия всех 49 вершин, кроме одной, если мы не будем включать `CA`, `DC`, `FL`, `IL`, `LA`, `MI`, `MN`, `MT`, `SC`, `TN`, `UT`, `WA` или `WV`. Имеются также двенадцать случаев максимального покрытия 47 вершин; например, если все вершины, кроме `NE` и `NM`, покрыты 18 кликами, то ни один из указанных штатов не покрыт. Необычный пример максимального покрытия кликами приведен на рисунке: если 29 "черных" штатов покрыты 12 кликами, то не будет покрыт ни один из "белых" штатов.

240. (а) Фактически подформула $f(x) = \bigwedge_v (x_v \vee \bigvee_{u-v} x_u)$ из (68) точно характеризует доминирующие множества x. А если удален любой элемент ядра, оно не доминирует над другими. [C. Berge, *Théorie des graphes et ses applications* (1958), 44.]

(б) Булева формула из п. (а) дает НДР с $Z(f) = 888$ после примерно 1.5 Мμ вычислений; затем другие 1.5 Мμ с алгоритмом MINMAL из ответа к упр. 237 дают минимальные элементы с $Z(f^\downarrow) = 2082$.

* Вспомните обсуждение различия терминов "максимальный" и "наибольший" в разделе 7 на с. 56. — *Примеч. пер.*

Более ясный путь заключается в том, чтобы начать с $h = \bigcup_v (e_v \sqcup \bigsqcup_{u-v} e_u)$, а затем вычислить $h^\sharp$, поскольку $h^\sharp = f^\downarrow$. Однако ясность в данном случае не главное: для вычисления h достаточно около 80 Кμ, но вычисление $h^\sharp$ с помощью алгоритма MINHIT стоит около 350 Мμ.

В любом случае мы выводим, что имеется ровно 7 798 658 минимальных доминирующих множеств. Говоря точнее, производящая функция имеет вид $192z^{11} + 58855z^{12} + \cdots + 4170z^{18} + 40z^{19}$ (что можно сравнить с $80z^{11} + 7851z^{12} + \cdots + 441z^{18} + 18z^{19}$ для ядер).

(в) Действуя так, как в ответе к упр. 239, мы можем определить множества вершин d_k, над которыми доминируют подмножества размеров $k = 1, 2, 3, \ldots$, поскольку $d_{k+1} = d_k \sqcup d_1$. Это гораздо быстрее, чем начинать с $d_1 = \wp \sqcap h$ вместо $d_1 = h$, несмотря на то что $Z(\wp \sqcap h) = 313$ при $Z(h) = 213$, поскольку нас не интересуют подробности о маломощных членах d_k. Используя тот факт, что производящая функция для d_7 имеет вид $\cdots + 61z^{42} + z^{43}$, можно убедиться в том, что приведенное на рисунке решение единственное. (Общая стоимость вычислений ≈ 300 Мμ.)

Примечание переводчика. В случае графа для Украины имеется десять наименьших доминирующих множеств размером 5, одно из которых (единственное, включающее столицу) показано на рисунке. Заметим, что каждое из решений включает Винницу, Львов и Херсон.

241. Пусть g представляет собой семейство всех 728 ребер. Тогда, как в предыдущих упражнениях, $f = \wp \searrow g$ представляет собой семейство независимых множеств, а кликами являются $c = \wp \searrow (((\bigcup_v e_v) \S 2) \setminus g)$. Мы имеем $Z(g) = 699$, $Z(f) = 20244$, $Z(c) = 1882$.

(а) Среди $|f| = 118\,969$ независимых множеств имеется $|f^\uparrow| = 10\,188$ ядер с $Z(f^\uparrow) = 8577$ и производящей функцией $728z^5 + 6912z^6 + 2456z^7 + 92z^8$. 92 максимальных независимых множества представляют собой знаменитые решения классической задачи о восьми ферзях, которую мы изучим в разделе 7.2.2; пример (C1) является единственным решением, в котором никакие три ферзя не находятся на одной прямой, как было замечено Сэмом Лойдом (Sam Loyd) в *Brooklyn Daily Eagle* (20 Dec 1896). $728 = 91 \times 8$ минимальных ядер были впервые перечислены К. Ф. де Янишем (C. F. de Jaenisch), *Traité des applications de l'analyse math. au jeu des échecs* **3** (1863), 255–259, который приписал их некоему "M^{r} de R***". Верхний левый ферзь в (C0) может быть заменен королем, слоном или пешкой, что не повлияет на доминирование над всеми открытыми полями [H. E. Dudeney, *The Weekly Dispatch* (3 Dec 1899)].

(б) В этом случае $Z(c^\uparrow) = 866$; 310 максимальных клик описаны в упр. 7–129.

(в) Эти подмножества вычислительно более сложные: НДР для всех доминирующих множеств d имеет $Z(d) = 12\,663\,505$, $|d| = 18\,446\,595\,708\,474\,987\,957$; минимальные имеют $Z(d^{\downarrow}) = 11\,363\,849$, $|d^{\downarrow}| = 28\,281\,838$, а производящая функция имеет вид $4860z^5 + 1075580z^6 + 14338028z^7 + 11978518z^8 + 873200z^9 + 11616z^{10} + 36z^{11}$. С использованием булевой алгебры можно вычислить НДР для d за 1.5 Gμ и затем НДР для $d^{\downarrow}$ еще за 680 Gμ; альтернативный, более "ясный" подход из ответа к упр. 240 получает $d^{\downarrow}$ за 775 Gμ без вычисления d. Расстановка 11 ферзей (C5) является единственным таким минимальным доминирующим множеством, ограниченным тремя рядами. Г. Э. Дьюдени (H. E. Dudeney) представил (C4), единственное решение, оставляющее пустым центральный ромб, в *Tit Bits* (1 Jan 1898), 257. Множество всех 4860 наименьших решений было впервые подсчитано K. фон Сили (K. von Szily) [*Deutsche Schachzeitung* **57** (1902), 199]; полный его список был приведен в W. Ahrens, *Math. Unterhaltungen und Spiele* **1** (1910), 313–318.

(г) Здесь достаточно вычислить $(c \cap d)^{\downarrow}$ вместо $c \cap (d^{\downarrow})$, если мы еще не знаем $d^{\downarrow}$, поскольку $c \sqcap \wp = c$. Мы имеем $Z(c \cap d^{\downarrow}) = 342$ и $|c \cap d^{\downarrow}| = 92$ с производящей функцией $20z^5 + 56z^6 + 16z^7$. И вновь Дьюдени был первым, кто открыл все 20 решений с 5 ферзями [*The Weekly Dispatch* (30 July 1899)].

(д) Мы имеем $Z(f \sqcup f) = 91\,780\,989$ ценой 24 Gμ; затем $Z((f \sqcup f)^{\uparrow}) = 11\,808\,436$ после очередных 290 Gμ. Имеется 27 567 390 максимальных порожденных двудольных подграфов с производящей функцией $109894z^{10} + 2561492z^{11} + 13833474z^{12} + 9162232z^{13} + 1799264z^{14} + 99408z^{15} + 1626z^{16}$. Любые 8 независимых ферзей можно скомбинировать с их зеркальным отражением и получить решение с 16 ферзями; так (C1) дает (C9). Но непересекающееся объединение минимальных ядер не всегда представляет собой максимальный порожденный двудольный подграф; например, рассмотрим объединение (C0) с его отражением:

Пункты (а), (б), (г) и, возможно, (в) можно решить и без применения НДР; см., например, упр. 7.1.3–132. Однако для п. (г), похоже, наилучшим является подход с применением НДР. А вычисление всех максимальных *трехдольных* подграфов Q_8 может оказаться за рамками *любого* реального алгоритма.

[В бóльших графах ферзей Q_n известно, что наименьшие ядра и наименьшие доминирующие множества имеют размеры либо $\lceil n/2 \rceil$, либо $\lceil n/2 \rceil + 1$ для $12 \le n \le 120$. См. P. R. J. Östergard and W. D. Weakley, *Electronic J. Combinatorics* **8** (2001), #R29; D. Finozhenok and W. D. Weakley, *Australasian J. Combinatorics* **37** (2007), 295–300. Наибольшие минимальные доминирующие множества были исследованы в A. P. Burger, E. J. Cockayne and C. M. Mynhardt, *Discrete Mathematics* **163** (1997), 47–66.]

242. Это ядра интересного 3-регулярного гиперграфа с 1544 ребрами. 4 113 975 079 его независимых подмножеств f (т. е. его подмножеств, в которых не имеется трех коллинеарных точек) имеют $Z(f) = 52\,322\,105$, вычислимое примерно за 12 миллиардов обращений к памяти с использованием алгебры семейств, как в ответе к упр. 236, (д). Другие 575 Gμ вычисляют ядра $f^{\uparrow}$, для которых мы имеем $Z(f^{\uparrow}) = 31\,438\,750$ и $|f^{\uparrow}| = 66\,509\,584$; производящая функция имеет вид $228z^8 + 8240z^9 + 728956z^{10} + 9888900z^{11} + 32215908z^{12} + 20739920z^{13} + 2853164z^{14} + 73888z^{15} + 380z^{16}$.

[Задача поиска независимого множества размером 16 была впервые поставлена Г. Э. Дьюдени (H. E. Dudeney) в *The Weekly Dispatch* (29 апреля и 13 мая 1900 года), где он привел

шаблон, крайний слева из показанных выше. Позже, в лондонской *Tribune* (7 ноября 1906), Дьюдени поставил перед любителями головоломок задачу найти второй шаблон с двумя точками в центре. Полное множество максимальных ядер, включая 57 различных с учетом симметрии, было найдено в работе M. A. Adena, D. A. Holton and P. A. Kelly, *Lecture Notes in Math.* **403** (1974), 6–17, авторы которой также обратили внимание на существование 8-точечного ядра. Средний из показанных выше шаблонов является одним из таких ядер, у которого все точки находятся в центральном квадрате 4×4. Два другие шаблона дают ядра, которые имеют соответственно $(8, 8, 10, 10, 12, 12, 12)$ точек в решетках $n \times n$ для $n = (8, 9, \ldots, 14)$; они были найдены С. Айнли (S. Ainley) и описаны в письме Мартину Гарднеру (Martin Gardner) от 27 октября 1976 года.]

243. (а) Этот результат легко проверить даже для бесконечных множеств. (Заметим, что как булева функция $f^{\cap}$ является наименьшей функцией Хорна, которая представляет собой $\supseteq f$ согласно теореме 7.1.1H.)

(б) Можно образовать $f^{(2)} = f \sqcap f$, затем $f^{(4)} = f^{(2)} \sqcap f^{(2)}$, ... и далее до $f^{(2^{k+1})} = f^{(2^k)}$, используя упр. 205. Однако быстрее разработать рекуррентное соотношение, быстро приводящее к пределу. Если $f = f_0 \cup (e_1 \sqcup f_1)$, мы имеем $f^{\cap} = f' \cup (e_1 \sqcup f_1^{\cap})$, где $f' = f_0^{\cap} \cup (f_0^{\cap} \sqcap f_1^{\cap})$. [Альтернативной формулой является $f' = (f_0 \cup f_1)^{\cap} \setminus (f_1^{\cap} \nearrow f_0)$; см. S. Minato and H. Arimura, *Transactions of the Japanese Society for Artificial Intelligence* **22** (2007), 165–172.]

(в) При применении первого предложения из п. (б) вычисление $F^{(2)}$, $F^{(4)}$ и $F^{(8)} = F^{(4)}$ стоит около $(610+450+460)$ миллионов обращений к памяти. В этом примере оказывается, что $F^{(4)} = F^{(3)}$ и что только три шаблона принадлежат $F^{(3)} \setminus F^{(2)}$, а именно `c***f`, `*k*t*` и `***sp`. (Словами, соответствующими шаблону `***sp`, являются `clasp`, `crisp` и `grasp`.) Непосредственное вычисление $F^{\cap}$ с помощью рекуррентного соотношения, основанного на $f_0^{\cap} \sqcap f_1^{\cap}$, стоит только 320 M$\mu$; в данном примере альтернативное рекуррентное соотношение, основанное на $(f_0 \cup f_1)^{\cap}$, имеет стоимость 470 Mμ. Производящая функция имеет вид $1 + 124z + 2782z^2 + 7753z^3 + 4820z^4 + 5757z^5$.

244. Чтобы преобразовать рис. 22 из БДР в НДР, мы добавляем соответствующие узлы с LO = HI, где связи перепрыгивают через уровни, приводя к z-профилю (1, 2, 2, 4, 4, 5, 5, 5, 5, 5, 2, 2, 2). Чтобы преобразовать его из НДР в БДР, мы добавляем узлы в те же места, но с HI = $\boxed{\bot}$, получая профиль (1, 2, 2, 4, 4, 5, 5, 5, 5, 5, 2, 2, 2). (Фактически функция связности и функция остовного дерева представляют собой Z-преобразование друг друга; см. упр. 192.)

245. См. упр. 7.1.1–26. (Было бы интересно сравнить производительность алгоритма Фредмана–Хачияна из упр. 7.1.1–27 с НДР-алгоритмом MINHIT из ответа к упр. 237 для множества различных функций.)

246. Если неконстантная функция не зависит от x_1, можно заменить x_1 в формулах на x_v, как в (50). Пусть P и Q — простые импликанты функций p и q. (Например, если $P = e_2 \cup (e_3 \sqcup e_4)$, то $p = x_2 \vee (x_3 \wedge x_4)$.) Согласно (137) и индукции по $|f|$ функция f, описанная в теореме, является легкой тогда и только тогда, когда p и q легкие и $\mathrm{PI}(f_0) \cap \mathrm{PI}(f_1) = \emptyset$. Последнее равенство выполняется тогда и только тогда, когда $p \subseteq q$.

247. Их можно охарактеризовать с помощью БДР, как в (49) и в упр. 75; но в этот раз

$$\sigma_n(x_1, \ldots, x_{2^n}) = \sigma_{n-1}(x_1, \ldots, x_{2^n-1}) \wedge \\ \Bigl((\bar{x}_2 \wedge \cdots \wedge \bar{x}_{2^n}) \vee \Bigl(\sigma_{n-1}(x_2, \ldots, x_{2^n}) \wedge \bigwedge_{j=0}^{2^{k-1}} \Bigl(\bar{x}_{2j+1} \vee \bigvee_{i \subset j} x_{2i+2}\Bigr)\Bigr)\Bigr).$$

Ответами $|\sigma_n|$ для $0 \le n \le 7$ являются (2, 3, 6, 18, 106, 2102, 456774, 7108935325). (Это вычисление строит БДР размером $B(\sigma_7) = 7\,701\,683$, выполняя около 900 миллионов обращений к памяти и используя 725 Мбайт памяти.)

248. Ложно; например, $(x_1 \vee x_2) \wedge (x_2 \vee x_3)$ не является легкой. (Но конъюнкция *является* легкой, если f и g зависят от непересекающихся множеств переменных или если x_1 является единственной переменной, от которой зависят обе функции.)

249. (Решение Шаддина Дахми (Shaddin Dughmi) и Яна Поста (Ian Post).) Ненулевая монотонная булева функция сверхлегкая тогда и только тогда, когда простые импликанты являются базами матроида; см. раздел 7.6.1. Расширяя ответ к упр. 247, можно определить количества сверхлегких функций $f(x_1, \ldots, x_n)$ для $0 \le n \le 7$: (2, 3, 6, 17, 69, 407, 3808, 75 165).

250. Исчерпывающий анализ показывает, что среднее $B(f) = 76726/7581 \approx 10.1$; среднее $Z(\mathrm{PI}(f)) = 71513/7581 \approx 9.4$; $\Pr\bigl(Z(\mathrm{PI}(f)) > B(f)\bigr) = 151/7581 \approx .02$; и максимальное $Z(\mathrm{PI}(f))/B(f) = 8/7$ достигается в единственном случае, когда f представляет собой $(x_1 \wedge x_4) \vee (x_1 \wedge x_5) \vee (x_2 \wedge x_3 \wedge x_4) \vee (x_2 \wedge x_5)$.

251. Говоря более строго, может ли выполняться $\limsup Z(\mathrm{PI}(f))/B(f) = 1$?

252. НДР должна описывать все слова из алфавита $\{e_1, e_1', \ldots, e_n, e_n'\}$, которые имеют ровно j букв без штриха и $k-j$ букв со штрихом и в одном слове не встречаются одновременно e_i и e_i' для некоторого множества пар (j, k). Например, если $n = 9$ и $f(x) = v_{\nu x}$, где $v = 110111011$, этими парами являются $(0, 8)$, $(3, 6)$ и $(8, 8)$. Независимо от множества пар, все элементы z-профиля будут $O(n^2)$, следовательно, $Z(\mathrm{PI}(f)) = O(n^3)$. (Мы упорядочиваем переменные так, чтобы x_i и x_i' были соседями.) А $f(x) = S_{\lfloor n/3 \rfloor, \ldots, \lfloor 2n/3 \rfloor}(x)$ имеет $Z(\mathrm{PI}(f)) = \Omega(n^3)$.

253. Пусть $\mathrm{I}(f)$ представляет собой семейство всех *импликант* f; тогда $\mathrm{PI}(f) = \mathrm{I}(f)^{\downarrow}$. Формулу $\mathrm{I}(f) = \mathrm{I}(f_0 \wedge f_1) \cup (e_1' \sqcup \mathrm{I}(f_0)) \cup (e_1 \sqcup \mathrm{I}(f_1))$ легко проверить. Таким образом, $\mathrm{I}(f)^{\downarrow} = A \cup (e_1' \sqcup (\mathrm{PI}(f_0) \searrow A)) \cup (e_1 \sqcup (\mathrm{PI}(f_1) \searrow A))$, как в упр. 237. Но $\mathrm{PI}(f_0) \searrow A = \mathrm{PI}(f_0) \setminus A$, поскольку $A \subseteq \mathrm{I}(f)$.

[Это рекуррентное соотношение для простых импликант было получено О. Кудером (O. Coudert) и Ж. К. Мадре (J. C. Madre), *ACM/IEEE Design Automation Conf.* **29** (1992), 36–39. Частичные результаты были сформулированы ранее в работе B. Reusch, *IEEE Trans.* **C-24** (1975), 924–930.]

254. Согласно (53) и (137) нам надо показать, что $\mathrm{PI}(g_h) \setminus \mathrm{PI}(f_h \cup g_l) = \bigl(\mathrm{PI}(g_h) \setminus \mathrm{PI}(g_l)\bigr) \setminus \bigl(\mathrm{PI}(f_h) \setminus \mathrm{PI}(f_l)\bigr)$. Но обе стороны выражения равны $\mathrm{PI}(g_h) \setminus (\mathrm{PI}(f_h) \cup \mathrm{PI}(g_l))$, поскольку $f_l \subseteq f_h \subseteq g_h$ и $f_l \subseteq g_l \subseteq g_h$.

[Это рекуррентное соотношение строит НДР непосредственно из БДР f и g и дает $\mathrm{PI}(g)$, когда $f = 0$. Таким образом, его проще реализовать, чем (137), где требуется также оператор разности множеств для НДР. Кроме того, на практике оно часто работает быстрее.]

255. (а) Типичный элемент α наподобие $e_2 \sqcup e_5 \sqcup e_6$ имеет очень простую НДР. Можно легко разработать подпрограмму BUMP, которая устанавливает $g \leftarrow g \oplus \alpha$ и возвращает $[\alpha \in g]$ для заданных НДР g и α.

Для вставки α в мультисемейство f начинаем с $k \leftarrow c \leftarrow 0$; затем, пока $c = 0$, устанавливаем $c \leftarrow \mathrm{BUMP}(f_k)$ и $k \leftarrow k+1$. Для удаления α, в предположении его наличия, начинаем с $k \leftarrow 0$ и $c \leftarrow 1$; пока $c = 1$, устанавливаем $c \leftarrow \mathrm{BUMP}(f_k)$ и $k \leftarrow k+1$.

(б) Предположим, что f_k и g_k представляют собой $\emptyset$ для $k \ge m$. Установить $k \leftarrow 0$ и $t \leftarrow \emptyset$ (НДР $\boxed{\perp}$). Пока $k < m$, устанавливать $h_k \leftarrow f_k \oplus g_k \oplus t$ и $t \leftarrow \langle f_k g_k t \rangle$. Наконец установить $h_m \leftarrow t$.

[Это представление и алгоритм вставки разработаны Ш. Минато (S. Minato) и Х. Аримура (H. Arimura), *Proc. Workshop, Web Information Retrieval and Integration* (IEEE, 2005), 4–11.]

256. (а) Зеркально отобразим бинарное представление слева направо и добавим нули, чтобы количество битов стало равным 2^n для некоторого n. Результат представляет собой таблицу истинности соответствующей булевой функции $f(x_1,\ldots,x_n)$ с x_k, соответствующим $2^{2^{n-k}} \in U$. Когда $x = 41$, например, 10010100 представляет собой таблицу истинности $(x_1 \wedge \bar{x}_2 \wedge x_3) \vee (\bar{x}_1 \wedge x_2 \wedge x_3) \vee (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3)$.

(б) Если $x < 2^{2^n}$, мы имеем $Z(x) \le U_n = O(2^n/n)$ согласно (79) и упр. 192.

(в) Имеется простая рекурсивная подпрограмма $\mathrm{ADD}(x,y,c)$, которая получает "бит переноса" c и указатели на НДР для x и y и возвращает указатель на НДР для $x+y+c$. Эта подпрограмма вызывается не более чем $4Z(x)Z(y)$ раз.

(г) Мы не можем утверждать, что $Z(x \dot{-} y) = O(Z(x)Z(y))$, поскольку $Z(x \dot{-} y) = n+1$ и $Z(x) = 3$ и $Z(y) = 1$ при $x = 2^{2^n}$ и $y = 1$. Но путем вычисления $x \dot{-} y = (x + 1 + ((2^{2^n} - 1) \oplus y)) - 2^{2^n}$ при $y \le x < 2^{2^n}$ можно показать, что $Z(x \dot{-} y) = O(Z(x)Z(y)\log\log x)$. (См. узлы t_j НДР в ответе к упр. 201.) Так что окончательный ответ — "да".

(д) Нет. Например, если $x = (2^{2^{2^k+k}} - 1)/(2^{2^k} - 1)$, мы имеем $Z(x) = 2^k + 1$, но $Z(x^2) = 3 \cdot (2^{2^k} - 1) = U_{2^k+k+1} - 2$, где U_{2^k+k+1} — наибольший возможный размер НДР для чисел с $\lg\lg x^2 < 2^k + k + 1$ (см. п. (б)).

[На это упражнение натолкнули исследования Жана Вюллемина (Jean Vuillemin), который начал эксперименты с такими разреженными целыми числами около 1993 года. К сожалению, важные в комбинаторных вычислениях числа — такие, как числа Фибоначчи, факториалы, биномиальные коэффициенты и т. п. — на практике редко оказываются разреженными.]

257. См. *Proc. Europ. Design and Test Conf.* (IEEE, 1995), 449–454. В случае знаковых коэффициентов можно использовать $\{-2, 4, -8, \ldots\}$ вместо $\{2, 4, 8, \ldots\}$, как в негабинарной арифметике.

[В частном случае, когда степень каждой переменной не превышает 1 и когда сложение выполняется по модулю 2, полиномы из этого упражнения эквивалентны мультилинейному представлению булевых функций (см. 7.1.1–(19)) и НДР эквивалентны "бинарным моментальным диаграммам" (binary moment diagrams — BMD). См. R. E. Bryant and Y.-A. Chen, *ACM/IEEE Design Automation Conf.* **32** (1995), 535–541.]

258. Если n нечетно, БДР должна зависеть от всех своих переменных, и их должно быть как минимум $\lceil \lg n \rceil$. Таким образом, $B(f) \ge \lceil \lg n \rceil + 2$ при $n > 1$, и худые функции из упр. 170, (в) достигают этой границы. Если n четно, добавим неиспользуемую переменную к решению для $n/2$.

Как легко видеть, вопрос об НДР эквивалентен поиску кратчайшей суммирующей цепочки, как в разделе 4.6.3. Таким образом, наименьшее значение $Z(f)$ для $|f| = n$ равно $l(n) + 1$, включая $\boxed{\top}$.

259. Теория вложенных скобок (см., например, упр. 2.2.1–3) говорит нам, что $N_n(x) = 1$ тогда и только тогда, когда $\bar{x}_1 + \cdots + \bar{x}_k \ge x_1 + \cdots + x_k$ для $0 \le k \le 2n$, причем равенство достигается при $k = 2n$. Или, что эквивалентно, $k - n \le x_1 + \cdots + x_k \le k/2$ для $0 \le k \le 2n$. Так что БДР для N_n во многом похожа на БДР для $S_n(x)$, но проще; фактически элементами профиля являются $b_k = \lfloor k/2 \rfloor + 1$ для $0 \le k \le n$ и $b_k = n + 1 - \lceil k/2 \rceil$ для $n \le k < 2n$. Следовательно, $B(N_n) = b_0 + \cdots + b_{2n-1} + 2 = \binom{n+2}{2} + 1$. z-профиль имеет $z_k = b_k - [k \text{ четно}]$ для $0 \le k < 2n$, поскольку ветви HI на четных уровнях ведут в $\boxed{\bot}$; следовательно, $Z(N_n) = B(N_n) - n$.

[Интересная база БДР для $n+1$ булевых функций, соответствующих C_{nn}, $C_{(n-1)(n+1)}$, $\ldots$, $C_{0(2n)}$ из 7.2.1.6–(21), может быть построена по аналогии с упр. 49.]

260. (а, б) Расположим переменные $x_{n,0}$, $x_{n,1}$, $\ldots$, $x_{n,n-1}$, $x_{n-1,0}$, $\ldots$, $x_{1,0}$ сверху вниз. Тогда ветвь HI из корня НДР для R_n представляет собой корень НДР R_{n-1}. (Это упорядочение, как оказывается, минимизирует $Z(R_n)$ для $n \le 6$ и, вероятно, также для всех n.) z-профиль имеет вид 1, $\ldots$, 1; $n-2$, $\ldots$, 2, 1, 1; $n-3$, $\ldots$, 2, 1, 1; $\ldots$; следовательно, $Z(R_n) = \binom{n}{3} + 2n + 1 \approx \frac{1}{6}n^3$ и $Z(R_{100}) = 161\,901$. Обычный профиль представляет собой 1, 2, 2, 3, 4, $\ldots$, $n-1$; $n-1$, $2n-4$, $2n-5$, $\ldots$, $n-1$; $n-2$, $2n-6$, $\ldots$, $n-2$; $\ldots$; в целом $B(R_n) = 3\binom{n}{3} + \binom{n+1}{2} + 3$ для $n \ge 2$ и $B(R_{100}) = 490\,153$.

[См. I. Semba and S. Yajima, *Trans. Inf. Proc. Soc. Japan* **35** (1994), 1666–1667. Кстати, метод из упр. 7.2.1.5–26 приводит к НДР для разбиений множества, которая имеет всего лишь $\binom{n}{2}$ переменных и $\binom{n}{2} + 1$ узлов. Но связь между этим представлением и разбиениями менее непосредственная; таким образом, внести ограничения естественным путем оказывается труднее.]

(в) Теперь при $n = 100$ имеется 573 переменные вместо 5050; количество переменных в общем случае равно $nl - 2^l + 1$, где $l = \lceil \lg n \rceil$ согласно 5.3.1–(3). Мы рассматриваем биты чисел a_n, a_{n-1}, $\ldots$, начиная со старшего бита. Тогда $B(R'_{100}) = 31\,861$, и можно показать, что $B(R'_n) = \binom{n}{2}l - \frac{1}{6}4^l - \frac{1}{2}2^l - \nu(n-1) + l + \frac{8}{3}$ для $n > 2$. Размер НДР более сложен, и оказывается больше примерно на 60%; мы имеем $Z(R'_{100}) = 50\,154$.

261. Для заданной булевой функции $f(x_1, \ldots, x_n)$ множество всех бинарных строк $x_1 \ldots x_n$, таких, что $f(x_1, \ldots, x_n) = 1$, является конечным языком, так что он является регулярным. Детерминированный автомат с минимальным количеством состояний $\mathcal{A}$ для этого языка представляет собой КДР для f. (В общем случае, когда L регулярен, состояние $\mathcal{A}$ после чтения $x_1 \ldots x_k$ принимает язык $\{\alpha \mid x_1 \ldots x_k \alpha \in L\}$.)

[Цитируемая теорема была открыта в более общем контексте Д. А. Хаффманом (D. A. Huffman), *Journal of the Franklin Institute* **257** (1954), 161–190, и независимо от него Э. Ф. Муром (E. F. Moore), *Annals of Mathematics Studies* **34** (1956), 129–153.]

Интересный пример связи между этой теорией и теорией БДР можно найти в ранних работах Юрия Брейтбарта, подытоженных в *Доклады Акад. наук СССР* **180** (1968), 1053–1055. Лемма 7 в статье Брейтбарта гласит, по сути, что $B_{\min}(\psi) = \Omega(2^{n/4})$, где ψ является функцией от $2n$ переменных $x = (x_1, \ldots, x_n)$, а $y = (y_1, \ldots, y_n)$ определяется с помощью $\psi(x, y) = x_{\nu y} \oplus y_{\nu x}$ с учетом того, что $x_0 = y_0 = 0$. (Заметим, что ψ представляет собой "двухстороннюю" скрыто взвешенную битовую функцию.)

262. (а) Если a обозначает функцию или подфункцию f, можно, например, использовать $C(a) = a \oplus 1$ для обозначения $\bar{f}$ в предположении, что каждый узел занимает четное число байтов. Тогда $C(C(a)) = a$ и связь с a обозначает не нормальную функцию тогда и только тогда, когда a нечетно; $a \mathbin{\&} -2$ всегда указывает на узел, который всегда представляет нормальную функцию.

Указатель LO каждого узла четный, поскольку нормальная функция остается нормальной при замене любой переменной нулем. Но указатель HI любого узла может быть дополнен; также может быть дополнен и внешний корневой указатель на любую функцию нормализованной базы БДР. Заметим, что сток $\boxed{\top}$ теперь невозможен.

(б) Уникальность очевидна из-за связи с таблицами истинности: бусина является либо нормальной (т. е. начинается с 0), либо дополнением нормальной бусины.

(в) На приведенных диаграммах каждая дополняющая связь указана пунктиром.

(г) Имеется $2^{2^m-1} - 2^{2^{m-1}-1}$ нормальных бусин порядка m. Наихудший случай, $B^0(f) \le B^0(f_n) = 1 + \sum_{k=0}^{n-1} \min(2^k, 2^{2^{n-k}-1} - 2^{2^{n-k-1}-1}) = (U_{n+1} - 1)/2$, осуществляется для

функций из ответа к упр. 110. Чтобы найти средний нормализованный профиль, заменим $2^{2^{n-k}}-1$ в (80) на $2^{2^{n-k}}-2$ и разделим всю формулу на 2; средний случай оказывается очень близким к наихудшему. Например, вместо (81) мы имеем

$$(1.0, 2.0, 4.0, 8.0, 16.0, 32.0, 64.0, 127.3, 103.9, 6.0, 1.0, 1.0).$$

(д) Мы экономим $\boxed{\top}$, один 6, два 5 и три 4, оставляя 45 нормализованных узлов.

(е) Вероятно, лучше всего иметь подпрограммы И, ИЛИ, НО-НЕ для случая, когда известно, что f и g являются нормальными, вместе с подпрограммой GAND для общего случая. Подпрограмма GAND(f,g) возвращает И(f,g), если f и g четны, НО-НЕ$(f,C(g))$, если f четно, но g нечетно, НО-НЕ$(g,C(f))$, если g четно, но f нечетно, C(ИЛИ$(C(f),C(g)))$, если f и g нечетны. Подпрограмма И(f,g) подобна (55) за исключением того, что $r_h \leftarrow$ GAND(f_h,g_h); только случаи $f=0$, $g=0$ и $f=g$ должны быть протестированы как "очевидные" значения.

Примечания. Дополняющие связи были предложены С. Акерсом (S. Akers) в 1978 году и независимо от него Ж. П. Биллоном (J. P. Billon) в 1987 году. Хотя такие связи использовались во всех основных пакетах БДР, их трудно рекомендовать, так как компьютерные программы становятся слишком сложными. Экономия памяти обычно получается незначительной, и никогда не более чем в два раза; кроме того, эксперименты автора демонстрируют малый выигрыш и во времени работы.

При использовании НДР вместо БДР "нормальное семейство" функций представляет собой семейство, не содержащее пустого множества. Шин-ичи Минато (Shin-ichi Minato) предложил использовать при работе с НДР обозначение $C(a)$ вместо $\bar{f}$ для семейства $f\oplus\epsilon$.

263. (а) Если $Hx=0$ и $x\neq 0$, у нас не может быть $\nu x=1$ или 2, поскольку столбцы H ненулевые и различные. [R. W. Hamming, *Bell System Tech. J.* **29** (1950), 147–160.]

(б) Пусть r_k представляет собой ранг первых k столбцов H, а s_k — ранг последних k столбцов. Тогда $b_k=2^{r_k+s_{n-k}-r_n}$ для $0\le k<n$, поскольку это количество элементов в пересечении векторных пространств, охватываемых первыми k и последними $n-k$ столбцами. В коде Хэмминга $r_k=1+\lambda k$ и $s_k=\min(m,2+\lambda(k-1))$ для $k>1$; так что мы находим $B(f)=(n^2+5)/2$. [См. G. D. Forney, Jr., *IEEE Trans.* **IT-34** (1988), 1184–1187.]

(в) Пусть $q_k=1-p_k$. Максимизация $\prod_{k=1}^n p_k^{[x_k=y_k]}q_k^{[x_k\neq y_k]}$ такова же, как и максимизация $\sum_{k=1}^n w_kx_k$, где $w_k=(2y_k-1)\log(p_k/q_k)$, так что мы можем использовать алгоритм В.

Примечания. Ученые в области теории кодов, начиная с неопубликованных работ Форни (Forney) в 1967 году, разработали идею так называемой *решетки* (*trellis*) кода. В бинарном случае решетка представляет собой то же самое, что и КДР для f, но все узлы для константной подфункции 0 из нее удалены. (Полезные коды имеют расстояние >1; в этом случае решетка также представляет собой БДР для f, но с удаленным $\boxed{\bot}$.) Изначально Форни хотел показать, что декодирующий алгоритм А. Витерби (A. Viterbi) [*IEEE Trans.* **IT-13** (1967), 260–269] оптимален для сверточных кодов. Несколько лет спустя в работе L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, *IEEE Trans.* **IT-20** (1974), 284–287, структура решетки была расширена до линейных блочных кодов и были представлены более оптимальные алгоритмы. См. также статьи G. B. Horn and F. R. Kschischang, *IEEE Trans.* **IT-42** (1996), 2042–2048; J. Lafferty and A. Vardy, *IEEE Trans.* **C-48** (1999), 971–986.

264. Процедуры, объединяющие "восходящие" методы алгоритма В с "нисходящими" методами, которые выполняют оптимизацию над предшественниками узла, могут оказаться более эффективными по сравнению с методами, работающими строго в одном направлении.

265. Вычислим количества c_j в восходящем направлении, как в алгоритме С, с использованием n-битовой арифметики. Затем будем работать в нисходящем направлении, начиная с $k\leftarrow s-1$, $j\leftarrow 1$, $m\leftarrow m-1$ и повторяя следующие шаги (во время которых мы

будем иметь $0 \le m < 2^{v_k-j}c_k$): если $v_k > j$, установить $x_j \leftarrow \lfloor m/2^{v_k-j-1}c_k \rfloor$, $m \leftarrow m \bmod 2^{v_k-j-1}c_k$, $j \leftarrow j+1$; в противном случае при $k=1$ завершить работу алгоритма; в противном случае установить $l \leftarrow l_k$, $h \leftarrow h_k$ и, если $m < 2^{v_l-v_k-1}c_l$, установить $x_j \leftarrow 0$, $k \leftarrow l$, $j \leftarrow j+1$; в противном случае установить $x_j \leftarrow 1$, $m \leftarrow m - 2^{v_l-v_k-1}c_l$, $k \leftarrow h$, $j \leftarrow j+1$.

266. Фактически НДР получается непосредственно из стандартного бинарного дерева "левый сын, правый брат" для F (см. 7.2.1.6–(4)), если использовать левую дочернюю связь для HI, и правую "братскую" связь для LO; нулевые связи изменяются таким образом, чтобы указывать на $\boxed{\top}$, за исключением того, что связь LO корня крайнего справа дерева (последнего узла в обратном порядке обхода) должна указывать на $\boxed{\bot}$.

267. Размер НДР для $d(F)$ может быть вычислен следующим образом с использованием вспомогательной функции $\zeta(T)$, рекурсивно определяемой на деревьях: если $|T|=1$ (т. е. если T имеет только один узел), $\zeta(T)=1$. В противном случае T состоит из корня вместе с $k \ge 1$ поддеревьями, $T_1, \ldots, T_k$, и мы определяем $\zeta(T) = 1+\zeta(T_1)+\cdots+\zeta(T_k)+|T|-|T_k|$. Тогда, если F состоит из $k \ge 1$ деревьев $T_1, \ldots, T_k$, мы имеем $Z(d(F)) = 1+\zeta(T_1)+\cdots+\zeta(T_k)+[|T_k|=1]$.

Минимальный размер, n, очевидно, получается, когда F состоит из n одноузловых деревьев. Максимальный размер, $\lfloor n^2/4 \rfloor + 2n + 1$, достигается для $n = 2m-1$ в дереве, в котором узел k имеет два дочерних узла, $k+1$ и $k+m$ для $1 \le k < m$; случай $n = 2m$ аналогичен.

Для того чтобы найти средний размер, рассмотрим производящую функцию $Z(w,z) = \sum w^{\zeta(T)} z^{|T|}$ с суммированием по всем деревьям T. Определение ζ дает функциональное уравнение $Z(w,z) = wz + w^2 z Z(w,z)/(1-Z(w,wz))$. Дифференцирование по w и по z с последующей установкой $w=1$ говорит нам, что $Z(1,z) = (1-s)/2$, $Z_w(1,z) = z/s + z/s^2$ и $Z_z(1,z) = 1/s$, где $s = \sqrt{1-4z}$. Производящая функция $\sum_F w^{Z(d(F))} z^{|F|}$ с суммированием по всем непустым лесам F равна $(wZ(w,z) + w^3 z - w^2 z)/(1-Z(w,z))$. Дифференцируя по w и устанавливая $w \leftarrow 1$, мы получаем $z/(1-4z) + 2z/\sqrt{1-4z}$; следовательно, среднее значение $Z(d(F))$ равно

$$(4^{n-1} + 2nC_{n-1})/C_n = \tfrac{1}{4}\sqrt{\pi}n^{3/2} + \tfrac{n}{2} + O(n^{1/2}),$$

где C_n — число Каталана (7.2.1.6–(15)).

РАЗДЕЛ 7.2.1.1

1. Пусть $m_j = u_j - l_j + 1$, и в алгоритме M вместо $(a_1, \ldots, a_n)$ посетим $(a_1+l_1, \ldots, a_n+l_n)$. Или заменим в этом алгоритме '$a_j \leftarrow 0$' на '$a_j \leftarrow l_j$' и '$a_j = m_j - 1$' на '$a_j = u_j$' и установим $l_0 \leftarrow 0$, $u_0 \leftarrow 1$ на шаге M1.

2. $(0,0,1,2,3,0,2,7,0,9)$.

3. Шаг M4 выполняется $m_1 m_2 \ldots m_k$ раз при $j=k$; следовательно, общее количество равно $\sum_{k=0}^{n} \prod_{j=1}^{k} m_j = m_1 \ldots m_n(1 + 1/m_n + 1/m_n m_{n-1} + \cdots + 1/m_n \ldots m_1)$. Если все m_j не меньше 2, это значение меньше $2m_1 \ldots m_n$. [Таким образом, мы должны иметь в виду, что методы на основе кодов Грея, которые изменяют при каждом посещении только одну цифру, в действительности уменьшают общее количество изменений цифр не более чем в два раза.]

4. **N1.** [Инициализация.] Установить $a_j \leftarrow m_j - 1$ для $0 \le j \le n$, где $m_0 = 2$.

N2. [Посещение.] Посетить n-кортеж $(a_1, \ldots, a_n)$.

N3. [Подготовка к вычитанию 1.] Установить $j \leftarrow n$.

N4. [Заем при необходимости.] Если $a_j = 0$, установить $a_j \leftarrow m_j - 1$, $j \leftarrow j-1$, и повторить данный шаг.

N5. [Уменьшение или завершение.] Если $j = 0$, завершить работу алгоритма. В противном случае установить $a_j \leftarrow a_j - 1$ и вернуться к шагу N2. ▌

5. Обращение битов легко выполняется на машине наподобие MMIX, но на других компьютерах эту задачу можно решить следующим образом.

Z1. [Инициализация.] Установить $j \leftarrow k \leftarrow 0$.

Z2. [Обмен.] Обменять $A[j+1] \leftrightarrow A[k+2^{n-1}]$. Кроме того, если $j < k$, обменять $A[j] \leftrightarrow A[k]$ и $A[j+2^{n-1}+1] \leftrightarrow A[k+2^{n-1}+1]$.

Z3. [Продвижение k.] Установить $k \leftarrow k+2$ и завершить работу алгоритма, если $k \geq 2^{n-1}$.

Z4. [Продвижение j.] Установить $h \leftarrow 2^{n-2}$. Если $j \geq h$, многократно устанавливать $j \leftarrow j-h$ и $h \leftarrow h/2$ до тех пор, пока не будет выполнено условие $j < h$. Затем установить $j \leftarrow j+h$. (Теперь $j = (b_0 \ldots b_{n-1})_2$, если $k = (b_{n-1} \ldots b_0)_2$.) Вернуться к шагу Z2. ▌

6. Если $g\big((0b_{n-1}\ldots b_1b_0)_2\big) = \big(0(b_{n-1})\ldots(b_2 \oplus b_1)(b_1 \oplus b_0)\big)_2$, то $g\big((1b_{n-1}\ldots b_1b_0)_2\big) = 2^n + g\big((0\overline{b}_{n-1}\ldots\overline{b}_1\overline{b}_0)_2\big) = \big(1(\overline{b}_{n-1})\ldots(\overline{b}_2 \oplus \overline{b}_1)(\overline{b}_1 \oplus \overline{b}_0)\big)_2$, где $\overline{b} = b \oplus 1$.

7. Для $2r$ секторов можно использовать $g(k)$ для $2^n - r \leq k < 2^n + r$, где $n = \lceil \lg r \rceil$, поскольку $g(2^n - r) \oplus g(2^n + r - 1) = 2^n$ согласно (5). [G. C. Tootill, *Proc. IEE* **103**, Part B Supplement (1956), 434.] См. также упр. 26.

8. Воспользуйтесь алгоритмом G с $n \leftarrow n-1$ и добавьте справа бит четности a_∞. (Это дает $g(0)$, $g(2)$, $g(4)$, $\ldots$.)

9. Одеть крайнее справа кольцо, поскольку $\nu(1011000)$ нечетно.

10. $A_n + B_n = g^{[-1]}(2^n - 1) = \lfloor 2^{n+1}/3 \rfloor$ и $A_n = B_n + n$. Следовательно, $A_n = \lfloor 2^n/3 + n/2 \rfloor$ и $B_n = \lfloor 2^n/3 - n/2 \rfloor$.

Историческая справка. Японский математик Ёриюки Арима (Yoriyuki Arima) (1714–1783) рассмотрел эту задачу в своей работе *Shūki Sanpō* (1769), задача 44, заметив, что головоломка с n кольцами сводится к головоломке с $(n-1)$ кольцами после определенного ряда шагов. Пусть $C_n = A_n - A_{n-1} = B_n - B_{n-1} + 1$ — количество снятий колец в процессе такого приведения. Арима заметил, что $C_n = 2C_{n-1} - [n \text{ четно}]$; таким образом, он мог бы вычислить $A_n = C_1 + C_2 + \cdots + C_n$ для $n = 9$ без знания формулы $C_n = \lceil 2^{n-1}/3 \rceil$.

Более чем двумя столетиями ранее Кардано (Cardano) уже упоминал "complicati annuli"* в своей *De Subtilitate Libri XXI* (Nuremberg: 1550), Book 15. Он считал их "бесполезными и замечательно неуловимыми", ошибочно указав, что для снятия семи колец достаточно 95 шагов, и еще 95, чтобы вернуть их на место. Джон Уоллис (John Wallis) посвятил этой головоломке семь страниц в латинском издании своей книги *Algebra* **2** (Oxford: 1693), Chapter 111, представив детально описанные, но не очень эффективные методы решения головоломки с девятью кольцами. Он добавил к операциям снятия и одевания кольца операцию его перемещения по стержню и указал, что возможны улучшения описанного метода, но найти наилучшее решение не пытался.

11. Решением соотношения $S_n = S_{n-2} + 1 + S_{n-2} + S_{n-1}$ при $S_1 = S_2 = 1$ является $S_n = 2^{n-1} - [n \text{ четно}]$. [*Math. Quest. Educational Times* **3** (1865), 66–67.]

12. (a) Теория $n-1$ китайских колец доказывает, что бинарный код Грея дает композицию в удобном порядке (4, 31, 211, 22, 112, 1111, 121, 13).

C1. [Инициализация.] Установить $t \leftarrow 0$, $j \leftarrow 1$, $s_1 \leftarrow n$. (Считаем, что $n > 1$.)

C2. [Посещение.] Посетить $s_1 \ldots s_j$. Затем установить $t \leftarrow 1 - t$ и перейти к шагу C4, если $t = 0$.

---

* Сложные отмены (итал.). — *Примеч. пер.*

C3. [Нечетный шаг.] Если $s_j > 1$, установить $s_j \leftarrow s_j - 1$, $s_{j+1} \leftarrow 1$, $j \leftarrow j + 1$; в противном случае установить $j \leftarrow j - 1$ и $s_j \leftarrow s_j + 1$. Вернуться к шагу C2.

C4. [Четный шаг.] Если $s_{j-1} > 1$, установить $s_{j-1} \leftarrow s_{j-1} - 1$, $s_{j+1} \leftarrow s_j$, $s_j \leftarrow 1$, $j \leftarrow j + 1$; в противном случае установить $j \leftarrow j - 1$, $s_j \leftarrow s_{j+1}$, $s_{j-1} \leftarrow s_{j-1} + 1$ (но завершить работу алгоритма, если $j - 1 = 0$). Вернуться к шагу C2. ▌

(б) Теперь $q_1, \ldots, q_{t-1}$ представляют кольца на стержне.

B1. [Инициализация.] Установить $t \leftarrow 1$, $q_0 \leftarrow n$. (Считаем, что $n > 1$.)

B2. [Посещение.] Установить $q_t \leftarrow 0$ и посетить $(q_0 - q_1) \ldots (q_{t-1} - q_t)$. Перейти к шагу B4, если t четно.

B3. [Нечетный шаг.] Если $q_{t-1} = 1$, установить $t \leftarrow t - 1$; в противном случае установить $q_t \leftarrow 1$ и $t \leftarrow t + 1$. Вернуться к шагу B2.

B4. [Четный шаг.] Если $q_{t-2} = q_{t-1} + 1$, установить $q_{t-2} \leftarrow q_{t-1}$ и $t \leftarrow t - 1$ (но завершить работу алгоритма, если $t = 2$); в противном случае установить $q_t \leftarrow q_{t-1}$, $q_{t-1} \leftarrow q_t + 1$, $t \leftarrow t + 1$. Вернуться к шагу B2. ▌

Эти алгоритмы [см. J. Misra, *ACM Trans. Math. Software* **1** (1975), 285] не содержат циклов даже на этапе инициализации.

13. На шаге C1 установите также $C \leftarrow 1$. На шаге C3 установите $C \leftarrow s_j C$, если $s_j > 1$, в противном случае $C \leftarrow C/(s_{j-1} + 1)$. На шаге C4 установите $C \leftarrow s_{j-1} C$, если $s_{j-1} > 1$, в противном случае $C \leftarrow C/(s_{j-2} + 1)$. Аналогичные изменения следует внести и в шаги B1, B3, B4. В случае значения $C = n!$ для композиции $1 \ldots 1$ требуется определенное расширение понятия алгоритма без циклов; примем, что арифметические операции выполняются за единичное время.

14. **V1.** [Инициализация.] Установить $j \leftarrow 0$.

V2. [Посещение.] Посетить строку $a_1 \ldots a_j$.

V3. [Удлинение.] Если $j < n$, установить $j \leftarrow j + 1$, $a_j \leftarrow 0$ и вернуться к шагу V2.

V4. [Увеличение.] Если $a_j < m_j - 1$, установить $a_j \leftarrow a_j + 1$ и вернуться к шагу V2.

V5. [Сокращение.] Установить $j \leftarrow j - 1$ и вернуться к шагу V4, если $j > 0$. ▌

15. **J1.** [Инициализация.] Установить $j \leftarrow 0$.

J2. [Четное посещение.] Если j четно, посетить строку $a_1 \ldots a_j$.

J3. [Удлинение.] Если $j < n$, установить $j \leftarrow j + 1$, $a_j \leftarrow 0$ и вернуться к шагу J2.

J4. [Нечетное посещение.] Если j нечетно, посетить строку $a_1 \ldots a_j$.

J5. [Увеличение.] Если $a_j < m_j - 1$, установить $a_j \leftarrow a_j + 1$ и вернуться к шагу J2.

J6. [Сокращение.] Установить $j \leftarrow j - 1$ и вернуться к шагу J4, если $j > 0$. ▌

Этот алгоритм не содержит циклов, хотя, на первый взгляд, может показаться, что это не так. Последовательные посещения разделяют не более четырех шагов. Основная идея связана с упр. 2.3.1–5 и прямо-обратным обходом (алгоритм 7.2.1.6Q).

16. Предположим, что $\texttt{LINK}(j-1) = j + nb_j$ для $1 \le j \le n$ и $\texttt{LINK}(j-1+n) = j + n(1-b_j)$ для $1 < j \le n$. Эти связи представляют $(a_1, \ldots, a_n)$ тогда и только тогда, когда $g(a_1 \ldots a_n) = b_1 \ldots b_n$, так что для получения требуемого результата можно воспользоваться бесцикловым генератором бинарного кода Грея.

17. Поместите на пересечении j-й строки и k-го столбца $(0 \le j, k < 8)$ конкатенацию трехбитовых кодов $\big(g(j), g(k)\big)$. [Нетрудно доказать, что это, по сути, *единственное* решение, за исключением перестановок и/или дополнения позиций координат, и/или циклических сдвигов строк, поскольку координаты, изменяющиеся при перемещении по вертикали,

зависят только от строк (аналогичное утверждение применимо и к столбцам). Изоморфизм Карно между четырехмерным кубом и тором 4×4 описан в *The Design of Switching Circuits* by W. Keister, A. E. Ritchie, and S. H. Washburn (1951), page 174. Кстати, Кейстер (Keister) придумал интересный вариант "Китайских колец" под названием *SpinOut*, а также его обобщение *The Hexadecimal Puzzle*, *U.S. Patents 3637215–3637216* (1972).]

18. Воспользуйтесь двухбитовым кодом Грея для представления цифр $u_j = (0, 1, 2, 3)$ соответственно как битовых пар $u'_{2j-1}u'_{2j} = (00, 01, 11, 10)$. [Ч. Ю. Ли (C. Y. Lee) предложил свою метрику в *IRE Trans.* **IT-4** (1958), 77–82. Аналогичная $m/2$-битовая кодировка работает для четных значений m; например, при $m = 8$ можно представить $(0, 1, 2, 3, 4, 5, 6, 7)$ в виде $(0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000)$. Но такая схема оставляет в стороне некоторые из бинарных шаблонов при $m > 4$.]

19. (а) Алгоритм модульного кватернарного кода Грея требует чуть меньше вычислений, чем алгоритм M, но это не имеет значения в силу малости числа 256. Результат имеет вид $z_0^8 + z_1^8 + z_2^8 + z_3^8 + 14(z_0^4 z_2^4 + z_1^4 z_3^4) + 56 z_0 z_1 z_2 z_3 (z_0^2 + z_2^2)(z_1^2 + z_3^2)$.

(б) Замена (z_0, z_1, z_2, z_3) на $(1, z, z^2, z)$ дает $1+112z^6+30z^8+112z^{10}+z^{16}$; таким образом, все ненулевые веса Ли ≥ 6. Теперь можно воспользоваться этой конструкцией в предыдущем упражнении для преобразования каждого вектора $(u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_\infty)$ в 16-битовое число.

20. Восстановите кватернарный вектор $(u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_\infty)$ из u' и используйте алгоритм 4.6.1D для поиска остатка от деления $u_0+u_1x+\cdots+u_6x^6$ на $g(x)$ по модулю 4; этот алгоритм можно использовать, несмотря на тот факт, что коэффициенты не принадлежат полю, поскольку $g(x)$ нормирован. Выразим остаток как $x^j + 2x^k$ (по модулю $g(x)$ и 4) и положим $d = (k - j) \bmod 7$, $s = (u_0 + \cdots + u_6 + u_\infty) \bmod 4$.

Случай 1. $s = 1$. Если $k = \infty$, то ошибка — x^j (другими словами, корректный вектор содержит $u_j \leftarrow (u_j - 1) \bmod 4$); в противном случае имеется три или более ошибок.

Случай 2. $s = 3$. Если $j = k$, то ошибка — $-x^j$; в противном случае имеется ≥ 3 ошибок.

Случай 3. $s = 0$. Если $j = k = \infty$, ошибок нет; если $j = \infty$ и $k < \infty$, имеется как минимум четыре ошибки. В противном случае ошибками являются $x^a - x^b$, где $a = \big(j + (\infty, 6, 5, 2, 3, 1, 4, 0)\big) \bmod 7$, и соответственно $d = (0, 1, 2, 3, 4, 5, 6, \infty)$ и $b = (j+2d) \bmod 7$.

Случай 4. $s = 2$. Если $j = \infty$, то ошибки — $2x^k$. В противном случае ошибками являются

$$\begin{aligned}
x^j + x^\infty&, \text{ если } k = \infty;\\
-x^j - x^\infty&, \text{ если } d = 0;\\
x^a + x^b&, \text{ если } d \in \{1, 2, 4\},\ a = (j - 3d) \bmod 7,\ b = (j - 2d) \bmod 7;\\
-x^a - x^b&, \text{ если } d \in \{3, 5, 6\},\ a = (j - 3d) \bmod 7,\ b = (j - d) \bmod 7.
\end{aligned}$$

Для заданного $u' = (1100100100001111)_2$ мы имеем $u = (2, 0, 3, 1, 0, 0, 2, 2)$ и $2 + 3x^2 + x^3 + 2x^6 \equiv 1 + 3x + 3x^2 \equiv x^5 + 2x^6$; а также $s = 2$. Таким образом, ошибками являются $x^2 + x^3$, а ближайшее безошибочное кодовое слово — $(2, 0, 2, 0, 0, 0, 2, 2)$. Алгоритм 4.6.1D говорит нам о том, что $2 + 2x^2 + 2x^6 \equiv (2 + 2x + 2x^3)g(x)$ (по модулю 4); так что восемь информационных битов соответствуют $(v_0, v_1, v_2, v_3) = (2, 2, 0, 2)$. [Более интеллектуальный алгоритм мог бы заметить, что это первые 16 бит числа π.]

Об обобщениях других эффективных схем кодирования на основе кватернарных векторов можно прочесть в классической статье Hammons, Kumar, Calderbank, Sloane, and Solé, *IEEE Trans.* **IT-40** (1994), 301–319.

21. (а) $C(\epsilon) = 1$, $C(0\alpha) = C(1\alpha) = C(\alpha)$ и $C(*\alpha) = 2C(\alpha) - [10\ldots0 \in \alpha]$. Итерирование этой последовательности дает $C(\alpha) = 2^t - 2^{t-1}e_t - 2^{t-2}e_{t-1} - \cdots - 2^0 e_1$, где $e_j = [10\ldots0 \in \alpha_j]$, а α_j является суффиксом α, следующим за j-й звездочкой. В приведенном примере $\alpha_1 =$

$*10**0*$, $\alpha_2 = 10**0*$, $\ldots$, $\alpha_5 = \epsilon$; таким образом, $e_1 = 0$, $e_2 = 1$, $e_3 = 1$, $e_4 = 0$ и $e_5 = 1$ (по соглашению), следовательно, $C(**10**0*) = 2^5 - 2^4 - 2^2 - 2^1 = 10$.

(б) Можно удалить завершающие звездочки, так что $t = t'$. Тогда из $e_t = 1$ вытекает $e_{t-1} = \cdots = e_1 = 0$. [Случай $C(\alpha) = 2^{t'-1}$ осуществляется тогда и только тогда, когда α заканчивается $10^j{*}^k$.]

(в) Чтобы вычислить сумму $C(\alpha)$ по всем t-подкубам, заметим, что $\binom{n}{t}$ кластеров начинаются с n-кортежа $0\ldots0$, а $\binom{n-1}{t}$ кластеров — с последующего n-кортежа (а именно по одному кластеру для каждого t-подкуба, содержащего этот n-кортеж и определяющего изменяемый бит). Таким образом, среднее равно $\bigl(\binom{n}{t} + (2^n - 1)\binom{n-1}{t}\bigr)/2^{n-t}\binom{n}{t} = 2^t(1 - t/n) + 2^{t-n}(t/n)$. [Формула в (в) справедлива для *любого* n-битового пути Грея, но (а) и (б) характерны только для рефлексивного бинарного кода Грея. Эти результаты получены в работе C. Faloutsos, *IEEE Trans.* **SE-14** (1988), 1381–1393.]

22. Пусть $\alpha{*}^j$ и $\beta{*}^k$ являются последовательными листьями бинарного луча Грея, где α и β представляют собой бинарные строки, и $j \le k$. Тогда последние $k - j$ бит α представляют собой строку α', такую, что α и $\beta\alpha'$ — последовательные элементы бинарного кода Грея, а следовательно, смежны. [Интересные приложения этого свойства к кубически связанным (cube-connected) параллельным компьютерам со связью путем передачи сообщений рассматриваются в William J. Dally, *A VLSI Architecture for Concurrent Data Structures* (Kluwer, 1987), глава 3.]

23. Из $2^j = g(k) \oplus g(l) = g(k \oplus l)$ вытекает, что $l = k \oplus g^{[-1]}(2^j) = k \oplus (2^{j+1} - 1)$. Другими словами, если $k = (b_{n-1}\ldots b_0)_2$, то $l = (b_{n-1}\ldots b_{j+1}\overline{b}_j\ldots\overline{b}_0)_2$.

24. Определяя $g(k) = k \oplus \lfloor k/2 \rfloor$ как обычно, найдем $g(k) = g(-1-k)$; следовательно, существуют *два* таких 2-адических целых числа k, что $g(k)$ имеет заданное 2-адическое значение l. Одно из них четно, второе — нечетно. Удобнее определить, что $g^{[-1]}$ — четное решение; тогда (8) заменяется на $b_j = a_{j-1} \oplus \cdots \oplus a_0$ для $j \ge 0$. Например, согласно этому определению $g^{[-1]}(1) = -2$; если l — обычное целое число, то "знак" $g^{[-1]}(l)$ — четность l.

25. Пусть $p = k \oplus l$; из упр. 7.1.3–3 известно, что $2^{\lfloor \lg p \rfloor + 1} - p \le |k - l| \le p$. Имеем $\nu(g(p)) = \nu(g(k) \oplus g(l)) = t$ тогда и только тогда, когда существуют натуральные числа $j_1, \ldots, j_t$, такие, что $p = (1^{j_1}0^{j_2}1^{j_3}\ldots(0 \text{ or } 1)^{j_t})_2$. Наибольшее возможное число $p < 2^n$ получается при $j_1 = n + 1 - t$ и $j_2 = \cdots = j_t = 1$, достигая $p = 2^n - \lceil 2^t/3 \rceil$. Наименьшее возможное число $q = 2^{\lfloor \lg p \rfloor + 1} - p = (1^{j_2}0^{j_3}\ldots(1 \text{ or } 0)^{j_t})_2 + 1$ получается при $j_2 = \cdots = j_t = 1$, достигая $q = \lceil 2^t/3 \rceil$. [C. K. Yuen, *IEEE Trans.* **IT-20** (1974), 668; S. R. Cavior, *IEEE Trans.* **IT-21** (1975), 596. Аналогичная граница для модульного m-арного кода Грея равна $\lceil m^t/(m^2 - 1) \rceil$, и эта формула справедлива и для рефлексивного m-арного кода Грея при четном m; см. van Zanten and Suparta, *IEEE Trans.* **IT-49** (2003), 485–487; *Proc. South East Asian Math. Soc. Conf.* (Yogyakarta: Gadjah Mada University, 2003), 98–105.]

26. Пусть $N = 2^{n_t} + \cdots + 2^{n_1}$, где $n_t > \cdots > n_1 \ge 0$; пусть также Γ_n — произвольный код Грея для $\{0, 1, \ldots, 2^n - 1\}$, который начинается с нуля и заканчивается единицей, за исключением Γ_0, просто равного нулю. Используйте

$$\Gamma^R_{n_t},\ 2^{n_t} + \Gamma_{n_{t-1}},\ \ldots,\ 2^{n_t} + \cdots + 2^{n_3} + \Gamma^R_{n_2},\ 2^{n_t} + \cdots + 2^{n_2} + \Gamma_{n_1},\ \text{если } t \text{ четно};$$
$$\Gamma_{n_t},\ 2^{n_t} + \Gamma^R_{n_{t-1}},\ \ldots,\ 2^{n_t} + \cdots + 2^{n_3} + \Gamma^R_{n_2},\ 2^{n_t} + \cdots + 2^{n_2} + \Gamma_{n_1},\ \text{если } t \text{ нечетно}.$$

27. В общем случае, если $k = (b_{n-1}\ldots b_0)_2$, $(k+1)$-й наибольший элемент S_n равен

$$1/\bigl(2 - (-1)^{a_{n-1}}/\bigl(2 - \cdots/(2 - (-1)^{a_1}/(2 - (-1)^{a_0}))\ldots\bigr)\bigr)$$

со структурой знаков $g(k) = (a_{n-1}\ldots a_0)_2$. Таким образом, по рангу можно вычислить любой элемент S_n за $O(n)$ шагов. Задав $k = 2^{100} - 10^{10}$ и $n = 100$, мы получим ответ

373065177/1113604409. [Всякий раз, когда $f(x)$ является положительной монотонной функцией, 2^n элементов $f(\pm f(\ldots \pm f(\pm x)\ldots))$ упорядочены в соответствии с бинарным кодом Грея, как показал H. E. Salzer, *CACM* **16** (1973), 180. В данном частном случае, однако, ответ можно получить и другим путем, поскольку мы также имеем $S_n = //2, \pm 2, \ldots, \pm 2, \pm 1//$ (используя обозначения из раздела 4.5.3). Непрерывная дробь в этом виде упорядочена дополняющими чередующимися битами k.]

28. (а) Так как $t = 1, 2, \ldots,$ бит a_j медианы (G_t) пробегает периодическую последовательность

$$0, \ldots, 0, *, 1, \ldots, 1, *, 0, \ldots, 0, *, \ldots$$

со звездочками на каждом 2^{1+j}-м шаге. Таким образом, строки, соответствующие двоичному представлению $\lfloor (t-1)/2 \rfloor$ и $\lfloor t/2 \rfloor$, являются медианами. И эти строки в действительности являются "экстремальными" в том смысле, что все медианы согласуются с общими битами $\lfloor (t-1)/2 \rfloor$ и $\lfloor t/2 \rfloor$, следовательно, звездочки встречаются там, где согласования нет. Например, при $t = 100 = (01100100)_2$ и $n = 8$ мы получаем медиану G_{100}, равную $001100**$.

(б) Поскольку $G_{2t} = 2G_t \cup (2G_t + 1)$, можно считать, что $t = (a_{n-2} \ldots a_1 a_0 1)_2$ нечетно. Если α равно $g(p)$, а β равно $g(q)$ в бинарном коде Грея, то мы имеем $p = (p_{n-1} \ldots p_0)_2$ и $q = (p_{n-1} \ldots p_{j+1} \overline{p}_j \ldots \overline{p}_0)_2$; а $a_{n-1} a_{n-2} = 01 = p_{n-1} p_{n-2}$. У нас не может быть $p < t \le q$, поскольку это повлекло бы за собой $j = n-1$ и $p_{n-3} = p_{n-4} = \cdots = p_0 = 1$. [См. A. J. Bernstein, K. Steiglitz, and J. E. Hopcroft, *IEEE Trans.* **IT-12** (1966), 425–430.]

29. В предположении, что $p \ne 0$, обозначим $l = \lfloor \lg p \rfloor$ и $S_a = \{s \mid 2^l a \le s < 2^l (a+1)\}$ для $0 \le a < 2^{n-l}$. Тогда $(k \oplus p) - k$ имеет постоянный знак для всех $k \in S_a$ и

$$\sum_{k \in S_a} \Big| (k \oplus p) - k \Big| = 2^l |S_a| = 2^{2l}.$$

Кроме того, $g^{[-1]}(g(k) \oplus p) = k \oplus g^{[-1]}(p)$ и $\lfloor \lg g^{[-1]}(p) \rfloor = \lfloor \lg p \rfloor$. Следовательно,

$$\frac{1}{2^n} \sum_{k=0}^{2^n-1} \Big| g^{[-1]}(g(k) \oplus p) - k \Big| = \frac{1}{2^n} \sum_{a=0}^{2^{n-l}-1} \sum_{k \in S_a} \Big| (k \oplus g^{[-1]}(p)) - k \Big| = \frac{1}{2^n} \sum_{a=0}^{2^{n-l}-1} 2^{2l} = 2^l.$$

[См. Morgan M. Buchner, Jr., *Bell System Tech. J.* **48** (1969), 3113–3130.]

30. Цикл, в котором содержится $k > 1$, имеет длину $2^{\lfloor \lg \lg k \rfloor + 1}$, поскольку из уравнения (7) легко показать, что если $k = (b_{n-1} \ldots b_0)_2$, то

$$g^{[2^l]}(k) = (c_{n-1} \ldots c_0)_2, \qquad \text{где } c_j = b_j \oplus b_{j+2^l}.$$

Чтобы переставить все элементы k, такие, что $\lfloor \lg k \rfloor = t$, есть два варианта: если t является степенью 2, цикл, содержащий $2\lfloor k/2 \rfloor$, содержит также и $2\lfloor k/2 \rfloor + 1$, так что для $t-1$ нужно удвоить ведущие элементы цикла. В противном случае цикл, содержащий $2\lfloor k/2 \rfloor$, не пересекается с циклом, содержащим $2\lfloor k/2 \rfloor + 1$, так что $L_t = (2L_{t-1}) \cup (2L_{t-1}+1) = (L_{t-1}*)_2$. Это рассуждение, открытое Йоргом Арндтом (Jörg Arndt) в 2001 году, дает подсказку и приводит к следующему алгоритму.

P1. [Инициализация.] Установить $t \leftarrow 1$, $m \leftarrow 0$. (Можно считать, что $n \ge 2$.)

P2. [Цикл по ведущим элементам.] Установить $r \leftarrow m$. Выполнить алгоритм Q с $k = 2^t + r$; затем, если $r > 0$, установить $r \leftarrow (r-1) \mathbin{\&} m$ и повторять эти действия, пока не будет выполнено условие $r = 0$. [См. упр. 7.1.3–79.]

P3. [Увеличение $\lg k$.] Установить $t \leftarrow t + 1$. Завершить работу алгоритма, если t равно n; в противном случае установить $m \leftarrow 2m + [t \mathbin{\&} (t-1) \ne 0]$ и вернуться к шагу P2. ▮

Q1. [Начало цикла.] Установить $s \leftarrow X_k$, $l \leftarrow k$, $j \leftarrow l \oplus \lfloor l/2 \rfloor$.

Q2. [Выполнение цикла.] Пока $j \neq k$, устанавливать $X_l \leftarrow X_j$, $l \leftarrow j$ и $j \leftarrow l \oplus \lfloor l/2 \rfloor$. Затем установить $X_l \leftarrow s$. ▌

31. Поле из f_n получается тогда и только тогда, когда можно получить поле из $f_n^{[2]}$, которая превращает $(a_{n-1} \ldots a_0)_2$ в $\big((a_{n-1} \oplus a_{n-2})(a_{n-1} \oplus a_{n-3})(a_{n-2} \oplus a_{n-4}) \ldots (a_2 \oplus a_0)(a_1)\big)_2$. Пусть $c_n(x)$ — характеристический полином матрицы A, определяющей это преобразование, по модулю 2; тогда $c_1(x) = x + 1$, $c_2(x) = x^2 + x + 1$, а $c_{j+1}(x) = xc_j(x) + c_{j-1}(x)$. Поскольку $c_n(A)$ — нулевая матрица по теореме Кэли–Гамильтона, поле получается тогда и только тогда, когда $c_n(x)$ — примитивный полином, а это условие можно проверить так, как описано в разделе 3.2.2. Первыми такими значениями n являются 1, 2, 3, 5, 6, 9, 11, 14, 23, 26, 29, 30, 33, 35, 39, 41, 51, 53, 65, 69, 74, 81, 83, 86, 89, 90, 95.

[Прокручивая это рекуррентное соотношение назад, можно показать, что $c_{-j-1}(x) = c_j(x)$, следовательно, $c_j(x)$ является делителем $c_{(2j+1)k+j}(x)$; например, $c_{3k+1}(x)$ всегда кратно $x+1$. Таким образом, все числа n вида $2jk + j + k$ исключаются при $j > 0$ и $k > 0$. Полиномы $c_{18}(x)$, $c_{50}(x)$, $c_{98}(x)$ и $c_{99}(x)$ неприводимы, но не примитивны.]

32. Почти всегда истинно, но ложно в точках, где $w_k(x)$ меняет знак. (Уолш (Walsh) изначально предположил, что функция $w_k(x)$ в таких точках должна быть нулевой, но принятое здесь соглашение лучше, так как делает простые формулы наподобие (15)–(19) корректными при всех x.)

33. По индукции по k имеем

$$w_k(x) = w_{\lfloor k/2 \rfloor}(2x) = r_1(2x)^{b_1+b_2} r_2(2x)^{b_2+b_3} \ldots = r_1(x)^{b_0+b_1} r_2(x)^{b_1+b_2} r_3(x)^{b_2+b_3} \ldots$$

для $0 \le x < \frac{1}{2}$, поскольку $r_j(2x) = r_{j+1}(x)$ и $r_1(x) = 1$ в этом диапазоне. А когда $\frac{1}{2} \le x < 1$,

$$\begin{aligned} w_k(x) = (-1)^{\lceil k/2 \rceil} w_{\lfloor k/2 \rfloor}(2x-1) &= r_1(x)^{b_0+b_1} r_1(2x-1)^{b_1+b_2} r_2(2x-1)^{b_2+b_3} \ldots \\ &= r_1(x)^{b_0+b_1} r_2(x)^{b_1+b_2} r_3(x)^{b_2+b_3} \ldots \end{aligned}$$

поскольку $\lceil k/2 \rceil \equiv b_0 + b_1$ (по модулю 2) и $r_j(2x-1) = r_{j+1}(x - \frac{1}{2}) = r_{j+1}(x)$ для $j \ge 1$.

34. $p_k(x) = \prod_{j \ge 0} r_{j+1}^{b_j}(x)$; так что $w_k(x) = p_k(x) p_{\lfloor k/2 \rfloor}(x) = p_{g(k)}(x)$. [R. E. A. C. Paley, *Proc. London Math. Soc.* (2) **34** (1932), 241–279.]

35. Если $j = (a_{n-1} \ldots a_0)_2$ и $k = (b_{n-1} \ldots b_0)_2$, то элемент на пересечении строки j и столбца k представляет собой $(-1)^{f(j,k)}$, где $f(j,k)$ — сумма всех $a_r b_s$, таких, что: $r = s$ (матрица Адамара); $r + s = n - 1$ (матрица Пейли); $r + s = n$ или $n - 1$ (матрица Уолша).

Пусть R_n, F_n и G_n — матрицы перестановок, которые преобразуют $j = (a_{n-1} \ldots a_0)_2$ в $k = (a_0 \ldots a_{n-1})_2$, $k = 2^n - 1 - j = (\overline{a}_{n-1} \ldots \overline{a}_0)_2$ и $k = g^{[-1]}(j) = \big((a_{n-1}) \ldots (a_{n-1} \oplus \cdots \oplus a_0)\big)_2$ соответственно. Тогда, используя прямое произведение матриц, мы получим рекурсивные формулы

$$R_{n+1} = \begin{pmatrix} R_n \otimes (1\ 0) \\ R_n \otimes (0\ 1) \end{pmatrix}, \qquad F_{n+1} = F_n \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \qquad G_{n+1} = \begin{pmatrix} G_n & 0 \\ 0 & G_n F_n \end{pmatrix},$$

$$H_{n+1} = H_n \otimes \begin{pmatrix} 1 & 1 \\ 1 & \overline{1} \end{pmatrix}, \qquad P_{n+1} = \begin{pmatrix} P_n \otimes (1\ 1) \\ P_n \otimes (1\ \overline{1}) \end{pmatrix}, \qquad W_{n+1} = \begin{pmatrix} W_n \otimes (1\ 1) \\ F_n W_n \otimes (1\ \overline{1}) \end{pmatrix}.$$

Таким образом, $W_n = G_n^T P_n = P_n G_n$; $H_n = P_n R_n = R_n P_n$; и $P_n = W_n G_n^T = G_n W_n = H_n R_n = R_n H_n$.

36. T1. [Преобразование Адамара.] Для $k = 0, 1, \ldots, n-1$ заменить пару (X_j, X_{j+2^k}) парой $(X_j + X_{j+2^k}, X_j - X_{j+2^k})$ для всех j с четным $\lfloor j/2^k \rfloor$, $0 \le j < 2^n$. (Эти операции эффективно устанавливают $X^T \leftarrow H_n X^T$.)

T2. [Обращение битов.] Примените алгоритм из упр. 5 к вектору X. (Эти операции эффективно устанавливают $X^T \leftarrow R_n X^T$ в обозначениях упр. 35.)

T3. [Бинарная перестановка Грея.] Примените алгоритм из упр. 30 к вектору X. (Эти операции эффективно устанавливают $X^T \leftarrow G_n^T X^T$.) ▌

Если n имеет одно из частных значений из упр. 31, может оказаться быстрее объединить шаги T2 и T3 в один шаг перестановки.

37. Если $k = 2^{e_1} + \cdots + 2^{e_t}$, где $e_1 > \cdots > e_t \ge 0$, то изменения знака происходят в $S_{e_1} \cup \cdots \cup S_{e_t}$, где $S_0 = \left\{\frac{1}{2}\right\}$, $S_1 = \left\{\frac{1}{4}, \frac{3}{4}\right\}$, $\ldots$, $S_e = \left\{\frac{2j+1}{2^{e+1}} \;\middle|\; 0 \le j < 2^e\right\}$. Следовательно, количество изменений знака в $(0\,..\,x)$ равно $\sum_{j=1}^{t} \lfloor 2^{e_j} x + \frac{1}{2} \rfloor$. Подстановка $x = l/(k+1)$ дает $l + O(t)$ изменений; так что l-е изменение находится от $l/(k+1)$ на расстоянии не более $O(\nu(k))/2^{\lfloor \lg k \rfloor}$.

[Это рассуждение позволяет утверждать, что имеется бесконечно много пар (k, l), таких, что $|z_{kl} - l/(k+1)| = \Omega((\log k)/k)$. Однако способ явного построения таких "плохих" пар пока не обнаружен.]

38. Пусть $t_0(x) = 1$ и $t_k(x) = \omega^{\lfloor 3x \rfloor \lceil 2k/3 \rceil} t_{\lfloor k/3 \rfloor}(3x)$, где $\omega = e^{2\pi i/3}$. Тогда $t_k(x)$ обходит вокруг начала координат $\frac{2}{3}k$ раза при возрастании x от 0 до 1. Если $s_k(x) = \omega^{\lfloor 3^k x \rfloor}$ — тернарный аналог функции Радемахера $r_k(x)$, то $t_k(x) = \prod_{j \ge 0} s_{j+1}(x)^{b_j - b_{j+1}}$ при $k = (b_{n-1} \ldots b_0)_3$, как в модульном тернарном коде Грея.

39. (а) Воспользуемся вместо $\{a, b, c, d, e, f, g, h\}$ символами $\{x_0, x_1, \ldots, x_7\}$. Мы хотим найти перестановку p множества $\{0, 1, \ldots, 7\}$, такую, что матрица с $(-1)^{j \cdot k} x_{p(j) \oplus k}$ в строке j и столбце k имеет ортогональные строки; это условие эквивалентно требованию, что

$$(j \oplus j') \cdot (p(j) \oplus p(j')) \equiv 1 \ \text{(по модулю 2)} \qquad \text{для } 0 \le j < j' < 8.$$

Одно решение имеет вид $p(0) \ldots p(7) = 0\,1\,7\,2\,5\,6\,3\,4$, что дает тождество $(a^2 + b^2 + c^2 + d^2 + e^2 + f^2 + g^2 + h^2)(A^2 + B^2 + C^2 + D^2 + E^2 + F^2 + G^2 + H^2) = \mathcal{A}^2 + \mathcal{B}^2 + \mathcal{C}^2 + \mathcal{D}^2 + \mathcal{E}^2 + \mathcal{F}^2 + \mathcal{G}^2 + \mathcal{H}^2$, где

$$\begin{pmatrix} \mathcal{A} \\ \mathcal{B} \\ \mathcal{C} \\ \mathcal{D} \\ \mathcal{E} \\ \mathcal{F} \\ \mathcal{G} \\ \mathcal{H} \end{pmatrix} = \begin{pmatrix} a & b & c & d & e & f & g & h \\ b & -a & d & -c & f & -e & h & -g \\ h & g & -f & -e & d & c & -b & -a \\ c & -d & -a & b & g & -h & -e & f \\ f & e & h & g & -b & -a & -d & -c \\ g & -h & e & -f & -c & d & -a & b \\ d & c & -b & -a & -h & -g & f & e \\ e & -f & -g & h & -a & b & c & -d \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{pmatrix}.$$

[Это тождество было открыто в C. F. Degen, *Mémoires de l'Acad. Sci. St. Petersbourg* (5) **8** (1818), 207–219. Связанные октонионы рассматриваются в интересном обзоре J. C. Baez, *Bull. Amer. Math. Soc.* **39** (2002), 145–205; **42** (2005), 213, 229–243. См. также J. H. Conway and D. A. Smith, *On Quaternions and Octonions* (2003).]

(б) Решения 16×16 не существует. Ближайшее решение имеет вид

$$p(0)\ \ldots\ p(15) \ = \ 0\ 1\ 11\ 2\ 14\ 15\ 13\ 4\ 9\ 10\ 7\ 12\ 5\ 6\ 3\ 8,$$

который нарушается тогда и только тогда, когда $j \oplus j' = 5$. (См. *Philos. Mag.* **34** (1867), 461–475. В §9, §10, §11 и §13 этой статьи Сильвестер сформулировал и доказал основные результаты, которые теперь известны как преобразование Адамара, хотя сам Адамар (Hadamard) ссылается на Сильвестера [*Bull. des Sciences Mathématiques* (2) **17** (1893), 240–246]. Кроме того, Сильвестер вводит преобразование m^n элементов в §14, используя m-е корни единицы.)

40. Да; такое изменение действительно проходило бы через переставленные подмножества в лексикографическом бинарном порядке, а не в порядке бинарного кода Грея. (Любая матрица размером 5×5, состоящая из нулей и единиц и не являющаяся сингулярной по модулю 2, будет генерировать все 32 возможности при проходе по всем линейным комбинациям ее строк.) В таких случаях, как здесь, где любое количество a_j может меняться одновременно при одинаковых затратах, наиболее важным является появление линеечной функции или некоторой другой дельта-последовательности кода Грея, а не тот факт, что на каждом шаге выполняется только одно изменение a_j.

41. Не более 16; например, `fired`, `fires`, `finds`, `fines`, `fined`, `fares`, `fared`, `wares`, `wards`, `wands`, `wanes`, `waned`, `wines`, `winds`, `wires`, `wired`. Те же 16 слов мы также получим из пар `paced/links` и `paled/mints`; вероятно, то же самое получится и из слова, смешиваемого с несловом-антиподом.

42. Предположим, что $n \le 2^{2^r} + r + 1$, и пусть $s = 2^r$. Мы используем вспомогательную таблицу из 2^{r+s} бит f_{jk} для $0 \le j < 2^s$ и $0 \le k < s$, представляющую фокусные указатели, как в алгоритме L, совместно со вспомогательным s-битовым "регистром" $j = (j_{s-1} \ldots j_0)_2$ и $(r+2)$-битовым "счетчиком программы" $p = (p_{r+1} \ldots p_0)_2$. На каждом шаге мы проверяем счетчик программы и, возможно, регистр j и один из битов f; затем, на основе просмотренных битов, мы выполняем дополнение бита кода Грея, дополняем бит счетчика программы и, возможно, изменяем бит j или f, тем самым эмулируя шаг L3 по отношению к старшим $n - r - 2$ битам.

Например, вот построение при $r = 1$.

$p_2p_1p_0$	Изменить	Установить	
0 0 0	a_0, p_0	$j_0 \leftarrow f_{00}$	$j \leftarrow f_0$
0 0 1	a_1, p_1	$j_1 \leftarrow f_{01}$	
0 1 1	a_0, p_0	$f_{00} \leftarrow 0$	$f_0 \leftarrow 0$
0 1 0	a_2, p_2	$f_{01} \leftarrow 0$	

$p_2p_1p_0$	Изменить	Установить	
1 1 0	a_0, p_0	$f_{j0} \leftarrow f_{(j+1)0}$	$f_j \leftarrow f_{j+1}$
1 1 1	a_1, p_1	$f_{j1} \leftarrow f_{(j+1)1}$	
1 0 1	a_0, p_0	$f_{(j+1)0} \leftarrow (j+1)_0$	$f_{j+1} \leftarrow j+1$
1 0 0	a_{j+3}, p_2	$f_{(j+1)1} \leftarrow (j+1)_1$	

Процесс останавливается при попытке изменить бит a_n.

[На самом деле на каждом шаге нужно изменять только *один* вспомогательный бит, если мы позволим себе исследовать вместе со вспомогательными некоторые биты бинарного кода Грея, поскольку $p_r \ldots p_0 = a_r \ldots a_0$, и можно установить $f_0 \leftarrow 0$ более интеллектуальным способом, когда j не принимает свое окончательное значение $2^s - 1$. Это построение, предложенное Фредманом в 2001 году, усовершенствует другое его построение, опубликованное в *SICOMP* **7** (1978), 134–146. С помощью улучшенного построения можно снизить количество вспомогательных битов до $O(n)$.]

43. Это количество было оценено Сильверманом (Silverman), Виккерсом (Vickers) и Сэмпсоном (Sampson) [*IEEE Trans.* **IT-29** (1983), 894–901] как приближенно равное 7×10^{22}. Г. Хаанпаа (H. Haanpää) /and П. Р. И. Остергард (P. R. J. Östergard) нашли точное значение $d(6) = 71\,676\,427\,445\,141\,767\,741\,440$ в 2011 году, используя симметрию и "склеивая" непересекающиеся пути, конечные точки x которых имеют $\nu x = 3$, а внутренние вершины — $\nu x \le 3$.

44. Каждый n-битовый цикл Грея определяет пару совершенных паросочетаний (см. упр. 55).

45. (а) (`000 002 012 010 090 094 0b4 0b6 2b6 2be 2ae 2a6 226 22e 23e 23c 33c 33e 32e 32c 3ac 3a8 388 38a 18a 182 192 19a 11a 112 102 100`) — всего 32 элемента в шестнадцатеричной записи. Обратите внимание, что сигнатуры элементов в каждом цикле обходят код Грея Γ_4.

(б) Заземленной вершине v в ее цикле предшествует ее брат $v \oplus 2$. Если v — заземленная вершина в другом цикле из ее брата $u = v \oplus 1$, то можно объединить эти циклы путем

удаления $\{u\oplus 2 — u,\ v\oplus 2 — v\}$ и вставки $\{u — v,\ u\oplus 2 — v\oplus 2\}$. Повторите эти действия для всех заземленных вершин v.

(в) Рассмотрим мультиграф G', вершины которого являются циклами, а ребра идут от цикла v к циклу $v+1$ для всех четных заземленных вершин v. Каждая вершина G' имеет четную степень, так что ребра представляют собой объединение циклов в G'. Таким образом, любое ребро G' может быть удалено без изменения связанных компонентов.

(г) Нетрудно построить путь $P = v^{(0)} — v^{(1)} — v^{(2)} — \cdots$, проходящий через вершины G, где $v_0 = v_{-1} = 0$, который проходит через все такие v, для которых $\sigma(v) \le 1$, и через такие, что $\sigma(v^{(i)}) \in \{0, 1, 2, 4, 8\}$ для всех i. Возьмем цикл из (б), который содержит $v^{(0)}$, и назовем его рабочим циклом W. Затем для $i = 1, 2, \ldots$, пока W не будет включать все вершины, будем выполнять следующие действия. Если $v = v^{(i)} \notin W$, положим, что $u = v^{(i-1)}$ имеет $u_l \ne v_l$. *Случай 1.* $u\oplus c — u$ является ребром W для $c = 1$ или $c = 2$. Возьмем цикл для класса эквивалентности v, который содержит ребро $v\oplus c — v$. Удалим эти ребра и вставим $\{u — v,\ u\oplus c — v\oplus c\}$. *Случай 2.* Иначе на предыдущем шаге к $w = v^{(i-2)}$ и u должен быть применен случай 1. Если $c = 1$, то W содержит ребро $u\oplus 2 — u\oplus 3$. Мы можем найти цикл с $v\oplus 2 — v\oplus 3$ и заменить эти ребра на $\{u\oplus 2 — v\oplus 3,\ u\oplus 3 — v\oplus 3\}$. Аналогичный обмен ребрами работает и при $c = 2$.

(д) Последний цикл W позволяет нам восстановить $\mathcal{M}_{l(v)}(v)$. Когда $l(v) \ne 0$, функция $\mathcal{M}_{l(v)}$ эквивалентна $t = 2^{3r-1}$ независимым паросочетаниям r-мерного куба, поскольку существует t способов выбрать v_i $(i \ne l)$ с корректной сигнатурой. Так что количество различных циклов — не менее $M(r)^{12t}$ (см. упр. 44).

46. Имеются k-битовые сигнатуры $\sigma(v)$. Когда в бинарном коде Грея $\sigma(v) = g(j)$, $l(v) = (\rho(j+1) + [j \ne 2^k - 1])[j + 2$ не является степенью $2]$. Возникает как минимум $M(r)^{(2^k-k)t}$ циклов, где $t = 2^{(k-1)(r-1)+2}$. [*Information Processing Letters* **109** (2009), 267–272.]

47. Границы $(\frac{r}{e})^{2^{r-1}} < 2^{r-1}!/(2^{r-1}/r)^{2^{r-1}} \le M(r) \le r!^{2^{r-1}/r} = (\frac{r}{e} + O(\log r))^{2^{r-1}}$ доказываются в разделе 7.5.1. Следовательно, $d(n)^{1/2^n} \le n/e + O(\log n)$ в соответствии с упр. 44.

Если G_j считать r_j-мерным кубом, то из упр. 46 вытекает нижняя граница

$$\bigl(M(r_1)^{2^{n-r_1-k+1}}\bigr)^{2^{k-1}-k} \cdot \bigl(M(r_2)^{2^{n-r_2-k+1}}\bigr)^{2^{k-2}} \cdot \bigl(M(r_3)^{2^{n-r_3-k+1}}\bigr)^{2^{k-3}} \cdot \ldots \cdot \bigl(M(r_{k-1})^{2^{n-r_{k-1}-k+1}}\bigr)^{2} \cdot \bigl(M(r_k)^{2^{n-r_k-k+1}}\bigr)^{2};$$

но лучше выбрать $r_j \approx (n-2)/2^{j-[j=k]}$ для $1 \le j \le k$ вместо использования кубов грубо того же размера. Пусть $\alpha_j = r_j/e$ — нижняя граница $M(r_j)^{2^{1-r_j}}$. Нижняя граница $d(n)^{1/2^n}$ упрощается до

$$\alpha_1^{1/2-k/2^k}\alpha_2^{1/4}\alpha_3^{1/8}\ldots\alpha_{k-1}^{1/2^{k-1}}\alpha_k^{1/2^{k-1}} = 2^{-2+(k-4)/2^k}\left(\frac{n-2}{e}\right)^{1-k/2^k}\left(1+O\left(\frac{k}{n}\right)\right),$$

что равно $n/(4e) + O(\log n)^2$ при $k = \lg n + O(1)$.

49. Возьмите любой путь Гамильтона P от $0\ldots0$ до $1\ldots1$ в $(2n-1)$-мерном кубе, такой, как код Сэведж–Винклера, и используйте $0P$, $1\overline{P}$. (При $n = 1$ или $n = 2$ с помощью данной конструкции получаются все такие циклы, но при $n > 2$ имеется гораздо больше разных возможностей.)

50. $\alpha_1(n{+}1)\alpha_1^R n\alpha_1 j_1\alpha_2 n\alpha_2^R(n{+}1)\alpha_2\ldots j_{l-1}\alpha_l n\alpha_l^R(n{+}1)\alpha_l n\alpha_l^R j_{l-1}\ldots j_1\alpha_1^R n.$

51. Пусть $c_j = 2\lfloor(2^{n-1}+j)/n\rfloor$ и $c'_j = 2\lfloor(2^{n+1}+j)/(n+2)\rfloor$. Если $n \ne 3$, нетрудно проверить, что $4c_j \ge 8\lfloor 2^{n-1}/n\rfloor > 2\lceil 2^{n+1}/(n+2)\rceil \ge c'_k$ для $0 \le j < n$ и $0 \le k < n+2$. Следовательно, можно применить теорему D к любому n-битовому циклу Грея с количеством переходов c_j, подчеркивая b_j копий j и размещая подчеркнутый 0 последним, где $b_j = 2c_j - \frac{1}{2}c'_{(j+2+d) \bmod (n+2)} - [j=0]$, а d выбирается так, что $c'_d = c'_{d+1}$. Эта конструкция

работает, поскольку $l = b_0 + \cdots + b_{n-1} = 2(c_0 + \cdots + c_{n-1}) - \frac{1}{2}(c'_0 + \cdots + c'_{n+1} - c'_d - c'_{d+1}) - 1 = c'_d - 1$ нечетно. [Следствие B было открыто Т. Бакосом (T. Bakos) в 1950-х годах и подробно доказано в работе A. Ádám, *Truth Functions* (Budapest: 1968), 28–37. В книге Адама (Ádám) представлено также доказательство Г. Поллака (G. Pollák) того, что $c'_0 = c'_1$ для всех n; следовательно, можно взять $d = 0$. См. также J. P. Robinson and M. Cohn, *IEEE Trans.* **C-30** (1981), 17–23.]

52. Количество различных шаблонов кода в наименьших положениях координаты j не превышает $c_0 + \cdots + c_{j-1}$.

53. Теорема D дает только циклы с $c_j = c_{j+1}$ для некоторого j, так что она не может давать количества $(2, 4, 6, 8, 12)$. Расширение в упр. 50 дает также $c_j = c_{j+1} - 2$, но не может дать $(6, 10, 14, 18, 22, 26, 32)$. Эти множества чисел, удовлетворяющие условиям из упр. 52, в точности совпадают с теми, которые получаются, если начать с $\{2, 2, 4, \ldots, 2^{n-1}\}$ и многократно заменять некоторые пары $\{c_j, c_k\}$, в которых $c_j < c_k$, парой $\{c_j + 2, c_k - 2\}$.

54. Предположим, что значениями являются $\{p_1, \ldots, p_n\}$, и пусть x_{jk} — количество раз, когда p_j встречается в $(a_1, \ldots, a_k)$. Тогда для некоторых $k < l$ должно выполняться $(x_{1k}, \ldots, x_{nk}) \equiv (x_{1l}, \ldots, x_{nl})$ (по модулю 2). Но если p — простые числа, изменяющиеся как дельта-последовательность n-битового цикла Грея, то единственным решением будет $k = 0$ и $l = 2^n$. [*AMM* **60** (1953), 418; **83** (1976), 54.]

55. В действительности для каждого заданного совершенного паросочетания Q в графе K_{2^n} можно за $O(2^n)$ шагов найти совершенное паросочетание R в n-мерном кубе, такое, что $Q \cup R$ представляет собой Гамильтонов цикл графа K_{2^n}. [См. J. Fink, *J. Comb. Theory* **B97** (2007), 1074–1076; *Elect. Notes Disc. Math.* **29** (2007), 345–351.]

56. [*Bell System Tech. J.* **37** (1958), 815–826.] 112 канонических дельта-последовательностей дают следующее.

Класс	Пример	t	Класс	Пример	t	Класс	Пример	t
A	0102101302012023	2	D	0102013201020132	4	G	0102030201020302	8
B	0102303132101232	2	E	0102032021202302	4	H	0102101301021013	8
C	0102030130321013	2	F	0102013102010232	4	I	0102013121012132	1

Здесь B — сбалансированный код (рис. 33, (б)), G — стандартный бинарный код Грея (рис. 30, (б)), а H — комплементарный код (рис. 33, (а)). Класс H эквивалентен также модульному $(4, 4)$ коду Грея в соответствии с упр. 18. Класс с t автоморфизмами соответствует $32 \times 24/t$ из 2688 различных дельта-последовательностей $\delta_0\delta_1 \ldots \delta_{15}$.

Аналогично (см. упр. 7.2.3–00), 5-битовые циклы Грея подразделяются на 237 675 различных классов эквивалентности.

57. В случае только типа 1 изолированными оказываются 480 вершин, а именно вершины из классов D, F, G из предыдущего упражнения. В случае типа 2 граф состоит из 384 компонентов, 288 из которых представляют собой изолированные вершины классов F и G. Имеется 64 компонента с 9 вершинами каждый, из которых 3 принадлежат классу E, а 6 — классу A; 16 компонентов размером 30, в каждом из которых 6 вершин принадлежат классу H, а 24 — классу C; и 16 компонентов размером 84, в каждом из которых 12 вершин принадлежат классу D, 24 — классу B и 48 — классу I. В случае только типа 3 (или 4) весь граф оказывается связным. [Аналогично все 91 392 4-битовых *путей* Грея оказываются связанными, если путь $\alpha\beta$ рассматривать как смежный с путем $\alpha^R\beta$. Виккерс (Vickers) и Сильверман (Silverman) в *IEEE Trans.* **C-29** (1980), 329–331, предположили, что изменений третьего типа будет достаточно для связывания графа n-битовых циклов Грея для всех $n \geq 3$.]

58. Если некоторая непустая подстрока $\beta\beta$ включает каждую координату четное число раз, то такая подстрока не может иметь длину $|\beta|$, так что некоторый циклический сдвиг

β имеет префикс γ с тем же свойством четности. Но тогда α не определяет цикл Грея, поскольку мы можем изменить каждое n в γ обратно в 0.

59. Если α является нелокальным в упр. 58, то таким же является и $\beta\beta$, при условии, что $q > 1$ и что 0 встречается в α больше $q+1$ раз. Следовательно, начиная с α из (30), но с переставленными местами 0 и 1, мы получим нелокальные циклы для $n \geq 5$, в которых координата 0 изменяется ровно 6 раз. [Mark Ramras, *Discrete Math.* **85** (1990), 329–331.] С другой стороны, 4-битовый цикл Грея не может быть нелокальным, поскольку в нем всегда содержится серия длиной 2; если $\delta_k = \delta_{k+2}$, элементы $\{v_k, v_{k+1}, v_{k+2}, v_{k+3}\}$ образуют 2-подкуб.

60. Воспользуйтесь конструкцией из упр. 58 с $q = 1$.

61. Идея заключается в чередовании m-битового цикла $U = (u_0, u_1, u_2, \ldots)$ и n-битового цикла $V = (v_0, v_1, v_2, \ldots)$ с образованием конкатенаций

$$W = (u_{i_0}v_{j_0},\ u_{i_1}v_{j_1},\ u_{i_2}v_{j_2},\ \ldots), \qquad i_k = \overline{a}_0 + \cdots + \overline{a}_{k-1}, \quad j_k = a_0 + \cdots + a_{k-1},$$

где $a_0a_1a_2\ldots$ представляет собой периодическую строку контрольных битов $\alpha\alpha\alpha\ldots$; мы переходим к следующему элементу U при $a_k = 0$, в противном случае — к следующему элементу V.

Если α — произвольная строка длиной $2^m \leq 2^n$, содержащая s нулевых битов и $t = 2^m - s$ единичных, то W представляет собой $(m+n)$-битовый цикл Грея при нечетных s и t. Мы имеем $i_{k+l} \equiv i_k$ (по модулю 2^m) и $j_{k+l} \equiv j_k$ (по модулю 2^n), только если l кратно 2^m, поскольку $i_k + j_k = k$. Допустим, что $l = 2^m c$; тогда $j_{k+l} = j_k + tc$, так что c кратно 2^n.

(а) Пусть $\alpha = 0111$; тогда серии длиной 8 находятся в двух левых битах, а серии длиной $\geq \lfloor \frac{4}{3} r(n) \rfloor$ — в n правых битах.

(б) Пусть s — наибольшее нечетное число $\leq 2^m r(m)/(r(m)+r(n))$. Пусть также $t = 2^m - s$ и $a_k = \lfloor (k+1)t/2^m \rfloor - \lfloor kt/2^m \rfloor$, так что $i_k = \lceil ks/2^m \rceil$ и $j_k = \lfloor kt/2^m \rfloor$. Если серия длиной l находится в m левых битах, то мы имеем $i_{k+l+1} \geq i_k + r(m) + 1$, следовательно, $l + 1 > 2^m r(m)/s \geq r(m) + r(n)$. А если она находится в n правых битах, то $j_{k+l+1} \geq j_k + r(n) + 1$, следовательно,

$$\begin{aligned} l + 1 &> 2^m r(n)/t > 2^m r(n)/\bigl(2^m r(n)/(r(m)+r(n)) + 2\bigr) \\ &= r(m) + r(n) - \frac{2(r(m)+r(n))^2}{2^m r(n) + 2(r(m)+r(n))} > r(m) + r(n) - 1 \end{aligned}$$

поскольку $r(m) \leq r(n)$.

Эта конструкция часто работает и в менее ограниченных случаях. Более подробно серии в кодах Грея рассматриваются в работе L. Goddyn, G. M. Lawrence, and E. Nemeth, *Utilitas Math.* **34** (1988), 179–192.

63. Установите $a_k \leftarrow k \bmod 4$ для $0 \leq k < 2^{10}$, за исключением $a_k = 4$ при $k \bmod 16 = 15$ или $k \bmod 64 = 42$, или $k \bmod 256 = 133$. Установите также $(j_0, j_1, j_2, j_3, j_4) \leftarrow (0, 2, 4, 6, 8)$. Затем для $k = 0, 1, \ldots, 1023$, установите $\delta_k \leftarrow j_{a_k}$ и $j_{a_k} \leftarrow 1 + 4a_k - j_{a_k}$. (Эта конструкция обобщает метод из упр. 61.)

64. (а) Каждый элемент u_k появляется вместе с $\{v_k, v_{k+2^m}, \ldots, v_{k+2^m(2^{n-1}-1)}\}$ и $\{v_{k+1}, v_{k+1+2^m}, \ldots, v_{k+1+2^m(2^{n-1}-1)}\}$. Таким образом, перестановка $\sigma_0 \ldots \sigma_{2^m-1}$ должна быть 2^{n-1}-циклом, содержащим n-битовые четные вершины и умноженным на произвольную перестановку других вершин. Это условие является и достаточным.

(б) Пусть τ_j является такой перестановкой, что $v \mapsto v \oplus 2^j$, и пусть $\pi_j(u, w)$ представляет собой перестановку $(uw)\tau_j$. Если $u \oplus w = 2^i + 2^j$, то $\pi_j(u, w)$ выполняет отображения $u \mapsto u \oplus 2^i$ и $w \mapsto w \oplus 2^i$, тогда как $v \mapsto v \oplus 2^j$ для всех других вершин v, так что каждая вершина переходит в соседнюю.

Если S — произвольное множество $\subseteq \{0,\ldots,n-1\}$, то пусть $\sigma(S)$ — поток всех перестановок τ_j для всех $j \in \{0,\ldots,n-1\} \setminus S$, в порядке возрастания j, повторенный дважды; например, если $n=5$, мы имеем $\sigma(\{1,2\}) = \tau_0\tau_3\tau_4\tau_0\tau_3\tau_4$. Тогда поток Грея

$$\Sigma(i,j,u) = \sigma(\{i,j\})\pi_j(u, u{\oplus}2^i{\oplus}2^j)\sigma(\{i,j\})\tau_j\sigma(\{j\})$$

состоит из $6n-8$ перестановок, произведение которых представляет собой транспозицию $(u\ u{\oplus}2^i{\oplus}2^j)$. Более того, когда этот поток применяется к любой n-битовой вершине v, то все его серии имеют длину не менее $n-2$.

Можно считать, что $n \ge 5$. Пусть $\delta_0\ldots\delta_{2^n-1}$ представляет собой дельта-последовательность для n-битового цикла Грея $(v_0, v_1, \ldots, v_{2^n-1})$, в которой все серии имеют длину не менее 3. Тогда произведение всех перестановок в

$$\Sigma = \prod_{k=1}^{2^{n-1}-1} \bigl(\Sigma(\delta_{2k-1}, \delta_{2k}, v_{2k-1})\,\Sigma(\delta_{2k}, \delta_{2k+1}, v_{2k})\bigr)$$

равно $(v_1\,v_3)(v_2\,v_4)\ldots(v_{2^n-3}\,v_{2^n-1})(v_{2^n-2}\,v_0) = (v_{2^n-1}\ldots v_1)(v_{2^n-2}\ldots v_0)$, т. е. удовлетворяет условию цикла (а).

Более того, все степени $(\sigma(\emptyset)\Sigma)^t$ дают при применении к произвольной вершине v серии длиной $\ge n-2$. Повторяя отдельные множители $\sigma(\{i,j\})$ или $\sigma(\{j\})$ в Σ столько раз, сколько нам нужно, мы можем настроить длину $\sigma(\emptyset)\Sigma$, получив $2n+(2^{n-1}-1)(12n-16)+2(n-2)a+2(n-1)b$ для произвольных целых чисел $a,b \ge 0$; таким образом, можно увеличить ее длину ровно до 2^m при условии, что $2^m \ge 2n+(2^{n-1}-1)(12n-16)+2(n^2-5n+6)$, согласно упр. 5.2.1– 21.

(в) Граница $r(n) \ge n - 4\lg n + 8$ может быть доказана для $n \ge 5$ следующим образом. Сначала заметим, что она справедлива для $5 \le n < 33$, согласно методу из упр. 60–63. Затем заметим, что каждое целое число $N \ge 33$ может быть записано как $N = m+n$ или $N = m+n+1$ для некоторого $m \ge 20$, где $n = m - \lfloor 4\lg m\rfloor + 10$. Если $m \ge 20$, то 2^m достаточно велико для того, чтобы построение из п. (б) было корректно; следовательно,

$$\begin{aligned} r(N) \ge r(m+n) \ge 2\min\bigl(r(m), n-2\bigr) &\ge 2\bigl(m - \lfloor 4\lg m\rfloor + 8\bigr) \\ &= m+n+1-\lfloor 4\lg N - 1 + \epsilon\rfloor + 8 \\ &\ge N - 4\lg N + 8, \end{aligned}$$

где $\epsilon = 4\lg(2m/N) < 1 + [N = m+n]$. [*Electronic Journal of Combinatorics* **10** (2003), #R27, 1–10.] Рекурсивное применение (б) дает $r(1024) \ge 1000$.

65. Компьютерный поиск дает восемь существенно различных возможных шаблонов (и обратных к ним). Один из них имеет дельта-последовательность 0102031420302404 1234214103234103, достаточно близкую к двум другим.

66. (Решение Марка Кука (Mark Cooke).) Одной из подходящих дельта-последовательностей является 0123456070121324356576071021353462670153741236256701731426206570134214656057310246453757102043537614073630464273703564027132750541210275641502403654250136025416156043125760325720431576243217604520417516354767035647570625437242132624161523417514367143164314. (Решения для $n > 8$ пока что неизвестны.)

67. Пусть $v_{2k+1} = \overline{v}_{2k}$ и $v_{2k} = 0u_k$, где $(u_0, u_1, \ldots, u_{2^{n-1}-1})$ — произвольный $(n-1)$-битовый цикл Грея. [См. Robinson and Cohn, *IEEE Trans.* **C-30** (1981), 17–23.]

68. Да. Вероятно, простейшим способом будет взять $(n-1)$-значный модульный тернарный код Грея и добавить к каждой его строке $0\ldots0$, $1\ldots1$, $2\ldots2$ по модулю 3. Например, при $n=3$ получится код 000, 111, 222, 001, 112, 220, 002, 110, 221, 012, 120, 201, ..., 020, 101, 212.

69. (а) Необходимо проверить только те изменения в h, когда одновременно дополняются биты $b_{j-1}\ldots b_0$, для $j = 1, 2, \ldots$; этими изменениями являются соответственно $(1110)_2$, $(1101)_2$, $(0111)_2$, $(1011)_2$, $(10011)_2$, $(100011)_2$, ... Чтобы доказать, что встречаются все n-кортежи, заметим, что $0 \le h(k) < 2^n$, когда $0 \le k < 2^n$ и $n > 3$; также $h^{[-1]}((a_{n-1}\ldots a_0)_2) = (b_{n-1}\ldots b_0)_2$, где $b_0 = a_0 \oplus a_1 \oplus a_2 \oplus \cdots$, $b_1 = a_0$, $b_2 = a_2 \oplus a_3 \oplus a_4 \oplus \cdots$, $b_3 = a_0 \oplus a_1 \oplus a_3 \oplus \cdots$ и $b_j = a_j \oplus a_{j+1} \oplus \cdots$ для $j \ge 4$.

(б) Пусть $h(k) = (\ldots a_2 a_1 a_0)_2$, где $a_j = b_j \oplus b_{j+1} \oplus b_0[j \le t] \oplus b_{t-1}[t-1 \le j \le t]$.

70. Как и в (32) и (33), можно удалить множитель $n!$, полагая, что строки с весом 1 упорядочены. Тогда для $n = 5$ имеется 14 решений, начинающихся с 00000, и 21 решение, начинающееся с 00001. Когда $n = 6$, имеется 46 935 кодов каждого типа (связанных обращением и дополнением). При $n = 7$ количество монотонных кодов гораздо, неимоверно большее, но при этом все равно очень маленькое по сравнению с общим количеством 7-битовых кодов Грея.

71. Предположим, что $\alpha_{n(j+1)}$ отличается от α_{nj} в координате t_j, для $0 \le j < n-1$. Тогда $t_j = j\pi_n$ в соответствии с (44) и (38). Теперь (34) говорит нам, что $t_0 = n-1$; и если $0 < j < n-1$, мы имеем $t_j = ((j-1)\pi_{n-1})\pi_{n-1}$ согласно (40). Таким образом, $t_j = j\sigma_n\pi_{n-1}^2$ для $0 \le j < n-1$, а значение $(n-1)\pi_n$ определяется тем, что осталось. (Обозначения для перестановок весьма запутанны, так что всегда разумно выполнить тщательную проверку для нескольких малых случаев.)

72. Дельта-последовательность имеет вид 0102132430201234012313041021323.

73. Пусть $Q_{nj} = P_{nj}^R$, и обозначим последовательности (41) и (42) как S_n и T_n. Таким образом, $S_n = P_{n0}Q_{n1}P_{n2}\ldots$ и $T_n = Q_{n0}P_{n1}Q_{n2}\ldots$, если опустить запятые. Итак, мы имеем

$$S_{n+1} = 0P_{n0}\ 0Q_{n1}\ 1Q_{n0}^{\pi}\ 1P_{n1}^{\pi}\ 0P_{n2}\ 0Q_{n3}\ 1Q_{n2}^{\pi}\ 1P_{n3}^{\pi}\ 0P_{n4}\ \ldots,$$
$$T_{n+1} = 0Q_{n0}\ 1P_{n0}^{\pi}\ 0P_{n1}\ 0Q_{n2}\ 1Q_{n1}^{\pi}\ 1P_{n2}^{\pi}\ 0P_{n3}\ 0Q_{n4}\ 1Q_{n3}^{\pi}\ \ldots,$$

где $\pi = \pi_n$, которые демонстрируют относительно простую взаимную рекурсию между дельта-последовательностями Δ_n и E_n для S_n и T_n. А именно, если записать

$$\Delta_n = \phi_1\, a_1\, \phi_2\, a_2 \ldots \phi_{n-1}\, a_{n-1}\, \phi_n, \qquad E_n = \psi_1\, b_1\, \psi_2\, b_2 \ldots \psi_{n-1}\, b_{n-1}\, \psi_n,$$

где каждое ϕ_j и ψ_j представляет собой строку длиной $2\binom{n-1}{j-1} - 1$, то следующие за ними последовательности имеют вид

$$\Delta_{n+1} = \phi_1\ a_1\ \phi_2\ n\ \psi_1\pi\ b_1\pi\ \psi_2\pi\ n\ \phi_3\ a_3\ \phi_4\ n\ \psi_3\pi\ b_3\pi\ \psi_4\pi\ n\ \ldots$$
$$E_{n+1} = \psi_1\ n\ \phi_1\pi\ n\ \psi_2\ b_2\ \psi_3\ n\ \phi_2\pi\ a_2\pi\ \phi_3\pi\ n\ \psi_4\ b_4\ \psi_5\ n\ \phi_4\pi\ a_4\pi\ \phi_5\pi\ n\ \ldots$$

Например, имеем $\Delta_3 = 0\,\underline{1}\,0\,2\,1\,\underline{0}\,1$ и $E_3 = 0\,\underline{2}\,1\,2\,0\,\underline{2}\,1$, если подчеркнуть a и b, чтобы отличать их от ϕ и ψ; и

$$\Delta_4 = 0\ 1\ 0\ 2\ 1\ 3\ 0\pi\ 2\pi\ 1\pi\ 2\pi\ 0\pi\ 3\ 1\ 3\ 1\pi = 0\ \underline{1}\ 0\ 2\ 1\ 3\ 2\ \underline{1}\ 0\ 1\ 2\ 3\ 1\ \underline{3}\ 0,$$
$$E_4 = 0\ 3\ 0\pi\ 3\ 1\ 2\ 0\ 2\ 1\ 3\ 0\pi\ 2\pi\ 1\pi\ 0\pi\ 1\pi = 0\ \underline{3}\ 2\ 3\ 1\ 2\ 0\ \underline{2}\ 1\ 3\ 2\ 1\ 0\ \underline{2}\ 0;$$

здесь $a_3\phi_4$ и $b_3\psi_4$ — пустые. Элементы подчеркнуты для следующего шага.

Таким образом, можно вычислить дельта-последовательности в памяти следующим образом. Здесь $p[j] = j\pi_n$ для $1 \le j < n$; $s_k = \delta_k$, $t_k = \varepsilon_k$ и $u_k = [\delta_k$ и ε_k подчеркнуты$]$ для $0 \le k < 2^n - 1$.

X1. [Инициализация.] Установить $n \leftarrow 1$, $p[0] \leftarrow 0$, $s_0 \leftarrow t_0 \leftarrow u_0 \leftarrow 0$.

X2. [Продвижение n.] Выполнить приведенный ниже алгоритм Y, который вычисляет массивы s', t' и u' для следующего значения n; затем установить $n \leftarrow n + 1$.

X3. [Готово?] Если n достаточно большое, искомая дельта-последовательность Δ_n находится в массиве s'; работу алгоритма следует завершить. В противном случае продолжить выполнение алгоритма.

X4. [Вычисление π_n.] Установить $p'[0] = n-1$ и $p'[j] = p[p[j-1]]$ для $1 \le j < n$.

X5. [Подготовка к продвижению.] Установить $p[j] \leftarrow p'[j]$ для $0 \le j < n$; установить $s_k \leftarrow s'_k$, $t_k \leftarrow t'_k$ и $u_k \leftarrow u'_k$ для $0 \le k < 2^n - 1$. Вернуться к шагу X2. ▌

В приведенных далее шагах фраза "Передавать `нечто`(l, j), пока $u_j = 0$" представляет собой сокращение для "если $u_j = 0$, многократно выполнять `нечто`(l, j), $l \leftarrow l+1$, $j \leftarrow j+1$, пока не будет выполнено условие $u_j \ne 0$".

Y1. [Подготовка к вычислению Δ_{n+1}.] Установить $j \leftarrow k \leftarrow l \leftarrow 0$ и $u_{2^n-1} \leftarrow -1$.

Y2. [Продвижение j.] Передавать $s'_l \leftarrow s_j$ и $u'_l \leftarrow 0$, пока $u_j = 0$. Затем перейти к шагу Y5, если $u_j < 0$.

Y3. [Продвижение j и k.] Установить $s'_l \leftarrow s_j$, $u'_l \leftarrow 1$, $l \leftarrow l+1$, $j \leftarrow j+1$. Затем передавать $s'_l \leftarrow s_j$ и $u'_l \leftarrow 0$, пока $u_j = 0$. Затем установить $s'_l \leftarrow n$, $u'_l \leftarrow 0$, $l \leftarrow l+1$. Затем передавать $s'_l \leftarrow p[t_k]$ и $u'_l \leftarrow 0$, пока $u_k = 0$. Затем установить $s'_l \leftarrow p[t_k]$, $u'_l \leftarrow 1$, $l \leftarrow l+1$, $k \leftarrow k+1$. И еще раз передавать $s'_l \leftarrow p[t_k]$ и $u'_l \leftarrow 0$, пока $u_k = 0$.

Y4. [Готово с Δ_{n+1}?] Если $u_k < 0$, перейти к шагу Y6. В противном случае установить $s'_l \leftarrow n$, $u'_l \leftarrow 0$, $l \leftarrow l+1$, $j \leftarrow j+1$, $k \leftarrow k+1$ и вернуться к шагу Y2.

Y5. [Завершение Δ_{n+1}.] Установить $s'_l \leftarrow n$, $u'_l \leftarrow 1$, $l \leftarrow l+1$. Затем передавать $s'_l \leftarrow p[t[k]]$ и $u'_l \leftarrow 0$, пока $u_k = 0$.

Y6. [Подготовка к вычислению E_{n+1}.] Установить $j \leftarrow k \leftarrow l \leftarrow 0$. Передавать $t'_l \leftarrow t_k$, пока $u_k = 0$. Затем установить $t'_l \leftarrow n$, $l \leftarrow l+1$.

Y7. [Продвижение j.] Передавать $t'_l \leftarrow p[s_j]$, пока $u_j = 0$. Завершить работу, если $u_j < 0$; в противном случае установить $t'_l \leftarrow n$, $l \leftarrow l+1$, $j \leftarrow j+1$, $k \leftarrow k+1$.

Y8. [Продвижение k.] Передавать $t'_l \leftarrow t_k$, пока $u_k = 0$. Затем перейти к шагу Y10, если $u_k < 0$.

Y9. [Продвижение k и j.] Установить $t'_l \leftarrow t_k$, $l \leftarrow l+1$, $k \leftarrow k+1$. Затем передавать $t'_l \leftarrow t_k$, пока $u_k = 0$. Затем установить $t'_l \leftarrow n$, $l \leftarrow l+1$. Затем передавать $t'_l \leftarrow p[s_j]$, пока $u_j = 0$. Затем установить $t'_l \leftarrow p[s_j]$, $l \leftarrow l+1$, $j \leftarrow j+1$. Вернуться к шагу Y7.

Y10. [Завершение E_{n+1}.] Установить $t'_l \leftarrow n$, $l \leftarrow l+1$. Затем передавать $t'_l \leftarrow p[s_j]$, пока $u_j = 0$. ▌

Для генерации монотонного кода Сэведж–Винклера для достаточно больших n можно сначала сгенерировать, скажем, Δ_{10} и E_{10} или даже Δ_{20} и E_{20}. Используя эти таблицы, подходящая рекурсивная процедура сможет достигнуть высоких значений n с очень небольшими средними накладными вычислительными расходами на один шаг.

74. Если монотонный путь имеет вид $v_0, \ldots, v_{2^n-1}$ и если v_k имеет вес j, то

$$2\sum_{t>0}\binom{n}{j-2t} + ((j+\nu(v_0)) \bmod 2) \le k \le 2\sum_{t\ge 0}\binom{n}{j-2t} + ((j+\nu(v_0)) \bmod 2) - 2.$$

Следовательно, максимальное расстояние между вершинами с весами j и $j+1$ составляет $2\left(\binom{n-1}{j-1} + \binom{n-1}{j} + \binom{n-1}{j+1}\right) - 1$. Максимальное значение, приближенно равное $3 \cdot 2^{n+1}/\sqrt{2\pi n}$, осуществляется при j, приближенно равном $n/2$. [Оно всего лишь примерно в три раза

больше минимального значения, достижимого при *любом* упорядочении вершин и равного $\sum_{j=0}^{n-1}\binom{j}{\lfloor j/2\rfloor}$ согласно упр. 7.10–00.]

75. Все канонические дельта-последовательности без трендов дают *циклы* Грея.

$$\begin{array}{l}
0\,1\,2\,3\,0\,1\,2\,4\,2\,1\,0\,3\,2\,1\,0\,1\,2\,1\,0\,3\,2\,1\,0\,4\,0\,1\,2\,3\,0\,1\,2\,(1)\\
0\,1\,2\,3\,0\,1\,2\,4\,2\,1\,0\,3\,2\,1\,0\,1\,3\,0\,1\,2\,3\,0\,1\,4\,1\,0\,3\,2\,1\,0\,3\,(1)\\
0\,1\,2\,3\,0\,1\,2\,4\,2\,1\,0\,3\,2\,1\,0\,2\,0\,3\,2\,1\,0\,3\,2\,4\,2\,3\,0\,1\,2\,3\,0\,(2)\\
0\,1\,2\,3\,0\,1\,2\,4\,2\,1\,0\,3\,2\,1\,0\,2\,1\,2\,3\,0\,1\,2\,3\,4\,3\,2\,1\,0\,3\,2\,1\,(2)\\
0\,1\,2\,3\,0\,1\,2\,4\,2\,3\,0\,1\,2\,3\,0\,2\,0\,1\,2\,3\,0\,1\,2\,4\,2\,3\,0\,1\,2\,3\,0\,(2)\\
0\,1\,2\,3\,4\,1\,0\,1\,2\,1\,0\,3\,0\,1\,4\,3\,2\,1\,0\,3\,0\,1\,4\,1\,0\,1\,2\,3\,4\,1\,0\,(3)
\end{array}$$

(Вторая и четвертая последовательности циклически эквивалентны.)

76. Если $v_0, \ldots, v_{2^n-1}$ не имеет тренда, то то же самое справедливо и в отношении $(n+1)$-битового цикла $0v_0, 1v_0, 1v_1, 0v_1, 0v_2, 1v_2, \ldots, 1v_{2^n-1}, 0v_{2^n-1}$. На рис. 34, (ж) показана более интересная конструкция, которая обобщает первое решение упр. 75 до $(n+2)$-битового цикла

$$00\Gamma''^R,\ 01\Gamma'^R,\ 11\Gamma',\ 10\Gamma'',\ 10\Gamma,\ 11\Gamma''',\ 01\Gamma'''^R,\ 00\Gamma^R,$$

где Γ — n-битовая последовательность $g(1), \ldots, g(2^{n-1})$ и $\Gamma' = \Gamma \oplus g(1)$, $\Gamma'' = \Gamma \oplus g(2^{n-1})$, $\Gamma''' = \Gamma \oplus g(2^{n-1}+1)$. [$n$-битовая структура без тренда, которая является *почти* кодом Грея, имея только четыре шага, где $\nu(v_k \oplus v_{k+1}) = 2$, была найдена для всех $n \geq 3$ Ч. С. Ченгом (C. S. Cheng), *Proc. Berkeley Conf. Neyman and Kiefer* **2** (Hayward, Calif.: Inst. of Math. Statistics, 1985), 619–633.]

77. В описании шага H1 замените массив $(o_{n-1}, \ldots, o_0)$ массивом ограничителей $(s_{n-1}, \ldots, s_0)$ с $s_j \leftarrow m_j - 1$. На шаге H4 установите $a_j \leftarrow (a_j + 1) \bmod m_j$. Если на шаге H5 выполняется условие $a_j = s_j$, установите $s_j \leftarrow (s_j - 1) \bmod m_j$, $f_j \leftarrow f_{j+1}$, $f_{j+1} \leftarrow j + 1$.

78. В случае (50) обратите внимание, что B_{j+1} представляет собой количество отражений в координате j, поскольку координата j пропускается на шагах, кратных $m_j \ldots m_0$. Следовательно, если $b_j < m_j - 1$, увеличение b_j на 1 вызывает соответствующее увеличение или уменьшение a_j на 1. Кроме того, если $b_i = m_i - 1$ для $0 \leq i < j$, замена всех этих b_i на 0 при увеличении b_j приведет к увеличению каждого из $B_0, \ldots, B_j$ на 1, тем самым оставляя значения $a_0, \ldots, a_{j-1}$ в (50) неизменными.

В случае (51) заметим, что $B_j = m_j B_{j+1} + b_j \equiv m_j B_{j+1} + a_j + (m_j - 1)B_{j+1} \equiv a_j + B_{j+1}$ (по модулю 2); следовательно, $B_j \equiv a_j + a_{j+1} + \cdots$ и очевидно, что (51) эквивалентно (50).

В модульном коде Грея для оснований $(m_{n-1}, \ldots, m_0)$ примем

$$\bar{g}(k) = \begin{bmatrix} a_{n-1}, & \ldots, & a_2, & a_1, & a_0 \\ m_{n-1}, & \ldots, & m_2, & m_1, & m_0 \end{bmatrix},$$

когда k определяется в (46). Тогда $a_j = (b_j - B_{j+1}) \bmod m_j$, поскольку координата j увеличивается по модулю m_j ровно $B_j - B_{j+1}$ раз, если начать с $(0, \ldots, 0)$. Обратная функция, которая определяет значения b из модульного кода Грея a, в частном случае, когда каждое m_j является делителем m_{j+1} (например, если все m_j равны) имеет вид $b_j = (a_j + a_{j+1} + a_{j+2} + \cdots) \bmod m_j$. Но в общем случае обратная функция простого вида не имеет; ее можно вычислить с помощью рекуррентных соотношений $b_j = (a_j + B_{j+1}) \bmod m_j$, $B_j = m_j B_{j+1} + b_j$ для $j = n-1, \ldots, 0$, начиная с $B_n = 0$.

[Рефлексивные коды Грея для основания $m > 2$ были предложены Айвеном Флоресом (Ivan Flores) в *IRE Trans.* **EC-5** (1956), 79–82; он вывел (50) и (51) для случая, когда все m_j равны. Модульные коды Грея с общими смешанными основаниями неявно рассмотрены Джозефом Розенбаумом (Joseph Rosenbaum) в *AMM* **45** (1938), 694–696, но без формул преобразований; формулы преобразований, когда все m_j имеют общее значение m, были опубликованы Мартином Коном (Martin Cohn), *Information and Control* **6** (1963), 70–78.]

79. (а) У последнего n-кортежа всегда $a_{n-1} = m_{n-1} - 1$, так что он находится на расстоянии одного шага от $(0, \ldots, 0)$, только если $m_{n-1} = 2$. Это условие достаточно, чтобы сделать последним n-кортежем $(1, 0, \ldots, 0)$. [Аналогично последний подлес, выводимый алгоритмом K, смежен с начальным тогда и только тогда, когда крайнее слева дерево является изолированной вершиной.]

(б) Последний n-кортеж имеет вид $(m_{n-1} - 1, 0, \ldots, 0)$ тогда и только тогда, когда $m_{n-1} \ldots m_{j+1} \bmod m_j = 0$ для $0 \le j < n-1$, поскольку $b_j = m_j - 1$ и $B_j = m_{n-1} \ldots m_j - 1$.

80. Пройдите все $p_1^{a_1} \ldots p_t^{a_t}$ с использованием рефлексивного кода Грея с основаниями $m_j = e_j + 1$.

81. Первый цикл содержит ребро от (x, y) до $(x, (y+1) \bmod m)$ тогда и только тогда, когда $(x+y) \bmod m \ne m-1$, тогда и только тогда, когда второй цикл содержит ребро от (x, y) до $((x+1) \bmod m, y)$.

82. Имеется два 4-битовых цикла Грея $(u_0, \ldots, u_{15})$ и $(v_0, \ldots, v_{15})$, которые покрывают все ребра четырехмерного куба. (В самом деле, неребра классов A, B, D, H и I в упр. 56 образуют циклы Грея в тех же классах, что и их дополнения.) Следовательно, с помощью 16-ричного модульного кода Грея можно образовать четыре искомых цикла $(u_0u_0, u_0u_1, \ldots, u_0u_{15}, u_1u_{15}, \ldots, u_{15}u_0)$, $(u_0u_0, u_1u_0, \ldots, u_{15}u_0, u_{15}u_1, \ldots, u_0u_{15})$, $(v_0v_0, \ldots, v_{15}v_0)$, $(v_0v_0, \ldots, v_0v_{15})$.

Аналогично можно показать, что существует $n/2$ n-битовых циклов Грея с непересекающимися ребрами при n, равном 16, 32, 64 и т. д. [*Abhandlungen Math. Sem. Hamburg* **20** (1956), 13–16.] Ж. Обер (J. Aubert) и Б. Шнайдер (B. Schneider) [*Discrete Math.* **38** (1982), 7–16] доказали, что то же свойство выполняется для *всех* четных значений $n \ge 4$, но неизвестно ни одной простой конструкции.

83. Марк Кук (Mark Cooke) нашел следующее несимметричное решение в декабре 2002 года.

(1) 2737465057320265612316546743610525106052042416314372145101421737
2506246064173213107351607103156205713172463452102434643207054702
4147356146737625047350745130620656415073123731427376432561240264
3016735467532402524637475217640270736065105215106073575463253105

(2) 0616713417232175171671540460247164742473202531621673531632736052
6710141503047313570615453627623241426465272021632075363710750740
3157674761545652756510451024023107353424651230406545306213710537
2620501752453406703437343531502602463045627674152752406021610434

(3) 3701063751507131236243765735103012042353747207410473621617247324
6505132565057121565024570473247421427640231034362703262764130574
0560620341745613151756314702721725205613212604053506260460173642
6717641743513401245360241730636545061563027414535676432625745051

(4) 6706546435672147236210405432054510737405170532145431636430504673
4560621206416201320742373627204506473140171020514126107452343672
1320452752353410515426370601363567307105420163151210535061731236
4272537165617217542510760215462375452674257037346403647376271657

(Каждая из этих дельта-последовательностей должна начинаться с одной и той же вершины куба.) Имеется ли симметричное решение поставленной задачи?

84. Если обозначить исходное положение как $(2, 2)$, то 8-шаговое решение будет иметь вид, показанный на рис. A.1, т. е. представляет собой последовательность, доходящую до $(0, 0)$. Например, на первом шаге передняя половина веревки обходит вокруг и под правой "расческой", затем проходит через большую правую петлю. Среднюю линию на рисунке

Рис. А.1. Решение головоломки.

нужно рассматривать справа налево. Обобщение на случай n пар петель требует $3^n - 1$ шагов.

[Происхождение этой замечательной головоломки не совсем ясное. В книге *The Book of Ingenious & Diabolical Puzzles* Джерри Слокама (Jerry Slocum) и Джека Ботерманса (Jack Botermans) (1994) показаны версия головоломки с двумя петлями, вырезанная из рога и, вероятно, сделанная в Китае около 1850 года [с. 101], а также современная версия с шестью петлями из Малайзии (около 1988 года) [с. 93]. У Слокама имеется версия с четырьмя петлями, сделанная из бамбука в Англии примерно в 1884 году. Он нашел ее в книгах Henry Novra, *Catalogue of Conjuring Tricks and Puzzles* (1858 или 1859) и W. H. Cremer, *Games, Amusements, Pastimes and Magic* (1867), а также в каталоге Хамли (Hamley) 1895 года под названием "Удивительное каноэ". См. также *U.S. Patents 2091191* (1937), *D172310* (1954), *3758114* (1973), *D406866* (1999). Дикман (Dyckman) заметил связь данной головоломки с рефлексивным тернарным кодом Грея в письме Мартину Гарднеру (Martin Gardner), датированном 2 августа 1972 года.]

85. Согласно (50) элемент $\left[\begin{smallmatrix} b, & b' \\ t, & t' \end{smallmatrix}\right]$ произведения $\Gamma \wr \Gamma'$ представляет собой $\alpha_a \alpha'_{a'}$, если $\hat{g}(\left[\begin{smallmatrix} b, & b' \\ t, & t' \end{smallmatrix}\right]) = \left[\begin{smallmatrix} a, & a' \\ t, & t' \end{smallmatrix}\right]$ в рефлексивном коде Грея для оснований (t, t'). Теперь можно показать, что элемент $\left[\begin{smallmatrix} b, & b', & b'' \\ t, & t', & t'' \end{smallmatrix}\right]$ обоих произведений — и $(\Gamma \wr \Gamma') \wr \Gamma''$, и $\Gamma \wr (\Gamma' \wr \Gamma'')$ — представляет собой $\alpha_a \alpha'_{a'} \alpha''_{a''}$, если $\hat{g}(\left[\begin{smallmatrix} b, & b', & b'' \\ t, & t', & t'' \end{smallmatrix}\right]) = \left[\begin{smallmatrix} a, & a', & a'' \\ t, & t', & t'' \end{smallmatrix}\right]$ в рефлексивном коде Грея для оснований (t, t', t''). См. упр. 4.1–10, а также обратите внимание на правило смешанных оснований

$$m_1 \ldots m_n - 1 - \begin{bmatrix} x_1, & \ldots, & x_n \\ m_1, & \ldots, & m_n \end{bmatrix} = \begin{bmatrix} m_1 - 1 - x_1, & \ldots, & m_n - 1 - x_n \\ m_1, & \ldots, & m_n \end{bmatrix}.$$

В общем случае рефлексивный код Грея для оснований $(m_1, \ldots, m_n)$ представляет собой $(0, \ldots, m_1 - 1) \wr \cdots \wr (0, \ldots, m_n - 1)$. [*Information Processing Letters* **22** (1986), 201–205.]

86. Пусть Γ_{mn} — рефлексивный m-арный код Грея, который можно определить как $\Gamma_{m0} = \epsilon$ и

$$\Gamma_{m(n+1)} = (0, 1, \ldots, m-1) \wr \Gamma_{mn}, \qquad n \geq 0.$$

Этот путь проходит от $(0, 0, \ldots, 0)$ до $(m-1, 0, \ldots, 0)$ при четном m. Рассмотрим путь Грея Π_{mn}, определяемый как $\Pi_{m0} = \emptyset$ и

$$\Pi_{m(n+1)} = \begin{cases} (0, 1, \ldots, m-1) \wr \Pi_{mn},\ m\Gamma^R_{(m+1)n}, & \text{если } m \text{ нечетно;} \\ (0, 1, \ldots, m) \wr \Pi_{mn},\ m\Gamma^R_{mn}, & \text{если } m \text{ четно.} \end{cases}$$

Этот путь проходит по всем $(m+1)^n - m^n$ неотрицательным целым n-кортежам, для которых $\max(a_1, \ldots, a_n) = m$, начиная с $(0, \ldots, 0, m)$ и заканчивая $(m, 0, \ldots, 0)$. Искомый бесконечный путь Грея имеет вид Π_{0n}, Π^R_{1n}, Π_{2n}, Π^R_{3n}, $\ldots$.

87. Это невозможно при нечетном n, поскольку n-кортежи с $\max(|a_1|, \ldots, |a_n|) = 1$ включают $\frac{1}{2}(3^n + 1)$ кортежей с нечетной четностью, и $\frac{1}{2}(3^n - 3)$ — с четной. При $n = 2$ можно использовать спираль Σ_0, Σ_1, Σ_2, $\ldots$, где Σ_m закручивается против часовой стрелки от $(m, 1-m)$ до $(m, -m)$ при $m > 0$. Для четных значений $n \geq 2$, если T_m представляет собой путь n-кортежей от $(m, 1-m, m-1, 1-m, \ldots, m-1, 1-m)$ до $(m, -m, m, -m, \ldots, m, -m)$, можно использовать $\Sigma_m \wr (T_0, \ldots, T_{m-1})^R, (\Sigma_0, \ldots, \Sigma_m)^R \bar{\wr} T_m$ для $(n+2)$-кортежей с тем же свойством, где $\bar{\wr}$ — дуальная операция

$$\Gamma \bar{\wr} \Gamma' = (\alpha_0\alpha'_0, \ldots, \alpha_{t-1}\alpha'_0, \alpha_{t-1}\alpha'_1, \ldots, \alpha_0\alpha'_1, \alpha_0\alpha'_2, \ldots, \alpha_{t-1}\alpha'_2, \alpha_{t-1}\alpha'_3, \ldots).$$

[Бесконечные n-мерные коды Грея *без* ограничения величины впервые были построены в работе E. Vázsonyi, *Acta Litterarum ac Scientiarum*, sectio Scientiarum Mathematicarum **9** (Szeged: 1938), 163–173.]

88. Он вновь посетит все подлеса, но уже в обратном порядке, заканчивая $(0, \ldots, 0)$ и возвращаясь в состояние, которое имел после инициализирующего шага K1. (Этот принцип отражения является ключевым в понимании работы алгоритма K.)

89. (а) Пусть $M_0 = \epsilon$, $M_1 = \bullet$ и $M_{n+2} = \bullet M^R_{n+1}$, $-M^R_n$. Такая конструкция работает, поскольку последний элемент M^R_{n+1} является первым элементом M_{n+1}, а именно точкой, за которой следует первый элемент M^R_n.

(б) Для заданной строки $d_1 \ldots d_l$, где каждое d_j является $\bullet$ или $-$, можно найти ее преемника, полагая $k = l - [d_l = \bullet]$ и действуя следующим образом: если k нечетно и $d_k = \bullet$, заменить $d_k d_{k+1}$ на $-$; если k четно и $d_k = -$, заменить d_k на $\bullet\bullet$; в противном случае уменьшить k на 1 и повторять описанные действия до тех пор, пока не произойдут изменения или пока не будет достигнуто значение $k = 0$. Следующим за указанным в условии словом является слово $\bullet - - \bullet\bullet\bullet\bullet - \bullet - \bullet$.

90. Цикл может существовать только тогда, когда количество слов кода четно, поскольку число тире изменяется на каждом шаге на ± 1. Таким образом, должно быть $n \bmod 3 = 2$. Пути Грея M_n из упр. 89 не подходят; они начинаются с $(\bullet -)^{\lfloor n/3 \rfloor} \bullet^{n \bmod 3}$ и заканчиваются $(- \bullet)^{\lfloor n/3 \rfloor} \bullet^{[n \bmod 3 = 1]} -^{[n \bmod 3 = 2]}$. Но $M_{3k+1}\bullet$, $M^R_{3k}-$ представляет собой гамильтонов цикл в графе кода Морзе при $n = 3k + 2$.

91. Указанное условие эквивалентно тому, что n-кортежи $a_1\bar{a}_2a_3\bar{a}_4\ldots$ не имеют двух последовательных единиц. Такие n-кортежи соответствуют последовательностям кода Морзе длиной $n + 1$, если добавить 0, а затем представить $\bullet$ и $-$ соответственно как 0 и 10. В таком случае можно преобразовать путь M_{n+1} из упр. 89 в процедуру наподобие алгоритма K с каймой, содержащей индексы начала каждой точки и тире (за исключением последней точки).

U1. [Инициализация.] Установить $a_j \leftarrow \lfloor ((j-1) \bmod 6)/3 \rfloor$ и $f_j \leftarrow j$ для $1 \leq j \leq n$. Также установить $f_0 \leftarrow 0$, $r_0 \leftarrow 1$, $l_1 \leftarrow 0$, $r_j \leftarrow j + (j \bmod 3)$ и $l_{j+(j \bmod 3)} \leftarrow j$ для

$1 \le j \le n$, за исключением того, что если $j + (j \bmod 3) > n$, то установить $r_j \leftarrow 0$ и $l_0 \leftarrow j$. (Сейчас "кайма" содержит 1, 2, 4, 5, 7, 8, ...)

U2. [Посещение.] Посетить n-кортеж $(a_1, \ldots, a_n)$.

U3. [Выбор p.] Установить $q \leftarrow l_0$, $p \leftarrow f_q$, $f_q \leftarrow q$.

U4. [Проверка a_p.] Завершить работу алгоритма, если $p = 0$. В противном случае установить $a_p \leftarrow 1 - a_p$ и перейти к шагу U6, если сейчас $a_p + p$ четно.

U5. [Вставка $p + 1$.] Если $p < n$, установить $q \leftarrow r_p$, $l_q \leftarrow p + 1$, $r_{p+1} \leftarrow q$, $r_p \leftarrow p + 1$, $l_{p+1} \leftarrow p$. Перейти к шагу U7.

U6. [Удаление $p + 1$.] Если $p < n$, установить $q \leftarrow r_{p+1}$, $r_p \leftarrow q$, $l_q \leftarrow p$.

U7. [Пассивизация p.] Установить $f_p \leftarrow f_{l_p}$ и $f_{l_p} \leftarrow l_p$. Вернуться к шагу U2. ▮

Этот алгоритм можно также получить как частный случай существенно более общего метода Ганга Ли (Gang Li), Фрэнка Раски (Frank Ruskey) и Д. Э. Кнута (D. E. Knuth), который расширяет алгоритм K, позволяя пользователю указать для каждой пары (p, q) типа (родитель, потомок) либо $a_p \ge a_q$, либо $a_p \le a_q$. [См. Knuth and Ruskey, *Lecture Notes in Computer Science* **2635** (2004), 183–204.] Обобщение в другом направлении, которое порождает все строки длиной n, которые не содержат определенных подстрок, открыто М. Б. Сквайром (M. B. Squire), *Electronic J. Combinatorics* **3** (1996), #R17, 1–29.

Кстати, забавно отметить, что отображение $k \mapsto g(k)/2$ представляет собой взаимно однозначное соответствие между всеми бинарными n-кортежами, в которых нет серий единиц нечетной длины, и всеми бинарными n-кортежами, в которых нет последовательных единиц.

92. Да, поскольку ориентированный граф всех $(n-1)$-кортежей $(x_1, \ldots, x_{n-1})$ с $x_1, \ldots, x_{n-1} \le m$ и с дугами $(x_1, \ldots, x_{n-1}) \to (x_2, \ldots, x_n)$ в случае $\max(x_1, \ldots, x_n) = m$ является связным и сбалансированным; см. теорему 2.3.4.2G. Действительно, такая последовательность получается из алгоритма F, если заметить, что последние k^n элементов простых строк длины, делящей n, при вычитании из $m-1$ одни и те же для всех $m \ge k$. Например, при $n = 4$ первой 81 цифрой последовательности Φ_4 является $2-\alpha^R = 0\,0001\,01\,0011\ldots$, где α — строка (62). [Существуют также бесконечные m-арные последовательности, первые m^n элементов которых представляют собой циклы де Брейна для всех n для заданного фиксированного $m \ge 3$. См. L. J. Cummings and D. Wiedemann, *Cong. Numerantium* **53** (1986), 155–160.]

93. Цикл, генерируемый $f()$, является циклической перестановкой $\alpha 1$, где α имеет длину $m^n - 1$ и завершается 1^{n-1}. Цикл, генерируемый алгоритмом R, представляет собой циклическую перестановку $\gamma = c_0 \ldots c_{m^{n+1}-1}$, где $c_k = (c_0 + b_0 + \cdots + b_{k-1}) \bmod m$ и $b_0 \ldots b_{m^{n+1}-1} = \beta = \alpha^m 1^m$.

Если $x_0 \ldots x_n$ имеется в γ (скажем, $x_j = c_{k+j}$ для $0 \le j \le n$), то $y_j = b_{k+j}$ для $0 \le j < n$, где $y_j = (x_{j+1} - x_j) \bmod m$. [Это связь с модульным m-арным кодом Грея; см. упр. 78.] Теперь, если $y_0 \ldots y_{n-1} = 1^n$, то имеем $m^{n+1} - m - n < k \le m^{n+1} - n$; в противном случае существует индекс k', такой, что $-n < k' < m^n - n$ и $y_0 \ldots y_{n-1}$ встречается в β в позициях $k = (k' + r(m^n - 1)) \bmod m^{n+1}$ для $0 \le r < m$. В обоих случаях m вариантов выбора k имеют различные значения x_0, поскольку сумма всех элементов в α равна $m-1$ (по модулю m) для $n \ge 2$. [Алгоритм R корректен и при $n = 1$, если $m \bmod 4 \ne 2$, поскольку в этом случае $m \perp \sum \alpha$.]

94. 0<u>10</u>20304<u>1</u><u>21</u>3142<u>3</u>243344. (Подчеркнутые цифры вставлены в чередование 00112234 с 34. Алгоритм D можно использовать в общем случае при $n = 1$ и $r = m - 2 \ge 0$; однако это бессмысленно ввиду (54).)

95. (a) Пусть $c_0 c_1 c_2 \ldots$ имеет период r. Если r нечетно, то $p = q = r$, так что $r = pq$ только в тривиальном случае $p = q = 1$ и $a_0 = b_0$. В противном случае, согласно 4.5.2–(10),

$r/2 = \operatorname{lcm}(p,q) = pq/\gcd(p,q)$, следовательно, $\gcd(p,q) = 2$. В последнем случае $2n$-кортежи $c_l c_{l+1} \ldots c_{l+2n-1}$ будут иметь вид $a_j b_k \ldots a_{j+n-1} b_{k+n-1}$ для $0 \le j < p$, $0 \le k < q$, $j \equiv k$ (по модулю 2) и $b_k a_j \ldots b_{k+n-1} a_{j+n-1}$ для $0 \le j < p$, $0 \le k < q$, $j \not\equiv k$ (по модулю 2).

(б) В выводе будут перемежаться две последовательности, $a_0 a_1 \ldots$ и $b_0 b_1 \ldots$, периоды которых соответственно равны $m^n + r$ и $m^n - r$; последовательность из a образует цикл $f()$ с x^n, замененными на x^{n+1}, а последовательность из b образует цикл $f'()$ с x^n, замененными на x^{n-1}, для $0 \le x < r$. Согласно (58) и п. (а) период равен $m^{2n} - r^2$ и каждый $2n$-кортеж встречается с исключением $(xy)^n$ для $0 \le x, y < r$.

(в) Реальный шаг D6 изменяет поведение (б) переходом к D3 при $t \ge n$, $t' = n$ и $0 \le x' = x < r$; это изменение приводит к выводу дополнительного x сразу после вывода x^{2n-1} и перед выводом b, где b представляет собой цифру, следующую в цикле за x^n. D6 также позволяет передать управление D7, а затем D3 с $t' = n$ в случае $t \ge n$ и $x < x' < r$; это поведение приводит к выводу дополнительного $x'x$ сразу после вывода $(xx')^{n-1}x$ с последующим выводом b. Эти r^2 дополнительных цифр дают r^2 недостающих $2n$-кортежей из (б).

96. (а) Например, когда $n = 5$, сопрограмма верхнего уровня типа R вызывает сопрограмму типа D для $n = 4$, которая вызывает две сопрограммы типа S для $n = 2$; следовательно, $R_5 = D_5 = 1$ и $S_5 = 2$. Рекуррентные соотношения $R_2 = 0$, $R_{2n+1} = 1 + R_{2n}$, $R_{2n} = 2R_n$, $D_2 = 0$, $D_{2n+1} = D_{2n} = 1 + 2D_n$, $S_2 = 1$, $S_{2n+1} = S_{2n} = 2S_n$ имеют решение $R_n = n - 2S_n$, $D_n = S_n - 1$, $S_n = 2^{\lfloor \lg n \rfloor - 1}$. Таким образом, $R_n + D_n + S_n = n - 1$.

(б) Каждый вывод верхнего уровня обычно включает $\lfloor \lg n \rfloor - 1$ D-активизаций и $\nu(n) - 1$ R-активизаций, плюс одну базовую активизацию на нижнем уровне. Однако имеется исключение: алгоритм R может вызвать свою сопрограмму $f()$ дважды, если первая активизация завершает последовательность 1^n; и иногда алгоритму R вызов $f()$ не требуется вовсе. Алгоритм D может вызывать свои сопрограммы $f'()$ дважды, если первая активизация завершает последовательность $(x')^n$ для $x' < r$; но иногда алгоритму D вовсе не требуются вызовы $f()$ или $f'()$.

Алгоритм R завершает последовательность x^{n+1} тогда и только тогда, когда его дочерняя сопрограмма $f()$ только что завершила последовательность 0^n. Алгоритм D завершает последовательность x^{2n} для $x < r$ тогда и только тогда, когда он только что совершил переход от D6 к D3 без вызова какой-либо дочерней сопрограммы.

Из этих наблюдений можно заключить, что, когда сопрограмма для m^n-цикла производит последнюю цифру серии x^n или первую цифру, следующую за такой серией, никаких исключений не возникает. Следовательно, наихудший случай реализуется тогда, когда сопрограмма верхнего уровня активизирует подсопрограмму дважды, всего выполняя $2\lfloor \lg n \rfloor + 2\nu(n) - 3$ активизаций.

97. (а) (0011), (00011101), (0000101001111011) и (00000110001011011111001110101001). Таким образом, $j_2 = 2$, $j_3 = 3$, $j_4 = 9$, $j_5 = 15$.

(б) Очевидно, что $f_{n+1}(k) = \Sigma f_n(k) \bmod 2$ для $0 \le k < j_n + n$. Следующее значение, $f_{n+1}(j_n + n)$, зависит от того, выполняется ли переход от шага R4 к шагу R2 после вычисления $y = f_n(j_n + n - 1)$. Если выполняется (а именно, если $f_{n+1}(j_n + n - 1) \ne 0$), то мы имеем $f_{n+1}(k) \equiv 1 + \Sigma f_n(k+1)$ для $j_n + n \le k < 2^n + j_n + n$; в противном случае $f_{n+1}(k) \equiv 1 + \Sigma f_n(k-1)$ для таких значений k. В частности, $f_{n+1}(k) = 1$ при $2^n \le k + \delta_n \le 2^n + n$. Предложенная формула, которая имеет более простые диапазоны для индекса k, выполняется потому, что $1 + \Sigma f_n(k \pm 1) \equiv \Sigma f_n(k)$, когда $j_n < k < j_n + n$ или $2^n + j_n < k < 2^n + j_n + n$.

(в) Перемежаемый цикл имеет $c_n(2k) = f_n^+(k)$ и $c_n(2k+1) = f_n^-(k)$, где

$$f_n^+(k) = \begin{cases} f_n(k-1), & \text{если } 0 < k \le j_n + 1; \\ f_n(k-2), & \text{если } j_n + 1 < k \le 2^n + 2; \end{cases} \qquad f_n^-(k) = \begin{cases} f_n(k+1), & \text{если } 0 \le k < j_n; \\ f_n(k+2), & \text{если } j_n \le k < 2^n - 2; \end{cases}$$

$f_n^+(k) = f_n^+(k \bmod (2^n{+}2))$, $f_n^-(k) = f_n^-(k \bmod (2^n{-}2))$. Следовательно, последовательность 1^{2n-1} начинается с позиции $k_n = (2^{n-1}-2)(2^n+2)+2j_n+2$ в цикле c_n; это делает j_{2n} нечетным. Подпоследовательность $(01)^{n-1}0$ начинается с позиции $l_n = (2^{n-1}+1)(j_n-1)$, если $j_n \bmod 4 = 1$, и с позиции $l_n = (2^{n-1}+1)(2^n+j_n-3)$, если $j_n \bmod 4 = 3$. Кроме того, $k_2 = 6$, $l_2 = 2$.

(г) Алгоритм D вставляет четыре элемента в цикл c_n; следовательно,

$$\text{когда } j_n \bmod 4{<}3\ (l_n{<}k_n):\qquad \text{когда } j_n \bmod 4{=}3\ (k_n{<}l_n):$$

$$f_{2n}(k)=\begin{cases} c_n(k{-}1), & \text{если } 0{<}k{\le}l_n{+}2;\\ c_n(k{-}3), & \text{если } l_n{+}2{<}k{\le}k_n{+}3;\\ c_n(k{-}4), & \text{если } k_n{+}3{<}k{\le}2^{2n};\end{cases} =\begin{cases} c_n(k{-}1), & \text{если } 0{<}k{\le}k_n{+}1;\\ c_n(k{-}2), & \text{если } k_n{+}1{<}k{\le}l_n{+}3;\\ c_n(k{-}4), & \text{если } l_n{+}3{<}k{\le}2^{2n}.\end{cases}$$

(д) Следовательно, $j_{2n} = k_n + 1 + 2[j_n \bmod 4 < 3]$. Действительно, элементы, предшествующие 1^{2n}, состоят из $2^{n-2}-1$ полных периодов $f_n^+()$, перемежаемых 2^{n-2} полными периодами $f_n^-()$, с одним вставленным элементом 0, а также с элементом 10, вставленным, если $l_n < k_n$, после чего следует $f_n(1)f_n(1)f_n(2)f_n(2)\ldots f_n(j_n-1)f_n(j_n-1)$. Сумма всех этих элементов нечетна, если только не выполняется условие $l_n < k_n$; следовательно, $\delta_{2n} = 1 - 2[j_n \bmod 4 = 3]$.

Пусть $n = 2^t q$, где q нечетно и $n > 2$. Эти рекурентные соотношения подразумевают, что, если $q = 1$, мы имеем $j_n = 2^{n-1} + b_t$, где $b_t = 2^t/3 - (-1)^t/3$. А если $q > 1$, то мы имеем $j_n = 2^{n-1} \pm b_{t+2}$, где знак + выбирается тогда и только тогда, когда $\lfloor \lg q \rfloor + [\lfloor 4q/2^{\lfloor \lg q \rfloor} \rfloor = 5]$ четно.

98. Если $f(k) = g(k)$, когда k лежит в определенном диапазоне, имеется константа C, такая, что $\Sigma f(k) = C + \Sigma g(k)$ для k в этом диапазоне. Следовательно, можно продолжить и почти без привлечения головного мозга вывести дополнительные рекуррентные соотношения: если $n > 1$, то

$$\Sigma f_{2n}(k),\ \text{когда } j_n \bmod 4 < 3\ (l_n < k_n):\qquad \text{когда } j_n \bmod 4 = 3\ (k_n < l_n):$$

$$\equiv\begin{cases} \Sigma c_n(k{-}1), & \text{если } 0 < k \le l_n{+}2;\\ 1{+}\Sigma c_n(k{-}3), & \text{если } l_n{+}2 < k \le k_n{+}3;\\ \Sigma c_n(k{-}4), & \text{если } k_n{+}3 < k \le 2^{2n};\end{cases} \equiv\begin{cases} \Sigma c_n(k{-}1), & \text{если } 0 < k \le k_n{+}1;\\ 1{+}\Sigma c_n(k{-}2), & \text{если } k_n{+}1 < k \le l_n{+}3;\\ \Sigma c_n(k{-}4), & \text{если } l_n{+}3 < k \le 2^{2n}.\end{cases}$$

$$\Sigma c_n(k) \equiv \Sigma f_n^+(\lceil k/2 \rceil) + \Sigma f_n^-(\lfloor k/2 \rfloor).$$

$$\Sigma f_n^+(k)\equiv\begin{cases} \Sigma f_n(k{-}1), & \text{если } 0{<}k{\le}j_n{+}1;\\ 1{+}\Sigma f_n(k{-}2), & \text{если } j_n{+}1{<}k{\le}2^n{+}2;\end{cases}\qquad \Sigma f_n^-(k)\equiv\begin{cases} \Sigma f_n(k{+}1), & \text{если } 0{\le}k{<}j_n;\\ 1{+}\Sigma f_n(k{+}2), & \text{если } j_n{\le}k{<}2^n{-}2;\end{cases}$$

$$\Sigma f_n^{\pm}(k) \equiv \lfloor k/(2^n \pm 2) \rfloor + \Sigma f_n^{\pm}(k \bmod (2^n \pm 2)); \quad \Sigma f_n(k) \equiv \Sigma f_n(k \bmod 2^n).$$

$$\Sigma f_{2n+1}(k) \equiv \begin{cases} \Sigma\Sigma f_{2n}(k), & \text{если } 0 < k \le j_{2n} \text{ или } 2^{2n} + j_{2n} < k \le 2^{2n+1};\\ 1 + k + \Sigma\Sigma f_{2n}(k + \delta_{2n}), & \text{если } j_{2n} < k \le 2^{2n} + j_{2n}.\end{cases}$$

$$\Sigma\Sigma f_{2n}(k),\ \text{когда } j_n \bmod 4{<}3\ (l_n{<}k_n):\qquad \text{когда } j_n \bmod 4{=}3\ (k_n{<}l_n):$$

$$\equiv\begin{cases} \Sigma\Sigma c_n(k{-}1), & \text{если } 0{<}k{\le}l_n{+}2;\\ 1{+}k{+}\Sigma\Sigma c_n(k{-}3), & \text{если } l_n{+}2{<}k{\le}k_n{+}3;\\ \Sigma\Sigma c_n(k{-}4), & \text{если } k_n{+}3{<}k{\le}2^{2n};\end{cases} \equiv\begin{cases} \Sigma\Sigma c_n(k{-}1), & \text{если } 0{<}k{\le}k_n{+}1;\\ 1{+}k{+}\Sigma\Sigma c_n(k{-}2), & \text{если } k_n{+}1{<}k{\le}l_n{+}3;\\ 1{+}\Sigma\Sigma c_n(k{-}4), & \text{если } l_n{+}3{<}k{\le}2^{2n}.\end{cases}$$

$$\Sigma\Sigma f_{2n}(k) \equiv [j_n \bmod 4 < 3]\lfloor k/2^{2n} \rfloor + \Sigma\Sigma f_{2n}(k \bmod 2^{2n}).$$

И тогда имеется замыкание:

$$\Sigma\Sigma c_n(2k) = \Sigma f_n^+(k), \qquad \Sigma\Sigma c_n(2k+1) = \Sigma f_n^-(k).$$

Если $n = 2^t q$, где q нечетно, то время вычисления $f_n(k)$ с помощью этой системы рекурсивных формул составляет $O(t + S(q))$, где $S(1) = 1$, $S(2k) = 1 + 2S(k)$ и $S(2k+1) =$

$1 + S(k)$. Ясно, что $S(k) < 2k$, так что вычисление включает не более $O(n)$ простых операций с n-битовыми числами. На самом деле этот метод часто значительно быстрее: если усреднить $S(k)$ по всем k, таким, что $\lfloor \lg k \rfloor = s$, то мы получим $(3^{s+1} - 2^{s+1})/2^s$, что меньше, чем $3k^{\lg(3/2)} < 3k^{0.59}$. (Кстати, если $k = 2^{s+1} - 1 - (2^{s-e_1} + 2^{s-e_2} + \cdots + 2^{s-e_t})$, где $0 < e_1 < \cdots < e_t$, мы получим $S(k) = s + 1 + e_t + 2e_{t-1} + 4e_{t-2} + \cdots + 2^{t-1}e_1$.)

99. Строка, начинающаяся с позиции k в $f_n()$, начинается с позиции $k^+ = k + 1 + [k > j_n]$ в $f_n^+()$ и с позиции $k^- = k - 1 - [k > j_n]$ в $f_n^-()$, с тем исключением, что 0^n и 1^n дважды встречаются в $f_n^+()$, но их вовсе нет в $f_n^-()$.

Чтобы найти $\gamma = a_0 b_0 \ldots a_{n-1} b_{n-1}$ в цикле $f_{2n}()$, обозначим $\alpha = a_0 \ldots a_{n-1}$ и $\beta = b_0 \ldots b_{n-1}$. Предположим, что в $f_n()$ α начинается с позиции j, а β — с позиции k и что ни α, ни β не представляет собой 0^n или 1^n. Если $j^+ \equiv k^+$ (по модулю 2), положим, что $l/2$ является решением уравнения $j^+ + (2^n + 2)x = k^- + (2^n - 2)y$; можно получить $l/2 = k + (2^n - 2)(2^{n-3}(j - k) \bmod (2^{n-1} + 1))$, если $j \geq k$, в противном случае $l/2 = j + (2^n + 2)(2^{n-3}(k - j) \bmod (2^{n-1} - 1))$. Иначе положим $(l - 1)/2 = k^+ + (2^n + 2)x = j^- + (2^n - 2)y$. Тогда γ начинается с позиции l в цикле $c_n()$; следовательно, она начинается с позиции $l + 1 + [l \geq k_n] + 2[l \geq l_n]$ в цикле $f_{2n}()$. Аналогичные формулы выполняются, когда справедливо условие $\alpha \in \{0^n, 1^n\}$ или $\beta \in \{0^n, 1^n\}$ (но не оба условия одновременно). Наконец 0^{2n}, 1^{2n}, $(01)^n$ и $(10)^n$ нечинаются соответственно в позициях 0, j_{2n}, $l_n + 1 + [k_n < l_n]$ и $l_n + 2 + [k_n < l_n]$.

Чтобы найти $\beta = b_0 b_1 \ldots b_n$ в $f_{n+1}()$ для четного n, предположим, что n-битовая строка $(b_0 \oplus b_1) \ldots (b_{n-1} \oplus b_n)$ начинается с позиции j в $f_n()$. Тогда β начинается с позиции $k = j - \delta_n[j \geq j_n] + 2^n[j = j_n][\delta_n = 1]$, если $f_{n+1}(k) = b_0$, в противном случае с позиции $k + (2^n - \delta_n, \delta_n, 2^n + \delta_n)$ для соответственно $(j < j_n, j = j_n, j > j_n)$.

Время работы этой рекурсии удовлетворяет уравнению $T(n) = O(n) + 2T(\lfloor n/2 \rfloor)$, так что оно представляет собой $O(n \log n)$. [Упр. 97–99 основаны на работе Д. Тулиани (J. Tuliani), который также разработал методы для некоторых бóльших значений m; см. *Discrete Math.* **226** (2001), 313–336.]

100. Даже если очевидные дефекты не заметны, прежде чем рекомендовать любую последовательность, следует провести тщательное тестирование. Напротив, цикл де Брейна, неявно создаваемый алгоритмом F, является неудачным источником предположительно случайных битов, даже если он имеет n-распределение в смысле определения 3.5D, поскольку в начале доминируют нули. Действительно, когда n — простое число, биты $tn + 1$ этой последовательности нулевые для $0 \leq t < (2^n - 2)/n$.

101. (а) Пусть β — собственный суффикс $\lambda\lambda'$ с $\beta \leq \lambda\lambda'$. Либо β является суффиксом λ', откуда $\lambda < \lambda' \leq \beta$, либо $\beta = \alpha\lambda'$ и мы имеем $\lambda < \alpha < \beta$.

Теперь $\lambda < \beta \leq \lambda\lambda'$ влечет за собой то, что $\beta = \lambda\gamma$ для некоторого $\gamma \leq \lambda'$. Но γ является суффиксом β с $1 \leq |\gamma| = |\beta| - |\lambda| < |\lambda'|$; следовательно, γ является собственным суффиксом λ' и $\lambda' < \gamma$. Получаем противоречие.

(б) Любая строка единичной длины — простая. Объединим смежные простые строки согласно п. (а) в любом порядке, пока не будет невозможно никакое дальнейшее объединение. [Более общие результаты можно найти в работе M. P. Schützenberger, *Proc. Amer. Math. Soc.* **16** (1965), 21–24.]

(в) Если $t \neq 0$, то пусть λ является наименьшим суффиксом $\lambda_1 \ldots \lambda_t$. Тогда λ — простая строка по определению и имеет вид $\beta\gamma$, где β — непустой суффикс некоторой строки λ_j. Следовательно, $\lambda_t \leq \lambda_j \leq \beta \leq \beta\gamma = \lambda \leq \lambda_t$, так что должно выполняться $\lambda = \lambda_t$. Удалим λ_t и будем повторять эти действия до тех пор, пока не окажется выполненным условие $t = 0$.

(г) Истинно. Если $\alpha = \lambda\beta$ для некоторой простой строки λ с $|\lambda| > |\lambda_1|$, то можно было бы добавить множители β и получить другое разложение α.

(д) $3 \cdot 14159265358979323846264338327 95 \cdot 02884197$. (Эффективный алгоритм представлен в упр. 106. Знание большего количества цифр π не приведет к изменению первых двух множителей. Бесконечное десятичное расширение любого числа, являющегося "нормальным" в смысле Бореля (Borel) (см. раздел 3.5) раскладывается на простые строки конечной длины.)

102. Мы должны получить $1/(1-mz) = 1/\prod_{n=1}^{\infty}(1-z^n)^{L_m(n)}$. Отсюда вытекает (60), как в упр. 4.6.2–4.

103. Когда $n = p$ простое, (59) гласит, что $L_m(1) + pL_m(p) = m^p$, и мы также имеем $L_m(1) = m$. [Это комбинаторное доказательство интересно контрастирует с традиционным алгебраическим доказательством теоремы 1.2.4F.]

104. 4483 непростыми строками являются `abaca`, `agora`, `ahead`, ...; 1274 простыми строками являются ..., `rusts`, `rusty`, `rutty`. (Поскольку строка `prime` (`простая`) не является простой, стоило бы называть простые строки `lowly` (`скромными`).)

105. (а) Пусть α' представляет собой α с увеличенной последней буквой, и предположим, что $\alpha' = \beta\gamma'$, где $\alpha = \beta\gamma$ и $\beta \neq \epsilon$, $\gamma \neq \epsilon$. Пусть θ — префикс α с $|\theta| = |\gamma|$. Согласно предположению существует строка ω, такая, что $\alpha\omega$ — простая строка; следовательно, $\theta \le \alpha\omega < \gamma\omega$, так что должно выполняться $\theta \le \gamma$. Поэтому $\theta < \gamma'$, и мы имеем $\alpha' < \gamma'$.

(б) Пусть $\alpha = \lambda_1\beta = a_1 \ldots a_n$, где $\lambda_1\beta\omega$ — простая строка. Из условия $\lambda_1\beta\omega < \beta\omega$ вытекает, что $a_j \le a_{j+r}$ для $1 \le j \le n-r$, где $r = |\lambda_1|$. Но $a_j < a_{j+r}$ не может быть; в противном случае α начиналась бы с простой строки, более длинной, чем λ_1, что противоречит упр. 101, (г).

(в) Если α представляет собой n-расширение и λ, и λ', где $|\lambda| > |\lambda'|$, то мы должны получить $\lambda = (\lambda')^q\theta$, где θ — непустой префикс λ'. Но тогда $\theta \le \lambda' < \lambda < \theta$.

106. **E1.** [Инициализация.] Установить $a_1 \leftarrow \cdots \leftarrow a_n \leftarrow m-1$, $a_{n+1} \leftarrow -1$ и $j \leftarrow 1$.

E2. [Посещение.] Посетить $(a_1, \ldots, a_n)$ с индексом j.

E3. [Вычитание единицы.] Завершить работу алгоритма, если $a_j = 0$. В противном случае установить $a_j \leftarrow a_j - 1$ и $a_k \leftarrow m-1$ для $j < k \le n$.

E4. [Подготовка к разложению.] (Согласно упр. 105, (б), нужно найти первый простой множитель λ_1 строки $a_1 \ldots a_n$.) Установить $j \leftarrow 1$ и $k \leftarrow 2$.

E5. [Поиск нового j.] (Сейчас $a_1 \ldots a_{k-1}$ представляет собой $(k-1)$-расширение простой строки $a_1 \ldots a_j$.) Если $a_{k-j} > a_k$, вернуться к шагу E2. В противном случае, если $a_{k-j} < a_k$, установить $j \leftarrow k$. Затем увеличить k на 1 и повторить этот шаг. ▌

Эффективный алгоритм разложения на шагах E4 и E5 разработан Ж.-П. Дювалем (J. P. Duval), *J. Algorithms* **4** (1983), 363–381. Более подробную информацию можно найти в Cattell, Ruskey, Sawada, Serra and Miers, *J. Algorithms* **37** (2000), 267–282.

107. Количество посещенных n-кортежей равно $P_m(n) = \sum_{j=1}^{n} L_m(j)$. Поскольку $L_m(n) = \frac{1}{n}m^n + O(m^{n/2}/n)$, имеем $P_m(n) = Q(m,n) + O(Q(\sqrt{m}, n))$, где

$$Q(m,n) = \sum_{k=1}^{n} \frac{m^k}{k} = \frac{m^n}{n} R(m,n);$$

$$R(m,n) = \sum_{k=0}^{n-1} \frac{m^{-k}}{1-k/n} = \sum_{k=0}^{n/2} \frac{m^{-k}}{1-k/n} + O(nm^{-n/2})$$

$$= \frac{m}{m-1} \sum_{j=0}^{t-1} \frac{1}{n^j} \sum_{l} \left\{ {j \atop l} \right\} \frac{l!}{(m-1)^l} + O(n^{-t}) \quad \text{для всех } t.$$

Таким образом, $P_m(n) \sim m^{n+1}/((m-1)n)$. Основной вклад во время выполнения дают циклы на шагах F3 и F5 стоимостью $n-j$ для каждой простой строки длиной j. Следовательно, общее время равно $nP_m(n) - \sum_{j=1}^{n} jL_m(j) = m^{n+1}(1/((m-1)^2 n) + O(1/(mn^2)))$. Это меньше времени, необходимого для вывода m^n отдельных цифр цикла де Брейна.

108. (а) Если $\alpha \neq 9\ldots9$, то имеем $\lambda_{k+1} \le \beta 9^{|\alpha|}$, поскольку последняя строка является простой.

(б) Можно считать, что β не состоит только из нулей, поскольку $9^j 0^{n-j}$ является подстрокой $\lambda_{t-1}\lambda_t\lambda_1\lambda_2 = 89^n 0^n 1$. Пусть k — минимальное значение, для которого $\beta \le \lambda_k$; тогда $\lambda_k \le \beta\alpha$, так что β представляет собой префикс λ_k. Поскольку β — предпростая строка, она является $|\beta|$-расширением некоторого простого числа $\beta' \le \beta$. Предпростая строка, посещенная алгоритмом F непосредственно перед β', согласно упр. 106 — $(\beta' - 1)9^{n-|\beta'|}$, где $\beta'-1$ означает десятичное число, на единицу меньшее β'. Таким образом, если β' не является λ_{k-1}, из указания (которое также следует из упр. 106) вытекает, что λ_{k-1} заканчивается как минимум $n-|\beta'| \ge n-|\beta|$ девятками, а α является суффиксом λ_{k-1}. С другой стороны, если $\beta' = \lambda_{k-1}$, α является суффиксом λ_{k-2}, а β — префиксом $\lambda_{k-1}\lambda_k$.

(в) Если $\alpha \neq 9\ldots9$, имеем $\lambda_{k+1} \le (\beta\alpha)^{d-1}\beta 9^{|\alpha|}$, поскольку последняя строка простая. В противном случае λ_{k-1} заканчивается как минимум $(d-1)|\beta\alpha|$ девятками, а $\lambda_{k+1} \le (\beta\alpha)^{d-1}9^{|\beta\alpha|}$, так что $(\alpha\beta)^d$ является подстрокой $\lambda_{k-1}\lambda_k\lambda_{k+1}$.

(г) Указанные подстроки появляются в простых строках 135899 135914, 787899 787979, 129999 13 131314, 09 090911, 089999 09 090911, 118999 119 119122.

(д) Да: во всех случаях позиция $a_1\ldots a_n$ предшествует позиции подстроки $a_1\ldots a_{n-1}(a_n+1)$, если $0 \le a_n < 9$ (и если мы считаем, что строки наподобие 9^j0^{n-j} находятся в начале). Кроме того, 9^j0^{n-j-1} располагается только после появления $9^{j-1}0^{n-j}a$ для $1 \le a \le 9$, так что мы не должны размещать 0 после 9^j0^{n-j-1}.

109. Предположим, что мы хотим найти подматрицу

$$\begin{pmatrix} (w_{n-1}\ldots w_1w_0)_2 & (x_{n-1}\ldots x_1x_0)_2 \\ (y_{n-1}\ldots y_1y_0)_2 & (z_{n-1}\ldots z_1z_0)_2 \end{pmatrix}.$$

Бинарный случай $n=1$ уже рассмотрен в качестве примера в условии упражнения, а если $n>1$, то по индукции можно считать, что требуется найти только старшие биты a_{2n-1}, a_{2n-2}, b_{2n-1} и b_{2n-2}. Случай $n=3$ типичен: нужно решить систему уравнений

$$\begin{array}{rrrrl}
b_5 = w_2, & b_4 = x_2, & a_5 \oplus b_5 = y_2, & a_4 \oplus b_4 = z_2, & \text{если } a_0 = 0,\ b_0 = 0; \\
b_4 = w_2, & b'_5 = x_2, & a_4 \oplus b_4 = y_2, & a_5 \oplus b'_5 = z_2, & \text{если } a_0 = 0,\ b_0 = 1; \\
a_5 \oplus b_5 = w_2, & a_4 \oplus b_4 = x_2, & b_5 = y_2, & b_4 = z_2, & \text{если } a_0 = 1,\ b_0 = 0; \\
a_4 \oplus b_4 = w_2, & a_5 \oplus b'_5 = x_2, & b_4 = y_2, & b'_5 = z_2, & \text{если } a_0 = 1,\ b_0 = 1;
\end{array}$$

здесь $b'_5 = b_5 \oplus b_4b_3b_2b_1$ учитывают переносы, когда j становится равным $j+1$.

110. Пусть $a_0a_1\ldots a_{m^2-1}$ является m-арным циклом де Брейна, таким, как первые m^2 элементов (54). Если m нечетно, положим $d_{ij} = a_j$ для четного i и $d_{ij} = a_{(j+(i-1)/2) \bmod m^2}$ для нечетного. [Первым из множества изобретателей этого построения, похоже, был Джон К. Кок (John C. Cock), который также построил торы де Брейна других форм и размеров в *Discrete Math.* **70** (1988), 209–210.]

Если $m = m'm''$, где $m' \perp m''$, воспользуемся китайским алгоритмом остатков для определения

$$d_{ij} \equiv d'_{ij} \ \text{(по модулю } m') \qquad \text{и} \qquad d_{ij} \equiv d''_{ij} \ \text{(по модулю } m'')$$

через матрицы, которые решают эту задачу для m' и m''. Таким образом, предыдущее упражнение приводит к решению для произвольного m.

Другое интересное решение для четных значений m было найдено Анталом Ивани (Antal Iványi) и Золтаном Тотом (Zoltán Tóth) [*2nd Conf. Automata, Languages, and Programming Systems* (1988), 165–172; см. также Hurlbert and Isaak, *Contemp. Math.* **178** (1994), 153–160]. Первые m^2 элементов a_j бесконечной последовательности

$$0011\,021331203223\,04152435534251405445\,0617263746577564\ldots07667\,08\ldots$$

определяют цикл де Брейна с тем свойством, что расстояние между появлениями ab и ba всегда четно. Тогда можно положить $d_{ij} = a_j$ при четном $i+j$ и $d_{ij} = a_i$ при нечетном. Например, для $m = 4$ мы имеем

$$\begin{pmatrix}
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
0&0&0&1&0&2&0&3&2&0&2&1&2&2&2&3\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
1&0&1&1&1&2&1&3&3&0&3&1&3&2&3&3\\
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
0&2&0&3&0&0&0&1&2&2&2&3&2&0&2&1\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
1&2&1&3&1&0&1&1&3&2&3&3&3&0&3&1\\
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
2&0&2&1&2&2&2&3&0&0&0&1&0&2&0&3\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
3&0&3&1&3&2&3&3&1&0&1&1&1&2&1&3\\
0&0&1&0&0&2&1&2&2&0&3&0&2&2&3&2\\
2&2&2&3&2&0&2&1&0&2&0&3&0&0&0&1\\
0&1&1&1&0&3&1&3&2&1&3&1&2&3&3&3\\
3&2&3&3&3&0&3&1&1&2&1&3&1&0&1&1
\end{pmatrix}\quad\text{(упр. 109);}\qquad
\begin{pmatrix}
0&0&1&0&0&0&1&0&3&0&2&0&3&0&2&0\\
0&0&0&1&0&2&0&3&0&1&0&0&0&2&0&3\\
0&1&1&1&0&1&1&1&3&1&2&1&3&1&2&1\\
1&0&1&1&1&2&1&3&1&1&1&0&1&2&1&3\\
0&0&1&0&0&0&1&0&3&0&2&0&3&0&2&0\\
2&0&2&1&2&2&2&3&2&1&2&0&2&2&2&3\\
0&1&1&1&0&1&1&1&3&1&2&1&3&1&2&1\\
3&0&3&1&3&2&3&3&3&1&3&0&3&2&3&3\\
0&3&1&3&0&3&1&3&3&3&2&3&3&3&2&3\\
1&0&1&1&1&2&1&3&1&1&1&0&1&2&1&3\\
0&2&1&2&0&2&1&2&3&2&2&2&3&2&2&2\\
0&0&0&1&0&2&0&3&0&1&0&0&0&2&0&3\\
0&3&1&3&0&3&1&3&3&3&2&3&3&3&2&3\\
2&0&2&1&2&2&2&3&2&1&2&0&2&2&2&3\\
0&2&1&2&0&2&1&2&3&2&2&2&3&2&2&2\\
3&0&3&1&3&2&3&3&3&1&3&0&3&2&3&3
\end{pmatrix}\quad\text{(Тот (Tóth)).}$$

111. (а) Пусть $d_j = j$ и $0 \le a_j < 3$ для $1 \le j \le 9$, $a_9 \ne 0$. Образуем последовательности s_j, t_j с помощью следующих правил: $s_1 = 0$, $t_1 = d_1$; $t_{j+1} = d_{j+1} + 10t_j[a_j = 0]$ для $1 \le j < 9$; $s_{j+1} = s_j + (0, t_j, -t_j)$ для $a_j = (0,1,2)$ и $1 \le j \le 9$. Тогда возможным результатом является s_{10}; нам нужно только запомнить наименьшие получающиеся значения. Можно сэкономить больше половины работы, если запретить $a_k = 2$ при $s_k = 0$, а затем использовать $|s_{10}|$ вместо s_{10}. Поскольку нужно испытать менее чем $3^8 = 6561$ возможность, вполне применима тернарная версия алгоритма M; для того, чтобы выяснить, что таким образом можно представить все целые числа, меньшие 211, но не само число 211, требуется меньше 24 000 обращений к памяти и 1 600 умножений. (Легко также выяснить, что наибольшим количеством способов — 46 — представляется число 9. — *Примеч. пер.*)

Другой подход, использующий код Грея для варьирования знаков после разбиения цифр на блоки всеми 2^8 возможными способами, снижает количество умножений до 255, но ценой около 500 дополнительных обращений к памяти. Таким образом, код Грея в данном случае не обладает никакими преимуществами.

(б) В этом случае (с помощью 73 000 обращений к памяти и 4 900 умножений) можно получить все числа, меньшие 241, но не само число 241. Число 100 можно получить 46 способами, в том числе замечательным способом $9 - 87 + 6 + 5 - 43 + 210$. (И в этом случае рекордсменом по количеству представлений является число 9, но на этот раз со 103 способами представления. — *Примеч. пер.*)

[Г. Э. Дьюдени (H. E. Dudeney) сформулировал эту “задачу века” в *The Weekly Dispatch* (4 and 18 June 1899). См. также *The Numerology of Dr. Matrix* Мартина Гарднера (Martin Gardner), глава 6; Steven Kahan, *J. Recreational Math.* **23** (1991), 19–25; а также упр. 7.2.1.6–122.]

112. Теперь метод из упр. 111 требует более 167 миллионов обращений к памяти и 10 миллионов умножений, поскольку 3^{16} гораздо больше, чем 3^8. Можно поступить существенно лучше (обойдясь 10.4 миллионами обращений к памяти и 1100 умножениями), если сначала протабулировать все возможные результаты, получаемые при использовании первых k и последних k цифр, для $1 \le k < 9$, а затем рассмотреть все блоки цифр, использующие 9. Имеется 60 318 представлений 100, а первое непредставимое число — 16 040.

(В этот раз девятка с ее 84 475 представлениями оказалась на втором месте, уступив первенство тройке, которую можно представить 84 483 способами. — *Примеч. пер.*)

РАЗДЕЛ 7.2.1.2

1. [Д. П. Н. Филлипс (J. P. N. Phillips), *Comp. J.* **10** (1967), 311.] В предположении, что $n \ge 3$, можно заменить шаги L2–L4 следующими.

L2′. [Простейший случай?] Установить $y \leftarrow a_{n-1}$ и $z \leftarrow a_n$. Если $y < z$, установить $a_{n-1} \leftarrow z$, $a_n \leftarrow y$ и вернуться к шагу L1.

L2.1′. [Следующий простейший случай?] Установить $x \leftarrow a_{n-2}$. Если $x \ge y$, перейти к шагу L2.2′. В противном случае установить $(a_{n-2}, a_{n-1}, a_n) \leftarrow (z, x, y)$, если $x < z$ и (y, z, x), если $x \ge z$. Вернуться к шагу L1.

L2.2′. [Поиск j.] Установить $j \leftarrow n-3$ и $y \leftarrow a_j$. Пока $y \ge x$, устанавливать $j \leftarrow j-1$, $x \leftarrow y$ и $y \leftarrow a_j$. Завершить работу алгоритма, если $j = 0$.

L3′. [Простое увеличение?] Если $y < z$, установить $a_j \leftarrow z$, $a_{j+1} \leftarrow y$, $a_n \leftarrow x$ и перейти к шагу L4.1′.

L3.1′. [Увеличение a_j.] Установить $l \leftarrow n-1$; если $y \ge a_l$, многократно уменьшать l на 1, пока не будет выполнено условие $y < a_l$. Затем установить $a_j \leftarrow a_l$ и $a_l \leftarrow y$.

L4′. [Начало обращения.] Установить $a_n \leftarrow a_{j+1}$ и $a_{j+1} \leftarrow z$.

L4.1′. [Обращение $a_{j+2} \ldots a_{n-1}$.] Установить $k \leftarrow j+2$ и $l \leftarrow n-1$. Затем, пока $k < l$, обменивать $a_k \leftrightarrow a_l$ и устанавливать $k \leftarrow k+1$, $l \leftarrow l-1$. Вернуться к шагу L1. ▮

Программа может работать еще быстрее, если a_t хранится в памяти по адресу $\mathtt{A}[n-t]$ для $0 \le t \le n$ или если используется обратный солексный порядок, как в следующем упражнении.

2. Вновь будем считать, что изначально $a_1 \le a_2 \le \cdots \le a_n$; однако генерируемыми для $\{1, 2, 2, 3\}$ перестановками будут 1223, 2123, 2213, ..., 2321, 3221. Пусть a_{n+1} — вспомогательный элемент, *больший* a_n.

M1. [Посещение.] Посетить перестановку $a_1 a_2 \ldots a_n$.

M2. [Поиск j.] Установить $j \leftarrow 2$. Если $a_{j-1} \ge a_j$, увеличивать j на 1, пока не будет выполнено условие $a_{j-1} < a_j$. Завершить работу алгоритма, если $j > n$.

M3. [Уменьшение a_j.] Установить $l \leftarrow 1$. Если $a_l \ge a_j$, увеличивать l, пока не будет выполнено условие $a_l < a_j$. Затем выполнить обмен $a_l \leftrightarrow a_j$.

M4. [Обращение $a_1 \ldots a_{j-1}$.] Установить $k \leftarrow 1$ и $l \leftarrow j-1$. Затем, если $k < l$, обменять $a_k \leftrightarrow a_l$, установить $k \leftarrow k+1$, $l \leftarrow l-1$ и повторять эти действия, пока не будет выполнено условие $k \ge l$. Вернуться к шагу M1. ▮

3. Пусть $C_1 \ldots C_n = c_{a_1} \ldots c_{a_n}$ — таблица инверсий, как в упр. 5.1.1–7. Тогда $\operatorname{rank}(a_1 \ldots a_n)$ представляет собой число со смешанным основанием $\left[\begin{smallmatrix} C_1, & \ldots, & C_{n-1}, & C_n \\ n, & \ldots, & 2, & 1 \end{smallmatrix}\right]$. [См. H. A. Rothe, *Sammlung combinatorisch-analytischer Abhandlungen* **2** (1800), 263–264; см. также пионерскую работу Сарнгадева (Śārṅgadeva) и Нараяна (Nārāyaṇa), ссылки на которую имеются в разделе 7.2.1.7.] Например, 314592687 имеет ранг $\left[\begin{smallmatrix} 2, & 0, & 1, & 1, & 4, & 0, & 0, & 1, & 0 \\ 9, & 8, & 7, & 6, & 5, & 4, & 3, & 2, & 1 \end{smallmatrix}\right] = 2 \cdot 8! + 6! + 5! + 4 \cdot 4! + 1! = 81577$; это факториальная система счисления, представленная в 4.1–(10).

4. Воспользуйтесь рекуррентным соотношением $\mathrm{rank}(a_1\ldots a_n) = \frac{1}{n}\sum_{j=1}^{t} n_j[x_j < a_1] \times \binom{n}{n_1,\ldots,n_t} + \mathrm{rank}(a_2\ldots a_n)$. Например, rank(314159265) равен

$$\tfrac{3}{9}\tbinom{9}{2,1,1,1,2,1,1} + 0 + \tfrac{2}{7}\tbinom{7}{1,1,1,2,1,1} + 0 + \tfrac{1}{5}\tbinom{5}{1,2,1,1} + \tfrac{3}{4}\tbinom{4}{1,1,1,1} + 0 + \tfrac{1}{2}\tbinom{2}{1,1} = 30991.$$

5. (а) Шаг L2 выполняется $n!$ раз. Вероятность того, что будет выполнено ровно k сравнений, равна $q_k - q_{k+1}$, где q_t — вероятность того, что $a_{n-t+1} > \cdots > a_n$, а именно $[t \le n]/t!$. Следовательно, среднее значение равно $\sum k(q_k - q_{k+1}) = q_1 + \cdots + q_n = \lfloor n!\,e\rfloor/n! - 1 \approx e - 1 \approx 1.718$, а дисперсия равна

$$\sum k^2(q_k - q_{k+1}) - \mathrm{mean}^2 = q_1 + 3q_2 + \cdots + (2n-1)q_n - (q_1 + \cdots + q_n)^2 \approx e(3-e) \approx 0.766.$$

[Информацию о более высоких моментах можно найти в работе R. Kemp, *Acta Informatica* **35** (1998), 17–89, Theorem 4.]

Кстати, таким образом среднее количество обменов на шаге L4 равно $\sum\lfloor k/2\rfloor(q_k - q_{k+1}) = q_2 + q_4 + \cdots \approx \cosh 1 - 1 = (e + e^{-1} - 2)/2 \approx 0.543$ (результат получен Р. Д. Орд-Смитом (R. J. Ord-Smith) [*Comp. J.* **13** (1970), 152–155].

(б) Шаг L3 выполняется только $n! - 1$ раз, но для удобства мы будем считать, что он выполняется на один раз больше (с нулем сравнений). Тогда вероятность того, что выполняется ровно k сравнений, равна $\sum_{j=k+1}^{n} 1/j!$ для $1 \le k < n$ и $1/n!$ для $k = 0$. Следовательно, среднее равно $\frac{1}{2}\sum_{j=0}^{n-2} 1/j! \approx e/2 \approx 1.359$; в упр. 1 это количество уменьшается на $\frac{2}{3}$. Дисперсия равна $\frac{1}{3}\sum_{j=0}^{n-3} 1/j! + \frac{1}{2}\sum_{j=0}^{n-2} 1/j! - \mathrm{mean}^2 \approx \frac{5}{6}e - \frac{1}{4}e^2 \approx 0.418$.

6. (а) Пусть $e_n(z) = \sum_{k=0}^{n} z^k/k!$; тогда количество различных префиксов $a_1\ldots a_j$ равно $j!\,[z^j]\,e_{n_1}(z)\ldots e_{n_t}(z)$. Это в $N = \binom{n}{n_1,\ldots,n_t}$ раз больше вероятности q_{n-j} того, что на шаге L2 делается как минимум $n - j$ сравнений. Следовательно, среднее значение равно $\frac{1}{N}w(e_{n_1}(z)\ldots e_{n_t}(z)) - 1$, где $w(\sum x_k z^k/k!) = \sum x_k$. В бинарном случае среднее равно $M/\binom{n}{s} - 1$, где $M = \sum_{l=0}^{s}\sum_{k=l}^{n-s+l}\binom{k}{l} = \sum_{l=0}^{s}\binom{n-s+l+1}{l+1} = \binom{n+2}{s+1} - 1 = \binom{n}{s}\bigl(2 + \frac{s}{n-s+1} + \frac{n-s}{s+1}\bigr) - 1$.

(б) Если $\{a_1,\ldots,a_j\} = \{n_1'\cdot x_1,\ldots,n_t'\cdot x_t\}$, префикс $a_1\ldots a_j$ дает вклад в общее количество сравнений, выполняемых на шаге L3, равный $\sum_{1\le k<l\le t}(n_k - n_k')[n_l > n_l']$. Таким образом, среднее равно $\frac{1}{N}\sum_{1\le k<l\le t} w(f_{kl}(z))$, где

$$f_{kl}(z) = \Biggl(\prod_{\substack{1\le m\le t\\ m\ne k,\, m\ne l}} e_{n_m}(z)\Biggr)\Biggl(\sum_{r=0}^{n_k}(n_k - r)\frac{z^r}{r!}\Biggr)e_{n_l-1}(z)$$

$$= e_{n_1}(z)\ldots e_{n_t}(z)\bigl(n_k - z\,r_k(z)\bigr)r_l(z), \qquad \text{где } r_k(z) = \frac{e_{n_k-1}(z)}{e_{n_k}(z)}.$$

В бинарном случае эта формула сводится к $\frac{1}{N}w\bigl((se_s(z) - ze_{s-1}(z))e_{n-s-1}(z)\bigr) = \frac{s}{N}\bigl(\binom{n+1}{s+1} - 1\bigr) - \frac{1}{N}\bigl(\binom{n+1}{s+1}(s - \frac{s+1}{n-s+1}) + 1\bigr) = \frac{1}{N}\bigl(-s - 1 + \binom{n+1}{s}\bigr) = \frac{n+1}{n-s+1} - \frac{s+1}{N}$.

7. При использовании обозначений из ответа к предыдущему упражнению значение $\frac{1}{N}w(e_{n_1}(z)\ldots e_{n_t}(z)) - 1$ равно

$$\frac{n_1 + \cdots + n_t}{n} + \frac{(n_1n_2 + n_1n_3 + \cdots + n_{t-1}n_t) + n_1(n_1-1) + \cdots + n_t(n_t-1)}{n(n-1)} + \cdots.$$

С применением 1.2.9–(38) можно показать, что предел равен $-1 + \exp\sum_{k\ge1} r_k/k$, где $r_k = \lim_{t\to\infty}(n_1^k + \cdots + n_t^k)/(n_1 + \cdots + n_t)^k$. В случаях (а) и (б) имеем $r_k = [k=1]$, так что предел равен $e - 1 \approx 1.71828$. В случае (в) имеем $r_k = 1/(2^k - 1)$, так что предел равен $-1 + \exp\sum_{k\ge1} 1/(k(2^k - 1)) \approx 2.46275$.

8. Будем считать, что изначально j равно нулю, и изменим шаг L1 на

L1′. [Посещение.] Посетить вариацию $a_1 \ldots a_j$. Если $j < n$, установить $j \leftarrow j + 1$ и повторить этот шаг. ▮

Этот алгоритм предложен Л. Й. Фишером (L. J. Fischer) и К. Х. Краузе (K. C. Krause), *Lehrbuch der Combinationslehre und der Arithmetik* (Dresden: 1812), 55–57.

Кстати, общее количество вариаций равно $w(e_{n_1}(z) \ldots e_{n_t}(z))$ в обозначениях ответа к упр. 6. Эта задача подсчета количества вариаций была впервые рассмотрена Якобом Бернулли (James Bernoulli) в его *Ars Conjectandi* (1713), Part 2, Chapter 9.

9. Считаем, что $r > 0$ и что мы начинаем с $a_0 < a_1 \le a_2 \le \cdots \le a_n$.

R1. [Посещение.] Посетить вариацию $a_1 \ldots a_r$. (В этот момент $a_{r+1} \le \cdots \le a_n$.)

R2. [Простой случай?] Если $a_r < a_n$, обменять $a_r \leftrightarrow a_j$, где j представляет собой наименьший индекс, такой, что $j > r$ и $a_j > a_r$, и вернуться к шагу R1.

R3. [Обращение.] Установить $(a_{r+1}, \ldots, a_n) \leftarrow (a_n, \ldots, a_{r+1})$, как на шаге L4.

R4. [Поиск j.] Установить $j \leftarrow r - 1$. Если $a_j \ge a_{j+1}$, уменьшать j на 1, пока не будет выполнено условие $a_j < a_{j+1}$. Завершить работу алгоритма, если $j = 0$.

R5. [Увеличение a_j.] Установить $l \leftarrow n$. Если $a_j \ge a_l$, уменьшать l на 1, пока не будет выполнено условие $a_j < a_l$. Затем обменять $a_j \leftrightarrow a_l$.

R6. [Опять обращение.] Установить $(a_{j+1}, \ldots, a_n) \leftarrow (a_n, \ldots, a_{j+1})$, как на шаге L4, и вернуться к шагу R1. ▮

Количество выводов алгоритма равно $r!\,[z^r]\,e_{n_1}(z) \ldots e_{n_t}(z)$; если все элементы различны, само собой разумеется, это значение равно $n^{\underline{r}}$.

10. $a_1 a_2 \ldots a_n = 213 \ldots n$, $c_1 c_2 \ldots c_n = 010 \ldots 0$, $o_1 o_2 \ldots o_n = 1(-1)1 \ldots 1$, если $n \ge 2$.

11. Шаг (P1, ..., P7) выполняется $(1, n!, n!, n! + x_n, n! - 1, (x_n + 3)/2, x_n)$ раз, где $x_n = \sum_{k=1}^{n-1} k!$, поскольку шаг P7 выполняется $(j-1)!$ раз при $2 \le j \le n$.

12. Нас интересует перестановка с рангом 999 999. Вот интересующие нас ответы. (а) 2783915460 согласно упр. 3; (б) 8750426319, поскольку отраженным числом со смешанным основанием, которое соответствует числу $\left[\begin{smallmatrix} 0, & 0, & 1, & 2, & 3, & 0, & 2, & 7, & 0, & 9 \\ 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & 10 \end{smallmatrix}\right]$, является $\left[\begin{smallmatrix} 0, & 0, & 1, & 3-2, & 3, & 5-0, & 2, & 7, & 8-0, & 9-9 \\ 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8, & 9, & 10 \end{smallmatrix}\right]$, в соответствии с 7.2.1.1–(50); (в) произведение $(0\ 1\ \ldots\ 9)^9 \times (0\ 1\ \ldots\ 8)^0 (0\ 1\ \ldots\ 7)^7 (0\ 1\ \ldots\ 6)^2 \ldots (0\ 1\ 2)^1$, а именно 9703156248.

13. Первое утверждение истинно для всех $n \ge 2$. Но когда 2 пересекает 1, т. е. когда c_2 изменяется с 0 на 1, имеем $c_3 = 2$, $c_4 = 3$, $c_5 = \cdots = c_n = 0$, и следующей перестановкой при $n \ge 5$ является $432156 \ldots n$. [См. *Time Travel* (1988), page 74.]

14. Истинно в начале шагов P4–P6, поскольку ровно $j - 1 - c_j + s$ элементов лежат слева от x_j, а именно $j - 1 - c_j$ элементов из $\{x_1, \ldots, x_{j-1}\}$ и s элементов из $\{x_{j+1}, \ldots, x_n\}$. (В определенном смысле эта формула является сутью алгоритма P.)

15. Если $\left[\begin{smallmatrix} b_{n-1}, & \ldots, & b_0 \\ 1, & \ldots, & n \end{smallmatrix}\right]$ соответствует рефлексивному коду Грея $\left[\begin{smallmatrix} c_1, & \ldots, & c_n \\ 1, & \ldots, & n \end{smallmatrix}\right]$, то мы переходим к шагу P6 тогда и только тогда, когда $b_{n-k} = k - 1$ для $j \le k \le n$ и B_{n-j+1} четно, в соответствии с 7.2.1.1–(50). Но из $b_{n-k} = k - 1$ для $j \le k \le n$ вытекает, что B_{n-k} нечетно для $j < k \le n$. Следовательно, на шаге P5 $s = [c_{j+1} = j] + [c_{j+2} = j + 1] = [o_{j+1} < 0] + [o_{j+2} < 0]$. [See *Math. Comp.* **17** (1963), 282–285.]

16. **P1′.** [Инициализация.] Установить $c_j \leftarrow j$ и $o_j \leftarrow -1$ для $1 \le j < n$; установить также $z \leftarrow a_n$.

P2′. [Посещение.] Посетить $a_1 \ldots a_n$. Затем перейти к шагу P3.5′, если $a_1 = z$.

P3′. [Колебание вниз.] Для $j \leftarrow n-1, n-2, \ldots, 1$ (в указанном порядке) установить $a_{j+1} \leftarrow a_j$, $a_j \leftarrow z$ и посетить $a_1 \ldots a_n$. Затем установить $j \leftarrow n-1$, $s \leftarrow 1$ и перейти к шагу P4′.

P3.5′. [Колебание вверх.] Для $j \leftarrow 1, 2, \ldots, n-1$ (в указанном порядке) установить $a_j \leftarrow a_{j+1}$, $a_{j+1} \leftarrow z$ и посетить $a_1 \ldots a_n$. Затем установить $j \leftarrow n-1$, $s \leftarrow 0$.

P4′. [Готовность к изменению?] Установить $q \leftarrow c_j + o_j$. Если $q = 0$, перейти к шагу P6′; если $q > j$, перейти к шагу P7′.

P5′. [Изменение.] Обменять $a_{c_j+s} \leftrightarrow a_{q+s}$. Затем установить $c_j \leftarrow q$ и вернуться к шагу P2′.

P6′. [Увеличение s.] Завершить работу алгоритма, если $j = 1$; в противном случае установить $s \leftarrow s+1$.

P7′. [Переключение направления.] Установить $o_j \leftarrow -o_j$, $j \leftarrow j-1$ и вернуться к шагу P4′. ▮

17. Изначально $a_j \leftarrow a'_j \leftarrow j$ для $1 \le j \le n$. Шаг P5 должен теперь устанавливать $t \leftarrow j - c_j + s$, $u \leftarrow j - q + s$, $v \leftarrow a_u$, $a_t \leftarrow v$, $a'_v \leftarrow t$, $a_u \leftarrow j$, $a'_j \leftarrow u$, $c_j \leftarrow q$. (См. упр. 14.)

Однако, как заметил Г. Эрлих (G. Ehrlich) [*JACM* **20** (1973), 505–506], требование доступности обратных перестановок обеспечивает возможность существенно упростить алгоритм, позволяя избежать переменной s, а управляющей таблице $c_1 \ldots c_n$ — использовать только убывания.

Q1. [Инициализация.] Установить $a_j \leftarrow a'_j \leftarrow j$, $c_j \leftarrow j-1$ и $o_j \leftarrow -1$ для $1 \le j \le n$. Установить также $c_0 = -1$.

Q2. [Посещение.] Посетить перестановку $a_1 \ldots a_n$ и обратную к ней $a'_1 \ldots a'_n$.

Q3. [Поиск k.] Установить $k \leftarrow n$. Затем, пока $c_k = 0$, устанавливать $c_k \leftarrow k-1$, $o_k \leftarrow -o_k$ и $k \leftarrow k-1$. Завершить работу алгоритма, если $k = 0$.

Q4. [Изменение.] Установить $c_k \leftarrow c_k - 1$, $j \leftarrow a'_k$ и $i = j + o_k$. Затем установить $t \leftarrow a_i$, $a_i \leftarrow k$, $a_j \leftarrow t$, $a'_t \leftarrow j$, $a'_k \leftarrow i$ и вернуться к шагу Q2. ▮

18. Установить $a_n \leftarrow n$ и использовать $(n-1)!/2$ итераций алгоритма P для генерации всех перестановок $\{1, \ldots, n-1\}$, таких, что 1 предшествует 2. [M. K. Roy, *CACM* **16** (1973), 312–313; см. также упр. 13.]

19. Например, можно воспользоваться идеей алгоритма P с n-кортежами $c_1 \ldots c_n$, изменяющимися, как в алгоритме 7.2.1.1H, по отношению к основаниям $(1, 2, \ldots, n)$. Этот алгоритм корректно поддерживает направления, хотя и нумерует индексы по-другому. Смещение s, требующееся алгоритму P, может быть вычислено, как в ответе к упр. 15; можно обойтись без него путем поддержки обратной перестановки, как это сделано в упр. 17. [См. G. Ehrlich, *CACM* **16** (1973), 690–691.] Без циклов можно также реализовать и другой алгоритм, подобный алгоритму Хипа.

(*Примечание.* В большинстве приложений генерации перестановок нас интересует уменьшение *общего* времени работы, а не максимального времени между последовательными посещениями; с этой точки зрения отсутствие циклов обычно нежелательно, если только речь не идет о работе на параллельном компьютере. Тем не менее нельзя сбрасывать со счетов чисто интеллектуальное удовлетворение от бесциклового решения задачи, независимое от его практичности.)

20. Например, когда $n = 3$, можно начать с 123, 132, 312, $\overline{3}12$, $1\overline{3}2$, $12\overline{3}$, $21\overline{3}$, …, 213, $\overline{2}13$, …. Если дельта-последовательность для n представляет собой $(\delta_1\delta_2 \ldots \delta_{2^n n!})$, то соответствующая последовательность для $n+1$ имеет вид $(\Delta_n\delta_1\Delta_n\delta_2 \ldots \Delta_n\delta_{2^n n!})$, где Δ_n — последовательность $2n+1$ операций $n\ n{-}1\ \ldots\ 1\ -\ 1\ \ldots\ n{-}1\ n$; здесь $\delta_k = j$ означает $a_j \leftrightarrow a_{j+1}$, а $\delta_k = -$ означает $a_1 \leftarrow -a_1$.

(Знаковые перестановки предстают в ином виде в упр. 5.1.4–43 и 5.1.4–44. Множество всех знаковых перестановок называется октаэдрической группой.)

21. Ясно, что $\mathtt{M} = 1$, следовательно, $\mathtt{O}$ должно быть 0, а $\mathtt{S}$ должно быть $b-1$. Тогда $\mathtt{N} = \mathtt{E}+1$, $\mathtt{R} = b-2$ и $\mathtt{D}+\mathtt{E} = b+\mathtt{Y}$. При этом остается ровно $\max(0, b-7-k)$ вариантов выбора для $\mathtt{E}$ при $\mathtt{Y} = k \ge 2$, следовательно, всего имеется $\sum_{k=2}^{b-7}(b-7-k) = \binom{b-8}{2}$ решений, когда $b \ge 8$. [*Math. Mag.* **45** (1972), 48–49. Кстати, Д. Эппштейн (D. Eppstein) доказал, что решение буквометика в системе счисления с заданным основанием является NP-полной задачей; см. *SIGACT News* **18**, 3 (1987), 38–40.]

22. $(\mathtt{X})_b + (\mathtt{X})_b = (\mathtt{XY})_b$ имеет решение только при $b = 2$.

23. Почти истинно, поскольку количество решений будет четным, *если только не выполняется условие* $[j \in F] \ne [k \in F]$. (Рассмотрите тернарный буквометик $\mathtt{X} + (\mathtt{XX})_3 + (\mathtt{YY})_3 + (\mathtt{XZ})_3 = (\mathtt{XYX})_3$.)

24. (а) $9283+7+473+1062 = 10825$. (б) $698392+3192 = 701584$. (в) $63952+69275 = 133227$. (г) $653924 + 653924 = 1307848$. (д) $5718 + 3 + 98741 = 104462$. (е) $127503 + 502351 + 3947539 + 46578 = 4623971$. (ж) $67432+704+8046+97364 = 173546$. (з) $59+577404251698+69342491650 + 49869442698 + 1504 + 40614 + 82591 + 344 + 41 + 741425 = 5216367650 + 691400684974$. [Все решения единственны. Источники для (б)–(ж): *J. Recreational Math.* **10** (1977), 115; **5** (1972), 296; **10** (1977), 41; **10** (1978), 274; **12** (1979), 133–134; **9** (1977), 207.]

(и) В этом случае имеется $\frac{8}{10}10! = 2903040$ решений, поскольку годится *любая* перестановка $\{0, 1, \ldots, 9\}$, за исключением тех из них, для которых $\mathtt{H}$ или $\mathtt{N}$ равны 0. (Хорошо продуманная программа решения аддитивных буквометиков должна в таких случаях сокращать объем выводимых данных.)

25. Можно считать, что $s_1 \le \cdots \le s_{10}$. Пусть i является наименьшим индексом $\notin F$. Установим $a_i \leftarrow 0$, после чего установим значения остальных элементов a_j в порядке возрастания j. Доказательство, аналогичное доказательству теоремы 6.1S, показывает, что эта процедура максимизирует $a \cdot s$. Аналогичная процедура дает минимум, поскольку $\min(a \cdot s) = -\max(a \cdot (-s))$.

26. $400739 + 63930 - 2379 - 1252630 + 53430 - 1390 + 738300$.

27. Читатели, вероятно, смогут улучшить следующие примеры: $\mathtt{BLOOD} + \mathtt{SWEAT} + \mathtt{TEARS} = \mathtt{LATER}$; $\mathtt{EARTH} + \mathtt{WATER} + \mathtt{WRATH} = \mathtt{HELLO} + \mathtt{WORLD}$; $\mathtt{AWAIT} + \mathtt{ROBOT} + \mathtt{ERROR} = \mathtt{SOBER} + \mathtt{WORDS}$; $\mathtt{CHILD} + \mathtt{THEME} + \mathtt{PEACE} + \mathtt{ETHIC} = \mathtt{IDEAL} + \mathtt{ALPHA} + \mathtt{METIC}$. (Это упражнение появилось после знакомства с буквометиком $\mathtt{WHERE} + \mathtt{SEDGE} + \mathtt{GRASS} + \mathtt{GROWS} = \mathtt{MARSH}$ [A. W. Johnson, Jr., *J. Recr. Math.* **15** (1982), 51], удивительно чистым, если бы не одинаковые сигнатуры $\mathtt{D}$ и $\mathtt{O}$.) Д. А. Браун (J. A. Brown), Й. Сабо (J. Szabó) и Т. Д. Троубридж (T. J. Trowbridge) предложили буквометики $\mathtt{GREAT} + \mathtt{GREAT} = \mathtt{LARGE}$ и $\mathtt{GREAT} + \mathtt{GREAT} = \mathtt{SMALL}$.

28. (а) $11 = 3+3+2+2+1$, $20 = 11+3+3+3$, $20 = 11+3+3+2+1$, $20 = 11+3+3+1+1+1$, $20 = 8+8+2+1+1$, $20 = 7+7+6$, $20 = 7+7+2+2+2$, $20 = 7+7+2+1+1+1+1$, $20 = 7+5+5+2+1$, $20 = 7+5+2+2+2+1+1$, $20 = 7+5+2+2+1+1+1+1$, $20 = 7+3+3+2+2+1+1+1$, $20 = 7+3+3+1+1+1+1+1+1+1$, $20 = 5+3+3+3+3+3$. [Эти четырнадцать решений были впервые вычислены Роем Чайлдсом (Roy Childs) в 1999. Следующими значениями n, имеющими дважды истинные разбиения, являются 30 (20 способами), затем 40 (94 способами), 41 (67 способов), 42 (57 способов), 50 (190 способов, включая $50 = 2+2+\cdots+2$) и т. д.]

(б) $51 = 20+15+14+2$, $51 = 15+14+10+9+3$, $61 = 19+16+11+9+6$, $65 = 17+16+15+9+7+1$, $66 = 20+19+16+6+5$, $69 = 18+17+16+10+8$, $70 = 30+20+10+7+3$, $70 = 20+16+12+9+7+6$, $70 = 20+15+12+11+7+5$, $80 = 50+20+9+1$, $90 = 50+12+11+9+5+2+1$, $91 = 45+19+11+10+5+1$. [Два способа для 51 вычислены Стивом Каганом (Steven Kahan); см. его книгу *Have Some Sums To Solve* (Farmingdale, New

York: Baywood, 1978), 36–37, 84, 112. Удивительные примеры с семнадцатью различными членами на итальянском языке и пятьюдесятью восемью членами, записанными римскими цифрами, найдены Джулио Чезаре (Giulio Cesare), *J. Recr. Math.* **30** (1999), 63.]

Примечания. Очень красивый пример THREE = TWO + ONE + ZERO [Richard L. Breisch, *Recreational Math. Magazine* **12** (December 1962), 24], к сожалению, не удовлетворяет поставленным условиям. Общее количество дважды истинных разбиений на различные части в английском языке, вероятно, конечно (хотя терминология для произвольно больших чисел не стандартизована). Удастся ли кому-то найти пример, больший чем

$$\mathtt{NINETYNINENONILLIONNINETYNINESEXTILLIONSIXTYONE} =$$
$$\mathtt{NINETYNINENONILLIONNINETYNINESEXTILLIONNINETEEN} + \mathtt{SIXTEEN} + \mathtt{ELEVEN} + \mathtt{NINE} + \mathtt{SIX}$$

(предложен Г. Гонзелес-Моррисом (G. González-Morris))?

(*Примечание переводчика.* Что касается русского и украинского языков, то они гораздо менее приспособлены для получения дважды истинных разбиений. Исчерпывающий поиск до 100 включительно дает только 4 дважды истинных разбиения на русском языке, причем все они для одного и того же числа — 40 (для экономии места суммы заменены умножениями, причем курсивом указывается количество суммируемых членов, т. е., например, $5*2$ означает $2+2+2+2+2$, а не $5+5$): $40 = 3*3+31*1 = 3+5*2+27*1 = 3+37*1 = 40*1$ (обратите внимание на красоту последнего разбиения). Благодаря близости русского и украинского языков эти же разбиения дважды истинны и на украинском языке, где они записываются совершенно так же, как и на русском, хотя и читаются несколько иначе. Кроме них, на украинском языке имеются еще три дважды истинных разбиения, два — для числа 40, а именно $40 = 5*8 = 3*8+2*7+2$, и еще одно — для числа 70: $70 = 3*10+2*9+8+2*7$ (70, 7, 8 и 9 на украинском языке записываются как *сімдесят*, *сім*, *вісім* и *дев'ять*).)

29. $10+7+1 = 9+6+3$, $11+10 = 8+7+6$, $12+7+6+5 = 11+10+9$, ..., $19+10+3 = 14+13+4+1$ (всего 31 пример).

30. (а) $567^2 = 321489$, $807^2 = 651249$ или $854^2 = 729316$. ($459^2 = 210681$ для русского варианта, $739^2 = 546121$ для украинского.) (б) $958^2 = 917764$. (в) $96 \times 7^2 = 4704$. (г) $51304/61904 = 7260/8760$. (д) $328509^2 = 4761^3$. [*Strand* **78** (1929), 91, 208; *J. Recr. Math* **3** (1970), 43; **13** (1981), 212; **27** (1995), 137; **31** (2003), 133. Решения (б)–(д) являются единственными. С помощью подхода, основанного на алгоритме X, ответы можно найти ценой (14, 13, 11, 3423, 42) тысяч обращений к памяти соответственно. Нобуюки Йошигара также заметил, что буквометик NORTH/SOUTH = WEST/EAST имеет единственное решение: $67104/27504 = 9320/3820$.]

31. (а) $5/34+7/68+9/12$(!). В единственности этого решения можно убедиться, воспользовавшись алгоритмом X с дополнительным условием $\mathtt{A} < \mathtt{D} < \mathtt{G}$ (при этом потребуется около 265 тысяч обращений к памяти). [*Quark Visual Science Magazine*, No. 136 (Tokyo: Kodansha, October 1993).] Забавно, что единственное решение имеет и похожая головоломка: $1/(3 \times 6) + 5/(8 \times 9) + 7/(2 \times 4) = 1$ [Scot Morris, *Omni* **17**, 4 (January 1995), 97].

(б) ABCDEFGHI = 381654729, решение с применением алгоритма X требует около 10 тысяч обращений к памяти.

32. Имеется одиннадцать способов, среди которых самый удивительный — $3 + 69258/714$. [См. *The Weekly Dispatch* (9 and 23 June 1901); *Amusements in Mathematics* (1917), 158–159.]

33. (а) 1, 2, 3, 4, 15, 18, 118, 146. (б) 6, 9, 16, 20, 27, 126, 127, 129, 136, 145. [*The Weekly Dispatch* (11 and 30 November, 1902); *Amusements in Math.* (1917), 159.]

В этом случае одна из подходящих стратегий заключается в поиске всех вариаций, где $a_k \ldots a_{l-1}/a_l \ldots a_9$ является целым числом, с последующей записью решений для всех перестановок $a_1 \ldots a_{k-1}$. Имеется 164 959 целых чисел с единственным представлением,

наибольшее из них — 9 876 533. Имеются решения для всех годов XXI века, за исключением 2091. Наибольшее количество решений (389) имеется для $n = 12\,221$; наибольший отрезок представимых чисел — $5109 < n < 7060$. Сам Дьюдени смог дать ответы только для небольших значений n путем "отбрасывания девяток".

34. (а) $x = 10^5$, $7378 + 155 + 92467 = 7178 + 355 + 92467 = 1016 + 733 + 98251 = 1014 + 255 + 98731 = 100000$.

(б) $x = 4^7$, $3036 + 455 + 12893 = 16384$ (решение единственное). Наиболее быстрый способ решения данной задачи, вероятно, заключается в том, чтобы начать со списка 2529 простых чисел, состоящих из пяти различных цифр (т. е. 10243, 10247, ..., 98731), и переставлять оставшиеся пять цифр.

Кстати, буквометик без ограничений $\texttt{EVEN} + \texttt{ODD} = \texttt{PRIME}$ имеет десять решений; и ODD, и PRIME являются простыми только в одном из них. [См. M. Arisawa, *J. Recr. Math.* **8** (1975), 153.]

35. В общем случае, если $s_k = |S_k|$ для $1 \le k < n$, имеется $s_1 \ldots s_{k-1}$ способов выбора каждого из нетождественных элементов S_k. Следовательно, ответ имеет вид $\prod_{k=1}^{n-1}(\prod_{j=1}^{k-1} s_j^{s_k-1})$, что в данном случае равно $2^2 \cdot 6^3 \cdot 24^{15} = 436\,196\,692\,474\,023\,836\,123\,136$.

(Но при перенумерации вершин значения s_k могут изменяться. Например, если вершины (0, 3, 5) в (12) обменять с (e, d, c), то получим $s_{14} = 1$, $s_{13} = 6$, $s_{12} = 4$, $s_{11} = 1$ и $4^5 \cdot 24^{15}$ таблиц Симса.)

36. Поскольку каждый из элементов $\{0, 3, 5, 6, 9, a, c, f\}$ находится на трех линиях, а каждый из остальных — только на двух, ясно, что можно положить $S_f = \{(), \sigma, \sigma^2, \sigma^3, \alpha, \alpha\sigma, \alpha\sigma^2, \alpha\sigma^3\}$, где $\sigma = (03fc)(17e8)(2bd4)(56a9)$ представляет собой поворот на 90°, а $\alpha = (05)(14)(27)(36)(8d)(9c)(af)(be)$ является кручением изнутри наружу. Кроме того, $S_e = \{(), \beta, \gamma, \beta\gamma\}$, где $\beta = (14)(28)(3c)(69)(7d)(be)$ является транспозицией, а $\gamma = (12)(48)(5a)(69)(7b)(de)$ — еще одним кручением; $S_d = \cdots = S_1 = \{()\}$. (Всего имеется $4^7 - 1$ альтернативных ответов.)

37. Множество S_k можно выбрать $k!^k$ способами (см. упр. 35), а его нетождественные элементы можно присвоить $\sigma(k,1)$, ..., $\sigma(k,k)$ еще $k!$ способами. Так что окончательный ответ — $A_n = \prod_{k=1}^{n-1} k!^{k+1} = n!^{\binom{n+1}{2}} / \prod_{k=1}^{n} k^{\binom{k+1}{2}}$. Например, $A_{10} \approx 1.148 \times 10^{170}$. По формуле суммирования Эйлера мы получаем

$$\sum_{k=1}^{n-1} \binom{k}{2} \ln k = \frac{1}{2} \int_1^n x(x-1) \ln x \, dx + O(n^2 \log n) = \frac{1}{6} n^3 \ln n + O(n^3),$$

так что $\ln A_n = \frac{1}{3} n^3 \ln n + O(n^3)$.

38. Вероятность того, что на шаге G4 потребуется $\phi(k)$, равна $1/k! - 1/(k+1)!$, для $1 \le k < n$; вероятность того, что мы вообще не попадем на шаг G4, составляет $1/n!$. Поскольку $\phi(k)$ выполняет $\lceil k/2 \rceil$ транспозиций, среднее значение равно $\sum_{k=1}^{n-1}(1/k! - 1/(k+1)!)\lceil k/2 \rceil = \sum_{k=1}^{n-1}(\lceil k/2 \rceil - \lceil (k-1)/2 \rceil)/k! - \lceil (n-1)/2 \rceil/n! = \sum_{k \text{ нечетно}} 1/k! + O(1/(n-1)!)$.

39. (а) 0123, 1023, 2013, 0213, 1203, 2103, 3012, 0312, 1302, 3102, 0132, 1032, 2301, 3201, 0231, 2031, 3021, 0321, 1230, 2130, 3120, 1320, 2310, 3210; (б) 0123, 1023, 2013, 0213, 1203, 2103, 3102, 1302, 0312, 3012, 1032, 0132, 0231, 2031, 3021, 0321, 2301, 3201, 3210, 2310, 1320, 3120, 2130, 1230.

40. По индукции находим $\sigma(1,1) = (0\ 1)$, $\sigma(2,2) = (0\ 1\ 2)$,

$$\sigma(k,k) = \begin{cases} (0\ k)(k{-}1\ k{-}2\ \ldots\ 1), & \text{если } k \ge 3 \text{ нечетно}, \\ (0\ k{-}1\ k{-}2\ 1\ \ldots\ k{-}3\ k), & \text{если } k \ge 4 \text{ четно}; \end{cases}$$

также $\omega(k) = (0\ k)$, когда k четно, $\omega(k) = (0\ k{-}2\ \ldots\ 1\ k{-}1\ k)$, когда $k \geq 3$ нечетно. Таким образом, когда $k \geq 3$ нечетно, $\sigma(k,1) = (k\ k{-}1\ 0)$, а $\sigma(k,j)$ отображает $k \mapsto j-1$ для $1 < j < k$; когда $k \geq 4$ четно, $\sigma(k,j) = (0\ k\ k{-}3\ \ldots\ 1\ k{-}2\ k{-}1)^j$ для $1 \leq j \leq k$.

Примечания. Первая схема, которая заставляет алгоритм G генерировать все перестановки с помощью одних транспозиций, была разработана Марком Уэллсом (Mark Wells) [*Math. Comp.* **15** (1961), 192–195], но при этом была существенно более сложной. В. Липски-мл. (W. Lipski, Jr.) изучал такие схемы в общем случае и обнаружил ряд других методов [*Computing* **23** (1979), 357–365].

41. Можно считать, что $r < n$. Алгоритм G будет генерировать r-вариации для любой таблицы Симса, если просто заменить '$k \leftarrow 1$' на '$k \leftarrow n-r$' на шаге G3, при условии переопределения $\omega(k)$ как $\sigma(n-r, n-r)\ldots\sigma(k,k)$ вместо использования (16).

Если $n-r$ нечетно, метод (27) остается корректным, хотя формулы в ответе к упр. 40 при $k < n-r+2$ следует пересмотреть. Новые формулы имеют вид $\sigma(k,j) = (k\ j{-}1\ \ldots\ 1\ 0)$ и $\omega(k) = (k\ \ldots\ 1\ 0)$, когда $k = n-r$; $\sigma(k,j) = (k\ \ldots\ 1\ 0)^j$, когда $k = n-r+1$.

Если $n-r$ четно, можно использовать (27), обратив четный и нечетный случаи при $r \leq 3$. Но когда $r \geq 4$, требуется более сложная схема, поскольку фиксированная транспозиция типа $(k\ 0)$ может быть использована для нечетного k, только если $\omega(k-1)$ является k-циклом, что означает, что $\omega(k-1)$ должна быть четной перестановкой; но $\omega(k)$ нечетна для $k \geq n-r+2$.

Следующая схема работает, когда $n-r$ четно. Пусть $\tau(k,j)\,\omega(k-1)^- = (k\ k{-}j)$ для $1 \leq j \leq k = n-r$, и используем (27) для $k > n-r$. Тогда при $k = n-r+1$ имеем $\omega(k{-}1) = (0\ 1\ \ldots\ k{-}1)$, следовательно, $\sigma(k,j)$ отображает $k \mapsto (2j{-}1) \bmod k$ для $1 \leq j \leq k$, и $\sigma(k,k) = (k\ k{-}1\ k{-}3\ \ldots\ 0\ k{-}2\ \ldots\ 1)$, $\omega(k) = (k\ \ldots\ 1\ 0)$, $\sigma(k+1,j) = (k{+}1\ \ldots\ 0)^j$.

42. Если $\sigma(k,j) = (k\ j{-}1)$, имеем $\tau(k,1) = (k\ 0)$ и $\tau(k,j) = (k\ j{-}1)(k\ j{-}2) = (k\ j{-}1\ j{-}2)$ для $2 \leq j \leq k$.

43. Конечно, $\omega(1) = \sigma(1,1) = \tau(1,1) = (0\ 1)$. Следующее построение создает $\omega(k) = (k{-}2\ k{-}1\ k)$ для всех $k \geq 2$. Пусть $\alpha(k,j) = \tau(k,j)\,\omega(k-1)^-$, где $\alpha(2,1) = (2\ 0)$, $\alpha(2,2) = (2\ 0\ 1)$, $\alpha(3,1) = \alpha(3,3) = (3\ 1)$, $\alpha(3,2) = (3\ 1\ 0)$; это делает $\sigma(2,2) = (0\ 2)$, $\sigma(3,3) = (0\ 3\ 1)$. Тогда для $k \geq 4$ положим $\alpha(k,1) = (k\ k{-}4)$, $\alpha(k,j) = (k\ k{-}3{-}j\ k{-}2{-}j)$ для $1 < j < k-2$, а

$$\begin{array}{rcccccc}
 & & k \bmod 3 = 0 & & k \bmod 3 = 1 & & k \bmod 3 = 2 \\
\alpha(k,k{-}2) & = & (k\ k{-}2\ 0) & \text{или} & (k\ k{-}3\ 0) & \text{или} & (k\ k{-}1\ 0), \\
\alpha(k,k{-}1) & = & (k\ k{-}2\ k{-}3) & \text{или} & (k\ k{-}3) & \text{или} & (k\ k{-}1\ k{-}3), \\
\alpha(k,k) & = & (k\ k{-}2) & \text{или} & (k\ k{-}3\ k{-}2) & \text{или} & (k\ k{-}2).
\end{array}$$

Это дает $\sigma(k,k) = (k{-}3\ k\ k{-}2)$, что и требовалось.

44. Нет, поскольку $\tau(k,j)$ является $(k+1)$-циклом, а не транспозицией. (См. (19) и (24).)

45. (а) 202280070, поскольку $u_k = \max\bigl(\{0,1,\ldots,a_k-1\} \setminus \{a_1,\ldots,a_{k-1}\}\bigr)$. (В действительности u_n никогда не устанавливается алгоритмом, но мы можем считать это значение нулем.) (б) 273914568.

46. Истинно (в предположении, что $u_n = 0$). Если либо $u_k > u_{k+1}$, либо $a_k > a_{k+1}$, то мы должны получить $a_k > u_k \geq a_{k+1} > u_{k+1}$.

47. Шаги (X1, X2, ..., X6) выполняются соответственно $(1, A, B, A-1, B-N_n, A)$ раз, где $A = N_0 + \cdots + N_{n-1}$ и $B = nN_0 + (n-1)N_1 + \cdots + 1N_{n-1}$.

48. Шаги (X2, X3, ..., X6) выполняются соответственно $A_n + (1, n!, 0, 0, 1)$ раз, где $A_n = \sum_{k=1}^{n-1} n^{\underline{k}} = n!\sum_{k=1}^{n-1} 1/k! \approx n!\,(e-1)$. Полагая их стоимости равными соответственно $(1,1,3,1,3)$ обращений к памяти, общая стоимость операций с a_j, l_j или u_j составляет около $9e-8 \approx 16.46$ обращений к памяти на перестановку.

Алгоритм L использует приблизительно $(e, 2+e/2, 2e+2e^{-1}-4)$ обращений к памяти на одну перестановку на шагах (L2, L3, L4), с общей стоимостью $3.5e+2e^{-1}-2 \approx 8.25$ (см. упр. 5).

Алгоритм X можно улучшить для данного случая, оптимизировав код, когда k близко к n. Но, как показано в упр. 1, то же самое можно сделать и с алгоритмом L.

49. Упорядочьте сигнатуры так, чтобы $|s_0| \geq \cdots \geq |s_9|$; подготовьте также таблицы $w_0 \ldots w_9$, $x_0 \ldots x_9$, $y_0 \ldots y_9$, так, чтобы сигнатуры $\{s_k, \ldots, s_9\}$ представляли собой $w_{x_k} \leq \cdots \leq w_{y_k}$. Например, когда SEND+MORE = MONEY, мы имеем $(s_0, \ldots, s_9) = (-9000, 1000, -900, 91, -90, 10, 1, -1, 0, 0)$ соответственно для букв (M, S, O, E, N, R, D, Y, A, B); также $(w_0, \ldots, w_9) = (-9000, -900, -90, -1, 0, 0, 1, 10, 91, 1000)$ и $x_0 \ldots x_9 = 0112233344$, $y_0 \ldots y_9 = 9988776554$. Еще одна таблица $f_0 \ldots f_9$ содержит $f_j = 1$, если цифра, соответствующая w_j, не может быть нулем; в этом случае $f_0 \ldots f_9 = 1000000001$. Эти таблицы упрощают вычисление самого большого и малого значений $s_k a_k + \cdots + s_9 a_9$ для всех вариантов выбора $a_k \ldots a_9$ из оставшихся цифр с использованием метода из упр. 25, поскольку связи l_j информируют нас об этих цифрах в порядке возрастания.

Этот метод требует очень затратных вычислений в каждом узле дерева поиска, но зато часто позволяет обойтись малым деревом. Например, первые восемь буквометиков из упр. 24 он решает ценой всего лишь 7, 13, 7, 9, 5, 343, 44 и 89 тысяч обращений к памяти; это существенное улучшение в случаях (а), (б), (д) и (з), хотя в случае (е) наблюдается значительное ухудшение. Еще одним плохим случаем является пример 'CHILD' из упр. 27, где подход слева направо требует 2947 тысяч обращений к памяти, в то время как подход справа налево — 588 тысяч обращений к памяти. Но он выигрывает в буквометиках BLOOD+SWEAT+TEARS (73 тысяч обращений к памяти против 360) и HELLO+WORLD (340 против 410).

50. Если α входит в группу перестановок, то в нее входят и все ее степени α^2, α^3, ..., включая $\alpha^{m-1} = \alpha^-$, где m является порядком α (наименьшее общее кратное длин циклов). А (32) эквивалентно $\alpha^- = \sigma_1\sigma_2 \ldots \sigma_{n-1}$.

51. Ложно. Например, и $\sigma(k,i)^-$, и $\sigma(k,j)^-$ могут отображать $k \mapsto 0$.

52. $\tau(k,j) = (k{-}j\ k{-}j{+}1)$ представляет собой смежный обмен, а $\omega(k) = (n{-}1\ \ldots\ 0)(n{-}2\ \ldots\ 0)\ldots(k\ \ldots\ 0) = \phi(n-1)\phi(k-1)$ является k-переворотом, за которым следует n-переворот. Перестановка, соответствующая управляющей таблице $c_0 \ldots c_{n-1}$ в алгоритме H, имеет c_j элементов справа от j, которые меньше j, для $0 \leq j < n$; так что она точно такая же, как и перестановка, соответствующая $c_1 \ldots c_n$ в алгоритме P, за исключением того, что индексы смещены на 1.

Единственным существенным отличием алгоритма P и данной версии алгоритма H является то, что алгоритм P для прохода по всем вариантам управляющей таблицы использует рефлексивный код Грея, в то время как алгоритм H выполняет обход чисел со смешанным основанием счисления в возрастающем (лексикографическом) порядке.

Действительно, код Грея можно использовать вместе с любой таблицей Симса, изменяя либо алгоритм G, либо алгоритм H. Тогда не имеют значения все переходы, выполняемые $\tau(k,j)$ или $\tau(k,j)^-$, а также перестановки $\omega(k)$.

53. В приведенном в разделе доказательстве того, что при $n = 4$ нельзя получить $n!-1$ транспозиций, также показано, что можно свести эту задачу от n до $n-2$ ценой одной транспозиции $(n{-}1\ n{-}2)$, которая в обозначениях доказательства была названа '$(3\,c)$'.

Таким образом можно генерировать все перестановки с помощью следующего преобразования на шаге H4: если $k = n-1$ или $k = n-2$, то выполнить транспонирование $a_{j \bmod n} \leftrightarrow a_{(j-1) \bmod n}$, где $j = c_{n-1} - 1$. Если $k = n-3$ или $k = n-4$, транспонировать $a_{n-1} \leftrightarrow a_{n-2}$, а также $a_{j \bmod (n-2)} \leftrightarrow a_{(j-1) \bmod (n-2)}$, где $j = c_{n-3} - 1$. И в общем случае, если $k = n-2t-1$ или $k = n-2t-2$, транспонировать $a_{n-2i+1} \leftrightarrow a_{n-2i}$ для $1 \leq i \leq t$, а также $a_{j \bmod (n-2t)} \leftrightarrow a_{(j-1) \bmod (n-2t)}$, где $j = c_{n-2t-1} - 1$. [См. *CACM* **19** (1976), 68–72.]

Перестановки соответствующей таблицы Симса могут быть записаны следующим образом, хотя они и не появляются явно в самóм алгоритме:

$$\sigma(k,j)^- = \begin{cases} (0\ 1\ \ldots\ j{-}1\ k), & \text{если } n-k \text{ нечетно;} \\ (0\ 1\ \ldots\ k)^j, & \text{если } n-k \text{ четно.} \end{cases}$$

Значение $a_{j \bmod (n-2t)}$ после обмена будет равно $n-2t-1$. Для повышения эффективности можно также использовать тот факт, что k обычно равно $n-1$. Общее количество транспозиций равно $\sum_{t=0}^{\lfloor n/2 \rfloor}(n-2t)! - \lfloor n/2 \rfloor - 1$.

54. Да; это преобразование может быть любым k-циклом на позициях $\{1,\ldots,k\}$.

55. (а) Поскольку $\rho_!(m) = \rho_!(m \bmod n!)$, когда $n > \rho_!(m)$, мы имеем $\rho_!(n!+m) = \rho_!(m)$ для $0 < m < n \cdot n! = (n+1)! - n!$. Следовательно, $\beta_{n!+m} = \sigma_{\rho_!(n!+m)} \ldots \sigma_{\rho_!(n!+1)} \beta_{n!} = \sigma_{\rho_!(m)} \ldots \sigma_{\rho_!(1)} \beta_{n!} = \beta_m \beta_{n!}$ для $0 \le m < n \cdot n!$, и мы, в частности, имеем

$$\beta_{(n+1)!} = \sigma_{n+1} \beta_{(n+1)!-1} = \sigma_{n+1} \beta_{n!-1} \beta_{n!}^n = \sigma_{n+1} \sigma_n^- \beta_{n!}^{n+1}.$$

Аналогично $\alpha_{n!+m} = \beta_{n!}^- \alpha_m \beta_{n!} \alpha_{n!}$ для $0 \le m < n \cdot n!$.

Поскольку $\beta_{n!}$ коммутирует с τ_n и τ_{n+1}, мы находим $\alpha_{n!} = \tau_n \alpha_{n!-1}$ и

$$\begin{aligned} \alpha_{(n+1)!} = \tau_{n+1} \alpha_{(n+1)!-1} &= \tau_{n+1} \beta_{n!}^- \alpha_{(n+1)!-1-n!} \beta_{n!} \alpha_{n!} = \cdots \\ &= \tau_{n+1} \beta_{n!}^{-n} \alpha_{n!-1} (\beta_{n!} \alpha_{n!})^n = \beta_{n!}^{-n-1} \tau_{n+1} \tau_n^- (\beta_{n!} \alpha_{n!})^{n+1} \\ &= \beta_{(n+1)!}^- \sigma_{n+1} \sigma_n^- \tau_{n+1} \tau_n^- (\beta_{n!} \alpha_{n!})^{n+1}. \end{aligned}$$

(б) В этом случае $\sigma_{n+1}\sigma_n^- = (n\ n{-}1\ \ldots\ 1)$ и $\tau_{n+1}\tau_n^- = (n{+}1\ n\ 0)$, и по индукции мы имеем $\beta_{(n+1)!}\alpha_{(n+1)!} = (n{+}1\ n\ \ldots\ 0)$. Следовательно, $\alpha_{jn!+m} = \beta_{n!}^{-j}\alpha_m(n\ \ldots\ 0)^j$ для $0 \le j \le n$ и $0 \le m < n!$. Достигаются все перестановки $\{0,\ldots,n\}$, поскольку $\beta_{n!}^{-j}\alpha_m$ фиксирует n и $(n\ \ldots\ 0)^j$ отображает $n \mapsto n-j$.

56. Если в предыдущем упражнении установить $\sigma_k = (k{-}1\ k{-}2)(k{-}3\ k{-}4)\ldots$, то по индукции найдем, что $\beta_{n!}\alpha_{n!}$ представляет собой $(n+1)$-цикл $(0\ n\ n{-}1\ n{-}3\ \ldots$ (2 или 1) (1 или 2) $\ldots\ n{-}4\ n{-}2)$.

57. Те же рассуждения, что и в упр. 5, дают нам $\sum_{k=2}^{n-1}[k \text{ нечетно}]/k! - (\lfloor n/2 \rfloor - 1)/n! = \sinh 1 - 1 - O(1/(n-1)!)$.

58. Истинно. Согласно формулам из упр. 55, $\alpha_{n!-1} = (0\ n)\beta_{n!}^-(n\ \ldots\ 0)$ и $0 \mapsto n-1$, поскольку $\beta_{n!}$ фиксирует n. (Следовательно, алгоритм E определит гамильтонов *цикл* на графе из упр. 66 тогда и только тогда, когда $\beta_{n!} = (n{-}1\ \ldots\ 2\ 1)$, а это справедливо тогда и только тогда, когда длина каждого цикла $\beta_{(n-1)!}$ является делителем n. Последнее утверждение истинно для $n = 2$, 3, 4, 6, 12, 20 и 40, но не для прочих $n \le 250\,000$.)

59. Граф Кейли с генераторами $(\alpha_1,\ldots,\alpha_k)$ в определении в тексте данного раздела изоморфен графу Кейли с генераторами $(\alpha_1^-,\ldots,\alpha_k^-)$ в альтернативном определении, поскольку $\pi \to \alpha_j\pi$ в первом определении тогда и только тогда, когда $\pi^- \to \pi^-\alpha_j^-$ в последнем.

60. (а, б) Имеется 88 дельта-последовательностей, которые сводятся к четырем классам: $P = (32131231)^3$ (простые изменения, представлен восемью различными дельта-последовательностями); $Q = (32121232)^3$ (вариант двойного кода Грея для простых изменений с восемью представителями); $R = (121232321232)^2$ (двойной код Грея с 24 представителями); $S = 2\alpha 3\alpha^R$, $\alpha = 12321312121$ (48 представителей). Классы P и Q являются циклическими сдвигами своих дополнений; классы P, Q и S являются сдвигами своих обращений; класс R является сдвинутым обращением своего дополнения. [См. A. L. Leigh Silver, *Math. Gazette* **48** (1964), 1–16.]

61. Имеется $(26, 36, 20, 26, 28, 40, 40, 20, 26, 28, 28, 26)$ таких путей, заканчивающихся соответственно $(1243, 1324, 1432, 2134, 2341, 2413, 3142, 3214, 3421, 4123, 4231, 4312)$.

62. Для $n = 3$ есть только два пути, заканчивающихся соответственно 132 и 213. Но при $n \geq 4$ имеются коды Грея, ведущие от $12\ldots n$ к любой нечетной перестановке $a_1a_2\ldots a_n$. В упр. 61 это установлено для $n = 4$, и по индукции это можно доказать для $n > 4$ следующим образом.

Пусть $A(j)$ — множество всех перестановок, начинающихся с j, и пусть $A(j,k)$ — те из них, которые начинаются с jk. Если $(\alpha_0, \alpha_1, \ldots, \alpha_n)$ — любые нечетные перестановки, такие, что $\alpha_j \in A(x_j, x_{j+1})$, то $(1\,2)\alpha_j$ является четной перестановкой в $A(x_{j+1}, x_j)$. Следовательно, если $x_1x_2\ldots x_n$ — перестановка множества $\{1, 2, \ldots, n\}$, то существует как минимум один гамильтонов путь вида

$$(1\,2)\alpha_0 \text{ — } \cdots \text{ — } \alpha_1 \text{ — } (1\,2)\alpha_1 \text{ — } \cdots \text{ — } \alpha_2 \text{ — } \cdots \text{ — } (1\,2)\alpha_{n-1} \text{ — } \cdots \text{ — } \alpha_n;$$

подпуть от $(1\,2)\alpha_{j-1}$ до α_j включает все элементы $A(x_j)$.

Такое построение решает задачу по крайней мере $(n-2)!^n/2^{n-1}$ различными способами при $a_1 \neq 1$, поскольку можно принять $\alpha_0 = 2\,1\ldots n$ и $\alpha_n = a_1a_2\ldots a_n$; существует $(n-2)!$ способов выбора $x_2\ldots x_{n-1}$, и $(n-2)!/2$ способов выбора каждого из $\alpha_1, \ldots, \alpha_{n-1}$.

Наконец, если $a_1 = 1$, возьмем произвольный путь $12\ldots n$ — $\cdots$ — $a_1a_2\ldots a_n$, который проходит через все $A(1)$, и выберем произвольный шаг α — α' с $\alpha \in A(1,j)$ и $\alpha' \in A(1,j')$ для некоторого $j \neq j'$. Заменим этот шаг на

$$\alpha \text{ — } (1\,2)\alpha_1 \text{ — } \cdots \text{ — } \alpha_2 \text{ — } \cdots \text{ — } (1\,2)\alpha_{n-1} \text{ — } \cdots \text{ — } \alpha_n \text{ — } \alpha',$$

используя конструкцию наподобие приведенного выше гамильтонова пути, но теперь с $\alpha_1 = \alpha$, $\alpha_n = (1\,2)\alpha'$, $x_1 = 1$, $x_2 = j$, $x_n = j'$ и $x_{n+1} = 1$. (В этом случае все перестановки $\alpha_1, \ldots, \alpha_n$ могут быть четными.)

63. Оценки методом Монте-Карло с использованием методов из раздела 7.2.3 говорят о том, что общее количество классов эквивалентности можно грубо оценить как 1.2×10^{21}; большинство из этих классов будут содержать 480 циклов Грея.

64. Свойством двойного кода Грея обладают ровно 2 005 200 дельта-последовательностей; они принадлежат 4 206 классам эквивалентности по отношению к циклическому сдвигу, обращению и/или дополнению. Девять классов, таких как код $2\alpha2\alpha^R$, где

$$\alpha = 1234323432123212123232123212123434321212343212343212123232 1,$$

являются сдвигами своих обращений; 48 классов состоят из повторяющихся циклов длиной 60. Один из наиболее интересных классов последнего типа имеет вид $\alpha\alpha$, где

$$\alpha = \beta2\beta4\beta4\beta4\beta4, \qquad \beta = 32121232123.$$

65. Такой путь существует для любого заданного $N \leq n!$. Пусть N-я перестановка имеет вид $\alpha = a_1\ldots a_n$ и пусть $j = a_1$. Пусть также Π_k представляет собой множество всех перестановок $\beta = b_1\ldots b_n$, для которых $b_1 = k$ и $\beta \leq \alpha$. По индукции по N существует путь Грея P_1 для Π_j. Тогда можно построить пути Грея P_k для $\Pi_j \cup \Pi_1 \cup \cdots \cup \Pi_{k-1}$ для $2 \leq k \leq j$, последовательно комбинируя P_{k-1} с циклом Грея для Π_{k-1}. (См. построение из ответа к упр. 62. Фактически P_j будет *циклом* Грея при N, кратном 6.)

66. Определяя дельта-последовательность с помощью правила $\pi_{(k+1) \bmod n!} = (1\ \delta_k)\pi_k$, мы находим ровно 36 таких последовательностей, причем все они являются циклическими сдвигами шаблона типа $(xyzyzyxzyzyz)^2$. (Следующий случай, $n = 5$, вероятно, имеет около 10^{18} решений, не являющихся эквивалентными по отношению к циклическому сдвигу, обращению и перестановке координат, т. е. всего около 6×10^{21} различных дельта-последовательностей.) Кстати, Игорь Пак (Igor Pak) показал, что граф Кейли, сгенерированный звездными транспозициями, в общем случае является $(n-2)$-мерным тором.

67. Если положить π эквивалентным $\pi(12345)$, то можно получить приведенный граф с 24 вершинами, имеющий 40 768 гамильтоновых циклов, 240 из которых приводят к дельта-последовательностям вида α^5, в которых α использует каждую транспозицию 6 раз (например, $\alpha = 354232534234532454352452$). Общее количество решений этой задачи, вероятно, составляет около 10^{16}.

68. Если A не является связным, то связным не является и G. Если A — связный граф, можно считать, что он является свободным деревом. Более того, в этом случае можно доказать обобщение результата упр. 62: для $n \geq 4$ существует гамильтонов путь в G от тождественной перестановки к любой нечетной перестановке. Без потери общности можно считать, что A содержит ребро $1 — 2$, где 1 — лист дерева, и применить доказательство, подобное использованному в упр. 62.

[Эта элегантная конструкция принадлежит М. Чуэнте (M. Tchuente), *Ars Combinatoria* **14** (1982), 115–122. Широкие обобщения рассмотрены Раски (Ruskey) и Сэведж (Savage) в *SIAM J. Discrete Math.* **6** (1993), 152–166. См. также публикацию на русском языке *Кибернетика* **11**, 3 (1975), 17–21 (английский перевод — *Cybernetics* **11** (1975), 362–366).]

69. Если последовать указанию, модифицированный алгоритм будет вести себя при $n = 5$ так, как показано далее.

1234	1243	1423	4123	4132	1432	1342	1324	3124	3142	3412	4312
↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑
54321	24351	24153	54123	14523	14325	24315	24513	54213	14253	14352	54312
12345	15342	35142	32145	32541	52341	51342	31542	31245	35241	25341	21345
15432	12435	32415	35412←	31452	51432	52431	32451←	35421	31425	21435	25431
23451	53421	51423	21453→	25413	23415	13425	15423→	12453	52413	53412	13452
21543	51243	53241	23541	23145	25143	15243	13245	13542	53142	52143	12543
34512	34215	14235	14532	54132	34152	34251	54231	24531	24135	34125	34521
32154→	35124	15324→	12354	52314	32514←	31524	51324	21354→	25314	35214→	31254
45123←	42153	42351←	45321	41325	41523→	42513	42315	45312←	41352	41253←	45213
43215	43512←	41532	41235	45231→	43251	43152→	45132	42135	42531←	43521	43125
51234	21534→	23514	53214	13254←	15234	25134←	23154	53124	13524→	12534	52134
↓	↑	↓	↑	↓	↑	↓	↑	↓	↑	↓	↑

Здесь столбцы представляют множества перестановок, циклически поворачиваемых и/или отражаемых всеми $2n$ способами; следовательно, каждый столбец содержит ровно одну "четочную перестановку" (см. упр. 18). Для систематического перебора всех четочных перестановок можно прибегнуть к алгоритму P, зная, что пара xy находится в столбце перед yx, а применение τ' вместо ρ' будет перемещать нас вправо или влево. На шаге Z2 пропускается обмен $a_1 \leftrightarrow a_2$, тем самым заставляя перестановки $a_1 \ldots a_{n-1}$ повторять самих себя в обратном направлении. (Мы неявно используем тот факт, что в выходе алгоритма T $t[k] = t[n! - k]$.)

Теперь, если заменить $1 \ldots n$ на $24 \ldots 31$ и $A_1 \ldots A_n$ на $A_1 A_n A_2 A_{n-1} \ldots$, можно получить немодифицированный алгоритм, результат работы которого показан на рис. 42, (б).

Этот метод вдохновлен (неконструктивной) теоремой Э. Ш. Рапапорт (E. S. Rapaport), *Scripta Math.* **24** (1959), 51–58. Он иллюстрирует более общий факт, замеченный Карлой Сэведж (Carla Savage) в 1989 году, а именно то, что граф Кейли для *любой* группы, сгенерированной тремя инволюциями, ρ, σ, τ, имеет гамильтонов цикл, когда $\rho\tau = \tau\rho$. [См. I. Pak и R. Radoicić, *Discrete Math.* **309** (2009), 5501–5508.]

70. Нет; самый длинный цикл в этом ориентированном графе имеет длину 358. Но есть пары непересекающихся циклов длиной 180, из которых можно получить гамильтонов путь

длиной 720. Например, рассмотрим циклы $\alpha\sigma\beta\sigma$ и $\gamma\sigma\sigma$, где

$$\alpha = \tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^5\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^5\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2\tau\sigma^1\tau\sigma^1;$$
$$\beta = \sigma^3\tau\sigma^5\tau\sigma^2\tau\sigma^2\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^5\tau\sigma^1\tau\sigma^3\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^1\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2\tau\sigma^4;$$
$$\gamma = \sigma\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^5\tau\sigma^1\tau\sigma^5\tau\sigma^3\tau\sigma^2\tau\sigma^1\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^3\tau\sigma^2$$
$$\tau\sigma^5\tau\sigma^5\tau\sigma^5\tau\sigma^3\tau\sigma^5\tau\sigma^2\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^1\tau\sigma^5\tau\sigma^1\tau\sigma^3\tau\sigma^3\tau\sigma^5\tau\sigma^5\tau\sigma^1\tau\sigma^5\tau\sigma^2\tau\sigma^3\tau\sigma^1\tau\sigma^2.$$

Если мы начнем с 134526 и последуем $\alpha\sigma\beta\tau$, то достигнем 163452; следуя затем $\gamma\sigma\tau$, достигнем 126345; следуя затем $\sigma\gamma\tau$, достигнем 152634; следуя затем $\beta\sigma\alpha$, в итоге получим 415263.

71. Брендон Мак-Кей (Brendan McKay) и Фрэнк Раски (Frank Ruskey) нашли такие циклы с помощью компьютера для $n = 7$, 9 и 11, но какую-либо интересную структуру им выявить не удалось.

72. Любой гамильтонов путь включает $(n-1)!$ вершин, отображающих $y \mapsto x$, за каждой из которых (если она не последняя) следует вершина, отображающая $x \mapsto x$. Таким образом, одна вершина должна быть последней; в противном случае вершин, отображающих $x \mapsto x$, было бы $(n-1)! + 1$.

73. (а) Сначала предположим, что β является тождественной перестановкой (). Тогда каждый цикл α, который содержит элемент A, целиком лежит в A. Следовательно, циклы σ получаются путем опускания всех циклов α, не содержащих элементов A. Все остальные циклы имеют нечетную длину, так что σ представляет собой четную перестановку.

Если β не является тождественной перестановкой, применим эти рассуждения к $\alpha' = \alpha\beta^-$, $\beta' = ()$ и $\sigma' = \sigma\beta^-$, делая вывод о том, что σ' является четной перестановкой; таким образом, σ и β имеют один и тот же знак.

Аналогично одинаковые знаки имеют σ и α, поскольку $\beta\alpha^- = (\alpha\beta^-)^-$ имеет тот же порядок, что и $\alpha\beta^-$.

(б) Пусть X — вершины графа Кейли из теоремы R и пусть $\hat\alpha$ — перестановка X, отображающая вершину π в $\alpha\pi$; эта перестановка имеет g/a циклов длиной a. Аналогично определим перестановку $\hat\beta$. Тогда $\hat\alpha\hat\beta^-$ имеет g/c циклов длиной c. Если c нечетно, любой гамильтонов цикл графа определяет цикл $\hat\sigma$, который содержит все вершины и удовлетворяет гипотезе из п. (а). Следовательно, $\hat\alpha$ и $\hat\beta$ имеют нечетное количество циклов, поскольку знак перестановки n элементов с r циклами равен $(-1)^{n-r}$ (см. упр. 5.2.2–2).

[Это доказательство, которое показывает, что X не может быть объединением любого нечетного числа циклов, представлено Ранкином (Rankin) в *Proc. Cambridge Phil. Soc.* **62** (1966), 15–16.]

74. Представление $\beta^j\gamma^k$ единственное, если потребовать $0 \le j < g/c$ и $0 \le k < c$. Если мы имеем $\beta^j = \gamma^k$ для некоторого $0 < j < g/c$, то группа будет иметь не более jc элементов. Отсюда следует, что $\beta^{g/c} = \gamma^t$ для некоторого t.

Пусть $\hat\sigma$ является гамильтоновым циклом, как в ответе к предыдущему упражнению, и пусть $\hat\gamma = \hat\alpha\hat\beta^-$. Если $\pi\hat\sigma = \pi\hat\alpha$, то $\pi\hat\gamma\hat\sigma$ должно быть равно $\pi\hat\gamma\hat\alpha$, поскольку $\pi\hat\gamma\hat\beta = \pi\hat\alpha$. А если $\pi\hat\sigma = \pi\hat\beta$, то $\pi\hat\gamma\hat\sigma$ не может быть равно $\pi\hat\gamma\hat\alpha$, поскольку отсюда вытекало бы $\pi\hat\gamma^2\hat\sigma = \pi\hat\gamma^2\hat\alpha, \ldots, \pi\hat\gamma^c\hat\sigma = \pi\hat\gamma^c\hat\alpha$. Таким образом, все элементы π, $\pi\hat\gamma$, $\pi\hat\gamma^2$, ... имеют эквивалентное поведение по отношению к их преемникам в $\hat\sigma$.

Когда путь $\pi \longrightarrow \pi\hat\sigma \longrightarrow \cdots \longrightarrow \pi\hat\sigma^j$ имеет $k(j)$ шагов типа $\hat\alpha$, имеем $\pi\hat\sigma^j = \pi\hat\beta^j\hat\gamma^{k(j)}$. Таким образом, $\pi\hat\sigma^{g/c} = \pi\hat\gamma^{t+k(g/c)}$, и мы имеем $k(j + g/c) = k(j)$ для $j \ge 0$. Путь возвращается к π в первый раз за g шагов тогда и только тогда, когда $t + k(g/c)$ является взаимно простым с c.

75. Воспользуемся результатом предыдущего упражнения с $g = mn$, $a = m$, $b = n$, $c = mn/d$. Число t удовлетворяет условиям $t \equiv 0$ (по модулю m) и $t + d \equiv 0$ (по модулю n); отсюда следует, что $k + t \perp c$ тогда и только тогда, когда $(d-k)m/d \perp kn/d$.

Примечания. Модульный код Грея из упр. 7.2.1.1–78 представляет собой гамильтонов путь от $(0,0)$ до $(m-1, (-m) \bmod n)$, так что он является гамильтоновым циклом тогда и только тогда, когда m кратно n. Естественно было бы сделать предположение (увы, ложное), что при $d > 1$ имеется как минимум один гамильтонов цикл. Но П. Эрдеш (P. Erdös) и У. Т. Троттер (W. T. Trotter) обнаружили [*J. Graph Theory* **2** (1978), 137–142], что если p и $2p+1$ — нечетные простые числа, то при $m = p(2p+1)(3p+1)$ и $n = (3p+1)\prod_{q=1}^{3p} q^{[q\,\text{простое}][q\neq p][q\neq 2p+1]}$ подходящего k не существует.

Интересные факты о других типах циклов в $\vec{C_m} \times \vec{C_n}$ можно найти в работе J. A. Gallian, *Mathematical Intelligencer* **13**, 3 (Summer 1991), 40–43.

76. Можно считать, что обход начинается в левом нижнем углу. Решений при m и n, делящихся на 3, не существует, поскольку 2/3 ячеек в этом случае недостижимы. В противном случае положим $d = \gcd(m,n)$ и, рассуждая, как в предыдущем упражнении, но с $(x,y)\alpha = ((x+2) \bmod m, (y+1) \bmod n)$ и $(x,y)\beta = ((x+1) \bmod m, (y+2) \bmod n)$, найдем ответ

$$\sum_{k=0}^{d} \binom{d}{k} \bigl[\gcd((2d-k)m, (k+d)n) = d \text{ или } (mn \perp 3 \text{ и } \gcd((2d-k)m, (k+d)n) = 3d)\bigr].$$

77.
```
01  * Генератор перестановки по типу Хипа
02  N     IS    10              Значение n (3 или больше, но не слишком).
03  t     IS    $255
04  j     IS    $0              8j
05  k     IS    $1              8k
06  ak    IS    $2
07  aj    IS    $3

08        LOC   Data_Segment
09  a     GREG  @               Базовый адрес для a_0 ... a_{n-1}.
10  A0    IS    @
11  A1    IS    @+8
12  A2    IS    @+16
13        LOC   @+8*N           Память для a_0 ... a_{n-1}.
14  c     GREG  @-8*3           Ячейка 8c_0.
15        LOC   @-8*3+8*N       8c_3 ... 8c_{n-1}, изначально — нуль.
16        OCTA  -1              8c_n = -1, удобный ограничитель.
17  u     GREG  0               Содержимое a_0, кроме внутреннего цикла.
18  v     GREG  0               Содержимое a_1, кроме внутреннего цикла.
19  w     GREG  0               Содержимое a_2, кроме внутреннего цикла.

20        LOC   #100
21  1H    STCO  0,c,k       B-A   c_k ← 0.
22        INCL  k,8         B-A   k ← k+1.
23  0H    LDO   j,c,k         B   j ← c_k.
24        CMP   t,j,k         B
25        BZ    t,1B          B   Цикл, если c_k = k.
26        BN    j,Done        A   Завершить работу, если c_k < 0 (k = n).
27        LDO   ak,a,k      A-1   Выборка a_k.
28        ADD   t,j,8       A-1
29        STO   t,c,k       A-1   c_k ← j+1.
```

30		AND	t,k,#8	$A-1$	
31		CSZ	j,t,0	$A-1$	Установить $j \leftarrow 0$, если k четно.
32		LDO	aj,a,j	$A-1$	Выборка a_j.
33		STO	ak,a,j	$A-1$	Заменить на a_k.
34		CSZ	u,j,ak	$A-1$	Установить $u \leftarrow a_k$, если $j=0$.
35		SUB	j,j,8	$A-1$	$j \leftarrow j-1$.
36		CSZ	v,j,ak	$A-1$	Установить $v \leftarrow a_k$, если $j=0$.
37		SUB	j,j,8	$A-1$	$j \leftarrow j-1$.
38		CSZ	w,j,ak	$A-1$	Установить $w \leftarrow a_k$, если $j=0$.
39		STO	aj,a,k	$A-1$	Заменить a_k тем, что было a_j.
40	Inner	PUSHJ	0,Visit	A	
		...			(См. (42))
55		PUSHJ	0,Visit	A	
56		SET	t,u	A	Обмен $u \leftrightarrow w$.
57		SET	u,w	A	
58		SET	w,t	A	
59		SET	k,8*3	A	$k \leftarrow 3$.
60		JMP	OB	A	
61	Main	LDO	u,A0	1	
62		LDO	v,A1	1	
63		LDO	w,A2	1	
64		JMP	Inner	1	▮

78. Строки 31–38 дают $2r-1$ инструкций, строки 61–63 дают r, а строки 56–58 — $3+(r-2)[r$ четно$]$ инструкций (см. $\omega(r-1)$ в ответе к упр. 40). Общее время работы, таким образом, равно $((2r!+2)A+2B+r-5)\mu+((2r!+2r+7+(r-2)[r\text{ четно}])A+7B-r-4)\upsilon$, где $A=n!/r!$ и $B=n!(1/r!+\cdots+1/n!)$.

79. `SLU u,[#f],t; SLU t,a,4; XOR t,t,a; AND t,t,u; SRU u,t,4; OR t,t,u; XOR a,a,t;` здесь, как и в ответе к упр. 1.3.1′–34, запись '`[#f]`' означает регистр, который содержит константное значение `#f`. (См. аналогичный код в 7.1.3–(69).)

80. `SLU u,a,t; MXOR u,[#8844221188442211],u; AND u,u,[#ff000000]; SRU u,u,t; XOR a,a,u.` Это легкое жульничество, поскольку при $t=4$ код преобразует `#12345678` в `#13245678`, но (45) все равно работает.

Еще быстрее и хитрее была бы подпрограмма, аналогичная (42): рассмотрим

```
PUSHJ 0,Visit; MXOR a,a,c1; PUSHJ 0,Visit; ... MXOR a,a,c5; PUSHJ 0,Visit
```

где `c1`, ..., `c5` представляют собой константы, которые приводят к последовательному преобразованию `#12345678` в `#12783456`, `#12567834`, `#12563478`, `#12785634`, `#12347856`. Прочие инструкции, выполняемые с относительной частотой всего лишь 1/6 или 1/24, могут позаботиться о перетасовке полубайтов внутри байтов и между ними. Этот остроумный способ не лучше (46) из-за накладных расходов на выполнение `PUSHJ/POP`.

81.
```
k  IS  $0 ;kk IS $1 ;c IS $2 ;d IS $3
   SET k,1      k ← 1.
3H SRU d,a,60   d ← крайний слева полубайт.
   SLU a,a,4    a ← 16a mod 16^16.
   CMP c,d,k
   SLU kk,k,2
   SLU d,d,kk
   OR  t,t,d    t ← t + 16^k d.
```

```
PBNZ c,1B   Вернуться в основной цикл, если d ≠ k.
INCL k,1    k ← k + 1.
PBNZ a,3B   Вернуться во второй цикл, если k < n.  ▌
```

82. $\mu + \bigl(5n! + 11A - (n-1)! + 6\bigr)\upsilon = \bigl((5 + 10/n)\upsilon + O(n^{-2})\bigr)n!$ плюс время посещения, где $A = \sum_{k=1}^{n-1} k!$ — количество итераций цикла 3H.

83. При соответствующей инициализации и таблице из 13 октабайт потребуется только около десятка инструкций MMIX.

```
magic  GREG #8844221188442211
0H     ⟨Посетить регистр a.⟩
       PBN  c,Sigma
Tau    MXOR t,magic,a; ANDNL t,#ffff; JMP 1F
Sigma  SRU  t,a,20; SLU a,a,4; ANDNML a,#f00
1H     XOR  a,a,t; SLU c,c,1
2H     PBNZ c,0B; INCL p,8
3H     LDOU c,p,0; PBNZ c,0B                ▌
```

84. Считая, что все процессоры имеют одну и ту же скорость работы, мы можем предложить, чтобы k-й процессор генерировал все перестановки ранга $(k-1)n!/p \le r < kn!/p$, используя любые методы, основанные на управляющих таблицах $c_1 \ldots c_n$. Начальную и конечную управляющие таблицы легко вычислить путем преобразования их рангов в числа в системе счисления со смешанными основаниями (см. упр. 12).

85. Можно воспользоваться методом наподобие используемого в алгоритме 3.4.2P: для вычисления $k = r(\alpha)$ сначала установить $a'_{a_j} \leftarrow j$ для $1 \le j \le n$ (обратная перестановка). Затем установить $k \leftarrow 0$ и для $j = n, n-1, \ldots, 2$ (в указанном порядке) установить $t \leftarrow a'_j$, $k \leftarrow kj + t - 1$, $a_t \leftarrow a_j$, $a'_{a_j} \leftarrow t$. Для вычисления $r^{[-1]}(k)$ следует начать с $a_1 \leftarrow 1$. Затем для $j = 2, \ldots, n-1, n$ (в указанном порядке) установить $t \leftarrow (k \bmod j) + 1$, $a_j \leftarrow a_t$, $a_t \leftarrow j$, $k \leftarrow \lfloor k/j \rfloor$. [См. S. Pleszczyński, *Inf. Proc. Letters* **3** (1975), 180–183; W. Myrvold and F. Ruskey, *Inf. Proc. Letters* **79** (2001), 281–284.]

Если же нам нужно искать ранг объекта и объект по рангу только для $n^{\underline{m}}$ *вариаций* $a_1 \ldots a_m$ множества $\{1, \ldots, n\}$, то предпочтительнее использовать другой метод. Для вычисления $k = r(a_1 \ldots a_m)$ следует начать с $b_1 \ldots b_n \leftarrow b'_1 \ldots b'_n \leftarrow 1 \ldots n$; затем для $j = 1, \ldots, m$ (в указанном порядке) установить $t \leftarrow b'_{a_j}$, $b_t \leftarrow b_{n+1-j}$ и $b'_{b_t} \leftarrow t$; наконец установить $k \leftarrow 0$ и для $j = m, \ldots, 1$ (в указанном порядке) установить $k \leftarrow k \times (n+1-j) + b'_{a_j} - 1$. Для вычисления $r^{[-1]}(k)$ следует начать с $b_1 \ldots b_n \leftarrow 1 \ldots n$; затем для $j = 1, \ldots, m$ (в указанном порядке) установить $t \leftarrow \bigl(k \bmod (n+1-j)\bigr) + 1$, $a_j \leftarrow b_t$, $b_t \leftarrow b_{n+1-j}$, $k \leftarrow \lfloor k/(n+1-j) \rfloor$. (См. упр. 3.4.2–15 для случаев больши́х n и малых m.)

86. Если $x \prec y$ и $y \prec z$, алгоритм никогда не поместит ни y слева от x, ни z слева от y, так что он никогда не будет выяснять взаимоотношение x и z.

87. Они находятся в лексикографическом порядке; алгоритм P использует рефлексивный порядок Грея.

88. Сгенерируйте обратные перестановки с $a'_0 < a'_1 < a'_2$, $a'_3 < a'_4 < a'_5$, $a'_6 < a'_7$, $a'_8 < a'_9$, $a'_0 < a'_3$, $a'_6 < a'_8$.

89. (а) Пусть $d_k = \max\{j \mid 0 \le j \le k \text{ и } j \text{ нетривиален}\}$, где 0 считается нетривиальным. Такую таблицу легко предвычислить, поскольку j тривиален тогда и только тогда, когда он следует за $\{1, \ldots, j-1\}$. В алгоритм следует внести следующие изменения: установить $k \leftarrow d_n$ на шаге V2 и $k \leftarrow d_{k-1}$ на шаге V5. (Считаем $d_n > 0$.)

(б) Теперь $M = \sum_{j=1}^{n} t_j[j \text{ нетривиален}]$.

(в) Имеется не менее двух топологических сортировок $a_j \ldots a_k$ множества $\{j, \ldots, k\}$, и каждая из них может быть размещена после любой топологической сортировки $a_1 \ldots a_{j-1}$ множества $\{1, \ldots, j-1\}$.

(г) Алгоритм 2.2.3T многократно выводит минимальные элементы (т. е. элементы, не имеющие предшественников), удаляя их из графа отношений. Мы используем его в обратном порядке, повторно удаляя и присваивая наивысшие метки *максимальным* элементам (т. е. элементам, не имеющим преемников). Если существует только один максимальный элемент, он тривиален. Если и k, и l максимальны, они оба выводятся до любого элемента x, для которого справедливо отношение $x \prec k$ или $x \prec l$, поскольку шаги T5 и T7 хранят максимальные элементы в очереди, а не в стеке. Таким образом, если k нетривиален и выводится первым, то элемент l может стать тривиальным, но следующий нетривиальный элемент j не будет выводиться перед l; а k не связан никаким отношением с l.

(д) Пусть для нетривиальных t выполняются условия $s_1 < s_2 < \cdots < s_r = N$. Тогда согласно (в) имеем $s_j \geq 2s_{j-2}$. Следовательно, $M = s_2 + \cdots + s_r \leq s_r(1 + \frac{1}{2} + \frac{1}{4} + \cdots) + s_{r-1}(1 + \frac{1}{2} + \frac{1}{4} + \cdots) < 4s_r$.

(На самом деле истинна более сильная оценка, как заметил M. Печарски (M. Peczarski): пусть $s_0 = 1$, нетривиальными индексами являются $0 = k_1 < k_2 < \cdots < k_r$ и пусть $k'_j = \max\{k \mid 1 \leq k < k_j,\ k \not\prec k_j\}$ для $j > 1$. Тогда $k'_{j+1} \geq k_j$. Существует s_j топологических сортировок $\{1, \ldots, k_{j+1}\}$, которые заканчиваются k_{j+1}; и существует как минимум s_{j-1} сортировок, которые заканчиваются k'_{j+1}, поскольку каждая из s_{j-1} топологических сортировок $\{1, \ldots, k_j - 1\}$ может быть расширена. Следовательно,

$$s_{j+1} \;\geq\; s_j + s_{j-1} \qquad \text{для } 1 \leq j < r.$$

Пусть теперь $y_0 = 0$, $y_1 = F_2 + \cdots + F_r$ и $y_j = y_{j-2} + y_{j-1} - F_{r+1}$ для $1 < j < r$. Тогда

$$F_{r+1}(s_1 + \cdots + s_r) + \sum_{j=1}^{r-1} y_j(s_{r+1-j} - s_{r-j} - s_{r-1-j}) \;=\; (F_2 + \cdots + F_{r+1})s_r,$$

и каждое значение $y_j = F_{r+1} - 2F_j - (-1)^j F_{r+1-j}$ является неотрицательным. Следовательно, $s_1 + \cdots + s_r \leq ((F_2 + \cdots + F_{r+1})/F_{r+1})s_r \approx 2.6s_r$. В следующем упражнении показано, что данная граница является наилучшей возможной.)

90. Согласно упр. 5.2.1–25 количество таких перестановок N равно F_{n+1}. Следовательно, $M = F_{n+1} + \cdots + F_2 = F_{n+3} - 2 \approx \phi^2 N$. Кстати, заметим, что все такие перестановки удовлетворяют условию $a_1 \ldots a_n = a'_1 \ldots a'_n$, и их можно упорядочить в виде пути Грея (см. упр. 7.2.1.1–89).

91. Поскольку $t_j = (j-1)(j-3)\ldots(2 \text{ или } 1)$, находим, что $M = \bigl(1 + 2/\sqrt{\pi n} + O(1/n)\bigr)N$.

Примечание. Таблица инверсий $c_1 \ldots c_{2n}$ для перестановок, удовлетворяющих соотношениям (49), характеризуется условиями $c_1 = 0$, $0 \leq c_{2k} \leq c_{2k-1}$, $0 \leq c_{2k+1} \leq c_{2k-1} + 1$.

92. Общее количество пар (R, S), где R — частичное упорядочение, а S — линейное упорядочение, включающее R, равно P_n, умноженному на ожидаемое количество топологических сортировок; оно также равно Q_n, умноженному на $n!$. Таким образом, ответ имеет вид $n!\,Q_n/P_n$.

Вычисление P_n и Q_n будет рассматриваться в разделе 7.2.3. Для $1 \leq n \leq 12$ ожидаемые значения приближенно равны

$$(1, 1.33, 2.21, 4.38, 10.1, 26.7, 79.3, 262, 950, 3760, 16200, 74800).$$

Асимптотические значения при $n \to \infty$ были получены Брайтвеллом (Brightwell), Прёмелем (Prömel) и Стигер (Steger) [*J. Combinatorial Theory* **A73** (1996), 193–206], но предельное

поведение очень сильно отличается от поведения при практических значениях n. Значения Q_n были впервые получены для $n \le 5$ Ш. П. Аванном (S. P. Avann) [*?quationes Math.* **8** (1972), 95–102].

93. Основная идея заключается в введении фиктивных элементов $n+1$ и $n+2$, таких, что $j \prec n+1$ и $j \prec n+2$ для $1 \le j \le n$, и поиске для таких расширенных отношений всех топологических сортировок, осуществляющихся с помощью смежных обменов. Затем можно взять каждую *вторую* перестановку, игнорируя фиктивные элементы. Можно использовать алгоритм, аналогичный алгоритму V, но с рекурсией, сводящей n к $n-2$ путем вставки $n-1$ и n среди $a_1 \ldots a_{n-2}$ всеми возможными путями, в предположении, что $n-1 \not\prec n$, время от времени обменивая $n+1$ с $n+2$. [См. G. Pruesse and F. Ruskey, *SICOMP* **23** (1994), 373–386. Реализация без применения циклов описана в работе Canfield and Williamson, *Order* **12** (1995), 57–75.]

94. Случай $n = 3$ иллюстрирует общую идею шаблона, который начинается с $1 \ldots (2n)$ и заканчивается $1(2n)2(2n{-}1)\ldots n(n{+}1)$: 123456, 123546, 123645, 132645, 132546, 132456, 142356, 142536, 142635, 152634, 152436, 152346, 162345, 162435, 162534.

Совершенные паросочетания можно рассматривать как инволюции $\{1, \ldots, 2n\}$, имеющие n циклов. При таком представлении рассматриваемый шаблон включает две транспозиции на шаг.

Обратите внимание, что таблицы инверсий C только что перечисленных перестановок представляют собой соответственно 000000, 000100, 000200, 010200, 010100, 010000, 020000, 020100, 020200, 030200, 030100, 030000, 040000, 040100, 040200. В общем случае $C_1 = C_3 = \cdots = C_{2n-1} = 0$ и n-кортежи $(C_2, C_4, \ldots, C_{2n})$ проходят по рефлексивному коду Грея с основаниями $(2n-1, 2n-3, \ldots, 1)$. Таким образом, при желании процесс генерации легко сделать бесцикловым. [См. Timothy Walsh, *J. Combinatorial Math. and Combinatorial Computing* **36** (2001), 95–118, Section 1.]

Примечание. Алгоритмы для генерации всех совершенных паросочетаний восходят к работе И. Ф. Пфаффа (J. F. Pfaff) [*Abhandlungen Akad. Wissenschaften* (Berlin: 1814–1815), 124–125], который описал две такие процедуры. Его первый метод является лексикографическим, а также соответствует лексикографическому порядку таблиц инверсий C; его второй метод соответствует *солексному* порядку этих таблиц. В обоих случаях наблюдается чередование четных и нечетных перестановок.

95. Сгенерируйте обратные перестановки с $a'_1 < a'_n > a'_2 < a'_{n-1} > \cdots$ с помощью алгоритма V. (Количество решений приведено в упр. 5.1.4–23.)

96. Например, можно начать с $a_1 \ldots a_{n-1} a_n = 2 \ldots n1$ и $b_1 b_2 \ldots b_n b_{n+1} = 12 \ldots n1$ и использовать алгоритм P для генерации $(n-1)!$ перестановок $b_2 \ldots b_n$ множества $\{2, \ldots, n\}$. Сразу после того, как этот алгоритм выполнит обмен $b_i \leftrightarrow b_{i+1}$, мы устанавливаем $a_{b_{i-1}} \leftarrow b_i$, $a_{b_i} \leftarrow b_{i+1}$, $a_{b_{i+1}} \leftarrow b_{i+2}$ и посещаем $a_1 \ldots a_n$.

97. Воспользуйтесь алгоритмом X с $t_k(a_1, \ldots, a_k) =$ '$a_k \ne k$'.

98. Используя обозначения из упр. 47, мы имеем в соответствии с методом включения и исключения (упр. 1.3.3–26) $N_k = \sum \binom{k}{j}(-1)^j (n-j)^{\underline{k-j}}$. Если $k = O(\log n)$, то $N_{n-k} = (n!\,e^{-1}/k!)(1 + O(\log n)^2/n)$; следовательно, $A/n! \approx (e-1)/e$ и $B/n! \approx 1$. Количество обращений к памяти при предположениях из ответа к упр. 48, таким образом, составляет $\approx A + B + 3A + B - N_n + 3A \approx n!\,(9 - \frac{8}{e}) \approx 6.06n!$, около 16.5 на одно неупорядочение. [См. аналогичный метод в работе S. G. Akl, *BIT* **20** (1980), 2–7.]

99. Предположим, что L_n генерирует $D_n \cup D_{n-1}$, начиная с $(1\ 2\ \ldots\ n)$, затем $(2\ 1\ \ldots\ n)$ и заканчивая $(1\ \ldots\ n{-}1)$; например, $L_3 = (1\ 2\ 3), (2\ 1\ 3), (1\ 2)$. Тогда можно генерировать D_{n+1} как $K_{nn}, \ldots, K_{n2}, K_{n1}$, где $K_{nk} = (1\ 2\ \ldots\ n)^{-k}(n\ n{+}1)L_n(1\ 2\ \ldots\ n)^k$; например, D_4

представляет собой

$$(1\,2\,3\,4),\ (2\,1\,3\,4),\ (1\,2)(3\,4),\ (3\,1\,2\,4),\ (1\,3\,2\,4),\ (3\,1)(2\,4),\ (2\,3\,1\,4),\ (3\,2\,1\,4),\ (2\,3)(1\,4).$$

Обратите внимание, что K_{nk} начинается с цикла $(k{+}1\ \ldots\ n\ 1\ \ldots\ k\ n{+}1)$, а заканчивается $(k{+}1\ \ldots\ n\ 1\ \ldots\ k{-}1)(k\ n{+}1)$; так что умножение слева на $(k{-}1\ k)$ переносит нас от K_{nk} к $K_{n(k-1)}$. Кроме того, умножение слева на $(1\ n)$ возвращает нас от последнего элемента D_{n+1} к первому. Умножение слева на $(1\ 2\ n{+}1)$ возвращает нас от последнего элемента D_{n+1} к $(2\ 1\ 3\ \ldots\ n)$, откуда мы можем вернуться к $(1\ 2\ \ldots\ n)$, следуя циклу для D_n в обратном направлении, тем самым завершая список L_{n+1}, что и требовалось.

100. Используйте алгоритм X с $t_k(a_1,\ldots,a_k) =$ '$p > 0$ или $l[q] \neq k+1$'.

Примечания. Количество неприводимых перестановок равно $[z^n]\,(1 - 1/\sum_{k=0}^{\infty} k!\,z^k)$; см. Л. Комте (L. Comtet), *Comptes Rendus Acad. Sci.* **A275** (Paris, 1972), 569–572.

Э. Д. Кинг (A. D. King) [*Discrete Math.* **306** (2006), 508–516] показал, что неприводимые перестановки можно эффективно генерировать с помощью одной транспозиции на каждом шаге. В действительности для этого достаточно *смежных* транспозиций; например, когда $n = 4$, неприводимыми перестановками являются 3142, 3412, 3421, 3241, 2341, 2431, 4231, 4321, 4312, 4132, 4123, 4213, 2413.

101. Вот как выглядит лексикографический генератор инволюций, аналогичный алгоритму X.

Y1. [Инициализация.] Установить $a_k \leftarrow k$ и $l_{k-1} \leftarrow k$ для $1 \le k \le n$. Затем установить $l_n \leftarrow 0$, $k \leftarrow 1$.

Y2. [Вход на уровень k.] Если $k > n$, посетить $a_1 \ldots a_n$ и перейти к шагу Y3. В противном случае установить $p \leftarrow l_0$, $u_k \leftarrow p$, $l_0 \leftarrow l_p$, $k \leftarrow k+1$ и повторить данный шаг. (Мы решили положить $a_p = p$.)

Y3. [Уменьшение k.] Установить $k \leftarrow k-1$ и завершить работу алгоритма, если $k = 0$. В противном случае установить $q \leftarrow u_k$ и $p \leftarrow a_q$. Если $p = q$, установить $l_0 \leftarrow q$, $q \leftarrow 0$, $r \leftarrow l_p$ и $k \leftarrow k+1$ (подготовка к $a_p > p$). В противном случае установить $l_{u_{k-1}} \leftarrow q$, $r \leftarrow l_q$ (подготовка к $a_p > q$).

Y4. [Увеличение a_p.] Если $r = 0$, перейти к шагу Y5. В противном случае установить $l_q \leftarrow l_r$, $u_{k-1} \leftarrow q$, $u_k \leftarrow r$, $a_p \leftarrow r$, $a_q \leftarrow q$, $a_r \leftarrow p$, $k \leftarrow k+1$ и перейти к шагу Y2.

Y5. [Восстановление a_p.] Установить $l_0 \leftarrow p$, $a_p \leftarrow p$, $a_q \leftarrow q$, $k \leftarrow k-1$ и вернуться к шагу Y3. ▮

Пусть $t_{n+1} = t_n + nt_{n-1}$, $a_{n+1} = 1 + a_n + na_{n-1}$, $t_0 = t_1 = 1$, $a_0 = 0$, $a_1 = 1$ (см. 5.1.4–(40)). Шаг Y2 выполняется t_n раз с $k > n$ и a_n раз с $k \le n$. Шаг Y3 выполняется a_n раз с $p = q$, а всего — $a_n + t_n$ раз. Шаг Y4 выполняется $t_n - 1$ раз; шаг Y5 — a_n раз. Общее количество обращений к памяти для всех t_n выводов, таким образом, приближенно равно $11a_n + 12t_n$, где $a_n < 1.25331414t_n$. (Если скорость критична, данный алгоритм можно оптимизировать.)

102. Построим список L_n, который начинается с $()$ и заканчивается $(n{-}1\ n)$, начиная с $L_3 = (),\ (1\ 2),\ (1\ 3),\ (2\ 3)$. Если n нечетно, то L_{n+1} имеет вид $L_n,\ K_{n1}^R,\ K_{n2},\ \ldots,\ K_{nn}^R$, где $K_{nk} = (k\ \ldots\ n)^{-}\,L_{n-1}(k\ \ldots n)(k\ n{+}1)$. Например,

$$L_4 \;=\; (),\ (1\ 2),\ (1\ 3),\ (2\ 3),\ (2\ 3)(1\ 4),\ (1\ 4),\ (2\ 4),\ (1\ 3)(2\ 4),\ (1\ 2)(3\ 4),\ (3\ 4).$$

Если n четно, L_{n+1} имеет вид $L_n,\ K_{n(n-1)},\ K_{n(n-2)}^R,\ \ldots,\ K_{n1},\ (1\ n{-}2)\,L_{n-1}^R\,(1\ n{-}2)(n\ n{+}1)$.

Более подробно эта тема рассматривается в статье Уолша (Walsh), упомянутой в ответе к упр. 94.

103. Следующее элегантное решение Карлы Сэведж (Carla Savage) требует только $n-2$ различных операций ρ_j для $1<j<n$, где ρ_j заменяет $a_{j-1}a_ja_{j+1}$ на $a_{j+1}a_{j-1}a_j$ при четном j и на $a_ja_{j+1}a_{j-1}$ при j нечетном. Можно считать, что $n\ge 4$; пусть $A_4=(\rho_3\rho_2\rho_2\rho_3)^3$. В общем случае A_n будет начинаться и заканчиваться ρ_{n-1}, а всего будет содержать $2n-2$ появлений ρ_{n-1}. Для получения A_{n+1} заменим k-ю операцию ρ_{n-1} в A_n на $\rho_nA'_n\rho_n$, где $k=1, 2, 4, \ldots, 2n-2$, если n четно, и $k=1, 3, \ldots, 2n-3, 2n-2$, если n нечетно, и где A'_n равно A_n с удаленными первым и последним элементами. Тогда, если начать с $a_1\ldots a_n=1\ldots n$, операции ρ_{n-1} в A_n приведут к тому, что позиция a_n последовательно пройдет значения $n\to p_1\to n\to p_2\to\cdots\to p_{n-1}\to n$, где $p_1\ldots p_{n-1}=(n-1-[n\,\text{четно}])\ldots 4213\ldots(n-1-[n\,\text{нечетно}])$; последней перестановкой вновь будет $1\ldots n$.

104. (а) У хорошо сбалансированной перестановки сумма $\sum_{k=1}^n ka_k=n(n+1)^2/4$ представляет собой целое число.

(б) Замените k на a_k при суммировании по k.

(в) Достаточно быстрый способ подсчета при небольших n может быть основан на оптимизированном алгоритме простых изменений из упр. 16, поскольку значение $\sum ka_k$ при каждом смежном обмене изменяется простым образом и поскольку $n-1$ из каждых n шагов являются "колебаниями", которые могут быть выполнены очень быстро. Половину работы можно сэкономить, рассматривая только те перестановки, в которых 1 предшествует 2. Искомыми значениями для $1\le n\le 15$ являются 0, 0, 0, 2, 6, 0, 184, 936, 6688, 0, 420480, 4298664, 44405142, 0, 6732621476.

105. (а) Для каждой перестановки $a_1\ldots a_n$ вставьте $\prec$ между a_j и a_{j+1}, если $a_j>a_{j+1}$; вставьте между ними либо $\equiv$, либо $\prec$, если $a_j<a_{j+1}$. (Перестановка с k "подъемами", таким образом, дает 2^k слабых порядков. Слабые порядки иногда называют "предпочтительным упорядочением"; в упр. 5.3.1–4 показано, что их общее количество приблизительно равно $n!/(2(\ln 2)^{n+1})$. Код Грея для слабых порядков, в котором на каждом шаге выполняется обмен $\prec\leftrightarrow\equiv$ и/или $a_j\leftrightarrow a_{j+1}$, может быть получен путем комбинирования алгоритма P с бинарным кодом Грея на подъемах.

(б) Начнем с $a_1\ldots a_na_{n+1}=0\ldots 00$ и $a_0=-1$. Будем выполнять алгоритм L до тех пор, пока он не остановится с $j=0$. Найдем k, такое, что $a_1>\cdots>a_k=a_{k+1}$, и завершим работу, если $k=n$. В противном случае установим $a_l\leftarrow a_{k+1}+1$ для $1\le l\le k$ и перейдем к шагу L4. [См. M. Mor and A. S. Fraenkel, *Discrete Math.* **48** (1984), 101–112. Слабо упорядоченные последовательности характеризуются тем свойством, что если появляется k и $k>0$, то появляется и $k-1$.]

106. Все слабо упорядоченные последовательности можно получить с помощью последовательности элементарных операций $a_i\leftrightarrow a_j$ и $a_i\leftarrow a_j$. (Пожалуй, можно было бы ограничить и эти преобразования, допуская только $a_j\leftrightarrow a_{j+1}$ или $a_j\leftarrow a_{j+1}$ для $1\le j<n$.)

107. Каждый шаг, как заметил Г. С. Вильф (H. S. Wilf), увеличивает величину $\sum_{k=1}^n 2^k\times[a_k=k]$, так что в конечном счете игра должна завершиться. К решению поставленной задачи есть как минимум три подхода: плохой, хороший и еще лучше.

Плохой состоит в том, чтобы разыграть все 13! перетасовок и записать самую длинную. Этот метод не может не дать верный ответ; проблема только в том, что $13!=6\,227\,020\,800$, а в среднем игра продолжается ≈ 8.728 шагов.

Хороший подход [A. Pepperdine, *Math. Gazette* **73** (1989), 131–133] состоит в розыгрыше в обратном порядке, начиная с конечной позиции $1*\ldots*$, где $*$ означает карту, повернутую рубашкой вверх; такая карта переворачивается только в том случае, когда ее значение становится существенным. Для выполнения хода назад из заданной позиции $a_1\ldots a_n$ рассмотрим все $k>1$, такие, что либо $a_k=k$, либо $a_k=*$ и k еще не перевернута. Таким образом, позициями, предшествующими рассматриваемой, могут быть позиции $21*\ldots*$, $3*1*\ldots*$, $\ldots$, $n*\ldots*1$. Некоторые из позиций (наподобие $6{**}213$ для $n=6$)

предшественников не имеют, несмотря на то что перевернуты еще не все карты. Легко систематически исследовать дерево потенциальных предшествующих игр и показать, что количество узлов с t звездочками равно $(n-1)!/t!$. Следовательно, общее количество рассматриваемых узлов составляет $\lfloor (n-1)!\,e \rfloor$. При $n = 13$ оно равно 1 302 061 345.

Лучший способ заключается в игре в прямом направлении, начиная с позиции $* \ldots *$, переворачивая верхнюю карту, если она лежит рубашкой вверх, проходя по всем $(n-1)!$ перестановкам множества $\{2, \ldots, n\}$ по мере переворачивания карт. Обозначим через $f(n)$ наибольшую возможную длину игры topswops с n картами. Если известно, что нижние $n-m$ карт представляют собой $(m+1)(m+2)\ldots n$ в указанном порядке, то возможно не более $f(m)$ дальнейших шагов; таким образом, нет необходимости продолжать игру, если понятно, что она не сможет быть достаточно длинной, чтобы представлять интерес. Генератор перестановок наподобие алгоритма X позволяет нам совместно использовать вычисления для всех перестановок с одинаковыми префиксами и отбрасывать префиксы, не представляющие интереса. Карта в позиции j не обязательно должна принимать при перевороте значение j. Когда $n = 13$, этот метод требует рассмотрения только $(1, 11, 940, 6\,960, 44\,745, 245\,083, 1\,118\,216, 4\,112\,676, 11\,798\,207, 26\,541\,611, 44\,380\,227, 37\,417\,359)$ ветвей соответственно на уровнях $(1, 2, \ldots, 12)$ и выполнения в сумме только 482 663 902 ходов. Хотя некоторые линии игры и повторяются, раннее обрезание заведомо проигрышных ветвей ускоряет вычисления примерно в 11 раз по сравнению с обратным методом при $n = 13$.

Единственный рекордсмен — 80-ходовая игра, начинающаяся с расположения карт 2 9 4 5 11 12 10 1 8 13 3 6 7.

108. Этот результат справедлив для любой игры, в которой

$$a_1 \ldots a_n \to a_k a_{p(k,2)} \ldots a_{p(k,k-1)} a_1 a_{k+1} \ldots a_n$$

для $a_1 = k$, где $p(k,2)\ldots p(k,k-1)$ представляет собой произвольную перестановку множества $\{2, \ldots, k-1\}$. Предположим, что a_1 в процессе игры принимает ровно m различных значений $d(1) < \cdots < d(m)$; докажем, что при этом происходит не более F_{m+1} перестановок, включая исходное перетасовывание карт. Это утверждение очевидно при $m = 1$.

Пусть $d(j)$ представляет собой начальное значение $a_{d(m)}$, где $j < m$, и предположим, что $a_{d(m)}$ изменяется на шаге r. Если $d(j) = 1$, количество перестановок $r + 1 \le F_m + 1 \le F_{m+1}$. В противном случае $r \le F_{m-1}$, и за шагом r следует не более F_m дальнейших перестановок. [*SIAM Review* **19** (1977), 739–741.]

Значениями $f(n)$ для $1 \le n \le 16$ являются (0, 1, 2, 4, 7, 10, 16, 22, 30, 38, 51, 65, 80, 101, 113, 139), и они достижимы соответственно (1, 1, 2, 2, 1, 5, 2, 1, 1, 1, 1, 1, 1, 4, 6, 1) способами. Единственная приводящая к максимально длинной игре исходная перестановка для $n = 16$ имеет вид

9 12 6 7 2 14 8 1 11 13 5 4 15 16 10 3.

109. Остроумное построение, приведенное в работе [I. H. Sudborough and L. Morales, *Theoretical Comp. Sci.* **411** (2010), 3965–3970], доказывает, что $f(n) \ge \frac{19}{128}n^2 + O(1)$.

110. Для $0 \le j \le 9$ построим битовые векторы $A_j = [a_j \in S_1]\ldots[a_j \in S_m]$ и $B_j = [j \in S_1]\ldots[j \in S_m]$. Тогда количество j, таких, что $A_j = v$, должно быть равно количеству k, таких, что $B_k = v$, для всех битовых векторов v. А в таком случае значения $\{a_j \mid A_j = v\}$ должны быть назначены перестановкам $\{k \mid B_k = v\}$ всеми возможными способами.

Например, в приведенной задаче битовыми векторами являются

$$(A_0, \ldots, A_9) = (\mathtt{9}, \mathtt{6}, \mathtt{8}, \mathtt{b}, \mathtt{5}, \mathtt{4}, \mathtt{0}, \mathtt{a}, \mathtt{2}, \mathtt{0}), \qquad (B_0, \ldots, B_9) = (\mathtt{5}, \mathtt{0}, \mathtt{8}, \mathtt{6}, \mathtt{2}, \mathtt{a}, \mathtt{4}, \mathtt{b}, \mathtt{9}, \mathtt{0}),$$

в шестнадцатеричной записи; следовательно, $a_0 \ldots a_9 = 8327061549$ или 8327069541.

В задаче большего размера битовые векторы можно хранить в хеш-таблице. Было бы лучше дать ответ в терминах классов эквавалентности, а не перестановок; и в самом деле, данная задача имеет мало общего с перестановками.

111. В ориентированном графе с $n!/2$ вершинами $a_1 \ldots a_{n-2}$ и $n!$ дугами $a_1 \ldots a_{n-2} \to a_2 \ldots a_{n-1}$ (по одной для каждой перестановки $a_1 \ldots a_n$), входящая и исходящая степень каждой вершины равна 2. Кроме того, из путей наподобие $a_1 \ldots a_{n-2} \to a_2 \ldots a_{n-1} \to a_3 \ldots a_n \to a_4 \ldots a_n a_2 \to a_5 \ldots a_n a_2 a_1 \to \cdots \to a_2 a_1 a_3 \ldots a_{n-2}$ можно видеть, что любая вершина достижима из любой другой. Следовательно, согласно теореме 2.3.4.2G имеется цепь Эйлера, и такая цепь, очевидно, эквивалентна универсальному циклу перестановок. Лексикографически наименьшим примером для $n = 4$ является (123124132134214324314234).

[Г. Хёльберт (G. Hurlbert) и Г. Исаак (G. Isaak) в *Discrete Math.* **149** (1996), 123–129, предложили другой привлекательный подход: назовем *модульным универсальным циклом* перестановок цикл из $n!$ цифр $\{0, \ldots, n\}$, обладающий тем свойством, что каждая перестановка $a_1 \ldots a_n$ множества $\{1, \ldots, n\}$ получается из последовательных цифр $u_1 \ldots u_n$, если положить $a_j = (u_j - c) \bmod (n+1)$, где c — "отсутствующая" в $\{u_1, \ldots, u_n\}$ цифра. Например, модульный универсальный цикл (012032), по сути, единственен при $n = 3$; а лексикографически наименьший цикл для $n = 4$ имеет вид (012301420132014321430243). Если вершины $a_1 \ldots a_{n-2}$ и $a'_1 \ldots a'_{n-2}$ в ориентированном графе из предыдущего абзаца рассматривать как эквивалентные при $a_1 - a'_1 \equiv \cdots \equiv a_{n-2} - a'_{n-2}$ (по модулю n), то мы получим ориентированный граф из $(n-1)!/2$ вершин, цепи Эйлера которого соответствуют модульным универсальным циклам перестановок для $\{1, \ldots, n-1\}$.]

112. (а) Если цикл представляет собой $a_1 a_2 \ldots$, то на шаге j используйте σ, если подпоследовательность $a_j a_{j+1} \ldots a_{j+n-1}$ является перестановкой; в противном случае используйте ρ.

(б) Это утверждение следует непосредственно из упр. 72.

(в) Пусть $\Omega_2 = \sigma^2$, и будем получать Ω_{n+1} /from Ω_n путем подстановки $\sigma \mapsto \sigma^2 \rho^{n-1}$ и $\rho \mapsto \sigma^2 \rho^{n-2} \sigma$. Например, $\Omega_3 = (\sigma^2 \rho)^2$ и $\Omega_4 = ((\sigma^2 \rho^2)^2 \sigma^2 \rho \sigma)^2$. Будем генерировать перестановки, начиная с $n \ldots 21$ и применяя последовательно элементы Ω_n; например, последовательность при $n = 4$ имеет вид

$$4321, 3214, 2143, 1423, 4213, 2134, 1342, 3412, 4132, 1324, 3241, 2431,$$
$$4312, 3124, 1243, 2413, 4123, 1234, 2341, 3421, 4231, 2314, 3142, 1432,$$

а соответствующий универсальный цикл — (432142134132431241234231). Заметим, что n в этой последовательности перестановок перемещается циклически; а перестановки, начинающиеся с n, соответствуют последовательности, полученной из Ω_{n-1}.

[См. F. Ruskey and A. Williams, *ACM Trans. on Algorithms* **6** (2010), 45:1–45:12. Аналогичные методы, как говорят, были известны колокольным звонарям. Универсальные циклы можно также строить явно для перестановок произвольного *мультимножества* с помощью метода, аналогичного 7.2.1.1–(62); см. A. Williams, Ph.D. thesis (Univ. of Victoria, 2009).]

113. Согласно упр. 2.3.4.2–22 достаточно подсчитать ориентированные деревья с корнями в $12 \ldots (n-2)$ в ориентированном графе из ответа к предыдущему упражнению; а подсчитать их можно на основе упр. 2.3.4.2–19. Для $n \le 6$ числа U_n оказываются соблазнительно простыми: $U_2 = 1$, $U_3 = 3$, $U_4 = 2^7 \cdot 3$, $U_5 = 2^{33} \cdot 3^8 \cdot 5^3$, $U_6 = 2^{190} \cdot 3^{49} \cdot 5^{33}$. (Здесь (121323) считается тем же циклом, что и (213231), но отличным от (131232).)

Марк Кук (Mark Cooke) открыл следующий поучительный способ эффективного вычисления этих значений. Рассмотрим $n! \times n!$-матрицу $M = 2I - R - S$, где $R_{\pi\pi'} = [\pi' = \pi\rho]$ и $S_{\pi\pi'} = [\pi' = \pi\sigma]$. Существует матрица H, такая, что и $H^- RH$, и $H^- SH$ имеют блок диагонального вида, состоящий из k_λ копий $k_\lambda \times k_\lambda$-матриц R_λ и S_λ, для каждого

разбиения λ числа n, где k_λ равно $n!$, деленному на произведение длин уголков формы λ (теорема 5.1.4H), и где R_λ и S_λ являются матричным представлением ρ и σ на основе диаграммы Юнга. [Доказательство можно найти в Bruce Sagan, *The Symmetric Group* (Pacific Grove, Calif.: Wadsworth & Brooks/Cole, 1991).] Например, когда $n = 3$, мы имеем

$$R = \begin{pmatrix} 0&0&0&1&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0\\ 1&0&0&0&0&0\\ 0&0&1&0&0&0\\ 0&1&0&0&0&0 \end{pmatrix}, \quad S = \begin{pmatrix} 0&1&0&0&0&0\\ 0&0&1&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1\\ 0&0&0&1&0&0 \end{pmatrix}, \quad H = \begin{pmatrix} 1&1&1&-1&1&0\\ 1&1&-1&0&0&-1\\ 1&1&0&1&-1&1\\ 1&-1&-1&1&0&1\\ 1&-1&1&0&1&-1\\ 1&-1&0&-1&-1&0 \end{pmatrix},$$

$$H^- RH = \begin{pmatrix} 1&0&0&0&0&0\\ 0&-1&0&0&0&0\\ 0&0&0&1&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&0&1\\ 0&0&0&0&1&0 \end{pmatrix}, \qquad H^- SH = \begin{pmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&-1&0&0\\ 0&0&1&-1&0&0\\ 0&0&0&0&0&-1\\ 0&0&0&0&1&-1 \end{pmatrix},$$

где строки и столбцы индексируются соответствующими перестановками 1, σ, σ^2, ρ, $\rho\sigma$, $\rho\sigma^2$; здесь $k_3 = k_{111} = 1$ и $k_{21} = 2$. Следовательно, собственные значения M представляют собой объединение по λ k_λ-кратных повторяющихся собственных значений $k_\lambda \times k_\lambda$-матриц $2I - R_\lambda - S_\lambda$. В этом примере собственные значения (0), (2) и двух $\left(\begin{smallmatrix} 2 & 0 \\ -2 & 3 \end{smallmatrix}\right)$ представляют собой $\{0\}$, $\{2\}$ и два $\{2, 3\}$.

Собственные значения M непосредственно связаны с собственными значениями матрицы A в упр. 2.3.4.2–19. Действительно, каждый собственный вектор A дает собственный вектор M, если приравнять компоненты для перестановок π и $\pi\rho\sigma^-$, поскольку строки π и $\pi\rho\sigma^-$ матрицы $R + S$ равны. Например,

$$A = \begin{pmatrix} 2&-1&-1\\ -1&2&-1\\ -1&-1&2 \end{pmatrix} \text{ имеет собственные векторы } \begin{pmatrix} 1\\1\\1 \end{pmatrix}, \begin{pmatrix} 1\\-1\\0 \end{pmatrix}, \begin{pmatrix} 1\\0\\-1 \end{pmatrix} \text{ для собственных значений } 0,\ 3,\ 3,$$

что дает собственные векторы $(1, 1, 1, 1, 1, 1)^T$, $(1, -1, 0, 0, -1, 1)^T$, $(1, 0, -1, -1, 0, 1)^T$ матрицы M для тех же собственных значений. Матрица M имеет $n!/2$ дополнительных собственных векторов со всеми нулевыми компонентами, за исключением тех, которые проиндексированы π и $\pi\sigma^-\rho$ для некоторого π, поскольку только строки $\pi\rho^-$ и $\pi\sigma^-$ матрицы $R + S$ имеют ненулевые элементы в столбцах π и $\pi\sigma^-\rho$; такие векторы дают $n!/2$ дополнительных собственных значений, равных 2.

Следовательно, значение U_n, которое представляет собой $2/n!$, умноженное на произведение ненулевых собственных значений A, равно $2^{1-n!/2}/n!$, умноженному на произведение ненулевых собственных значений M.

К сожалению, закономерность малых простых сомножителей не продолжается, и U_7 равно $2^{1217}3^{123}5^{119}7^5 11^{28}43^{35}73^{20}79^{21}109^{35}$, а U_9 кратно 59229013196333^{168}.

РАЗДЕЛ 7.2.1.3

1. Для данного мультисочетания сформируем последовательность $e_t \ldots e_2 e_1$ путем перечисления справа налево сначала несовпадающих элементов, затем — элементов, встречающихся дважды, после — трижды и т. д. Установим $e_{-j} \leftarrow s - j$ для $0 \le j \le s = n - t$, так что каждый элемент e_j для $1 \le j \le t$ равен некоторому элементу справа от него в последовательности $e_t \ldots e_1 e_0 \ldots e_{-s}$. Если первый такой элемент равен $e_{c_j - s}$, мы получаем решение (3). И наоборот, каждое решение (3) дает единственное мультимножество $\{e_1, \ldots, e_t\}$, потому что $c_j < s + j$ для $1 \le j \le t$.

[Подобное соответствие было предложено Э. Каталаном (E. Catalan): если $0 \le e_1 \le \cdots \le e_t \le s$, примем

$$\{c_1, \ldots, c_t\} \;=\; \{e_1, \ldots, e_t\} \cup \{s + j \mid 1 \le j < t \text{ и } e_j = e_{j+1}\}.$$

См. *Mémoires de la Soc. roy. des Sciences de Liège* (2) **12** (1885), *Mélanges Math.*, 3.]

2. Начнем с нижнего левого угла и для каждого нуля битовой строки будем идти вверх, а для каждой единицы — вправо. В результате получится путь см. рисунок справа. И наоборот, мы можем легко "прочесть" представления a_i, b_i, c_i, d_i, p_i и q_i из (2)–(11) в любом заданном пути от $(0, 0)$ до (s, t).

3. В этом алгоритме переменная r представляет собой наименьший положительный индекс, такой, что $q_r > 0$.

N1. [Инициализация.] Установить $q_j \leftarrow 0$ для $1 \le j \le t$ и $q_0 \leftarrow s$. (Считаем, что $st > 0$.)

N2. [Посещение.] Посетить композицию $q_t \ldots q_0$. Перейти к шагу N4, если $q_0 = 0$.

N3. [Простой случай.] Установить $q_0 \leftarrow q_0 - 1$, $r \leftarrow 1$ и перейти к шагу N5.

N4. [Сложный случай.] Завершить работу алгоритма, если $r = t$. В противном случае установить $q_0 \leftarrow q_r - 1$, $q_r \leftarrow 0$, $r \leftarrow r + 1$.

N5. [Увеличение q_r.] Установить $q_r \leftarrow q_r + 1$ и вернуться к шагу N2. ▮

[См. *CACM* **11** (1968), 430; **12** (1969), 187. Задача генерации таких композиций в *уменьшающемся* лексикографическом порядке существенно сложнее.]

4. Можно обратить роли 0 и 1 в (14), так что $0^{q_t}10^{q_{t-1}}1 \ldots 10^{q_1}10^{q_0} = 1^{r_s}01^{r_{s-1}}0 \ldots 01^{r_1}01^{r_0}$. Это дает нам

$$0^1 10^0 10^2 10^2 10^4 10^0 10^0 10^0 10^0 10^0 10^1 10^0 10^1 10^0 = 1^0 01^2 01^0 01^1 01^0 01^1 01^0 01^0 01^0 01^6 01^2 01^1.$$

Лексикографический порядок $a_{n-1} \ldots a_1 a_0$ соответствует лексикографическому порядку $r_s \ldots r_1 r_0$.

Кстати, имеется также связь мультисочетаний: $\{d_t, \ldots, d_1\} = \{r_s \cdot s, \ldots, r_0 \cdot 0\}$. Например, $\{10, 10, 8, 6, 2, 2, 2, 2, 2, 2, 1, 1, 0\} = \{0 \cdot 11, 2 \cdot 10, 0 \cdot 9, 1 \cdot 8, 0 \cdot 7, 1 \cdot 6, 0 \cdot 5, 0 \cdot 4, 0 \cdot 3, 6 \cdot 2, 2 \cdot 1, 1 \cdot 0\}$.

5. (а) Установить $x_j = c_j - \lfloor (j-1)/2 \rfloor$ в каждом t-сочетании из $n + \lfloor t/2 \rfloor$. (б) Установить $x_j = c_j + j + 1$ в каждом t-сочетании из $n - t - 2$.

(Аналогичный подход позволяет найти все решения $(x_t, \ldots, x_1)$ неравенств $x_{j+1} \ge x_j + \delta_j$ для $0 \le j \le t$ для заданных значений x_{t+1}, $(\delta_t, \ldots, \delta_0)$ и x_0.)

6. Считаем, что $t > 0$. Мы добираемся до шага T3, когда $c_1 > 0$; до шага T5, когда $c_2 = c_1 + 1 > 1$; до шага T4 для тех $2 \le j \le t + 1$, для которых $c_j = c_1 + j - 1 \ge j$. Таким образом, соответствующие величины равны: T1, 1; T2, $\binom{n}{t}$; T3, $\binom{n-1}{t}$; T4, $\binom{n-2}{t-1} + \binom{n-3}{t-2} + \cdots + \binom{n-t-1}{0} = \binom{n-1}{t-1}$; T5, $\binom{n-2}{t-1}$; T6, $\binom{n-1}{t-1} + \binom{n-2}{t-1} - 1$.

7. Процедура немного проще, чем требуется для алгоритма T. Примем, что $s < n$.

S1. [Инициализация.] Установить $b_j \leftarrow j + n - s - 1$ для $1 \le j \le s$; затем установить $j \leftarrow 1$.

S2. [Посещение.] Посетить сочетание $b_s \ldots b_2 b_1$. Завершить работу алгоритма, если $j > s$.

S3. [Уменьшение b_j.] Установить $b_j \leftarrow b_j - 1$. Если $b_j < j$, установить $j \leftarrow j + 1$ и вернуться к шагу S2.

S4. [Сброс $b_{j-1} \ldots b_1$.] Если $j > 1$, установить $b_{j-1} \leftarrow b_j - 1$ и $j \leftarrow j - 1$ и повторять эти действия до тех пор, пока выполняется условие $j > 1$. Перейти к шагу S2. ▮

(См. S. Dvořák, *Comp. J.* **33** (1990), 188. Заметим, что если $x_k = n - b_k$ при $1 \le k \le s$, то этот алгоритм проходит по всем сочетаниям $x_s \ldots x_2 x_1$ из $\{1, 2, \ldots, n\}$ с $1 \le x_s < \cdots < x_2 < x_1 \le n$ в *возрастающем* лексикографическом порядке.)

8. **A1.** [Инициализация.] Установить $a_n \ldots a_0 \leftarrow 0^{s+1}1^t$, $q \leftarrow t$, $r \leftarrow 0$. (Считаем, что $0 < t < n$.)

A2. [Посещение.] Посетить сочетание $a_{n-1} \ldots a_1 a_0$. Перейти к шагу A4, если $q = 0$.

A3. [Замена $\ldots 01^q$ на $\ldots 101^{q-1}$.] Установить $a_q \leftarrow 1$, $a_{q-1} \leftarrow 0$, $q \leftarrow q-1$; затем, если $q = 0$, установить $r \leftarrow 1$. Вернуться к шагу A2.

A4. [Сдвиг блока единиц.] Установить $a_r \leftarrow 0$ и $r \leftarrow r + 1$. Затем, если $a_r = 1$, установить $a_q \leftarrow 1$, $q \leftarrow q + 1$, и повторить шаг A4.

A5. [Перенос влево.] Завершить работу алгоритма, если $r = n$; в противном случае установить $a_r \leftarrow 1$.

A6. [Нечетно?] Если $q > 0$, установить $r \leftarrow 0$. Вернуться к шагу A2. ▮

На шаге A2 на крайние справа 0 и 1 в $a_{n-1} \ldots a_0$ указывают соответственно q и r. Шаги A1, …, A6 выполняются с частотой 1, $\binom{n}{t}$, $\binom{n-1}{t-1}$, $\binom{n}{t} - 1$, $\binom{n-1}{t}$, $\binom{n-1}{t} - 1$.

9. (а) Первые $\binom{n-1}{t}$ строк начинаются с 0 и имеют $2A_{(s-1)t}$ изменений битов; другие $\binom{n-1}{t-1}$ строк начинаются с 1 и имеют $2A_{s(t-1)}$ изменений битов. А $\nu(01^t0^{s-1} \oplus 10^s1^{t-1}) = 2\min(s,t)$.

(б) Решение 1 (прямое). Пусть $B_{st} = A_{st} + \min(s,t) + 1$. Тогда

$$B_{st} = B_{(s-1)t} + B_{s(t-1)} + [s = t] \quad \text{при } st > 0; \qquad\qquad B_{st} = 1 \quad \text{при } st = 0.$$

Следовательно, $B_{st} = \sum_{k=0}^{\min(s,t)} \binom{s+t-2k}{s-k}$. Если $s \le t$, то эта величина $\le \sum_{k=0}^{s} \binom{s+t-k}{s-k} = \binom{s+t+1}{s} = \binom{s+t}{s}\frac{s+t+1}{t+1} < 2\binom{s+t}{t}$.

Решение 2 (непрямое). Алгоритм в ответе к упр. 8 выполняет $2(x + y)$ изменений битов, если шаги (A3, A4) выполняются (x, y) раз. Таким образом, $A_{st} \le \binom{n-1}{t-1} + \binom{n}{t} - 1 < 2\binom{n}{t}$.

[Следовательно, комментарий в ответе 7.2.1.1–3 применим и к сочетаниям.]

10. Каждый сценарий соответствует $(4,4)$-сочетанию $b_4b_3b_2b_1$ или $c_4c_3c_2c_1$, в котором A побеждает в играх $\{8 - b_4, 8 - b_3, 8 - b_2, 8 - b_1\}$, а N побеждает в играх $\{8 - c_4, 8 - c_3, 8 - c_2, 8 - c_1\}$, поскольку можно считать, что проигравшая команда побеждает в оставшихся сериях из 8 игр. (Или, что то же самое, можно сгенерировать все перестановки $\{\mathrm{A}, \mathrm{A}, \mathrm{A}, \mathrm{A}, \mathrm{N}, \mathrm{N}, \mathrm{N}, \mathrm{N}\}$ и опустить завершающие серии A и N.) Американская лига побеждает тогда и только тогда, когда $b_1 \ne 0$, что, в свою очередь, выполняется тогда и только тогда, когда $c_1 = 0$. Формула $\binom{c_4}{4} + \binom{c_3}{3} + \binom{c_2}{2} + \binom{c_1}{1}$ позволяет назначить каждому сценарию уникальный номер от 0 до 69.

Например, ANANAA $\iff a_7\ldots a_1a_0 = 01010011 \iff b_4b_3b_2b_1 = 7532 \iff c_4c_3c_2c_1 = 6410$, и этот сценарий имеет ранг $\binom{6}{4}+\binom{4}{3}+\binom{1}{2}+\binom{0}{1} = 19$ в лексикографическом порядке. (Член $\binom{c_j}{j}$ равен нулю тогда и только тогда, когда он соответствует завершающему N.)

11. Наиболее часто встречались сценарии AAAA (9 раз), NNNN (8 раз) и ANAAA (7 раз). Из 70 возможных серий ни разу не встречались 27, включая все четыре варианта, начинающиеся с NNNA. (Игры 1907, 1912 и 1922 годов игнорируются из-за наличия ничьих. Сценарий ANNAAA также следует исключить, так как он встретился только в 1920 году как часть последовательности ANNAAAA. Сценарий NNAAANN впервые осуществился в 2001 году.)

12. (а) Пусть V_j — подпространство $\{a_{n-1}\ldots a_0 \in V \mid a_k = 0$ для $0 \le k < j\}$, так что $\{0\ldots0\} = V_n \subseteq V_{n-1} \subseteq \cdots \subseteq V_0 = V$. Тогда $\{c_1,\ldots,c_t\} = \{c \mid V_c \ne V_{c+1}\}$, и α_k — единственный элемент $a_{n-1}\ldots a_0$ из V с $a_{c_j} = [j=k]$ для $1 \le j \le t$.

Кстати, матрица размером $t\times n$, соответствующая каноническому базису, называется *приведенной к построчно-эшелонному виду*. Ее можно получить с помощью стандартного алгоритма "триангуляции" (см. упр. 4.6.1–19 и алгоритм 4.6.2N).

(б) Необходимыми свойствами обладает 2-номиальный коэффициент $\binom{n}{t}_2 = 2^t\binom{n-1}{t}_2 + \binom{n-1}{t-1}_2$ из упр. 1.2.6–58, так как $2^t\binom{n-1}{t}_2$ бинарных векторных пространств имеют $c_t < n-1$, а $\binom{n-1}{t-1}_2$ имеют $c_t = n-1$. [Вообще говоря, количество канонических базисов с r звездочками равно количеству разбиений r не более чем на t частей, причем никакая часть не превосходит $n-t$, и это количество равно $[z^r]\binom{n}{t}_z$ согласно формуле 7.2.1.4–(51). См. D. E. Knuth, *J. Combinatorial Theory* **A10** (1971), 178–180.]

(в) В приведенном далее алгоритме предполагается, что $n > t > 0$ и что $a_{(t+1)j} = 0$ для $t \le j \le n$.

V1. [Инициализация.] Установить $a_{kj} \leftarrow [j=k-1]$ для $1 \le k \le t$ и $0 \le j < n$. Установить также $q \leftarrow t$, $r \leftarrow 0$.

V2. [Посещение.] (В этот момент мы имеем $a_{k(k-1)} = 1$ для $1 \le k \le q$, $a_{(q+1)q} = 0$ и $a_{1r} = 1$.) Посетить канонический базис $(a_{1(n-1)}\ldots a_{11}a_{10},\ldots,a_{t(n-1)}\ldots a_{t1}a_{t0})$. Если $q > 0$, перейти к шагу V4.

V3. [Поиск блока единиц.] Устанавливать $q \leftarrow 1, 2, \ldots$, пока не выполнится условие $a_{(q+1)(q+r)} = 0$. Если $q+r = n$, завершить работу алгоритма.

V4. [Добавление 1 к столбцу $q+r$.] Установить $k \leftarrow 1$. До тех пор, пока справедливо условие $a_{k(q+r)} = 1$, устанавливать $a_{k(q+r)} \leftarrow 0$ и $k \leftarrow k+1$. Затем, если $k \le q$, установить $a_{k(q+r)} \leftarrow 1$; в противном случае установить $a_{q(q+r)} \leftarrow 1$, $a_{q(q+r-1)} \leftarrow 0$, $q \leftarrow q-1$.

V5. [Сдвиг блока вправо.] Если $q = 0$, установить $r \leftarrow r+1$. В противном случае, если $r > 0$, установить $a_{k(k-1)} \leftarrow 1$ и $a_{k(r+k-1)} \leftarrow 0$ для $1 \le k \le q$, затем установить $r \leftarrow 0$. Перейти к шагу V2. ▮

Шаг V2 находит $q > 0$ с вероятностью $1-(2^{n-t}-1)/(2^n-1) \approx 1-2^{-t}$, так что можно сэкономить время, рассматривая этот случай отдельно.

(г) Поскольку $999999 = 4\binom{8}{4}_2 + 16\binom{7}{4}_2 + 5\binom{6}{3}_2 + 5\binom{5}{3}_2 + 8\binom{4}{3}_2 + 0\binom{3}{2}_2 + 4\binom{2}{2}_2 + 1\binom{1}{1}_2 + 2\binom{0}{1}_2$, миллионный вывод дает бинарные столбцы 4, 16/2, 5, 5, 8/2, 0, 4/2, 1, 2/2, а именно

$$\begin{aligned}
\alpha_1 &= 0\,0\,1\,1\,0\,0\,0\,1\,1,\\
\alpha_2 &= 0\,0\,0\,0\,0\,0\,1\,0\,0,\\
\alpha_3 &= 1\,0\,1\,1\,1\,0\,0\,0\,0,\\
\alpha_4 &= 0\,1\,0\,0\,0\,0\,0\,0\,0.
\end{aligned}$$

[*Источник*: E. Calabi and H. S. Wilf, *J. Combinatorial Theory* **A22** (1977), 107–109.]

13. Пусть $n = s + t$. Имеется $\binom{s-1}{\lceil (r-1)/2 \rceil}\binom{t-1}{\lfloor (r-1)/2 \rfloor}$ конфигураций, начинающихся с 0, и $\binom{s-1}{\lfloor (r-1)/2 \rfloor}\binom{t-1}{\lceil (r-1)/2 \rceil}$ конфигураций, начинающихся с 1, поскольку конфигурация Изинга, начинающаяся с 0, соответствует композиции s нулей на $\lceil (r+1)/2 \rceil$ частей и композиции t единиц на $\lfloor (r+1)/2 \rfloor$ частей. Все такие пары композиций могут быть сгенерированы и объединены в конфигурации. [См. E. Ising, *Zeitschrift für Physik* **31** (1925), 253–258; J. M. S. Simões Pereira, *CACM* **12** (1969), 562.]

14. Начнем с $l[j] \leftarrow j - 1$ и $r[j-1] \leftarrow j$ для $1 \le j \le n$; $l[0] \leftarrow n$, $r[n] \leftarrow 0$. Чтобы получить следующее сочетание в предположении, что $t > 0$, установим $p \leftarrow s$, если $l[0] > s$, в противном случае $p \leftarrow r[n] - 1$. Если $p \le 0$, завершаем работу алгоритма; в противном случае устанавливаем $q \leftarrow r[p]$, $l[q] \leftarrow l[p]$ и $r[l[p]] \leftarrow q$. Затем, если $r[q] > s$ и $p < s$, устанавливаем $r[p] \leftarrow r[n]$, $l[r[n]] \leftarrow p$, $r[s] \leftarrow r[q]$, $l[r[q]] \leftarrow s$, $r[n] \leftarrow 0$, $l[0] \leftarrow n$; в противном случае устанавливаем $r[p] \leftarrow r[q]$, $l[r[q]] \leftarrow p$. И наконец устанавливаем $r[q] \leftarrow p$ и $l[p] \leftarrow q$.

[См. Korsh and Lipschutz, *J. Algorithms* **25** (1997), 321–335, где эта идея распространена на алгоритм без циклов для перестановок мультимножеств. *Предупреждение*: это упражнение, как и упр. 7.2.1.1–16, носит в большей степени академический, чем практический характер, поскольку подпрограмма для обхода связанного списка может потребовать применения цикла, что сведет на нет все преимущества генерации без использования цикла.]

15. (Указанный факт верен, поскольку лексикографический порядок $c_t \ldots c_1$ соответствует лексикографическому порядку $a_{n-1} \ldots a_0$, который представляет собой обратный лексикографический порядок комплементарной последовательности $1 \ldots 1 \oplus a_{n-1} \ldots a_0$.) Согласно теореме L сочетание $c_t \ldots c_1$ посещается в точности *перед* посещением $\binom{b_s}{s} + \cdots + \binom{b_2}{2} + \binom{b_1}{1}$ других сочетаний, так что должно выполняться

$$\binom{b_s}{s} + \cdots + \binom{b_1}{1} + \binom{c_t}{t} + \cdots + \binom{c_1}{1} = \binom{s+t}{t} - 1.$$

Это общее тождество может быть записано как

$$\sum_{j=0}^{n-1} x_j \binom{j}{x_0 + \cdots + x_j} + \sum_{j=0}^{n-1} \bar{x}_j \binom{j}{\bar{x}_0 + \cdots + \bar{x}_j} = \binom{n}{x_0 + \cdots + x_{n-1}} - 1 ,$$

где каждое x_j представляет собой 0 или 1 и $\bar{x}_j = 1 - x_j$; это также следует из уравнения

$$x_n \binom{n}{x_0 + \cdots + x_n} + \bar{x}_n \binom{n}{\bar{x}_0 + \cdots + \bar{x}_n} = \binom{n+1}{x_0 + \cdots + x_n} - \binom{n}{x_0 + \cdots + x_{n-1}}.$$

16. Поскольку $999999 = \binom{1414}{2} + \binom{1008}{1} = \binom{182}{3} + \binom{153}{2} + \binom{111}{1} = \binom{71}{4} + \binom{56}{3} + \binom{36}{2} + \binom{14}{1} = \binom{43}{5} + \binom{32}{4} + \binom{21}{3} + \binom{15}{2} + \binom{6}{1}$, ответы к упражнению следующие: (а) 1414 1008; (б) 182 153 111; (в) 71 56 36 14; (г) 43 32 21 15 6; (д) 1000000 999999 ... 2 0.

17. По теореме L n_t — наибольшее целое, такое, что $N \ge \binom{n_t}{t}$; остальные члены образуют представление числа $N - \binom{n_t}{t}$ степени $(t-1)$.

Простой последовательный метод для $t > 1$ начинается с $x = 1$, $c = t$ и устанавливает $c \leftarrow c + 1$, $x \leftarrow xc/(c-t)$ нуль или несколько раз, пока не выполнится условие $x > N$; после этого первая фаза завершается установкой $x \leftarrow x(c-t)/c$, $c \leftarrow c - 1$, и мы получаем $x = \binom{c}{t} \le N < \binom{c+1}{t}$. Установим $n_t \leftarrow c$, $N \leftarrow N - x$. Алгоритм завершает работу установкой $n_1 \leftarrow N$ при $t = 2$; в противном случае установим $x \leftarrow xt/c$, $t \leftarrow t - 1$, $c \leftarrow c - 1$. Если $x > N$, устанавливаем $x \leftarrow x(c-t)/c$, $c \leftarrow c - 1$ и повторяем это действие, пока $x > N$. Этот метод требует $O(n)$ арифметических операций при $N < \binom{n}{t}$, так что он вполне подходит для решения поставленной задачи, если только t не слишком мало́, а N — не слишком велико.

При $t = 2$ упр. 1.2.4–41 говорит нам, что $n_2 = \lfloor\sqrt{2N+2}+\frac{1}{2}\rfloor$. В общем случае n_t равно $\lfloor x\rfloor$, где x — наибольший корень уравнения $x^{\underline{t}} = t!\,N$. Этот корень можно приближенно найти, обратив ряд $y = (x^{\underline{t}})^{1/t} = x - \frac{1}{2}(t-1) + \frac{1}{24}(t^2-1)x^{-1} + \cdots$, что дает нам $x = y + \frac{1}{2}(t-1) + \frac{1}{24}(t^2-1)/y + O(y^{-3})$. Если в этой формуле принять $y = (t!\,N)^{1/t}$, то получается хорошее приближение, после которого можно убедиться, что $\binom{\lfloor x\rfloor}{t} \le N < \binom{\lfloor x\rfloor+1}{t}$, или выполнить заключительное уточнение. [См. A. S. Fraenkel and M. Mor, *Comp. J.* **26** (1983), 336–343.]

18. Получится полное бинарное дерево с $2^n - 1$ узлами, с дополнительным узлом наверху наподобие "дерева проигравших" в сортировке выбором с замещением (рис. 63 в разделе 5.4.1). Таким образом, явные связи не являются необходимыми; правый дочерний узел узла k — это узел $2k+1$, а левый "братский" узел — узел $2k$ для $1 \le k < 2^{n-1}$.

Такое представление биномиального дерева имеет любопытное свойство, заключающееся в том, что узел $k = (0^a 1\alpha)_2$ соответствует сочетанию, бинарная строка которого $0^a 1\alpha^R$.

19. Это 11110100001001000100, бинарное представление $\mathrm{post}(1000000)$, где $\mathrm{post}(2^{k+1}-1) = 2^k$, а $\mathrm{post}(n) = 2^k + \mathrm{post}(n-2^k+1)$ при $2^k \le n < 2^{k+1}-1$, где $k \ge 0$. [Кстати, представление T_∞ с указателями на "левого ребенка" и "правого брата" представляет собой перевернутую пирамиду.]

20. $f(z) = (1+z^{w_{n-1}})\ldots(1+z^{w_1})/(1-z)$, $g(z) = (1+z^{w_0})f(z)$, $h(z) = z^{w_0}f(z)$.

21. Ранг $c_t\ldots c_2c_1$ равен $\binom{c_t+1}{t} - 1$ минус ранг $c_{t-1}\ldots c_2c_1$. [С. 40 диссертации Миллер; см. также H. Lüneburg, *Abh. Math. Sem. Hamburg* **52** (1982), 208–227.]

22. Поскольку $999999 = \binom{1415}{2} - \binom{406}{1} = \binom{183}{3} - \binom{98}{2} + \binom{21}{1} = \binom{72}{4} - \binom{57}{3} + \binom{32}{2} - \binom{27}{1} = \binom{44}{5} - \binom{40}{4} + \binom{33}{3} - \binom{13}{2} + \binom{3}{1}$, ответы к упражнению следующие: (а) 1414 405; (б) 182 97 21; (в) 71 56 31 26; (г) 43 39 32 12 3; (д) 1000000 999999 999998 999996 ... 0.

23. Имеется $\binom{n-r}{t-r}$ сочетаний с $j > r$ для $r = 1, 2, \ldots, t$. (Если $r = 1$, то $c_2 = c_1+1$; если $r = 2$, то $c_1 = 0$, $c_2 = 1$; если $r = 3$, то $c_1 = 0$, $c_2 = 1$, $c_4 = c_3+1$; и т. д.) Таким образом, среднее значение равно $\left(\binom{n}{t} + \binom{n-1}{t-1} + \cdots + \binom{n-t}{0}\right)\Big/\binom{n}{t} = \binom{n+1}{t}\Big/\binom{n}{t} = (n+1)/(n+1-t)$. Среднее время работы одного шага приблизительно пропорционально этой величине; таким образом, алгоритм достаточно быстр при малых t, но становится медленным при t, близком к n.

24. В действительности $j_k - 2 \le j_{k+1} \le j_k + 1$, когда $j_k \equiv t$ (по модулю 2), и $j_k - 1 \le j_{k+1} \le j_k + 2$, когда $j_k \not\equiv t$, поскольку R5 выполняется только тогда, когда $c_i = i-1$ для $1 \le i < j$.

Таким образом, в конце R2 можно добавить "если $j \ge 4$, установить $j \leftarrow j-1-[j$ нечетно$]$ и перейти к шагу R5", если t нечетно; и "если $j \ge 3$, установить $j \leftarrow j-1-[j$ четно$]$ и перейти к шагу R5", если t четно. В этом случае алгоритм не будет содержать циклов, поскольку R4 и R5 будут выполняться не более двух раз на одно посещение.

25. Положим, что $N > N'$, и $N - N'$ минимально; кроме того, пусть t и c_t минимальны при этих же предположениях. Тогда $c_t > c'_t$.

Если существует элемент $x \notin C \cup C'$, такой, что $0 \le x < c_t$, отобразим каждое t-сочетание $C \cup C'$ путем замены $j \mapsto j-1$ для $j > x$; или, если имеется элемент $x \in C \cap C'$, отобразим каждое t-сочетание, которое содержит x, на $(t-1)$-сочетание, опуская x и заменяя $j \mapsto x-j$ для $j < x$. В любом случае отображение сохраняет чередующийся лексикографический порядок; следовательно, величина $N - N'$ должна превосходить количество сочетаний между образами C и C'. Но c_t минимально, так что такое x не существует. Следовательно, $t = m$ и $c_t = 2m-1$.

Теперь, если $c'_m < c_m - 1$, мы можем уменьшить $N - N'$ путем увеличения c'_m. Поэтому $c'_m = 2m-2$, и задача сводится к поиску *максимума* $\mathrm{rank}(c_{m-1}\ldots c_1) - \mathrm{rank}(c'_{m-1}\ldots c'_1)$, где ранг вычисляется так, как в (30).

Пусть $f(s,t) = \max(\mathrm{rank}(b_s \ldots b_1) - \mathrm{rank}(c_t \ldots c_1))$ по всем $\{b_s, \ldots, b_1, c_t, \ldots, c_1\} = \{0, \ldots, s+t-1\}$. Тогда $f(s,t)$ удовлетворяет любопытному рекуррентному соотношению

$$f(s,0) = f(0,t) = 0; \qquad f(1,t) = t;$$
$$f(s,t) = \binom{s+t-1}{s} + \max\bigl(f(t-1,s-1), f(s-2,t)\bigr) \quad \text{если } st > 0 \text{ и } s > 1.$$

Когда $s+t = 2u+2$, решение принимает вид

$$f(s,t) = \binom{2u+1}{t-1} + \sum_{j=1}^{u-r}\binom{2u+1-2j}{r} + \sum_{j=0}^{r-1}\binom{2j+1}{j}, \qquad r = \min(s-2, t-1),$$

с максимумом в $f(t-1,s-1)$ при $s \le t$ и в $f(s-2,t)$ при $s \ge t+2$.

Следовательно, минимум $N - N'$ достигается при

$$C = \{2m-1\} \cup \{2m-2-x \mid 1 \le x \le 2m-2, \quad x \bmod 4 \le 1\},$$
$$C' = \{2m-2\} \cup \{2m-2-x \mid 1 \le x \le 2m-2, \quad x \bmod 4 \ge 2\}$$

и равен $\binom{2m-1}{m-1} - \sum_{k=0}^{m-2}\binom{2k+1}{k} = 1 + \sum_{k=1}^{m-1}\binom{2k}{k-1}$. [См. A. J. van Zanten, *IEEE Trans.* **IT-37** (1991), 1229–1233.]

26. (а) Да: первый элемент — $0^{n-\lceil t/2\rceil}1^{t \bmod 2}2^{\lfloor t/2\rfloor}$, последний — $2^{\lfloor t/2\rfloor}1^{t \bmod 2}0^{n-\lceil t/2\rceil}$; переходы представляют собой подстроки вида $02^a1 \leftrightarrow 12^a0$, $02^a2 \leftrightarrow 12^a1$, $10^a1 \leftrightarrow 20^a0$, $10^a2 \leftrightarrow 20^a1$.

(б) Нет: если $s = 0$, имеется большой переход от 02^t0^{r-1} к 20^r2^{t-1}.

27. Описанная далее процедура выбирает все сочетания $c_1 \ldots c_k$ из Γ_n, имеющие вес $\le t$. Начнем с $k \leftarrow 0$ и $c_0 \leftarrow n$. Посетим $c_1 \ldots c_k$. Если k четно и $c_k = 0$, установим $k \leftarrow k-1$; если k четно и $c_k > 0$, при $k = t$ установим $c_k \leftarrow c_k - 1$, в противном случае установим $k \leftarrow k+1$ и $c_k \leftarrow 0$. С другой стороны, если k нечетно и $c_k + 1 = c_{k-1}$, установим $k \leftarrow k-1$ и $c_k \leftarrow c_{k+1}$ (но если $k = 0$, завершим работу); если k нечетно и $c_k + 1 < c_{k-1}$, при $k = t$ установим $c_k \leftarrow c_k + 1$, в противном случае установим $k \leftarrow k+1$, $c_k \leftarrow c_{k-1}$, $c_{k-1} \leftarrow c_k + 1$. Повторим описанные действия.

(Этот алгоритм без циклов приводится к алгоритму из упр. 7.2.1.1–12, (б) при $t = n$ с небольшими изменениями в обозначениях.)

28. Истинно. Битовые строки $a_{n-1} \ldots a_0 = \alpha\beta$ и $a'_{n-1} \ldots a'_0 = \alpha\beta'$ соответствуют индексным последовательностям ($b_s \ldots b_1 = \theta\chi$, $c_t \ldots c_1 = \phi\psi$) и ($b'_s \ldots b'_1 = \theta\chi'$, $c'_t \ldots c'_1 = \phi\psi'$), таким, что все строки между $\alpha\beta$ и $\alpha\beta'$ начинаются с α тогда и только тогда, когда все элементы между $\theta\chi$ и $\theta\chi'$ начинаются с θ, а все элементы между $\phi\psi$ и $\phi\psi'$ начинаются с ϕ. Например, если $n = 10$, префикс $\alpha = 01101$ соответствует префиксам $\theta = 96$ и $\phi = 875$.

(Условие, что в облексном порядке находится только $c_t \ldots c_1$, гораздо более слабое. Например, при $t = 1$ *каждая* такая последовательность находится в облексном порядке.)

29. (а) $\texttt{-}^k\texttt{0}^{l+1}$ или $\texttt{-}^k\texttt{0}^{l+1}\texttt{+}\pm^m$, или $\pm^k$ для $k, l, m \ge 0$.

(б) Нет; преемник в уравновешенной троичной системе счисления всегда меньше.

(в) Для всех α и всех $k, l, m \ge 0$ имеем $\alpha\texttt{0-}^{k+1}\texttt{0}^l\texttt{+}\pm^m \to \alpha\texttt{-+}^k\texttt{0}^{l+1}\texttt{-}\pm^m$ и $\alpha\texttt{+-}^k\texttt{0}^{l+1}\texttt{+}\pm^m \to \alpha\texttt{0+}^{k+1}\texttt{0}^l\texttt{-}\pm^m$; также $\alpha\texttt{0-}^{k+1}\texttt{0}^l \to \alpha\texttt{-+}^k\texttt{0}^{l+1}$ и $\alpha\texttt{+-}^k\texttt{0}^{l+1} \to \alpha\texttt{0+}^{k+1}\texttt{0}^l$.

(г) Пусть j-й знак α_i — $(-1)^{a_{ij}}$ и пусть он находится в позиции b_{ij}. Тогда, если положить $b_{i0} = 0$, $(-1)^{a_{ij}+b_{i(j-1)}} = (-1)^{a_{(i+1)j}+b_{(i+1)(j-1)}}$ для $0 \le i < k$ и $1 \le j \le s$.

(д) Согласно первым трем ответам α принадлежит некоторой цепочке $\alpha_0 \to \cdots \to \alpha_k$, где α_k — конечный элемент (не имеющий последующего), а α_0 — первый элемент (не имеющий предшествующего). Согласно п. (г), каждая такая цепочка имеет не более $\binom{s+t}{t}$ элементов. Но согласно п. (а) всего имеется 2^s конечных строк и $2^s\binom{s+t}{t}$ строк с s знаками и t нулями; так что k должно быть равно $\binom{s+t}{t} - 1$.

Источник: *SICOMP* **2** (1973), 128–133.

30. Предположим, что $t > 0$. Начальные строки являются отрицательными конечными строками. Пусть σ_j — начальная строка $\mathtt{0}^t\mathtt{-}\tau_j$ для $0 \le j < 2^{s-1}$, причем k-й символ τ_j для $1 \le k < s$ представляет собой знак $(-1)^{a_k}$, где j — двоичное число $(a_{s-1}\ldots a_1)_2$; таким образом, $\sigma_0 = \mathtt{0}^t\mathtt{-++\ldots+}$, $\sigma_1 = \mathtt{0}^t\mathtt{--+\ldots+}$, $\ldots$, $\sigma_{2^{s-1}-1} = \mathtt{0}^t\mathtt{---\ldots-}$. Пусть ρ_j — конечная строка, полученная путем вставки $\mathtt{-0}^t$ после первой (возможно, пустой) серии минусов в τ_j; таким образом, $\rho_0 = \mathtt{-0}^t\mathtt{++\ldots+}$, $\rho_1 = \mathtt{--0}^t\mathtt{+\ldots+}$, $\ldots$, $\rho_{2^{s-1}-1} = \mathtt{--\ldots-0}^t$. Пусть также $\sigma_{2^{s-1}} = \sigma_0$ и $\rho_{2^{s-1}} = \rho_0$. Тогда по индукции можно доказать, что цепочка, начинающаяся с σ_j, заканчивается ρ_j, если t четно, и ρ_{j-1}, если t нечетно, для $1 \le j \le 2^{s-1}$. Следовательно, цепочка, начинающаяся с $-\rho_j$, заканчивается $-\sigma_j$ или $-\sigma_{j+1}$.

Пусть $A_j(s,t)$ — последовательность (s,t)-сочетаний, порожденная отображением цепочки, начинающейся с σ_j, и пусть $B_j(s,t)$ — аналогичная последовательность, порожденная из $-\rho_j$. Тогда для $1 \le j \le 2^{s-1}$ обратная последовательность $A_j(s,t)^R$ представляет собой последовательность $B_j(s,t)$ при четном t и $B_{j-1}(s,t)$ при нечетном t. Соответствующие рекуррентные соотношения при $st > 0$ имеют вид

$$A_j(s,t) = \begin{cases} 1A_j(s,t-1),\ 0A_{\lfloor(2^{s-1}-1-j)/2\rfloor}(s-1,t)^R, & \text{если } j+t \text{ четно;} \\ 1A_j(s,t-1),\ 0A_{\lfloor j/2\rfloor}(s-1,t), & \text{если } j+t \text{ нечетно;} \end{cases}$$

и, когда $st > 0$, все 2^{s-1} этих последовательностей различны.

Последовательность Чейза C_{st} представляет собой $A_{\lfloor 2^s/3\rfloor}(s,t)$, а $\widehat{C}_{st}$ — $A_{\lfloor 2^{s-1}/3\rfloor}(s,t)^R$. Следует заметить, что однородная последовательность K_{st} из (31) представляет собой $A_{2^{s-1}-[t\text{ четно}]}(s,t)^R$.

31. (а) $2^{\binom{s+t}{t}-1}$ является решением рекуррентного соотношения $f(s,t) = 2f(s-1,t)f(s,t-1)$ при $f(s,0) = f(0,t) = 1$. (б) Рекуррентное соотношение $f(s,t) = (s+1)!\,f(s,t-1)\ldots f(0,t-1)$ имеет решение

$$(s+1)!^t\, s!^{\binom{t}{2}}\,(s-1)!^{\binom{t+1}{3}}\ldots 2!^{\binom{s+t-2}{s}} = \prod_{r=1}^{s}(r+1)!^{\binom{s+t-1-r}{t-2}+[r=s]}.$$

32. (а) Простой формулы, похоже, не существует, но для малых s и t можно подсчитать количества последовательностей посредством вычисления количества облексных путей, которые проходят по всем строкам веса t из заданной начальной точки в заданную конечную, выполняя только перемещения со свойством двери-вертушки. Вот общие количества для $s+t \le 6$:

$$\begin{array}{ccccccccccccc}
 & & & & & & 1 & & & & & & \\
 & & & & & 1 & & 1 & & & & & \\
 & & & & 1 & & 2 & & 1 & & & & \\
 & & & 1 & & 4 & & 4 & & 1 & & & \\
 & & 1 & & 8 & & 20 & & 8 & & 1 & & \\
 & 1 & & 16 & & 160 & & 160 & & 16 & & 1 & \\
1 & & 32 & & 2264 & & 17152 & & 2264 & & 32 & & 1
\end{array}$$

$f(4,4) = 95\,304\,112\,865\,280$; $f(5,5) \approx 5.92646 \times 10^{48}$. [Этот класс генераторов сочетаний был впервые изучен Г. Эрлихом (G. Ehrlich), *JACM* **20** (1973), 500–513, но он не пытался их сосчитать.]

(б) Расширяя доказательство теоремы N, можно показать, что все такие последовательности или обратные к ним должны проходить от 1^t0^s до $0^a1^t0^{s-a}$ для некоторого $1 \le a \le s$. Более того, количество возможностей n_{sta} для данных s, t и a при $st > 0$ удовлетворяет условию $n_{1t1} = 1$ и соотношению

$$n_{sta} = \begin{cases} n_{s(t-1)1}\,n_{(s-1)t(a-1)}, & \text{если } a > 1; \\ n_{s(t-1)2}\,n_{(s-1)t1} + \cdots + n_{s(t-1)s}\,n_{(s-1)t(s-1)}, & \text{если } a = 1 < s. \end{cases}$$

Это рекуррентное соотношение имеет замечательное решение $n_{sta} = 2^{m(s,t,a)}$, где

$$m(s,t,a) = \begin{cases} \binom{s+t-3}{t} + \binom{s+t-5}{t-2} + \cdots + \binom{s-1}{2}, & \text{если } t \text{ четно;} \\ \binom{s+t-3}{t} + \binom{s+t-5}{t-2} + \cdots + \binom{s}{3} + s - a - [a<s], & \text{если } t \text{ нечетно.} \end{cases}$$

33. Сначала рассмотрим случай $t = 1$: количество близких к идеальным путей от i до $j > i$ равно $f(j - i - [i>0] - [j<n-1])$, где $\sum_j f(j)z^j = 1/(1 - z - z^3)$. (По случайному совпадению, та же последовательность $f(j)$ возникает и в многофазном слиянии Карона (Caron) для шести лент, табл. 5.4.2–2.) Сумма по $0 \le i < j < n$ равна $3f(n) + f(n-1) + f(n-2) + 2 - n$, и мы должны ее удвоить, чтобы охватить случаи $j > i$.

Когда $t > 1$, можно построить $\binom{n}{t} \times \binom{n}{t}$ матриц, которые подскажут нам, сколько облексных последовательностей начинаются и заканчиваются определенными сочетаниями. Элементы этих матриц представляют собой суммы произведений матриц для случая $t-1$, просуммированные по всем путям типов, рассмотренных для случая $t = 1$. Вот общие количества для $s + t \le 6$:

$$\begin{array}{ccccccc} & & & 1 & & & \\ & & 1 & & 1 & & \\ & 1 & & 2 & & 1 & \\ 1 & & 6 & & 2 & & 1 \\ 1 & 12 & & 10 & & 2 & 1 \\ 1 & 20 & 44 & & 10 & 2 & 1 \\ 1 & 34 & 238 & 68 & 10 & 2 & 1 \end{array} \qquad \begin{array}{ccccccc} & & & 1 & & & \\ & & 1 & & 1 & & \\ & 1 & & 2 & & 1 & \\ 1 & & 2 & & 0 & & 1 \\ 1 & 2 & & 2 & & 0 & 1 \\ 1 & 2 & 0 & & 0 & 0 & 1 \\ 1 & 2 & 6 & 0 & 0 & 0 & 1 \end{array}$$

Здесь в правом треугольнике показаны количества *циклов*, $g(s,t)$. Прочие значения включают $f(4,4) = 17\,736$; $f(5,5) = 9\,900\,888\,879\,984$; $g(4,4) = 96$; $g(5,5) = 30\,961\,456\,320$.

Имеется ровно 10 таких схем при $s = 2$ и $n \ge 4$. Например, когда $n = 7$ они пробегают от 43210 до 65431 или 65432, или от 54321 до 65420 или 65430, или 65432, или в обратном порядке.

34. Минимум может быть вычислен, как в предыдущем ответе, но с использованием вместо обычного матричного умножения $c_{ij} = \sum_k a_{ik}b_{kj}$ операции $c_{ij} = \min_k(a_{ik} + b_{kj})$. (Когда $s = t = 5$, облексный путь на рис. 46, (д) с 49 неидеальными переходами, по сути, единственный. Для $s = t = 5$ имеется облексный цикл, который содержит только 55 “неидеальностей”.)

35. Из рекуррентных соотношений (35) мы имеем $a_{st} = b_{s(t-1)} + [s>1][t>0] + a_{(s-1)t}$, $b_{st} = a_{s(t-1)} + a_{(s-1)t}$; следовательно, $a_{st} = b_{st} + [s>1][t \text{ нечетно}]$ и $a_{st} = a_{s(t-1)} + a_{(s-1)t} + [s>1][t \text{ нечетно}]$. Решение имеет следующий вид:

$$a_{st} = \sum_{k=0}^{t/2} \binom{s+t-2-2k}{s-2} - [s>1][t \text{ четно}];$$

эта сумма приблизительно равна $\frac{s}{s+2t}\binom{s+t}{t}$.

36. Рассмотрим бинарное дерево с корневым узлом (s,t) и с рекурсивно определяемыми поддеревьями с корнями $(s-1,t)$ и $(s,t-1)$, пока $st > 0$; при $st = 0$ узел (s,t) является листом. Тогда дерево с корнем (s,t) имеет $\binom{s+t}{t}$ листьев, соответствующих всем (s,t)-сочетаниям $a_{n-1}\ldots a_1a_0$. Узлы на уровне l соответствуют префиксам $a_{n-1}\ldots a_{n-l}$, а листья на уровне l представляют собой сочетания с $r = n - l$.

Любой облексный алгоритм для сочетаний $a_{n-1}\ldots a_1a_0$ соответствует прямому порядку обхода такого дерева, после того как дочерние узлы $\binom{s+t}{t} - 1$ узлов с ветвлениями упорядочены в некотором требуемом порядке; вот почему имеется $2^{\binom{s+t}{t}-1}$ таких облексных схем (упр. 31, (а)). Операция $j \leftarrow j+1$ выполняется в точности один раз для каждого узла с ветвлением, а именно после того, как обработаны оба дочерних узла.

Кстати, из упр. 7.2.1.2–6, (а) следует, что среднее значение r равно величине $s/(t+1)+t/(s+1)$, которая может представлять собой $\Omega(n)$; таким образом, дополнительное время, необходимое для отслеживания r, теряется не зря.

37. (а) В лексикографическом случае нам не нужно поддерживать таблицу w_j, поскольку a_j активно для $j \ge r$ тогда и только тогда, когда $a_j = 0$. После установки $a_j \leftarrow 1$ и $a_{j-1} \leftarrow 0$ имеется два случая, которые следует рассмотреть при $j > 1$: если $r = j$, установить $r \leftarrow j-1$; в противном случае установить $a_{j-2}\ldots a_0 \leftarrow 0^r 1^{j-1-r}$ и $r \leftarrow j-1-r$ (или $r \leftarrow j$, если r было равно $j-1$).

(б) Теперь переходы, обрабатываемые при $j > 1$, состоят в изменении $a_j \ldots a_0$ следующим образом: $01^r \to 1101^{r-2}$, $010^r \to 10^{r+1}$, $010^a1^r \to 110^{a+1}1^{r-1}$, $10^r \to 010^{r-1}$, $110^r \to 010^{r-1}1$, $10^a1^r \to 0^a1^{r+1}$; эти шесть случаев легко различимы. Значение r должно изменяться соответствующим образом.

(в) Случай $j = 1$ вновь тривиален. В противном случае $01^a0^r \to 101^{a-1}0^r$; $0^a1^r \to 10^a1^{r-1}$; $101^a0^r \to 01^{a+1}0^r$; $10^a1^r \to 0^a1^{r+1}$; имеется также один неоднозначный случай, который может произойти, только если $a_{n-1}\ldots a_{j+1}$ содержит как минимум один 0: пусть $k > j$ — минимальное, для которого $a_k = 0$. Тогда $10^r \to 010^{r-1}$, если k нечетно, и $10^r \to 0^r1$, если k четно.

38. Работает тот же алгоритм C с небольшими модификациями. (i) Шаг C1 устанавливает $a_{n-1}\ldots a_0 \leftarrow 01^t0^{s-1}$, если n нечетно или $s = 1$, $a_{n-1}\ldots a_0 \leftarrow 001^t0^{s-2}$, если n четно и $s > 1$, и соответствующее значение r; (ii) шаг C3 меняет роли четных и нечетных значений; (iii) шаг C5 переходит к шагу C4 также и при $j = 1$.

39. В общем случае работа начинается с $r \leftarrow 0$ и $j \leftarrow s+t-1$, и до тех пор, пока не будет выполнено условие $st = 0$, повторяются следующие действия:

$$r \leftarrow r + [w_j = 0]\binom{j}{s-a_j}, \quad s \leftarrow s - [a_j = 0], \quad t \leftarrow t - [a_j = 1], \quad j \leftarrow j-1.$$

Тогда r представляет собой ранг $a_{n-1}\ldots a_1a_0$. Так что ранг 110010010000111111101101010 равен $\binom{23}{12}+\binom{22}{11}+\binom{21}{9}+\binom{17}{8}+\binom{16}{7}+\binom{14}{5}+\binom{13}{3}+\binom{12}{3}+\binom{11}{3}+\binom{10}{3}+\binom{9}{3}+\binom{8}{3}+\binom{4}{3}+\binom{3}{1}+\binom{1}{0} = 2\,390\,131$.

40. Начнем с $N \leftarrow 999999$, $v \leftarrow 0$ и будем повторять следующие шаги до выполнения условия $st = 0$: если $v = 0$, то при $N < \binom{s+t-1}{s}$ устанавливаем $t \leftarrow t-1$ и $a_{s+t} \leftarrow 1$, а в противном случае — $N \leftarrow N - \binom{s+t-1}{s}$, $v \leftarrow (s+t) \bmod 2$, $s \leftarrow s-1$, $a_{s+t} \leftarrow 0$. Если же $v = 1$, то при $N < \binom{s+t-1}{t}$ устанавливаем $v \leftarrow (s+t) \bmod 2$, $s \leftarrow s-1$ и $a_{s+t} \leftarrow 0$, а в противном случае — $N \leftarrow N - \binom{s+t-1}{t}$, $t \leftarrow t-1$, $a_{s+t} \leftarrow 1$. В конце, если $s = 0$, устанавливаем $a_{t-1}\ldots a_0 \leftarrow 1^t$; если $t = 0$, устанавливаем $a_{s-1}\ldots a_0 \leftarrow 0^s$. Ответ для указанных в условии значений — $a_{25}\ldots a_0 = 11101001111110101001000001$.

41. Пусть $c(0), \ldots, c(2^n-1) = C_n$, где $C_{2n} = 0C_{2n-1}, 1C_{2n-1}$; $C_{2n+1} = 0C_{2n}, 1\widehat{C}_{2n}$; $\widehat{C}_{2n} = 1C_{2n-1}, 0\widehat{C}_{2n-1}$; $\widehat{C}_{2n+1} = 1\widehat{C}_{2n}, 0\widehat{C}_{2n}$; $C_0 = \widehat{C}_0 = \epsilon$. Тогда $a_j \oplus b_j = b_{j+1}\ \&\ (b_{j+2} \mid (b_{j+3}\&(b_{j+4}|\cdots)))$, если j четно, $b_{j+1}|(b_{j+2}\&(b_{j+3}|(b_{j+4}\&\cdots)))$, если j нечетно. Любопытно, что мы получаем также обратное соотношение $c((\ldots a_4\bar{a}_3a_2\bar{a}_1a_0)_2) = (\ldots b_4\bar{b}_3b_2\bar{b}_1b_0)_2$.

42. Уравнение (40) показывает, что левый контекст $a_{n-1}\ldots a_{l+1}$ не влияет на поведение алгоритма для $a_{l-1}\ldots a_0$, если $a_l = 0$ и $l > r$. Следовательно, мы можем проанализировать алгоритм C путем подсчета сочетаний, завершающихся определенным битовым шаблоном, и отсюда следует, что количество выполнений каждой операции может быть представлено как $[w^sz^t]\,p(w,z)/(1-w^2)^2(1-z^2)^2(1-w-z)$ при подходящем полиноме $p(w,z)$.

Например, алгоритм переходит от C5 к C4 по одному разу для каждого сочетания, которое оканчивается на $01^{2a+1}01^{2b+1}$ или имеет вид $1^{a+1}01^{2b+1}$ для целых $a, b \ge 0$; соответствующими производящими функциями являются $w^2z^2/(1-z^2)^2(1-w-z)$ и $w(z^2+z^3)/(1-z^2)^2$.

Вот как выглядят полиномы $p(w,z)$ для ключевых операций. Здесь $W = 1 - w^2$, $Z = 1 - z^2$.

C3 → C4:	$wzW(1+wz)(1-w-z^2)$;	C5($r \leftarrow 1$):	$w^2zW^2Z(1-wz-z^2)$;
C3 → C5:	$wzW(w+z)(1-wz-z^2)$;	C5($r \leftarrow j-1$):	$w^2z^3W^2(1-wz-z^2)$;
C3 → C6:	$w^2z^2W(w+z)$;	C6($j=1$):	w^2zW^2Z;
C3 → C7:	$w^2zW(1+wz)$;	C6($r \leftarrow j-1$):	$w^2z^3W^2$;
C4($j=1$):	$wzW^2Z(1-w-z^2)$;	C6($r \leftarrow j$):	w^3z^2WZ;
C4($r \leftarrow j-1$):	$w^3zWZ(1-w-z^2)$;	C7 → C6:	w^2zW^2;
C4($r \leftarrow j$):	$wz^2W^2(1+z-2wz-z^2-z^3)$;	C7($r \leftarrow j$):	w^4zWZ;
C5 → C4:	$wz^2W^2(1-wz-z^2)$;	C7($r \leftarrow j-2$):	$w^3z^2W^2$.
C5($r \leftarrow j-2$):	$w^4zWZ(1-wz-z^2)$;		

Асимптотическое значение равно $\binom{s+t}{t}\big(p(1-x,x)/(2x-x^2)^2(1-x^2)^2+O(n^{-1})\big)$ для фиксированного $0 < x < 1$, если $t = xn + O(1)$ при $n \to \infty$. Таким образом, мы находим, например, что четыре пути ветвления на шаге C3 имеют относительные частоты $x + x^2 - x^3 : 1 : x : 1 + x - x^2$.

Кстати, количество случаев нечетного j превосходит количество случаев четного j на

$$\sum_{k,l\ge 1}\binom{s+t-2k-2l}{s-2k}[2k+2l\le s+t]+[s\,\text{нечетно}][t\,\text{нечетно}]$$

для *любой* облексной схемы, использующей метод (39). Эта величина имеет интересную производящую функцию $wz/(1+w)(1+z)(1-w-z)$.

43. Тождество истинно для всех неотрицательных целых x, за исключением $x = 1$. (Кстати, $s(x) = f(x) \oplus 1$ и $p(x) = f(x \oplus 1)$, где $f(x) = (x \dot{-} 1) + ((x \mathbin{\&} 1) \ll 1)$.)

44. В действительности $C_t(n) - 1 = \widehat{C}_t(n-1)^R$ и $\widehat{C}_t(n) - 1 = C_t(n-1)^R$. (Следовательно, $C_t(n) - 2 = C_t(n-2)$ и т. д.)

45. В приведенном далее алгоритме r — наименьший индекс, для которого $c_r \ge r$.

CC1. [Инициализация.] Установить $c_j \leftarrow n - t - 1 + j$ и $z_j \leftarrow 0$ для $1 \le j \le t + 1$. Также установить $r \leftarrow 1$. (Считаем, что $0 < t < n$.)

CC2. [Посещение.] Посетить сочетание $c_t \ldots c_2c_1$. Затем установить $j \leftarrow r$.

CC3. [Ветвление.] Перейти к шагу CC5, если $z_j \ne 0$.

CC4. [Попытка уменьшения c_j.] Установить $x \leftarrow c_j + (c_j \bmod 2) - 2$. Если $x \ge j$, установить $c_j \leftarrow x$, $r \leftarrow 1$; в противном случае: если $c_j = j$, установить $c_j \leftarrow j - 1$, $z_j \leftarrow c_{j+1} - ((c_{j+1} + 1) \bmod 2)$, $r \leftarrow j$; если $c_j < j$, установить $c_j \leftarrow j$, $z_j \leftarrow c_{j+1} - ((c_{j+1}+1) \bmod 2)$, $r \leftarrow \max(1, j-1)$; иначе установить $c_j \leftarrow x$, $r \leftarrow j$. Вернуться к шагу CC2.

CC5. [Попытка увеличения c_j.] Установить $x \leftarrow c_j + 2$. Если $x < z_j$, установить $c_j \leftarrow x$; в противном случае при $x = z_j$ и $z_{j+1} \ne 0$ установить $c_j \leftarrow x - (c_{j+1} \bmod 2)$; иначе установить $z_j \leftarrow 0$, $j \leftarrow j + 1$ и перейти к шагу CC3 (но если $j > t$, завершить работу алгоритма). Если $c_1 > 0$, установить $r \leftarrow 1$; в противном случае установить $r \leftarrow j - 1$. Вернуться к шагу CC2. ▮

46. Из уравнения (40) следует, что $u_k = (b_j + k + 1) \bmod 2$, где j — минимальное значение, при котором выполняется условие $b_j > k$. Тогда (37) и (38) дают следующий алгоритм, в котором для удобства полагается, что $3 \le s < n$.

CB1. [Инициализация.] Установить $b_j \leftarrow j - 1$ для $1 \le j \le s$; установить также $z \leftarrow s + 1$, $b_z \leftarrow 1$. (Когда последующие шаги проверяют значение z, оно представляет собой наименьший индекс, такой, что $b_z \ne z - 1$.)

CB2. [Посещение.] Посетить дуальное сочетание $b_s \ldots b_2 b_1$.

CB3. [Ветвление.] Если b_2 нечетно: перейти к шагу CB4, если $b_2 \neq b_1 + 1$, иначе — к шагу CB5, если $b_1 > 0$, в противном случае к шагу CB6, если b_z нечетно. Перейти к шагу CB9, если b_2 четно и $b_1 > 0$. В противном случае перейти к шагу CB8, если $b_{z+1} = b_z + 1$, иначе перейти к шагу CB7.

CB4. [Увеличение b_1.] Установить $b_1 \leftarrow b_1 + 1$ и вернуться к шагу CB2.

CB5. [Смещение b_1 и b_2.] Если b_3 нечетно, установить $b_1 \leftarrow b_1 + 1$ и $b_2 \leftarrow b_2 + 1$; в противном случае установить $b_1 \leftarrow b_1 - 1$, $b_2 \leftarrow b_2 - 1$, $z \leftarrow 3$. Перейти к шагу CB2.

CB6. [Смещение влево.] Если z нечетно, установить $z \leftarrow z - 2$, $b_{z+1} \leftarrow z + 1$, $b_z \leftarrow z$; в противном случае установить $z \leftarrow z - 1$, $b_z \leftarrow z$. Перейти к шагу CB2.

CB7. [Смещение b_z.] Если b_{z+1} нечетно, установить $b_z \leftarrow b_z + 1$ и завершить работу алгоритма, если $b_z \geq n$; в противном случае установить $b_z \leftarrow b_z - 1$, после этого, если $b_z < z$, установить $z \leftarrow z + 1$. Перейти к шагу CB2.

CB8. [Смещение b_z и b_{z+1}.] Если b_{z+2} нечетно, установить $b_z \leftarrow b_{z+1}$, $b_{z+1} \leftarrow b_z + 1$ и завершить работу алгоритма, если $b_{z+1} \geq n$. В противном случае установить $b_{z+1} \leftarrow b_z$, $b_z \leftarrow b_z - 1$, после чего, если $b_z < z$, установить $z \leftarrow z + 2$. Перейти к шагу CB2.

CB9. [Уменьшение b_1.] Установить $b_1 \leftarrow b_1 - 1$, $z \leftarrow 2$ и вернуться к шагу CB2. ▮

Обратите внимание, что это — алгоритм *без циклов*. Чейз приводит похожую процедуру для последовательности $\widehat{C}^R_{st}$ в *Cong. Num.* **69** (1989), 233–237. Удивительно, что этот алгоритм определяет точное дополнение индексов $c_t \ldots c_1$, полученных с помощью алгоритма из предыдущего упражнения.

47. Мы можем, например, использовать алгоритм C и обратный к нему (упр. 38) с заменой w_j d-битовым числом, биты которого представляют активность на разных уровнях рекурсии. Для отслеживания r-значений на каждом уровне требуются отдельные указатели $r_0, r_1, \ldots, r_{d-1}$. (Возможно множество других решений.)

48. Имеются такие перестановки $\pi_1, \ldots, \pi_M$, что k-й элемент Λ_j равен $\pi_k \alpha_j \uparrow \beta_{k-1}$, а $\pi_k \alpha_j$ пробегают все перестановки $\{s_1 \cdot 1, \ldots, s_d \cdot d\}$ при изменении j от 0 до $N - 1$.

Историческая справка. Первая публикация однородной схемы двери-вертушки для (s,t)-сочетаний — [Éva Török, *Matematikai Lapok* **19** (1968), 143–146], посвященная генерации перестановок мультимножества. Позже многие исследователи опирались на условие однородности при подобных построениях, но данное упражнение показывает, что однородность не является необходимой.

49. Мы имеем $\lim_{z\to q}(z^{km+r} - 1)/(z^{lm+r} - 1) = 1$ при $0 < r < m$, а при $r = 0$ предел равен $\lim_{z\to q}(kmz^{km-1})/(lmz^{lm-1}) = k/l$. Так что мы можем объединить в пары множители числителя $\prod_{n-k<a\leq n}(z^a - 1)$ и знаменателя $\prod_{0<b\leq k}(z^b - 1)$, когда $a \equiv b$ (по модулю m).

Примечания. Эта формула открыта Г. Олив (G. Olive), *AMM* **72** (1965), 619. В частном случае $m = 2$, $q = -1$, второй множитель равен нулю тогда и только тогда, когда n четно, а k нечетно.

Формула $\binom{n}{k}_q = \binom{n}{n-k}_q$ выполняется для всех $n \geq 0$, но $\binom{\lfloor n/m\rfloor}{\lfloor k/m\rfloor}$ *не* всегда равно $\binom{\lfloor n/m\rfloor}{\lfloor (n-k)/m\rfloor}$. Причина этого в том, что второй множитель равен нулю, если только не выполняется условие $n \bmod m \geq k \bmod m$, и в этом случае $\lfloor k/m\rfloor + \lfloor (n-k)/m\rfloor = \lfloor n/m\rfloor$.

50. Указанный коэффициент равен нулю, когда $n_1 \bmod m + \cdots + n_t \bmod m \geq m$. В противном случае он равен

$$\binom{\lfloor (n_1 + \cdots + n_t)/m\rfloor}{\lfloor n_1/m\rfloor, \ldots, \lfloor n_t/m\rfloor} \binom{(n_1 + \cdots + n_t) \bmod m}{n_1 \bmod m, \ldots, n_t \bmod m}_q$$

согласно формуле 1.2.6–(43); здесь каждый верхний индекс равен сумме нижних.

51. Понятно, что все пути проходят между 000111 и 111000, поскольку эти вершины имеют степень 1. Все четырнадцать путей при указанных условиях эквивалентности сводятся к четырем. Путь из (50), эквивалентный самому себе при отражении и обращении, может быть описан дельта-последовательностью $A = 3452132523414354123$; три других класса — $B = 3452541453414512543$, $C = 3452541453252154123$, $D = 3452134145341432543$. Д. Г. Лемер (D. H. Lehmer) открыл путь C [*AMM* **72** (1965), Part II, 36–46]; D, по сути, является путем, построенным Идесом (Eades), Хикки (Hickey) и Ридом (Read).

(Кстати, идеальные схемы на самом деле не редкость, хотя, похоже, их трудно строить систематически. Для случая $(s,t) = (3,5)$ имеется 4 050 046 таких схем.)

52. Можно считать, что каждый элемент s_j не равен нулю и что $d > 1$. Тогда в соответствии с упр. 50 разница между перестановками с четными и нечетными количествами инверсий равна $\binom{\lfloor (s_0+\cdots+s_d)/2\rfloor}{\lfloor s_0/2\rfloor,\ldots,\lfloor s_d/2\rfloor} \geq 2$, за исключением случая, когда как минимум две из кратностей s_j нечетны.

И обратно: если как минимум две кратности нечетны, обобщенное построение Г. Стаховяка (G. Stachowiak) [*SIAM J. Discrete Math.* **5** (1992), 199–206] показывает, что идеальная схема существует. На самом деле его построение применимо к ряду задач топологической сортировки; в частном случае мультимножеств оно дает гамильтонов цикл во всех случаях при $d > 1$ и нечетном $s_0 s_1$, за исключением случаев $d = 2$, $s_0 = s_1 = 1$ и четного s_2.

53. См. *AMM* **72** (1965), Part II, 36–46.

54. В предположении, что $st \neq 0$, гамильтонов путь существует тогда и только тогда, когда s и t не являются одновременно четными; гамильтонов цикл существует тогда и только тогда, когда дополнительно ($s \neq 2$ и $t \neq 2$) или $n = 5$. [T. C. Enns, *Discrete Math.* **122** (1993), 153–165.]

55. (a) [Решение Аарона Вильямса (Aaron Williams).] Последовательность 0^s1^t, W_{st} обладает правильными свойствами, если

$$W_{st} \;=\; 0W_{(s-1)t},\ 1W_{s(t-1)},\ 10^s1^{t-1} \quad \text{для } st > 0; \qquad W_{0t} = W_{s0} = \emptyset.$$

А вот — удивительно эффективная реализация *без использования циклов*. Считаем, что $t > 0$.

W1. [Инициализация.] Установить $n \leftarrow s+t$, $a_j \leftarrow 1$ для $0 \leq j < t$ и $a_j \leftarrow 0$ для $t \leq j \leq n$. Затем установить $j \leftarrow k \leftarrow t-1$. (Это хитрый трюк, но он работает.)

W2. [Посещение.] Посетить (s,t)-сочетание $a_{n-1}\ldots a_1a_0$.

W3. [Обнуление a_j.] Установить $a_j \leftarrow 0$ и $j \leftarrow j+1$.

W4. [Простой случай?] Если $a_j = 1$, установить $a_k \leftarrow 1$, $k \leftarrow k+1$ и вернуться к шагу W2.

W5. [Оборот.] Если $j = n$, завершить работу. В противном случае установить $a_j \leftarrow 1$. Затем, если $k > 0$, установить $a_k \leftarrow 1$, $a_0 \leftarrow 0$, $j \leftarrow 1$ и $k \leftarrow 0$. Вернуться к шагу W2. ▮

После второго посещения j — наименьший индекс, для которого $a_ja_{j-1} = 10$, а k — наименьший индекс, для которого $a_k = 0$. Простой случай встречается ровно $\binom{s+t-1}{s} - 1$ раз; условие $k = 0$ на шаге W5 выполняется ровно $\binom{s+t-2}{t} + \delta_{t1}$ раз. Любопытно, что если N имеет комбинаторное представление (57), то сочетание ранга N в алгоритме L имеет ранг $N - t + \binom{n_v}{v-1} + v - 1$ в алгоритме W. [*Lecture Notes in Comp. Sci.* **3595** (2005), 570–576; в работе A. Williams, *SODA* **20** (2009), 987–996, имеется значительное обобщение, благодаря которому можно сгенерировать перестановки произвольного мультимножества без использования циклов, путем циклических сдвигов префиксов.]

```
(б) SET bits,(1<<t)-1                          (Предполагается, что s > 0 и t > 0.)
1H PUSHJ $0,Visit                              Посещение bits = (a_{s+t-1} ... a_1 a_0)_2.
   ADDU $0,bits,1; AND $0,$0,bits              Установить $0 ← bits & (bits + 1).
   SUBU $1,$0,1; XOR $1,$0,$1                  Установить $1 ← $0 ⊕ ($0 − 1).
   ADDU $0,$1,1; AND $1,$1,bits                Установить $0 ← $1 + 1, $1 ← $1 & bits.
   AND $0,$0,bits; ODIF $0,$0,1                Установить $0 ← ($0 & bits) ∸ 1.
   SUBU $1,$1,$0; ADDU bits,bits,$1            Установить bits ← bits + $1 − $0.
   SRU $0,bits,s+t; PBZ $0,1B                  Повторять до выполнения условия a_{s+t} = 1.
```
∎

56. [*Discrete Math.* **48** (1984), 163–171.] Эта задача эквивалентна "гипотезе средних уровней", которая утверждает, что существует путь Грея по всем бинарным строкам длиной $2t-1$ и весами $\{t-1, t\}$. В действительности такие строки могут быть сгенерированы с помощью дельта-последовательности специального вида $\alpha_0\alpha_1\ldots\alpha_{2t-2}$, в которой элементы α_k представляют собой элемент α_0, сдвинутый на k позиций по модулю $2t-1$. Например, при $t=3$ можно начать с $a_5a_4a_3a_2a_1a_0 = 000111$ и повторно выполнять обмены $a_0 \leftrightarrow a_\delta$, где δ пробегает цикл (4134 5245 1351 2412 3523). Известно, что гипотеза средних уровней справедлива для $t \le 15$ [см. I. Shields and C. D. Savage, *Cong. Num.* **140** (1999), 161–178].

57. Да; имеется близкое к идеальному облексное решение для всех m, n и t при $n \ge m > t$. Одна такая схема (при применении записи в виде битовых строк) представляет собой $1A_{(m-t)(t-1)}0^{n-m}$, $01A_{(m-t)(t-1)}0^{n-m-1}$, $\ldots$, $0^{n-m}1A_{(m-t)(t-1)}$, $0^{n-m+1}1A_{(m-1-t)(t-1)}$, $\ldots$, $0^{n-t}1A_{0(t-1)}$ с использованием последовательности A_{st} из (35).

58. Решите предыдущую задачу с m и n, уменьшенными на $t-1$, а затем добавьте $j-1$ к каждому c_j. (Случай (а), который особенно прост, вероятно, был известен Черны (Czerny).)

59. Производящая функция $G_{mnt}(z) = \sum g_{mntk}z^k$ для количества аккордов g_{mntk}, достижимых за k шагов от $0^{n-t}1^t$, удовлетворяет уравнениям $G_{mmt}(z) = \binom{m}{t}_z$ и $G_{m(n+1)t}(z) = G_{mnt}(z) + z^{tn-(t-1)m}\binom{m-1}{t-1}_z$, поскольку последний член учитывает случаи $c_t = n$ и $c_1 > n-m$. Идеальная схема возможна, только если $|G_{mnt}(-1)| \le 1$. Но если $n \ge m > t \ge 2$, это условие, согласно (49), выполняется только при $m = t+1$ или нечетном $(n-t)t$. Таким образом, идеального решения для $t=4$ и $m>5$ не существует. (Многие аккорды при $n=t+2$ имеют только двух соседей, так что можно легко исключить этот случай. Все случаи с $n \ge m > 5$ и $t=3$, вероятно, не имеют идеальных путей при четном n.)

60. В приведенном далее решении используется лексикографический порядок и принимаются меры для того, чтобы среднее количество вычислений на одно посещение было ограниченным. Можно считать, что $stm_s\ldots m_0 \ne 0$ и $t \le m_s + \cdots + m_1 + m_0$.

Q1. [Инициализация.] Установить $q_j \leftarrow 0$ для $s \ge j \ge 1$ и $x \leftarrow t$.

Q2. [Распределение.] Установить $j \leftarrow 0$. Затем, если $x > m_j$, установить $q_j \leftarrow m_j$, $x \leftarrow x - m_j$ и $j \leftarrow j+1$ и повторять эти действия, пока выполняется условие $x > m_j$. В конце установить $q_j \leftarrow x$.

Q3. [Посещение.] Посетить ограниченную композицию $q_s + \cdots + q_1 + q_0$.

Q4. [Выбор крайних справа единиц.] Если $j=0$, установить $x \leftarrow q_0 - 1$, $j \leftarrow 1$. В противном случае, если $q_0 = 0$, установить $x \leftarrow q_j - 1$, $q_j \leftarrow 0$ и $j \leftarrow j+1$, а иначе перейти к шагу Q7.

Q5. [Завершение?] Завершить работу, если $j > s$. В противном случае, если $q_j = m_j$, установить $x \leftarrow x + m_j$, $q_j \leftarrow 0$, $j \leftarrow j+1$ и повторить этот шаг.

Q6. [Увеличение q_j.] Установить $q_j \leftarrow q_j + 1$. Затем, если $x = 0$, установить $q_0 \leftarrow 0$ и вернуться к шагу Q3. (В этом случае $q_{j-1} = \cdots = q_0 = 0$.) В противном случае перейти к шагу Q2.

Q7. [Увеличение и уменьшение.] (Сейчас $q_i = m_i$ для $j > i \geq 0$.) Пока $q_j = m_j$, установить $j \leftarrow j + 1$ и повторять эти действия до тех пор, пока не будет выполнено условие $q_j < m_j$ (но если $j > s$, завершить работу алгоритма). Затем установить $q_j \leftarrow q_j + 1$, $j \leftarrow j - 1$, $q_j \leftarrow q_j - 1$. Если $q_0 = 0$, установить $j \leftarrow 1$. Вернуться к шагу Q3. ▮

Например, если $m_s = \cdots = m_0 = 9$, то после композиции $3 + 9 + 9 + 7 + 0 + 0$ идут $4+0+0+6+9+9$, $4+0+0+7+8+9$, $4+0+0+7+9+8$, $4+0+0+8+7+9$, ...

61. Пусть $F_s(t) = \emptyset$, если $t < 0$ или $t > m_s + \cdots + m_0$; в противном случае пусть $F_0(t) = t$, а

$$F_s(t) = 0{+}F_{s-1}(t),\ 1{+}F_{s-1}(t-1)^R,\ 2{+}F_{s-1}(t-2),\ \ldots,\ m_s{+}F_{s-1}(t-m_s)^{R^{m_s}}$$

при $s > 0$. Можно показать, что эта последовательность обладает требуемыми свойствами. Фактически она эквивалентна композициям, определенным однородной последовательностью K_{st} из (31) при использовании соответствия из упр. 4 при наложенных на подпоследовательность ограничениях, определяемых границами $m_s, \ldots, m_0$. [См. T. Walsh, *J. Combinatorial Math. and Combinatorial Computing* **33** (2000), 323–345, где эта последовательность генерируется без применения циклов.]

62. (а) Факторная таблица размером $2 \times n$ с суммами строк r и $c_1 + \cdots + c_n - r$ эквивалентна решению $r = a_1 + \cdots + a_n$ при $0 \leq a_1 \leq c_1, \ldots, 0 \leq a_n \leq c_n$.

(б) Ее можно вычислить, последовательно устанавливая $a_{ij} \leftarrow \min(r_i - a_{i1} - \cdots - a_{i(j-1)}, c_j - a_{1j} - \cdots - a_{(i-1)j})$ для $j = 1, \ldots, n$ и $i = 1, \ldots, m$. В качестве другого решения можно, если $r_1 \leq c_1$, установить $a_{11} \leftarrow r_1$, $a_{12} \leftarrow \cdots \leftarrow a_{1n} \leftarrow 0$ и работать с остальными строками с c_1, уменьшенным на r_1; если же $r_1 > c_1$, установить $a_{11} \leftarrow c_1$, $a_{21} \leftarrow \cdots \leftarrow a_{m1} \leftarrow 0$ и работать с остальными столбцами с r_1, уменьшенным на c_1. Из второго подхода видно, что ненулевыми являются не более $m + n - 1$ элементов. Можно указать и явную формулу

$$a_{ij} = \max(0, \min(r_i, c_j, r_1 + \cdots + r_i - c_1 - \cdots - c_{j-1}, c_1 + \cdots + c_j - r_1 - \cdots - r_{i-1})).$$

(в) Получается такая же матрица, как и в п. (б).

(г) Поменяйте местами левые и правые части в пп. (б) и (в); в обоих случаях ответ

$$a_{ij} = \max(0, \min(r_i, c_j, r_1 + \cdots + r_i - c_{j+1} - \cdots - c_n, c_j + \cdots + c_n - r_1 - \cdots - r_{i-1})).$$

(д) Давайте выберем, скажем, построчный порядок. Сгенерируем первую строку так же, как и для ограниченной композиции r_1 с границами $(c_1, \ldots, c_n)$; а для каждой строки $(a_{11}, \ldots, a_{1n})$ сгенерируем оставшиеся строки рекурсивно тем же способом, но с суммами столбцов $(c_1 - a_{11}, \ldots, c_n - a_{1n})$. Большинство действий этого алгоритма выполняются только с двумя нижними строками; когда же изменения затрагивают более высокие строки, то строки, идущие за ними, должны быть инициализированы заново.

63. Если a_{ij} и a_{kl} положительны, мы получаем другую факторную таблицу путем установки $a_{ij} \leftarrow a_{ij} - 1$, $a_{il} \leftarrow a_{il} + 1$, $a_{kj} \leftarrow a_{kj} + 1$, $a_{kl} \leftarrow a_{kl} - 1$. Рассмотрим граф G, вершины которого являются факторными таблицами для $(r_1, \ldots, r_m; c_1, \ldots, c_n)$. Эти вершины смежны, если они могут быть получены одна из другой с помощью указанного преобразования. Мы хотим показать, что этот граф G имеет гамильтонов путь.

При $m = n = 2$ граф G представляет собой простой путь. При $m = 2$ и $n = 3$, граф G имеет двумерную структуру, из которой видно, что каждая вершина является начальной по крайней мере для двух гамильтоновых путей с разными конечными точками. При $m = 2$ и $n \geq 4$ можно индуктивно показать, что граф G имеет гамильтоновы пути из каждой вершины в любую другую.

При $m \geq 3$ и $n \geq 3$ задачу можно свести от m к $m-1$ так же, как в ответе к упр. 62, (д), если быть достаточно аккуратным, чтобы не загнать себя в угол. В частности, следует

избегать состояний, когда ненулевые элементы в последних двух строках имеют вид $\left(\begin{smallmatrix}1 & a & 0\\ 0 & b & c\end{smallmatrix}\right)$ для некоторых a, b, $c > 0$ и когда изменения в строке $m-2$ приводят к виду $\left(\begin{smallmatrix}0 & a & 1\\ 0 & b & c\end{smallmatrix}\right)$. Предыдущий цикл изменений в строках $m-1$ и m может избежать данной ловушки, если только c не равно 1 и строки не начинаются с $\left(\begin{smallmatrix}0 & a+1 & 0\\ 1 & b-1 & 1\end{smallmatrix}\right)$ или $\left(\begin{smallmatrix}1 & a-1 & 1\\ 0 & b+1 & 0\end{smallmatrix}\right)$. Однако этих ситуаций также можно избежать.

(Облексный метод, основанный на упр. 61, существенно проще и почти всегда требует только четырех изменений на каждом шаге. Однако иногда ему требуется обновить одновременно $2\min(m,n)$ элементов.)

64. Пусть $x_1\ldots x_s$ — бинарная строка, A — список подкубов и пусть $A\oplus x_1\ldots x_s$ обозначает замену цифр $(a_1,\ldots,a_s)$ в каждом подкубе A на $(a_1\oplus x_1,\ldots,a_s\oplus x_s)$ слева направо. Например, $0{*}1{*}{*}10\oplus 1010 = 1{*}1{*}{*}00$. Тогда приведенная далее взаимная рекурсия определяет цикл Грея, поскольку A_{st} дает путь Грея от $0^s{*}^t$ до $10^{s-1}{*}^t$, а B_{st} дает путь Грея от $0^s{*}^t$ до ${*}01^{s-1}{*}^{t-1}$ при $st>0$:

$$\begin{aligned} A_{st} &= 0B_{(s-1)t},\ {*}A_{s(t-1)}\oplus 001^{s-2},\ 1B^R_{(s-1)t};\\ B_{st} &= 0A_{(s-1)t},\ 1B_{(s-1)t}\oplus 010^{s-2},\ {*}A_{s(t-1)}\oplus 1^s. \end{aligned}$$

Строки 001^{s-2} и 010^{s-2} при $s<2$ представляют собой просто 0^s; A_{s0} — бинарный код Грея; $A_{0t}=B_{0t}={*}^t$. (Кстати, несколько более простая конструкция

$$G_{st} = {*}G_{s(t-1)},\ a_tG_{(s-1)t},\ a_{t-1}G^R_{(s-1)t},\qquad a_t = t \bmod 2,$$

определяет симпатичный *путь* Грея от ${*}^t0^s$ до $a_{t-1}{*}^t0^{s-1}$.)

65. Если рассматривать путь P как эквивалентный путям P^R и $P\oplus x_1\ldots x_s$, то общее количество может быть вычислено систематически, как в упр. 33, со следующими результатами для $s+t\le 6$.

Пути	Циклы
1	1
1 1	1 1
1 2 1	1 1 1
1 3 3 1	1 1 1 1
1 5 10 4 1	1 2 1 1 1
1 6 36 35 5 1	1 2 3 1 1 1
1 9 310 4630 218 6 1	1 3 46 4 1 1 1

В общем случае имеется $t+1$ путей при $s=1$ и $\binom{\lceil s/2\rceil+2}{2}-(s \bmod 2)$ путей при $t=1$. Циклы при $s\le 2$ единственны. При $s=t=5$ имеется приблизительно 6.869×10^{170} путей и 2.495×10^{70} циклов.

66. Пусть $G(n,0)=\epsilon$; $G(n,t)=\emptyset$ при $n<t$; пусть также $G(n,t)$ для $1\le t\le n$ представляет собой

$$\hat g(0)G(n-1,t),\ \hat g(1)G(n-1,t)^R,\ \ldots,\ \hat g(2^t-1)G(n-1,t)^R,\ \hat g(2^t-1)G(n-1,t-1),$$

где $\hat g(k)$ — t-битовый столбец, содержащий на вершине бинарное число Грея $g(k)$ с наименьшим значащим битом. В этой общей формуле мы неявно добавляем строку нулей под базисом $G(n-1,t-1)$.

Это замечательное правило дает обычный бинарный код Грея при $t=1$ с пропущенным $0\ldots00$. Циклический код Грея невозможен, поскольку $\binom{n}{t}_2$ нечетно.

67. Из пути Грея для композиций, соответствующих алгоритму C, вытекает, что существует путь, в котором все переходы имеют вид $0^k1^l\leftrightarrow 1^l0^k$, где $\min(k,l)\le 2$. Возможно, существует цикл, для каждого перехода которого выполняется $\min(k,l)=1$.

68. (а) $\{\emptyset\}$; (б) $\emptyset$.

69. Наименьшее N, для которого $\kappa_t N < N$, равно $\binom{2t-1}{t} + \binom{2t-3}{t-1} + \cdots + \binom{1}{1} + 1 = \frac{1}{2}\bigl(\binom{2t}{t} + \binom{2t-2}{t-1} + \cdots + \binom{0}{0} + 1\bigr)$, поскольку $\binom{n}{t-1} \le \binom{n}{t}$ тогда и только тогда, когда $n \ge 2t-1$.

70. Воспользовавшись тем фактом, что из $t \ge 3$ вытекает

$$\kappa_t\bigl(\binom{2t-3}{t} + N'\bigr) - \bigl(\binom{2t-3}{t} + N'\bigr) = \kappa_t\bigl(\binom{2t-2}{t} + N'\bigr) - \bigl(\binom{2t-2}{t} + N'\bigr) = \binom{2t-2}{t}\frac{1}{t-1} + \kappa_{t-1}N' - N'$$

при $N' < \binom{2t-3}{t}$, заключаем, что максимум равен $\binom{2t-2}{t}\frac{1}{t-1} + \binom{2t-4}{t-1}\frac{1}{t-2} + \cdots + \binom{2}{2}\frac{1}{1}$ и что он достигается при 2^{t-1} значениях N при $t > 1$.

71. Пусть C_t — t-клики. Первые $\binom{1414}{t} + \binom{1009}{t-1}$ t-сочетаний, посещенных алгоритмом L, определяют граф из 1415 вершин с 1 000 000 ребер. Если значение $|C_t|$ превышает указанное, то $|\partial^{t-2}C_t|$ должно превосходить 1 000 000. Таким образом, единственный граф, определяемый $P_{(1000000)2}$, имеет наибольшее количество t-клик для всех $t \ge 2$.

72. $M = \binom{m_s}{s} + \cdots + \binom{m_u}{u}$ для $m_s > \cdots > m_u \ge u \ge 1$, где $\{m_s, \ldots, m_u\} = \{s+t-1, \ldots, n_v\} \setminus \{n_t, \ldots, n_{v+1}\}$. (Сравните с упр. 15, которое дает $\binom{s+t}{t} - 1 - N$.)

Если $\alpha = a_{n-1}\ldots a_0$ — битовая строка, соответствующая сочетанию $n_t \ldots n_1$, то v на 1 больше количества завершающих единиц в α, а u — длина крайней справа последовательности нулей. Например, для $\alpha = 1010001111$ получаем $s = 4$, $t = 6$, $M = \binom{8}{4} + \binom{7}{3}$, $u = 3$, $N = \binom{9}{6} + \binom{7}{5}$, $v = 5$.

73. A и B перекрестно пересекающиеся $\iff \alpha \not\subseteq U \setminus \beta$ для всех $\alpha \in A$ и $\beta \in B \iff A \cap \partial^{n-s-t}B^- = \emptyset$, где $B^- = \{U \setminus \beta \mid \beta \in B\}$ — множество $(n-t)$-сочетаний. Поскольку $Q^-_{Nnt} = P_{N(n-t)}$, имеем $|\partial^{n-s-t}B^-| \ge |\partial^{n-s-t}P_{N(n-t)}|$ и $\partial^{n-s-t}P_{N(n-t)} = P_{N's}$, где $N' = \kappa_{s+1}\ldots\kappa_{n-t}N$. Таким образом, если A и B перекрестно пересекающиеся, то $M + N' \le |A| + |\partial^{n-s-t}B^-| \le \binom{n}{s}$ и $Q_{Mns} \cap P_{N's} = \emptyset$.

И наоборот: если $Q_{Mns} \cap P_{N's} \ne \emptyset$, то $\binom{n}{s} < M + N' \le |A| + |\partial^{n-s-t}B^-|$, так что A и B не могут быть перекрестно пересекающимися.

74. $|\varrho Q_{Nnt}| = \kappa_{n-t}N$ (см. упр. 94). Кроме того, применяя те же рассуждения, что и в (58) и (59), в данном частном случае находим $\varrho P_{N5} = (n-1)P_{N5} \cup \cdots \cup 10P_{N5} \cup \{543210, \ldots, 987654\}$; в общем случае $|\varrho P_{Nt}| = (n+1-n_t)N + \binom{n_t+1}{t+1}$.

75. Тождество $\binom{n+1}{k} = \binom{n}{k} + \binom{n-1}{k-1} + \cdots + \binom{n-k}{0}$ из 1.2.6–(10) дает другое представление, если $n_v > v$. Но (60) при этом не затрагивается, поскольку $\binom{n+1}{k-1} = \binom{n}{k-1} + \binom{n-1}{k-2} + \cdots + \binom{n-k+1}{0}$.

76. Представим $N+1$ путем добавления $\binom{v-1}{v-1}$ к (57); затем воспользуемся предыдущим упражнением для вывода $\kappa_t(N+1) - \kappa_t N = \binom{v-1}{v-2} = v - 1$.

77. [D. E. Daykin, *Nanta Math.* **8**, 2 (1975), 78–83.] Мы работаем с расширенными представлениями $M = \binom{m_t}{t} + \cdots + \binom{m_u}{u}$ и $N = \binom{n_t}{t} + \cdots + \binom{n_v}{v}$, как и в упр. 75, называя их *некорректными* (*improper*), если последний индекс u или v равен нулю. Назовем N *гибким* (*flexible*), если оно имеет и корректное, и некорректное представления, т. е. если $n_v > v > 0$.

(а) Для данного целого числа S найдем $M + N$, такие, что $M + N = S$ и $\kappa_t M + \kappa_t N$ минимально; при этом M должно быть велико, насколько это возможно. Если $N = 0$, задача решена. В противном случае операция максимума-минимума сохраняет как $M + N$, так и $\kappa_t M + \kappa_t N$, так что можно считать, что в корректных представлениях M и N выполняется условие $v \ge u \ge 1$. Если N не гибко, $\kappa_t(M+1) + \kappa_t(N-1) = (\kappa_t M + u - 1) + (\kappa_t N - v) < \kappa_t M + \kappa_t N$ согласно упр. 76. Следовательно, N должно быть гибким. Но тогда можно применить операцию максимума-минимума к M и к некорректному представлению N, увеличивая M. Таким образом, получено противоречие.

Это доказательство показывает, что равенство выполняется тогда и только тогда, когда $MN = 0$ (факт, замеченный в 1927 году Ф. С. Маколеем (F. S. Macaulay)).

(б) Теперь попытаемся минимизировать $\max(\kappa_t M, N) + \kappa_{t-1}N$ при $M + N = S$, представляя в этот раз N как $\binom{n_{t-1}}{t-1} + \cdots + \binom{n_v}{v}$. Если $n_{t-1} < m_t$, все еще можно использовать операцию максимума-минимума. Оставляя m_t неизменным, она сохраняет $M + N$ и $\kappa_t M + \kappa_{t-1}N$, как и соотношение $\kappa_t M > N$. Если $N \neq 0$, мы приходим, как и в п. (а), к противоречию, так что можно считать, что $n_{t-1} \geq m_t$.

Если $n_{t-1} > m_t$, то имеем $N > \kappa_t M$, а также $\lambda_t N > M$; следовательно, $M + N < \lambda_t N + N = \binom{n_{t-1}+1}{t} + \cdots + \binom{n_v+1}{v+1}$, и мы получаем $\kappa_t(M + N) \leq \kappa_t(\lambda_t N + N) = N + \kappa_{t-1}N$.

Наконец, если $n_{t-1} = m_t = a$, положим $M = \binom{a}{t} + M'$ и $N = \binom{a}{t-1} + N'$. Тогда $\kappa_t(M + N) = \binom{a+1}{t-1} + \kappa_{t-1}(M' + N')$, $\kappa_t M = \binom{a}{t-1} + \kappa_{t-1}M'$ и $\kappa_{t-1}N = \binom{a}{t-2} + \kappa_{t-2}N'$. Требуемый результат получается по индукции по t.

78. [J. Eckhoff and G. Wegner, *Periodica Math. Hung.* **6** (1975), 137–142; A. J. W. Hilton, *Periodica Math. Hung.* **10** (1979), 25–30.] Пусть $M = |A_1|$ и $N = |A_0|$; можно считать, что $t > 0$ и $N > 0$. Тогда $|\partial A| = |\partial A_1 \cup A_0| + |\partial A_0| \geq \max(|\partial A_1|, |A_0|) + |\partial A_0| \geq \max(\kappa_t M, N) + \kappa_{t-1}N \geq \kappa_t(M + N) = |P_{|A|t}|$ по индукции по $m + n + t$.

И наоборот: пусть $A_1 = P_{Mt} + 1$ и $A_0 = P_{N(t-1)} + 1$; эта запись означает, например, что $\{210, 320\} + 1 = \{321, 431\}$. Тогда $\kappa_t(M + N) \leq |\partial A| = |\partial A_1 \cup A_0| + |(\partial A_0)0| = \max(\kappa_t M, N) + \kappa_{t-1}N$, поскольку $\partial A_1 = P_{(\kappa_t M)(t-1)} + 1$. [Шютценбергер (Schützenberger) в 1959 году заметил, что $\kappa_t(M + N) \leq \kappa_t M + \kappa_{t-1}N$ тогда и только тогда, когда $\kappa_t M \geq N$.]

Перейдем к первому неравенству. Пусть A и B — непересекающиеся множества t-сочетаний, для которых $|A| = M$, $|\partial A| = \kappa_t M$, $|B| = N$, $|\partial B| = \kappa_t N$. Тогда $\kappa_t(M + N) = \kappa_t|A \cup B| \leq |\partial(A \cup B)| = |\partial A \cup \partial B| = |\partial A| + |\partial B| = \kappa_t M + \kappa_t N$.

79. В действительности $\mu_t(M + \lambda_{t-1}M) = M$ и $\mu_t N + \lambda_{t-1}\mu_t N = N + (n_2 - n_1)[v = 1]$, если N задается формулой (57).

80. Если $N > 0$ и $t > 1$, представим N в таком виде, как в (57), и пусть $N = N_0 + N_1$, где

$$N_0 = \binom{n_t - 1}{t} + \cdots + \binom{n_v - 1}{v}, \qquad N_1 = \binom{n_t - 1}{t - 1} + \cdots + \binom{n_v - 1}{v - 1}.$$

Пусть $N_0 = \binom{y}{t}$ и $N_1 = \binom{z}{t-1}$. Тогда по индукции по t и $\lfloor x \rfloor$ имеем $\binom{x}{t} = N_0 + \kappa_t N_0 \geq \binom{y}{t} + \binom{y}{t-1} = \binom{y+1}{t}$; $N_1 = \binom{x}{t} - \binom{y}{t} \geq \binom{x}{t} - \binom{x-1}{t} = \binom{x-1}{t-1}$; а $\kappa_t N = N_1 + \kappa_{t-1}N_1 \geq \binom{z}{t-1} + \binom{z}{t-2} = \binom{z+1}{t-1} \geq \binom{x}{t-1}$.

[Ловас (Lovász) в действительности доказал более строгий результат; см. упр. 1.2.6–66. Аналогично $\mu_t N \geq \binom{x-1}{t-1}$; см. Björner, Frankl, and Stanley, *Combinatorica* **7** (1987), 27–28.]

81. Например, если наибольший элемент $\widehat{P}_{N5}$ — 66433, то

$$\widehat{P}_{N5} = \{00000, \ldots, 55555\} \cup \{60000, \ldots, 65555\} \cup \{66000, \ldots, 66333\} \cup \{66400, \ldots, 66433\}$$

так что $N = \binom{10}{5} + \binom{9}{4} + \binom{6}{3} + \binom{5}{2}$. Его нижняя тень,

$$\partial\widehat{P}_{N5} = \{0000, \ldots, 5555\} \cup \{6000, \ldots, 6555\} \cup \{6600, \ldots, 6633\} \cup \{6640, \ldots, 6643\},$$

имеет размер $\binom{9}{4} + \binom{8}{3} + \binom{5}{2} + \binom{4}{1}$.

Если наименьший элемент Q_{N95} — 66433, то

$$\widehat{Q}_{N95} = \{99999, \ldots, 70000\} \cup \{66666, \ldots, 66500\} \cup \{66444, \ldots, 66440\} \cup \{66433\}$$

так что $N = (\binom{13}{9} + \binom{12}{8} + \binom{11}{7}) + (\binom{8}{6} + \binom{7}{5}) + \binom{5}{4} + \binom{3}{3}$. Его верхняя тень,

$$\varrho\widehat{Q}_{N95} = \{999999, \ldots, 700000\} \cup \{666666, \ldots, 665000\}$$
$$\cup \{664444, \ldots, 664400\} \cup \{664333, \ldots, 664330\},$$

имеет размер $\left(\binom{14}{9}+\binom{13}{8}+\binom{12}{7}\right)+\left(\binom{9}{6}+\binom{8}{5}\right)+\binom{6}{4}+\binom{4}{3}=N+\kappa_9 N$. Размер t каждого сочетания, по сути, не важен, пока $N\le\binom{s+t}{t}$; например, в рассмотренном нами случае наименьший элемент $\widehat{Q}_{N98}$ — 99966433.

82. (а) Производная должна быть равна $\sum_{k>0} r_k(x)$, но этот ряд расходится.

[Говоря неформально, график $\tau(x)$ выглядит как множество "ям" с относительной глубиной 2^{-k} во всех точках, представляющих собой нечетные кратные 2^{-k}. Оригинальная статья Такаги (Takagi) в *Proc. Physico-Math. Soc. Japan* (2) **1** (1903), 176–177, была переведена на английский язык и опубликована в его *Collected Papers* (Iwanami Shoten, 1973).]

(б) Поскольку $r_k(1-t)=(-1)^{\lceil 2^k t\rceil}$ при $k>0$, имеем $\int_0^{1-x} r_k(t)\,dt=\int_x^1 r_k(1-u)\,du=-\int_x^1 r_k(u)\,du=\int_0^x r_k(u)\,du$. Второе уравнение следует из того факта, что $r_k(\frac12 t)=r_{k-1}(t)$. В п. (г) показано, что этих двух уравнений достаточно для определения $\tau(x)$ при рациональном x.

(в) Поскольку $\tau(2^{-a}x)=a2^{-a}x+2^{-a}\tau(x)$ при $0\le x\le 1$, имеем $\tau(\epsilon)=a\epsilon+O(\epsilon)$ при $2^{-a-1}\le\epsilon\le 2^{-a}$. Следовательно, $\tau(\epsilon)=\epsilon\lg\frac{1}{\epsilon}+O(\epsilon)$ при $0<\epsilon\le 1$.

(г) Предположим, что $0\le p/q\le 1$. Если $p/q\le 1/2$, имеем $\tau(p/q)=p/q+\tau(2p/q)/2$; в противном случае $\tau(p/q)=(q-p)/q+\tau(2(q-p)/q)/2$. Следовательно, можно считать, что q нечетно. Пусть в этом случае $p'=p/2$, если p четно, и $p'=(q-p)/2$, если p нечетно. Тогда $\tau(p/q)=2\tau(p'/q)-2p'/q$ при $0<p<q$; эта система $q-1$ уравнений имеет единственное решение. Например, для $q=3$, 4, 5, 6, 7 получаем значения 2/3, 2/3; 1/2, 1/2, 1/2; 8/15, 2/3, 2/3, 8/15; 1/2, 2/3, 1/2, 2/3, 1/2; 22/49, 30/49, 32/49, 32/49, 30/49, 22/49.

(д) Решения $<\frac12$: $x=\frac14,\ \frac14-\frac{1}{16},\ \frac14-\frac{1}{16}-\frac{1}{64},\ \frac14-\frac{1}{16}-\frac{1}{64}-\frac{1}{256},\ \ldots,\ \frac16$.

(е) Значение $\frac23$ достигается при несчетном множестве значений $x=\frac12\pm\frac18\pm\frac{1}{32}\pm\frac{1}{128}\pm\cdots$.

83. Для любых данных целых чисел $q>p>0$ рассмотрим пути, начинающиеся с 0, в ориентированном графе

$$\begin{array}{ccccccccccccc}
0 & \leftarrow & 1 & \leftarrow & 2 & \leftarrow & 3 & \leftarrow & 4 & \leftarrow & 5 & \leftarrow & \cdots\\
\updownarrow & & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow & & \updownarrow & & \\
1 & \rightarrow & 2 & \rightarrow & 3 & \rightarrow & 4 & \rightarrow & 5 & \rightarrow & 6 & \rightarrow & \cdots
\end{array}$$

Вычислим связанное с путем значение v. Изначально $v\leftarrow -p$; горизонтальные перемещения удваивают v: $v\leftarrow 2v$, вертикальные перемещения из узла a изменяют v следующим образом: $v\leftarrow 2(qa-v)$. Путь завершается, когда дважды достигается узел с одним и тем же значением v. Переход в верхний узел a не разрешен, если в нем $v\le -q$ или $v\ge qa$; в нижний узел a переход запрещен, если в нем $v\le 0$ или $v\ge q(a+1)$. Эти ограничения определяют большинство шагов пути. (Узел a в верхней строке означает "Решение уравнения $\tau(x)=ax-v/q$"; нижняя строка означает "Решение уравнения $\tau(x)=v/q-ax$".) Эмпирические тесты приводят к гипотезе о конечности всех таких путей. Тогда уравнение $\tau(x)=p/q$ имеет решения $x=x_0$, определяемые последовательностью x_0, x_1, x_2, $\ldots$, где $x_k=\frac12 x_{k+1}$ на горизонтальном шаге и $x_k=1-\frac12 x_{k+1}$ на вертикальном; в конечном итоге $x_k=x_j$ для некоторого $j<k$. Если $j>0$ и если q не является степенью 2, то это все решения уравнения $\tau(x)=p/q$ при $x>1/2$.

Например, эта процедура устанавливает, что $\tau(x)=1/5$ и $x>1/2$ только при x, равном 83581/87040; единственный путь дает нам $x_0=1-\frac12 x_1$, $x_1=\frac12 x_2$, $\ldots$, $x_{18}=\frac12 x_{19}$ и $x_{19}=x_{11}$. Аналогично можно определить, что имеется только два значения $x>1/2$, для которых $\tau(x)=3/5$ (со знаменателем $2^{46}(2^{56}-1)/3$).

Кроме того, похоже, все циклы ориентированного графа, которые проходят через узел 0, определяют значения p и q, такие, что уравнение $\tau(x)=p/q$ имеет несчетно много решений. Такими значениями, например, являются 2/3, 8/15, 8/21, соответствующие

циклам (01), (0121), (012321). Значение 32/63 соответствует как (012121), так и (012101234545454321), а также еще двум другим путям, не возвращающимся в 0.

84. [Frankl, Matsumoto, Ruzsa, and Tokushige, *J. Combinatorial Theory* **A69** (1995), 125–148.] Если $a \le b$, имеем

$$\binom{2t-1-b}{t-a}\Big/T = t^{\underline{a}}(t-1)^{\underline{b-a}}/(2t-1)^{\underline{b}} = 2^{-b}\bigl(1+f(a,b)t^{-1}+O(b^4/t^2)\bigr),$$

где $f(a,b) = a(1+b)-a^2-b(1+b)/4 = f(a+1,b)-b+2a$. Следовательно, если N имеет комбинаторное представление (57) и если положить $n_j = 2t-1-b_j$, то получим

$$\frac{t}{T}\Bigl(\kappa_t N - N\Bigr) = \frac{b_t}{2^{b_t}} + \frac{b_{t-1}-2}{2^{b_{t-1}}} + \frac{b_{t-2}-4}{2^{b_{t-2}}} + \cdots + \frac{O(\log t)^3}{t}.$$

Этими членами можно пренебречь, когда b_j превышает $2\lg t$. Можно также показать, что

$$\tau\Bigl(\sum_{j=0}^{l} 2^{-e_j}\Bigr) \;=\; \sum_{j=0}^{l}(e_j-2j)2^{-e_j}.$$

85. В соответствии с (63) $N-\lambda_{t-1}N$ имеет тот же асимптотический вид, что и $\kappa_t N - N$, поскольку $\tau(x) = \tau(1-x)$. То же относится и к $2\mu_t N - N$ с точностью до $O\bigl(T(\log t)^3/t^2\bigr)$, так как $\binom{2t-1-b}{t-a} = 2\binom{2t-2-b}{t-a}\bigl(1+O(\log t)/t\bigr)$ при $b < 2\lg t$.

86. $x \in X^{\circ\sim} \iff \bar{x} \notin X^\circ \iff \bar{x} \notin X$ или $\bar{x} \notin X+e_1$, или $\cdots$, или $\bar{x} \notin X+e_n \iff x \in X^\sim$ или $x \in X^\sim - e_1$, или $\cdots$, или $x \in X^\sim - e_n \iff x \in X^{\sim+}$.

87. Все три утверждения истинны. Воспользуемся тем фактом, что $X \subseteq Y^\circ$ тогда и только тогда, когда $X^+ \subseteq Y$: (а) $X \subseteq Y^\circ \iff X^\sim \supseteq Y^{\circ\sim} = Y^{\sim+} \iff Y^\sim \subseteq X^{\sim\circ}$. (б) $X^+ \subseteq X^+ \implies X \subseteq X^{+\circ}$; следовательно, $X^\circ \subseteq X^{\circ+\circ}$. Также $X^\circ \subseteq X^\circ \implies X^{\circ+} \subseteq X$; следовательно, $X^{\circ+\circ} \subseteq X^\circ$. (в) $\alpha M \le N \iff S_M^+ \subseteq S_N \iff S_M \subseteq S_N^\circ \iff M \le \beta N$.

88. Если $\nu x < \nu y$, то $\nu(x-e_k) < \nu(y-e_j)$, так что можно считать, что $\nu x = \nu y$ и что $x > y$ в лексикографическом порядке. Должно выполняться условие $y_j > 0$; в противном случае $\nu(y-e_j)$ превышало бы $\nu(x-e_k)$. Если $x_i = y_i$ при $1 \le i \le j$, ясно, что $k > j$ и $x - e_k \prec y - e_j$. В противном случае $x_i > y_i$ для некоторого $i \le j$; и мы вновь получаем $x - e_k \prec y - e_j$, если только не выполняется равенство $x - e_k = y - e_j$.

89. Из таблицы

j =	0	1	2	3	4	5	6	7	8	9	10	11
e_j+e_1 =	e_1	e_0	e_4	e_5	e_2	e_3	e_8	e_9	e_6	e_7	e_{11}	e_{10}
e_j+e_2 =	e_2	e_4	e_0	e_6	e_1	e_8	e_3	e_{10}	e_5	e_{11}	e_7	e_9
e_j+e_3 =	e_3	e_5	e_6	e_7	e_8	e_9	e_{10}	e_0	e_{11}	e_1	e_2	e_4

находим $(\alpha 0, \alpha 1, \ldots, \alpha 12) = (0,4,6,7,8,9,10,11,11,12,12,12,12)$; $(\beta 0, \beta 1, \ldots, \beta 12) = (0,0,0,0,1,1,2,3,4,5,6,8,12)$.

90. Пусть $Y = X^+$ и $Z = C_k X$ и пусть $N_a = |X_k(a)|$ для $0 \le a < m_k$. Тогда

$$\begin{aligned}|Y| = \sum_{a=0}^{m_k-1}|Y_k(a)| &= \sum_{a=0}^{m_k-1}\bigl|(X_k(a-1)+e_k)\cup(X_k(a)+E_k(0))\bigr| \\ &\ge \sum_{a=0}^{m_k-1}\max(N_{a-1},\alpha N_a),\end{aligned}$$

где $a-1$ означает $(a-1) \bmod m_k$, а функция α получается из $(n-1)$-мерного тора, поскольку по индукции $|X_k(a) + E_k(0)| \geq \alpha N_a$. Кроме того,

$$|Z^+| = \sum_{a=0}^{m_k-1} |Z_k^+(a)| = \sum_{a=0}^{m_k-1} \big|(Z_k(a-1) + e_k) \cup (Z_k(a) + E_k(0))\big|$$
$$= \sum_{a=0}^{m_k-1} \max(N_{a-1}, \alpha N_a),$$

поскольку как $Z_k(a-1) + e_k$, так и $Z_k(a) + E_k(0)$ — стандартные множества в $n-1$ измерениях.

91. Пусть в полностью сжатом массиве, где строка 0 находится внизу, имеется N_a точек; таким образом, $l = N_{-1} \geq N_0 \geq \cdots \geq N_{m-1} \geq N_m = 0$. Покажем сначала, что существует оптимальное значение X, для которого "плохое" условие $N_a = N_{a+1}$ никогда не выполняется, за исключением случаев $N_a = 0$ и $N_a = l$. Если a — наименьший плохой индекс, предположим, что $N_{a-1} > N_a = N_{a+1} = \cdots = N_{a+k} > N_{a+k+1}$. Тогда мы всегда можем уменьшить N_{a+k} на 1 и добавить 1 к некоторому N_b $(b \leq a)$ без увеличения $|X^+|$, за исключением случаев, когда $k = 1$ и $N_{a+2} = N_{a+1} - 1$, и $N_b = N_a + a - b < l$ для $0 \leq b \leq a$. Исследуя эти случаи далее, при $N_{c+1} < N_c = N_{c-1}$ для некоторого $c > a+1$, можно установить $N_c \leftarrow N_c - 1$ и $N_a \leftarrow N_a + 1$, таким образом либо уменьшая a, либо увеличивая N_0. В противном случае можно найти такой индекс d, что $N_c = N_{a+1} + a + 1 - c > 0$ для $a < c < d$, и либо $N_d = 0$, либо $N_d < N_{d-1} - 1$. Тогда можно уменьшить N_c на 1 для $a < c < d$ и после увеличить N_b на 1 для $0 \leq b < d - a - 1$. (Важно заметить, что если $N_d = 0$, то $N_0 \geq d - 1$; следовательно, из $d = m$ вытекает $l = m$.)

Повторяя такие преобразования до тех пор, пока не будет выполнено условие $N_a > N_{a+1}$, и при этом $N_a \neq l$ и $N_{a+1} \neq 0$, мы достигнем ситуации (86), и доказательство может быть завершено так, как это сделано в разделе.

92. Пусть $x + k$ — лексикографически наименьший элемент $T(m_1, \ldots, m_{n-1})$, который превышает x и имеет вес $\nu x + k$ (если таковой элемент существует). Например, если $m_1 = m_2 = m_3 = 4$ и $x = 211$, имеем $x + 1 = 212$, $x + 2 = 213$, $x + 3 = 223$, $x + 4 = 233$, $x + 5 = 333$, а $x + 6$ не существует; в общем случае $x + k + 1$ получается из $x + k$ путем увеличения крайнего справа компонента, который может быть увеличен. Если $x + k = (m_1 - 1, \ldots, m_{n-1} - 1)$, положим $x + k + 1 = x + k$. Тогда, если $S(k)$ — множество всех элементов $T(m_1, \ldots, m_{n-1})$, которые $\preceq x + k$, получаем $S(k+1) = S(k)^+$. Кроме того, элементы S, заканчивающиеся компонентом a, — это те элементы, первые $n-1$ компонентов которых находятся в $S(m - 1 - a)$.

Результат этого упражнения можно сформулировать более интуитивным образом: при генерации n-мерных стандартных множеств S_1, S_2, $\ldots$, $(n-1)$-мерные стандартные множества на всех уровнях становятся размахами друг друга после добавления каждой точки к уровню $m - 1$. Аналогично они становятся ядрами друг друга перед добавлением каждой точки к уровню 0.

93. (а) Предположим, что параметрами являются корректно отсортированные $2 \leq m_1' \leq m_2' \leq \cdots \leq m_n'$, и пусть k — наименьшее значение, для которого $m_k \neq m_k'$. Тогда получим $N = 1 + \operatorname{rank}(0, \ldots, 0, m_k' - 1, 0, \ldots, 0)$. (Мы должны считать, что $\min(m_1, \ldots, m_n) \geq 2$, поскольку параметры, равные 1, могут быть расположены где угодно.)

(б) Только в доказательстве для $n = 2$, скрытом в ответе к упр. 91. По индукции это доказательство распространяется и на случаи бóльших n.

94. Дополнение сохраняет лексикографический порядок и изменяет ϱ на ∂.

95. Для теоремы K положим $d = n - 1$ и $s_0 = \cdots = s_d = 1$. Для теоремы M положим $d = s$ и $s_0 = \cdots = s_d = t + 1$.

96. В таком представлении N является количеством t-мультисочетаний $\{s_0 \cdot 0, s_1 \cdot 1, s_2 \cdot 2, \ldots\}$, предшествующих $n_t n_{t-1} \ldots n_1$ в лексикографическом порядке, поскольку обобщенный коэффициент $\binom{S(n)}{t}$ дает количество мультисочетаний, крайний левый компонент которых $< n$.

Если мы оборвем представление, остановившись на крайнем справа ненулевом члене $\binom{S(n_v)}{v}$, то получим красивое обобщение (60):

$$|\partial P_{Nt}| = \binom{S(n_t)}{t-1} + \binom{S(n_{t-1})}{t-2} + \cdots + \binom{S(n_v)}{v-1}.$$

[См. G. F. Clements, *J. Combinatorial Theory* **A37** (1984), 91–97. Неравенства $s_0 \geq s_1 \geq \cdots \geq s_d$ необходимы для корректности следствия C, но не для вычисления $|\partial P_{Nt}|$. Некоторые члены $\binom{S(n_k)}{k}$ для $t \geq k > v$ могут быть нулевыми. Например, при $N = 1$, $t = 4$, $s_0 = 3$ и $s_1 = 2$ получим $N = \binom{S(1)}{4} + \binom{S(1)}{3} = 0 + 1$.]

97. (а) Тетраэдр имеет четыре вершины, шесть ребер и четыре грани: $(N_0, \ldots, N_4) = (1, 4, 6, 4, 1)$. Аналогично для октаэдра $(N_0, \ldots, N_6) = (1, 6, 8, 8, 0, 0, 0)$, а для икосаэдра $(N_0, \ldots, N_{12}) = (1, 12, 30, 20, 0, \ldots, 0)$. Гексаэдр, он же куб, имеет восемь вершин, двенадцать ребер и шесть квадратных граней; смещение разбивает каждый квадрат на два треугольника и добавляет новые ребра, так что мы получаем $(N_0, \ldots, N_8) = (1, 8, 18, 12, 0, \ldots, 0)$. Наконец смещение вершин, образующих пятиугольные грани додекаэдра, дает $(N_0, \ldots, N_{20}) = (1, 20, 54, 36, 0, \ldots, 0)$.

(б) $\{210, 310\} \cup \{10, 20, 21, 30, 31\} \cup \{0, 1, 2, 3\} \cup \{\epsilon\}$.

(в) $0 \leq N_t \leq \binom{n}{t}$ для $0 \leq t \leq n$ и $N_{t-1} \geq \kappa_t N_t$ для $1 \leq t \leq n$. Второе условие эквивалентно $\lambda_{t-1} N_{t-1} \geq N_t$ для $1 \leq t \leq n$, если мы определим $\lambda_0 1 = \infty$. Эти условия необходимы для теоремы K и достаточны, если $A = \bigcup P_{N_t t}$.

(г) Дополнения элементов, не входящих в симплициальный комплекс, а именно множества $\{\{0, \ldots, n-1\} \setminus \alpha \mid \alpha \notin C\}$, образуют симплициальный комплекс. (Можно также убедиться в выполнении необходимого и достаточного условия: $N_{t-1} \geq \kappa_t N_t \iff \lambda_{t-1} N_{t-1} \geq N_t \iff \kappa_{n-t+1} \overline{N}_{n-t+1} \leq \overline{N}_{n-t}$, поскольку $\kappa_{n-t} \overline{N}_{n-t+1} = \binom{n}{t} - \lambda_{t-1} N_{t-1}$ согласно упр. 94.)

(д) 00000 $\leftrightarrow$ 14641; 10000 $\leftrightarrow$ 14640; 11000 $\leftrightarrow$ 14630; 12000 $\leftrightarrow$ 14620; 13000 $\leftrightarrow$ 14610; 14000 $\leftrightarrow$ 14600; 12100 $\leftrightarrow$ 14520; 13100 $\leftrightarrow$ 14510; 14100 $\leftrightarrow$ 14500; 13200 $\leftrightarrow$ 14410; 14200 $\leftrightarrow$ 14400; 13300 $\leftrightarrow$ 14310; самодуальными являются 14300 и 13310.

98. Приведенная далее процедура предложена С. Линуссоном (S. Linusson) [*Combinatorica* **19** (1999), 255–266], который рассматривал более общую задачу для мультимножеств, и является существенно более быстрой, чем другие очевидные подходы к решению данной задачи. Пусть $L(n, h, l)$ — количество допустимых векторов с $N_t = \binom{n}{t}$ при $0 \leq t \leq l$, $N_{t+1} < \binom{n}{t+1}$ и $N_t = 0$ при $t > h$. Тогда $L(n, h, l) = 0$ для всех значений, за исключением $-1 \leq l \leq h \leq n$; кроме того, $L(n, h, h) = L(n, h, -1) = 1$ и $L(n, n, l) = L(n, n-1, l)$ для $l < n$. Когда $n > h \geq l \geq 0$, можно вычислить $L(n, h, l) = \sum_{j=l}^{h} L(n-1, h, j) L(n-1, j-1, l-1)$; данное рекуррентное соотношение следует из теоремы K. (Каждый вектор размеров соответствует комплексу $\bigcup P_{N_t t}$, причем $L(n-1, h, j)$ представляет сочетания, которые не содержат максимальный элемент $n-1$, а $L(n-1, j-1, l-1)$ — сочетания, которые его содержат.) Окончательное общее количество равно $L(n) = \sum_{l=1}^{n} L(n, n, l)$.

Вычисления дают $L(0)$, $L(1)$, $L(2)$, $\ldots$ = 2, 3, 5, 10, 26, 96, 553, 5461, 100709, 3718354, 289725509, $\ldots$; $L(100) \approx 3.2299 \times 10^{1842}$.

99. Максимальные элементы симплициального комплекса образуют клаттер; и обратно: сочетания, содержащиеся в элементах клаттера, образуют симплициальный комплекс. Таким образом, эти две концепции, по сути, эквивалентны.

(а) Если $(M_0, M_1, \ldots, M_n)$ — вектор размеров клаттера, то $(N_0, N_1, \ldots, N_n)$ представляет собой вектор размеров симплициального комплекса, если $N_n = M_n$ и $N_t = M_t +$

$\kappa_{t+1}N_{t+1}$ для $0 \le t < n$. И обратно: каждый такой вектор $(N_0, \ldots, N_n)$ дает $(M_0, \ldots, M_n)$, если использовать лексикографически первые N_t t-сочетаний. [Д. Ф. Клементс (G. F. Clements) распространил этот результат на мультимножества в своей работе *Discrete Math.* **4** (1973), 123–128.]

(б) В порядке ответа к упр. 97, (д) это 00000, 00001, 10000, 00040, 01000, 00030, 02000, 00120, 03000, 00310, 04000, 00600, 00100, 00020, 01100, 00210, 02100, 00500, 00200, 00110, 01200, 00400, 00300, 01010, 01300, 00010. Обратите внимание, что $(M_0, \ldots, M_n)$ является допустимым тогда и только тогда, когда допусти́м $(M_n, \ldots, M_0)$, так что в данной интерпретации мы сталкиваемся с дуальностью другого вида.

100. Представим A в виде подмножества $T(m_1, \ldots, m_n)$, как в доказательстве следствия C. Тогда максимальное значение νA достигается, когда A состоит из N лексикографически наименьших точек $x_1 \ldots x_n$.

Доказательство начинается с приведения к случаю сжатого A в том смысле, что его t-мультисочетания представляют собой $P_{|A \cap T_t|t}$ для каждого t. Тогда, если y — наибольший элемент $\in A$ и если x — наименьший элемент $\notin A$, можно доказать, что из $x < y$ вытекает $\nu x > \nu y$; следовательно, $\nu(A \setminus y \cup x) > \nu A$. Если $\nu x = \nu y - k$, то можно найти элемент $\partial^k y$, больший x, что приводит к противоречию с предположением о сжатости A.

101. (а) В общем случае $F(p) = N_0 p^n + N_1 p^{n-1}(1-p) + \cdots + N_n(1-p)^n$, если $f(x_1, \ldots, x_n)$ равна 1 для N_t бинарных строк $x_1 \ldots x_n$ с весом t. Таким образом, $G(p) = p^4 + 3p^3(1-p) + p^2(1-p)^2$; $H(p) = p^4 + p^3(1-p) + p^2(1-p)^2$.

(б) Монотонная формула f при соответствии $f(x_1, \ldots, x_n) = 1 \iff \{j-1 \mid x_j = 0\} \in C$ эквивалентна симплициальному комплексу C. Следовательно, функции $f(p)$ среди монотонных булевых функций те, которые удовлетворяют условию из упр. 97, (в), и мы получаем подходящую функцию путем выбора лексикографически последних N_{n-t} t-сочетаний (которые являются дополнениями первых N_s s-сочетаний): $\{3210\}$, $\{321, 320, 310\}$, $\{32\}$ дают $f(w,x,y,z) = wxyz \vee xyz \vee wyz \vee wxz \vee yz = wxz \vee yz$.

М. П. Шютценбергер (M. P. Schützenberger) заметил, что можно легко найти параметры N_t из $f(p)$, если обратить внимание на то, что $f(1/(1+u)) = (N_0 + N_1 u + \cdots + N_n u^n)/(1+u)^n$. Можно показать, что $H(p)$ не эквивалентна монотонной формуле при любом количестве переменных, поскольку из $(1+u+u^2)/(1+u)^4 = (N_0 + N_1 u + \cdots + N_n u^n)/(1+u)^n$ вытекает, что $N_1 = n-3$, $N_2 = \binom{n-3}{2} + 1$ и $\kappa_2 N_2 = n-2$.

Однако ответить на поставленный вопрос в общем случае не так просто. Например, функция $(1+5u+5u^2+5u^3)/(1+u)^5$ не соответствует ни одной монотонной формуле с пятью переменными, поскольку $\kappa_3 5 = 7$; однако она равна $(1+6u+10u^2+10u^3+5u^4)/(1+u)^6$, что дает решение с шестью переменными.

102. (а) Выберем N_t линейно независимых полиномов степени t из I; лексикографически упорядочим их члены и возьмем линейные комбинации так, чтобы лексикографически наименьшие члены были различными одночленами. Составим I' из всех произведений этих одночленов.

(б) Каждый одночлен степени t в I', по сути, представляет собой t-мультисочетание; например, $x_1^3 x_2 x_5^4$ соответствует 55552111. Если M_t — множество независимых одночленов степени t, свойство идеала эквивалентно тому, что $M_{t+1} \supseteq \varrho M_t$.

В данном примере $M_3 = \{x_0 x_1^2\}$; $M_4 = \varrho M_3 \cup \{x_0 x_1 x_2^2\}$; $M_5 = \varrho M_4 \cup \{x_1 x_2^4\}$, поскольку $x_2^2(x_0 x_1^2 - 2x_1 x_2^2) - x_1(x_0 x_1 x_2^2) = -2x_1 x_2^4$; соответственно, $M_{t+1} = \varrho M_t$.

(в) По теореме M можно считать, что $M_t = \widehat{Q}_{Mst}$. Пусть $N_t = \binom{n_{ts}}{s} + \cdots + \binom{n_{t2}}{2} + \binom{n_{t1}}{1}$, где $s + t \ge n_{ts} > \cdots > n_{t2} > n_{t1} \ge 0$; в этом случае $n_{ts} = s + t$ тогда и только тогда, когда $n_{t(s-1)} = s-2$, $\ldots$, $n_{t1} = 0$. Кроме того,

$$N_{t+1} \ge N_t + \kappa_s N_t = \binom{n_{ts} + [n_{ts} \ge s]}{s} + \cdots + \binom{n_{t2} + [n_{t2} \ge 2]}{2} + \binom{n_{t1} + [n_{t1} \ge 1]}{1}.$$

Таким образом, последовательность $(n_{ts} - t - \infty[n_{ts} < s], \ldots, n_{t2} - t - \infty[n_{t2} < 2], n_{t1} - t - \infty[n_{t1} < 1])$ лексикографически неубывающая при возрастании t, где '$-\infty$' вставляется в компоненты, у которых $n_{tj} = j - 1$. Такая последовательность не может увеличиваться бесконечно много раз без того, чтобы не превысить максимальное значение $(s, -\infty, \ldots, -\infty)$ согласно упр. 1.2.1–15, (г).

103. Пусть P_{Nst} — первые N элементов последовательности, определяемой следующим образом: для каждой бинарной строки $x = x_{s+t-1} \ldots x_0$ (в лексикографическом порядке) записываем $\binom{\nu x}{t}$ подкубов путем замены t единиц звездочками всеми возможными способами в лексикографическом порядке (считая $1 < *$). Например, если $x = 0101101$ и $t = 2$, мы генерируем подкубы 0101∗0∗, 010∗10∗, 010∗∗01, 0∗0110∗, 0∗01∗01, 0∗0∗101.

[См. B. Lindström, *Arkiv för Mat.* **8** (1971), 245–257; обобщение, аналогичное следствию C, появляется в K. Engel, *Sperner Theory* (Cambridge Univ. Press, 1997), Theorem 8.1.1.]

104. Первые N строк в перекрестном порядке обладают искомым свойством. [T. N. Danh and D. E. Daykin, *J. London Math. Soc.* (2) **55** (1997), 417–426.]

Примечание. Начав с наблюдения, что "1-тень" N лексикографически первых строк веса t (т. е. строк, полученных удалением только единичных битов) состоит из первых $\mu_t N$ строк веса t, Р. Альсведе (R. Ahlswede) и Н. Кай (N. Cai) распространили теорему Дана-Дейкина на вставку, удаление и/или перемещение битов [*Combinatorica* **17** (1997), 11–29; *Applied Math. Letters* **11**, 5 (1998), 121–126]. Уве Лек (Uwe Leck) доказал, что не имеется глобального упорядочения *тернарных строк* с аналогичным свойством минимальных теней [Preprint 98/6 (Univ. Rostock, 1998), 6 p.].

105. В цикле каждое число должно встречаться одинаковое количество раз, т. е. $\binom{n-1}{t-1}$ должно быть кратно t. Это необходимое условие, похоже, является одновременно и достаточным, если n не слишком мало́ по сравнению с t. Однако это пока что невозможно доказать. [См. Chung, Graham, and Diaconis, *Discrete Math.* **110** (1992), 55–57.]

В нескольких следующих упражнениях рассматриваются случаи $t = 2$ и $t = 3$, для которых известны элегантные решения. Аналогичные, но более сложные результаты были выведены для $t = 4$ и $t = 5$; случай $t = 6$ решен частично. В настоящее время случай $(n, t) = (12, 6)$ — минимальный, для которого неизвестно наличие универсального цикла.

106. Пусть разности последовательных элементов по модулю $(2m+1)$ равны 1, 2, …, m, 1, 2, …, m, …, повторенным $2m+1$ раз; например, цикл для $m = 3$ представляет собой (013602561450346235124). Этот способ работает потому, что $1 + \cdots + m = \binom{m+1}{2}$ является взаимно простым с $2m+1$. [*J. École Polytechnique* **4**, Cahier 10 (1810), 16–48.]

107. Существует 3^7 способов, которыми семь дублей ■■, ⚀⚀, …, ⚅⚅ могут быть вставлены в любой универсальный цикл 2-сочетаний множества $\{0, 1, 2, 3, 4, 5, 6\}$. Количество таких универсальных циклов равно количеству цепей Эйлера в полном графе K_7, что, как можно показать, составляет 129 976 320, если рассматривать цикл $(a_0 a_1 \ldots a_{20})$ как эквивалентный циклу $(a_1 \ldots a_{20} a_0)$, но не обратному циклу $(a_{20} \ldots a_1 a_0)$. Таким образом, окончательный ответ — 284 258 211 840.

[Эта задача впервые решена в 1859 году М. Рейссом (M. Reiss), метод которого был так сложен, что полученный им результат вызывал сомнения; см. *Nouvelles Annales de Mathématiques* **8** (1849), 74; **11** (1852), 115; *Annali di Matematica Pura ed Applicata* (2) **5** (1871–1873), 63–120. Существенно более простое решение, подтверждающее результат Рейсса, было найдено Ф. Жоливальдом (P. Jolivald) и Г. Тарри (G. Tarry), которые также подсчитали количество цепей Эйлера в K_9; см. *Comptes Rendus Association Francaise pour l'Avancement des Sciences* **15**, part 2 (1886), 49–53; É. Lucas, *Récréations Mathématiques* **4** (1894), 123–151. Брендон Мак-Кей (Brendan McKay) и Роберт Робинсон (Robert Robinson)

нашли еще лучший подход, позволивший им продолжить подсчет вплоть до K_{21} с использованием того факта, что количество путей равно

$$(m-1)!^{2m+1}\ [z_0^{2m} z_1^{2m-2} \ldots z_{2m}^{2m-2}]\ \det(a_{jk}) \prod_{1\le j<k\le 2m} (z_j^2 + z_k^2),$$

где $a_{jk} = -1/(z_j^2 + z_k^2)$ при $j \ne k$; $a_{jj} = -1/(2z_j^2) + \sum_{0\le k\le 2m} 1/(z_j^2 + z_k^2)$; см. *Combinatorics, Probability, and Computing* **7** (1998), 437–449.]

К. Фли Сан-Мари (C. Flye Sainte-Marie) в L'*Intermédiaire des Mathématiciens* **1** (1894), 164–165, заметила, что цепи Эйлера в K_7 включают 2×720 путей с 7-кратной симметрией относительно перестановок $\{0, 1, \ldots, 6\}$ (цикл Пуансо и обратный к нему), 32×1680 с 3-кратной симметрией и 25778×5040 асимметричных циклов.

108. При $n < 7$ решение невозможно, за исключением тривиального случая $n = 4$. При $n = 7$ имеется $12\,255\,208 \times 7!$ универсальных циклов, не считая эквивалентными циклы $(a_0 a_1 \ldots a_{34})$ и $(a_1 \ldots a_{34} a_0)$ и включая случаи 5-кратной симметрии наподобие цикла из упр. 105.

При $n \ge 8$ к решению задачи можно подойти систематическим путем, предложенным Б. Джексоном (B. Jackson) в *Discrete Math.* **117** (1993), 141–150; см. также G. Hurlbert, *SIAM J. Disc. Math.* **7** (1994), 598–604. Поместим каждое 3-сочетание в "стандартном циклическом порядке" $c_1c_2c_3$, где $c_2 = (c_1 + \delta) \bmod n$, $c_3 = (c_2 + \delta') \bmod n$, $0 < \delta, \delta' < n/2$, и либо $\delta = \delta'$, либо $\max(\delta, \delta') < n - \delta - \delta' \ne (n-1)/2$, либо ($1 < \delta < n/4$ и $\delta' = (n-1)/2$), либо ($\delta = (n-1)/2$ и $1 < \delta' < n/4$). Например, при $n = 8$ допустимыми значениями (δ, δ') являются $(1,1)$, $(1,2)$, $(1,3)$, $(2,1)$, $(2,2)$, $(3,1)$, $(3,3)$; при $n = 11$ это $(1,1)$, $(1,2)$, $(1,3)$, $(1,4)$, $(2,1)$, $(2,2)$, $(2,3)$, $(2,5)$, $(3,1)$, $(3,2)$, $(3,3)$, $(4,1)$, $(4,4)$, $(5,2)$, $(5,5)$. Затем строим ориентированный граф с вершинами (c, δ) для $0 \le c < n$ и $1 \le \delta < n/2$ и с дугами $(c_1, \delta) \to (c_2, \delta')$ для каждого сочетания $c_1c_2c_3$ в стандартном циклическом порядке. Этот ориентированный граф связный и сбалансированный, так что по теореме 2.3.4.2D он имеет цепь Эйлера. (Наличие в правилах $(n-1)/2$ делает ориентированный граф связным при нечетном n. Можно выбрать цепь Эйлера с n-кратной симметрией для $n = 8$, но не для $n = 12$.)

109. При $n = 1$ цикл (000) тривиален; при $n = 2$ цикла нет; имеется ровно два таких цикла при $n = 4$, а именно (00011122233302021313) и (00011120203332221313). При рассмотрении $n \ge 5$ примем, что мультисочетание $d_1d_2d_3$ находится в стандартном циклическом порядке, если $d_2 = (d_1 + \delta - 1) \bmod n$, $d_3 = (d_2 + \delta' - 1) \bmod n$, и (δ, δ') является допустимым для $n + 3$ в смысле предыдущего упражнения. Построим ориентированный граф с вершинами (d, δ) для $0 \le d < n$ и $1 \le \delta < (n+3)/2$ и с дугами $(d_1, \delta) \to (d_2, \delta')$ для каждого мультисочетания $d_1d_2d_3$ в стандартном циклическом порядке; затем найдем цепь Эйлера.

Вероятно, универсальный цикл t-мультисочетаний существует для $\{0, 1, \ldots, n-1\}$ тогда и только тогда, когда существует универсальный цикл t-сочетаний для $\{0, 1, \ldots, n + t - 1\}$.

110. Для проверки троп можно воспользоваться хорошим способом, который состоит в вычислении чисел $b(S) = \sum\{2^{p(c)} \mid c \in S\}$, где $(p(\mathtt{A}), \ldots, p(\mathtt{K})) = (1, \ldots, 13)$, с последующей установкой $l \leftarrow b(S)\ \&\ -b(S)$ и проверкой, что $b(S) + l = l \ll s$, а также что $((l \ll s) \mid (l \gg 1))\ \&\ a = 0$, где $a = 2^{p(c_1)} \mid \cdots \mid 2^{p(c_5)}$. Значения $b(S)$ и $\sum\{v(c) \mid c \in S\}$ легко поддерживаются при проходе S по всему 31 непустому подмножеству в порядке кода Грея. Ответами к упражнению для $x = (0, \ldots, 29)$ являются (1009008, 99792, 2813796, 505008, 2855676, 697508, 1800268, 751324, 1137236, 361224, 388740, 51680, 317340, 19656, 90100, 9168, 58248, 11196, 2708, 0, 8068, 2496, 444, 356, 3680, 0, 0, 0, 76, 4); таким образом, средний счет ≈ 4.769, а дисперсия ≈ 9.768.

Карты с нулевым счетом иногда в шутку называют "девятнадцать", поскольку такой счет невозможно получить ни при каком наборе карт.

— Д. Г. ДЭВИДСОН (G. H. DAVIDSON), *Справочник по игре в Крибедж (Dee's Hand-Book of Cribbage) (1839)*

Примечание. В ограниченном варианте криббеджа ("crib") не допускается "масть" из четырех карт. В этом случае распределение вычисляется немного проще и получается следующий ответ: (1022208, 99792, 2839800, 508908, 2868960, 703496, 1787176, 755320, 1118336, 358368, 378240, 43880, 310956, 16548, 88132, 9072, 57288, 11196, 2264, 0, 7828, 2472, 444, 356, 3680, 0, 0, 0, 76, 4); среднее значение и дисперсия при этом приблизительно равны 4.735 и 9.667.

111. $\partial^{n-2r}B$ представляет собой множество всех r-подмножеств B; эти подмножества не должны содержаться в A. Если $|A| = |B| = \binom{x}{n-r}$ для некоторого действительного $x > n-1$, то согласно упр. 80 должно выполняться $\binom{n}{r} \ge |A| + |\partial^{n-2r}B| \ge \binom{x}{n-r} + \binom{x}{r} > \binom{n-1}{n-r} + \binom{n-1}{r} = \binom{n}{r}$. [См. *Quart. J. Math. Oxford* **12** (1961), 313–320.]

РАЗДЕЛ 7.2.1.4

1.

m^n	$m^{\underline{n}}$	$m!\left\{{n \atop m}\right\}$
$\binom{m+n-1}{n}$	$\binom{m}{n}$	$\binom{n-1}{n-m}$
$\left\{{n \atop 0}\right\} + \cdots + \left\{{n \atop m}\right\}$	$[m \ge n]$	$\left\{{n \atop m}\right\}$
$\left\vert{m+n \atop m}\right\vert$	$[m \ge n]$	$\left\vert{n \atop m}\right\vert$

2. В общем случае для произвольных целых чисел $x_1 \ge \cdots \ge x_m$ мы получаем все целочисленные m-кортежи $a_1 \ldots a_m$, такие, что $a_1 \ge \cdots \ge a_m$, $a_1 + \cdots + a_m = x_1 + \cdots + x_m$ и $a_m \ldots a_1 \ge x_m \ldots x_1$, путем инициализации $a_1 \ldots a_m \leftarrow x_1 \ldots x_m$ и $a_{m+1} \leftarrow x_m - 2$. В частности, если c — произвольная целая константа, мы получим все целочисленные m-кортежи, такие, что $a_1 \ge \cdots \ge a_m \ge c$ и $a_1 + \cdots + a_m = n$, путем инициализации $a_1 \leftarrow n - mc + c$, $a_j \leftarrow c$ для $1 < j \le m$ и $a_{m+1} \leftarrow c - 2$, в предположении, что $n \ge cm$.

3. $a_j = \lfloor (n+m-j)/m \rfloor = \lceil (n+1-j)/m \rceil$ для $1 \le j \le m$; см. *CMath* §3.4.

4. Пусть $1 \le r \le n$. У нас должно выполняться $a_m \ge a_1 - 1$; следовательно, $a_j = \lfloor (n+m-j)/m \rfloor$ при $1 \le j \le m$, где m — наибольшее целое, для которого $\lfloor n/m \rfloor \ge r$, а именно $m = \lfloor n/r \rfloor$.

5. [См. Eugene M. Klimko, *BIT* **13** (1973), 38–49.]

C1. [Инициализация.] Установить $c_0 \leftarrow 1$, $c_1 \leftarrow n$, $c_2 \ldots c_n \leftarrow 0 \ldots 0$, $l_0 \leftarrow 1$, $l_1 \leftarrow 0$. (Считаем, что $n > 0$.)

C2. [Посещение.] Посетить разбиение, представленное количеством частей $c_1 \ldots c_n$ и связями $l_0 l_1 \ldots l_n$.

C3. [Ветвление.] Установить $j \leftarrow l_0$ и $k \leftarrow l_j$. Если $c_j = 1$, перейти к шагу C6; в противном случае, если $j > 1$, перейти к шагу C5.

C4. [Замена 1+1 на 2.] Установить $c_1 \leftarrow c_1 - 2$, $c_2 \leftarrow c_2 + 1$ и $l_{[c_1>0]} \leftarrow 2$. Если $k \ne 2$, установить также $l_2 \leftarrow k$. Вернуться к шагу C2.

C5. [Замена $j \cdot c_j$ на $(j+1) + 1 + \cdots + 1$.] Установить $c_1 \leftarrow j(c_j - 1) - 1$ и перейти к шагу C7.

C6. [Замена $k \cdot c_k + j$ на $(k+1) + 1 + \cdots + 1$.] Завершить работу алгоритма, если $k = 0$. В противном случае установить $c_j \leftarrow 0$; затем установить $c_1 \leftarrow k(c_k - 1) + j - 1$, $j \leftarrow k$ и $k \leftarrow l_k$.

C7. [Обновление связей.] Если $c_1 > 0$, установить $l_0 \leftarrow 1$, $l_1 \leftarrow j+1$; в противном случае установить $l_0 \leftarrow j+1$. Затем установить $c_j \leftarrow 0$ и $c_{j+1} \leftarrow c_{j+1}+1$. Если $k \neq j+1$, установить $l_{j+1} \leftarrow k$. Вернуться к шагу C2. ▌

Заметим, что этот алгоритм — *без циклов*, но он не быстрее алгоритма Р. Шаги C4, C5 и C6 выполняются $p(n-2)$, $2p(n)-p(n+1)-p(n-2)$ и $p(n+1)-p(n)$ раз соответственно. Таким образом, шаг C4 оказывается наиболее важным при больших n. (См. упр. 45 и детальный анализ Феннера (Fenner) и Лоизоу (Loizou) в *Acta Inf.* **16** (1981), 237–252.)

6. Считаем, что за каждым разбиением следует 0. Установить $j \leftarrow 0$, $k \leftarrow a_1$ и $b_{k+1} \leftarrow 0$. Затем, пока $k > 0$, устанавливать $j \leftarrow j+1$ и, пока $k > a_{j+1}$, устанавливать $b_k \leftarrow j$ и $k \leftarrow k-1$. (Мы использовали (11) в дуальной форме $a_j - a_{j+1} = d_j$, где $d_1 \ldots d_n$ — представление в виде количества частей $b_1 b_2 \ldots$. Этот алгоритм, по сути, проходит по краю диаграммы Ферररса; таким образом, время работы алгоритма грубо пропорционально $a_1 + b_1$, сумме количества частей во входных и выходных данных.)

7. Из дуальности (11) имеем $b_1 \ldots b_n = n^{a_n}(n-1)^{a_{n-1}-a_n} \ldots 1^{a_1-a_2} 0^{n-a_1}$.

8. Транспонирование диаграммы Ферररса соответствует зеркальному отражению и дополнению битовой строки (15). Так что можно просто поменять местами и обратить p и q, получив разбиение $(a_1 a_2 \ldots)^T = (q_t + \cdots + q_1)^{p_1}(q_t + \cdots + q_2)^{p_2} \ldots (q_t)^{p_t}$.

9. Доказательство по индукции: если $a_k = l-1$ и $b_l = k-1$, увеличение a_k и b_l сохраняет равенство.

10. (а) В первом дереве левый дочерний узел каждого узла получается путем добавления '1'. Правый дочерний узел получается путем увеличения крайней справа цифры; этот дочерний узел существует тогда и только тогда, когда родительский узел заканчивается разными цифрами. Все разбиения n на уровне n находятся в лексикографическом порядке.

(б) Во втором дереве левый дочерний узел получается путем замены '11' на '2'. Этот дочерний узел существует тогда и только тогда, когда родительский узел содержит по крайней мере две единицы. Правый дочерний узел получается путем удаления 1 и увеличения наименьшей части, превосходящей 1. Этот дочерний узел существует тогда и только тогда, когда в родительском узле имеется по меньшей мере одна единица и наименьшая превосходящая единицу часть ровно одна. Все разбиения n на m частей находятся на уровне $n-m$ в лексикографическом порядке; прямой порядок обхода всего дерева дает лексикографический порядок всех узлов. [T. I. Fenner and G. Loizou, *Comp. J.* **23** (1980), 332–337.]

11. $[z^{100}]\,1/((1-z)(1-z^2)(1-z^5)(1-z^{10})(1-z^{20})(1-z^{50})(1-z^{100})) = 4563$ и $[z^{100}](1+z+z^2)(1+z^2+z^4)\ldots(1+z^{100}+z^{200}) = 7$. [См. G. Pólya, *AMM* **63** (1956), 689–697.] В бесконечном произведении $\prod_{k\geq 0}\prod_{r\in\{10^k,2\cdot 10^k,5\cdot 10^k\}}(1+z^r+z^{2r})$ коэффициент при z^{10^n} равен $2^{n+1}-1$, а коэффициент при z^{10^n-1} равен 2^n.

Примечание переводчика. Для российского набора монет (1, 5, 10, 50 копеек и рубль) соответствующие количества способов — 159 и 2, украинского (1, 2, 5, 10, 25, 50 копеек и гривна) — 3954 и 6, советского (1, 2, 3, 5, 10, 15, 20, 50 копеек и рубль) — 66 949 и 49.

12. Для доказательства того, что $(1+z)(1+z^2)(1+z^3)\ldots = 1/((1-z)(1-z^3)(1-z^5)\ldots)$, запишем левую часть в виде

$$\frac{(1-z^2)}{(1-z)}\frac{(1-z^4)}{(1-z^2)}\frac{(1-z^6)}{(1-z^3)}\cdots$$

и сократим общие множители в числителе и знаменателе. Другой способ состоит в замене z на z^1, z^3, z^5, ... в тождестве $(1+z)(1+z^2)(1+z^4)(1+z^8)\ldots = 1/(1-z)$ и перемножении получившихся результатов. [*Novi Comment. Acad. Sci. Pet.* **3** (1750), 125–169, §47.]

13. Отобразим разбиение $c_1{\cdot}1 + c_2{\cdot}2 + c_3{\cdot}3 + \cdots$ на $r_1{\cdot}1 + \lfloor c_1/2\rfloor{\cdot}2 + r_3{\cdot}3 + \lfloor c_2/2\rfloor{\cdot}4 + r_5{\cdot}5 + \lfloor c_3/2\rfloor{\cdot}6 + \cdots$, где $r_m = (c_m \bmod 2) + 2(c_{2m} \bmod 2) + 4(c_{4m} \bmod 2) + 8(c_{8m} \bmod 2) + \cdots$; $433222211 \mapsto 64421111 \mapsto 8332211$. [*Johns Hopkins Univ. Circular* **2** (1882), 72.]

14. Соответствие Сильвестера легче всего понять, если представить его в виде диаграммы, в которой точки нечетных частей отцентрованы, а разбиения разделены на непересекающиеся "крюки". Например, разбиение $17+15+15+9+9+9+9+5+5+3+3$, имеющее пять различных нечетных частей, посредством диаграммы

ставится в соответствие разбиению с разными частями $19+18+16+13+12+9+5+4+3$ с четырьмя просветами.

Пусть в общем случае, когда "прямоугольник Дюрфи" (показанный на диаграмме) имеет t строк, справа от него имеется $a_1, \ldots, a_t$ дополнительных точек, а снизу — $b_t, \ldots, b_1$, $\ldots, b_t$ точек, где $a_1 \ge \cdots \ge a_t \ge 0$ и $b_1 \ge \cdots \ge b_t \ge 0$. Тогда различные части получаются как $2t-1+a_1+b_1$, $2t-2+a_1+b_2$, $\ldots$, $2+a_{t-1}+b_t$, $1+a_t+b_t$ и (если она не равна нулю) $0+a_t$. И наоборот, любое разбиение с $2t$ различными неотрицательными частями может быть записано в этом виде единственным образом.

Так, разбиения с нечетными частями для $n = 10$ представляют собой $9+1$, $7+3$, $7+1+1+1$, $5+5$, $5+3+1+1$, $5+1+1+1+1+1$, $3+3+3+1$, $3+3+1+1+1+1$, $3+1+\cdots+1$, $1+\cdots+1$, а соответствующие разбиения с разными частями — $6+4$, $5+4+1$, $7+3$, $4+3+2+1$, $6+3+1$, $8+2$, $5+3+2$, $7+2+1$, $9+1$, 10. [См. замечательную статью Сильвестера в *Amer. J. Math.* **5** (1882), 251–330; **6** (1883), 334–336.]

15. Каждое самосопряженное разбиение следа k соответствует разбиению n на k различных нечетных частей ("крюков"). Следовательно, можно записать производящую функцию либо как произведение $(1+z)(1+z^3)(1+z^5)\ldots$, либо как сумму $1 + z^1/(1{-}z^2) + z^4/((1{-}z^2)(1{-}z^4)) + z^9/((1{-}z^2)(1{-}z^4)(1{-}z^6)) + \cdots$. [*Johns Hopkins Univ. Circular* **3** (1883), 42–43.]

16. Квадрат Дюрфи содержит k^2 точек, а остающиеся точки соответствуют двум независимым разбиениям с наибольшей частью $\le k$. Таким образом, обозначая через w количество частей, а через z — количество точек, находим, что

$$\prod_{m=1}^{\infty} \frac{1}{1-wz^m} = \sum_{k=0}^{\infty} \frac{w^k z^{k^2}}{(1-z)(1-z^2)\ldots(1-z^k)(1-wz)(1-wz^2)\ldots(1-wz^k)}.$$

[Эта впечатляюще выглядящая формула оказывается всего лишь частным случаем $x = y = 0$ еще более впечатляющей формулы из упр. 19.]

17. (а) $((1+uvz)(1+uvz^2)(1+uvz^3)\ldots)/((1-uz)(1-uz^2)(1-uz^3)\ldots)$.

(б) Объединенное разбиение можно представить в виде обобщенной диаграммы Феррерса, в которой мы объединяем все части, помещая a_i над b_j, если $a_i \ge b_j$, а затем помечая крайнюю справа точку каждого b_j. Например, объединенное разбиение $(8,8,5;\ 9,7,5,2)$ имеет приведенную здесь диаграмму, на которой помеченные точки имеют вид '✦'. Метки

появляются только у угловых точек; таким образом, транспонированная диаграмма соответствует другому объединенному разбиению, в данном случае — $(7,6,6,4,3;\ 7,6,4,1)$. [См. J. T. Joichi and D. Stanton, *Pacific J. Math.* **127** (1987), 103–120; S. Corteel and J. Lovejoy, *Trans. Amer. Math. Soc.* **356** (2004), 1623–1635; Igor Pak, *The Ramanujan Journal* **12** (2006), 5–75.]

Каждое объединенное разбиение с $t > 0$ частями соответствует, таким образом, "сопряженному" разбиению с наибольшей частью t. Производящая функция для такого объединенного разбиения — $((1+vz)\ldots(1+vz^{t-1}))/((1-z)\ldots(1-z^t))$, умноженное на $(vz^t + z^t)$, где vz^t соответствует случаю $b_1 = t$, а z^t соответствует случаю $s = 0$ или $b_1 < t$.

(в) Таким образом, мы получили вариант общей z-номиальной теоремы из ответа к упр. 1.2.6–58:

$$\frac{(1+uvz)}{(1-uz)}\frac{(1+uvz^2)}{(1-uz^2)}\frac{(1+uvz^3)}{(1-uz^3)}\cdots = \sum_{t=0}^{\infty}\frac{(1+v)}{(1-z)}\frac{(1+vz)}{(1-z^2)}\cdots\frac{(1+vz^{t-1})}{(1-z^t)}\,u^t z^t.$$

18. Уравнения очевидным образом определяют a и b для заданных c и d, так что нужно показать, что c и d однозначно определяются величинами a и b. Приведенный далее алгоритм определяет c и d справа налево.

A1. [Инициализация.] Установить $i \leftarrow r$, $j \leftarrow s$, $k \leftarrow 0$ и $a_0 \leftarrow b_0 \leftarrow \infty$.

A2. [Ветвление.] Завершить работу, если $i + j = 0$. В противном случае перейти к шагу A4, если $a_i \geq b_j - k$.

A3. [Поглощение a_i.] Установить $c_{i+j} \leftarrow a_i$, $d_{i+j} \leftarrow 0$, $i \leftarrow i-1$, $k \leftarrow k+1$ и вернуться к шагу A2.

A4. [Поглощение b_j.] Установить $c_{i+j} \leftarrow b_j - k$, $d_{i+j} \leftarrow 1$, $j \leftarrow j-1$, $k \leftarrow k+1$ и вернуться к шагу A2. ▮

Имеется также метод, работающий слева направо.

B1. [Инициализация.] Установить $i \leftarrow 1$, $j \leftarrow 1$, $k \leftarrow r+s$ и $a_{r+1} \leftarrow b_{s+1} \leftarrow -\infty$.

B2. [Ветвление.] Завершить работу, если $k = 0$. В противном случае установить $k \leftarrow k-1$, затем перейти к шагу B4, если $a_i \leq b_j - k$.

B3. [Поглощение a_i.] Установить $c_{i+j-1} \leftarrow a_i$, $d_{i+j-1} \leftarrow 0$, $i \leftarrow i+1$ и вернуться к шагу B2.

B4. [Поглощение b_j.] Установить $c_{i+j-1} \leftarrow b_j - k$, $d_{i+j-1} \leftarrow 1$, $j \leftarrow j+1$ и вернуться к шагу B2. ▮

В обоих случаях ветвление обеспечивает удовлетворение полученной последовательности условию $c_1 \geq \cdots \geq c_{r+s}$. Заметим, что $c_{r+s} = \min(a_r, b_s)$ и $c_1 = \max(a_1, b_1 - r - s + 1)$.

Таким образом, мы доказали тождество из упр. 17, (в) другим способом. Расширение этой идеи приводит к комбинаторному доказательству "замечательной формулы с многими параметрами" Рамануджана:

$$\sum_{n=-\infty}^{\infty} w^n \prod_{k=0}^{\infty}\frac{1-bz^{k+n}}{1-az^{k+n}} = \prod_{k=0}^{\infty}\frac{(1-a^{-1}bz^k)(1-a^{-1}w^{-1}z^{k+1})(1-awz^k)(1-z^{k+1})}{(1-a^{-1}bw^{-1}z^k)(1-a^{-1}z^{k+1})(1-az^k)(1-wz^k)}.$$

[*Ссылки*: G. H. Hardy, *Ramanujan* (1940), Eq. (12.12.2); D. Zeilberger, *Europ. J. Combinatorics* **8** (1987), 461–463; A. J. Yee, *J. Comb. Theory* **A105** (2004), 63–77.]

19. [*Crelle* **34** (1847), 285–328.] Согласно упр. 17, (в), упомянутая в указании сумма по k равна

$$\Bigl(\sum_{l\geq 0} v^l \frac{(z-bz)\ldots(z-bz^l)}{(1-z)\ldots(1-z^l)}\frac{(1-uz)\ldots(1-uz^l)}{(1-auz)\ldots(1-auz^l)}\Bigr)\cdot\prod_{m=1}^{\infty}\frac{1-auz^m}{1-uz^m};$$

суммирование по l аналогично, с заменой $u \leftrightarrow v$, $a \leftrightarrow b$, $k \leftrightarrow l$. Дальнейшее суммирование по k и l приводит при $b = auz$ к

$$\prod_{m=1}^{\infty} \frac{(1-uvz^{m+1})(1-auz^m)}{(1-uz^m)(1-vz^m)}.$$

Положим теперь $u = wxy$, $v = 1/(yz)$, $a = 1/x$ и $b = wyz$. Приравняйте это бесконечное произведение к сумме по l.

20. Для получения $p(n)$ требуется сложить или вычесть примерно $\sqrt{8n/3}$ ранее вычисленных значений, большинство из которых имеют длину $\Theta(\sqrt{n})$ бит. Следовательно, $p(n)$ можно вычислить за $\Theta(n)$ шагов, т. е. общее время составляет $\Theta(n^2)$.

(Непосредственное использование (17) приводит к $\Theta(n^{5/2})$ шагам.)

21. Поскольку $\sum_{n=0}^{\infty} q(n)z^n = (1+z)(1+z^2)\ldots$ равно $(1-z^2)(1-z^4)\ldots P(z) = (1-z^2-z^4+z^{10}+z^{14}-z^{24}-\cdots)P(z)$, имеем

$$q(n) = p(n) - p(n-2) - p(n-4) + p(n-10) + p(n-14) - p(n-24) - \cdots.$$

[Имеется также "чисто рекуррентное соотношение", в котором используются только значения $q(n)$, аналогичные рекуррентному соотношению для $\sigma(n)$ из следующего упражнения.]

22. Из (21) имеем $\sum_{n=1}^{\infty} \sigma(n)z^n = \sum_{m,n\geq 1} mz^{mn} = z\frac{d}{dz}\ln P(z) = (z+2z^2-5z^5-7z^7+\cdots)/(1-z-z^2+z^5+z^7+\cdots)$. [*Bibliothèque Impartiale* **3** (1751), 10–31.]

23. (Решение Марка ван Леувена (Marc van Leeuwen).) Разделим (19) на $1-v$, получив

$$\begin{aligned}\prod_{k=1}^{\infty}(1-u^kv^{k-1})(1-u^kv^k)(1-u^kv^{k+1}) &= \sum_{n=0}^{\infty}(-1)^n u^{\binom{n+1}{2}}\left(\frac{v^{\binom{n}{2}}-v^{\binom{n+2}{2}}}{1-v}\right)\\ &= \sum_{n=0}^{\infty}(-1)^n u^{\binom{n+1}{2}}\sum_{k=0}^{2n} v^k;\end{aligned}$$

теперь установим $u = z$ и $v = 1$.

[См. §57 статьи Сильвестера, на которую имеется ссылка в ответе к упр. 14. Доказательство Якоби содержится в §66 его монографии *Fundamenta Nova Theori? Functionum Ellipticarum* (1829).]

24. (а) $[z^n]\,A(z) = \sum(-1)^{j+k}(2k+1)[3j^2+j+k^2+k=2n]$ (где сумма берется по всем целым числам j и всем неотрицательным целым числам k) в соответствии с (18) и упр. 23. Если $n \bmod 5 = 4$, ненулевые члены образуются только при $j \bmod 5 = 4$ и $k \bmod 5 = 2$; но тогда $(2k+1) \bmod 5 = 0$.

(б) В соответствии с формулой 4.6.2–(5), если p — простое число, то $B(z)^p \equiv B(z^p)$ (по модулю p).

(в) Поскольку $A(z) = P(z)^{-4}$, примем $B(z) = P(z)$. [*Proc. Cambridge Philos. Soc.* **19** (1919), 207–210. Аналогично доказывается, что $p(n)$ кратно 7 при $n \bmod 7 = 5$. Рамануджан дошел до получения красивых формул $p(5n+4)/5 = [z^n]\,P(z)^6/P(z^5)^5$; $p(7n+5)/7 = [z^n]\,(P(z)^4/P(z^7)^3 + 7zP(z)^8/P(z^7)^7)$. Аткин (Atkin) и Свиннертон-Дайр (Swinnerton-Dyer) в работе *Proc. London Math. Soc.* (3) **4** (1953), 84–106, показали, что разбиения $5n+4$ и $7n+5$ могут быть разделены на классы равного размера в соответствии со значениями (наибольшая часть – число частей) mod 5 или mod 7, как предположил Ф. Дайсон (F. Dyson). Имеется немного более сложное доказательство с применением комбинаторной статистики того, что $p(n) \bmod 11 = 0$ при $n \bmod 11 = 6$; см. F. G. Garvan, *Trans. Amer. Math. Soc.* **305** (1988), 47–77.]

25. [Соотношение из указания можно доказать, дифференцируя обе части приведенного тождества. Оно представляет собой частный случай $y = 1-x$ красивой формулы, открытой Н. Х. Абелем (N. H. Abel) в 1826 году:

$$\mathrm{Li}_2(x) + \mathrm{Li}_2(y) = \mathrm{Li}_2\Bigl(\frac{x}{1-y}\Bigr) + \mathrm{Li}_2\Bigl(\frac{y}{1-x}\Bigr) - \mathrm{Li}_2\Bigl(\frac{xy}{(1-x)(1-y)}\Bigr) - \ln(1-x)\ln(1-y).$$

См. статью Абеля *?uvres Complètes* **2** (Christiania: Grondahl, 1881), 189–193.]

(а) Пусть $f(x) = \ln(1/(1-e^{-xt}))$. Тогда $\int_1^x f(x)\,dx = -\mathrm{Li}_2(e^{-tx})/t$ и $f^{(n)}(x) = (-t)^n \times e^{tx}\sum_k \bigl\langle {n-1 \atop k} \bigr\rangle e^{ktx}/(e^{tx}-1)^n$, так что формула суммирования Эйлера позволяет получить $\mathrm{Li}_2(e^{-t})/t + \frac{1}{2}\ln(1/(1-e^{-t})) + O(1) = \bigl(\zeta(2) + t\ln(1-e^{-t}) - \mathrm{Li}_2(1-e^{-t})\bigr)/t - \frac{1}{2}\ln t + O(1) = \zeta(2)/t + \frac{1}{2}\ln t + O(1)$ при $t \to 0$.

(б) Имеем $\sum_{m,n\ge1} e^{-mnt}/n = \frac{1}{2\pi i}\sum_{m,n\ge1}\int_{1-i\infty}^{1+i\infty}(mnt)^{-z}\Gamma(z)\,dz/n$; суммирование дает $\frac{1}{2\pi i}\int_{1-i\infty}^{1+i\infty}\zeta(z+1)\zeta(z)t^{-z}\Gamma(z)\,dz$. Полюс в $z=1$ дает $\zeta(2)/t$; двойной полюс в $z=0$ дает $-\zeta(0)\ln t + \zeta'(0) = \frac{1}{2}\ln t - \frac{1}{2}\ln 2\pi$; полюс в $z=-1$ дает $-\zeta(-1)\zeta(0)t = B_2B_1t = -t/24$. Нули $\zeta(z+1)\zeta(z)$ сокращают другие полюса $\Gamma(z)$, так что результатом является $\ln P(e^{-t}) = \zeta(2)/t + \frac{1}{2}\ln(t/2\pi) - t/24 + O(t^M)$ для произвольно большого M.

26. Пусть $F(n) = \sum_{k=1}^\infty e^{-k^2/n}$. Можно воспользоваться формулой (25) либо с $f(x) = e^{-x^2/n}[x>0] + \frac{1}{2}\delta_{x0}$, либо с $f(x) = e^{-x^2/n}$ для всех x, поскольку $2F(n)+1 = \sum_{k=-\infty}^\infty e^{-k^2/n}$. Выберем второй способ. Тогда правая часть (25) при $\theta = 0$ представляет собой быстро сходящийся ряд

$$\lim_{M\to\infty}\sum_{m=-M}^{M}\int_{-\infty}^{\infty} e^{-2\pi miy-y^2/n}\,dy = \sum_{m=-\infty}^{\infty} e^{-\pi^2m^2n}\int_{-\infty}^{\infty} e^{-u^2/n}\,du,$$

если выполнить подстановку $u = y+\pi mni$; значение интеграла равно $\sqrt{\pi n}$. [Этот результат представляет собой формулу (15) на с. 420 оригинальной статьи Пуассона.]

27. Начнем с подстановки $u = y+b-ci/a$ и получим $\int_{-\infty}^\infty e^{-a(y+b)^2+2ciy}\,dy = e^{-c^2/a-2bci}\int_{-\infty}^\infty e^{-au^2}\,du$. А с помощью подстановки $t = au^2$ и результатов упр. 1.2.5–20 и 1.2.6–43 получим $\int_{-\infty}^\infty e^{-au^2}\,du = \int_0^\infty e^{-t}\,dt/\sqrt{at} = \Gamma(\frac{1}{2})/\sqrt{a} = \sqrt{\pi/a}$.

Теперь (30) вытекает из (29), так как для всех целых чисел m

$$g(3m+1) + g(-3m) = \sqrt{\frac{2\pi}{t}}(-1)^m e^{-6\pi^2(m+\frac{1}{6})^2/t}; \quad g(3m+2) + g(-3m-1) = 0.$$

[См. M. I. Knopp, *Modular Functions in Analytic Number Theory* (1970), Chapter 3.]

28. (а–г) См. *Trans. Amer. Math. Soc.* **43** (1938), 271–295. В действительности Лемер нашел явные формулы для $A_{p^e}(n)$ с использованием символов Якоби из упр. 4.5.4–23:

$$A_{2^e}(n) = (-1)^e\Bigl(\frac{-1}{m}\Bigr)2^{e/2}\sin\frac{4\pi m}{2^{e+3}}, \qquad \text{если } (3m)^2 \equiv 1-24n \text{ (по модулю } 2^{e+3});$$

$$A_{3^e}(n) = (-1)^{e+1}\Bigl(\frac{m}{3}\Bigr)\frac{2}{\sqrt3}3^{e/2}\sin\frac{4\pi m}{3^{e+1}}, \qquad \text{если } (8m)^2 \equiv 1-24n \text{ (по модулю } 3^{e+1});$$

$$A_{p^e}(n) = \begin{cases} 2\Bigl(\dfrac{3}{p^e}\Bigr)p^{e/2}\cos\dfrac{4\pi m}{p^e}, & \text{если } (24m)^2 \equiv 1-24n \text{ (по модулю } p^e),\ p\ge5, \\ & \quad \text{и } 24n \bmod p \ne 1; \\ \Bigl(\dfrac{3}{p^e}\Bigr)p^{e/2}[e=1], & \text{если } 24n \bmod p = 1 \text{ и } p\ge5. \end{cases}$$

(д) Если $k = 2^a 3^b p_1^{e_1} \dots p_t^{e_t}$, где $3 < p_1 < \dots < p_t$ и $e_1 \dots e_t \neq 0$, вероятность того, что $A_k(n) \neq 0$, равна $2^{-t}(1 + (-1)^{[e_1>1]}/p_1) \dots (1 + (-1)^{[e_t>1]}/p_t)$.

29. $z_1 z_2 \dots z_m / \bigl((1 - z_1)(1 - z_1 z_2) \dots (1 - z_1 z_2 \dots z_m)\bigr)$.

30. (а) $\left|{n+1 \atop m}\right|$ и (б) $\left|{m+n \atop m}\right|$ согласно (39).

31. *Первое решение* [Marshall Hall, Jr., *Combinatorial Theory* (1967), §4.1]: непосредственно из рекуррентного соотношения (39) можно показать, что для $0 \le r < k!$ существует полином $f_{k,r}(n) = n^{k-1}/(k!(k-1)!) + O(n^{k-2})$, такой, что $\left|{n \atop k}\right| = f_{n, n \bmod k!}(n)$.

Второе решение: поскольку $(1 - z) \dots (1 - z^m) = \prod_{p \perp q} (1 - e^{2\pi i p/q} z)^{\lfloor m/q \rfloor}$, где произведение берется по всем сокращенным дробям p/q с $0 \le p < q$, коэффициент при z^n в (41) может быть выражен как сумма корней единицы, умноженных на полином степени n, а именно как $\sum_{p \perp q} e^{2\pi i p n/q} f_{p,q}(n)$, где $f_{p,q}(n)$ — полином степени, меньшей, чем $\lfloor m/q \rfloor$. Таким образом, существуют константы, такие, что $\left|{n \atop 2}\right| = a_1 n + a_2 + (-1)^n a_3$; $\left|{n \atop 3}\right| = b_1 n^2 + b_2 n + b_3 + (-1)^n b_4 + \omega^n b_5 + \omega^{-n} b_6$, где $\omega = e^{2\pi i/3}$; и т. д. Эти константы определяются значениями для малых n, и для первых двух случаев

$$\left|{n \atop 2}\right| = \frac{1}{2}n - \frac{1}{4} + \frac{1}{4}(-1)^n; \qquad \left|{n \atop 3}\right| = \frac{1}{12}n^2 - \frac{7}{72} - \frac{1}{8}(-1)^n + \frac{1}{9}\omega^n + \frac{1}{9}\omega^{-n}.$$

Отсюда следует, что $\left|{n \atop 3}\right|$ — целое число, ближайшее к $n^2/12$. Аналогично $\left|{n \atop 4}\right|$ — целое число, ближайшее к $(n^3 + 3n^2 - 9n\,[n \text{ нечетно}])/144$.

[Точные формулы для $\left|{n \atop 2}\right|$, $\left|{n \atop 3}\right|$ и $\left|{n \atop 4}\right|$, без упрощения с помощью функции "пол", были впервые найдены Д. Ф. Малфатти (G. F. Malfatti), *Memorie di Mat. e Fis. Società Italiana* **3** (1786), 571–663. У. Д. А. Колмен (W. J. A. Colman) в *Fibonacci Quarterly* **21** (1983), 272–284, показал, что $\left|{n \atop 5}\right|$ — целое число, ближайшее к $(n^4 + 10n^3 + 10n^2 - 75n - 45n(-1)^n)/2880$, и привел аналогичные формулы для $\left|{n \atop 6}\right|$ и $\left|{n \atop 7}\right|$.]

32. Поскольку $\left|{m+n \atop m}\right| \le p(n)$, причем равенство выполняется тогда и только тогда, когда $m \ge n$, имеем $\left|{n \atop m}\right| \le p(n-m)$ с равенством тогда и только тогда, когда $2m \ge n$.

33. Разбиение на m частей соответствует не более чем $m!$ композициям; следовательно, $\binom{n-1}{m-1} \le m! \left|{n \atop m}\right|$. Соответственно, $p(n) \ge (n-1)!/((n-m)!\,m!\,(m-1)!)$, и при $m = \lfloor \sqrt{n} \rfloor$ приближение Стирлинга доказывает, что $\ln p(n) \ge 2\sqrt{n} - \ln n - \frac{1}{2} - \ln 2\pi$.

34. $a_1 > a_2 > \dots > a_m > 0$ тогда и только тогда, когда $a_1 - m + 1 \ge a_2 - m + 2 \ge \dots \ge a_m \ge 1$. Разбиения на m различных частей соответствуют $m!$ композициям. Таким образом, с учетом ответа к предыдущему упражнению

$$\frac{1}{m!}\binom{n-1}{m-1} \le \left|{n \atop m}\right| \le \frac{1}{m!}\binom{n + m(m-1)/2}{m-1}.$$

[См. H. Gupta, *Proc. Indian Acad. Sci.* **A16** (1942), 101–102. Подробная асимптотическая формула для $\left|{n \atop m}\right|$ при $n = \Theta(m^3)$ имеется в упр. 3.3.2–30.]

35. (а) $x = \frac{1}{C} \ln \frac{1}{C} \approx -0.194$.

(б) $x = \frac{1}{C} \ln \frac{1}{C} - \frac{1}{C} \ln \ln 2 \approx 0.092$; в общем случае $x = \frac{1}{C}(\ln \frac{1}{C} - \ln \ln \frac{1}{F(x)})$.

(в) $\int_{-\infty}^{\infty} x\,dF(x) = \int_0^{\infty} (Cu)^{-2} (\ln u) e^{-1/(Cu)}\,du = -\frac{1}{C} \int_0^{\infty} (\ln C + \ln v) e^{-v}\,dv = (\gamma - \ln C)/C \approx 0.256$.

(г) Аналогично $\int_{-\infty}^{\infty} x^2 e^{-Cx} \exp(-e^{-Cx}/C)\,dx = (\gamma^2 + \zeta(2) - 2\gamma \ln C + (\ln C)^2)/C^2 \approx 1.0656$. Таким образом, дисперсия равна $\zeta(2)/C^2 = 1$ (в точности (!)). [Распределение вероятности $e^{-e^{(a-x)/b}}$ обычно называется распределением Фишера–Типпетта; см. *Proc. Cambridge Phil. Soc.* **24** (1928), 180–190.]

36. Сумма по всем $j_r - (m+r-1) \ge \cdots \ge j_2 - (m+1) \ge j_1 - m \ge 1$ дает

$$\begin{aligned}\Sigma_r &= \sum_t \left|{t-rm-r(r-1)/2 \atop r}\right| \frac{p(n-t)}{p(n)}\\ &= \frac{\alpha}{1-\alpha}\frac{\alpha^2}{1-\alpha^2}\cdots\frac{\alpha^r}{1-\alpha^r}\alpha^{rm}(1+O(n^{-1/2+2\epsilon}))+E\\ &= \frac{n^{-1/2}}{\alpha^{-1}-1}\frac{n^{-1/2}}{\alpha^{-2}-1}\cdots\frac{n^{-1/2}}{\alpha^{-r}-1}\exp(-Crx+O(rn^{-1/2+2\epsilon}))+E,\end{aligned}$$

где E — член ошибки, учитывающий случаи $t > n^{1/2+\epsilon}$. Старший член $n^{-1/2}/(\alpha^{-j}-1)$ равен $\frac{1}{jC}(1+O(jn^{-1/2}))$. Легко убедиться, что $E = O(n^{\log n}e^{-Cn^\epsilon})$, даже при использовании грубой верхней границы $\left|{t-rm-r(r-1)/2 \atop r}\right| \le t^r$, поскольку

$$\sum_{t\ge xN} t^r e^{-t/N} = O\Bigl(\int_{xN}^\infty t^r e^{-t/N}\,dt\Bigr) = O(N^{r+1}x^r e^{-x}/(1-r/x)),$$

где $N = \Theta(\sqrt{n})$, $x = \Theta(n^\epsilon)$, $r = O(\log n)$.

37. Такое разбиение учитывается один раз в Σ_0, q раз в Σ_1, $\binom{q}{2}$ раз в Σ_2, ...; т. е. ровно $\sum_{j=0}^r (-1)^j\binom{q}{j} = (-1)^r\binom{q-1}{r}$ раз в частичной сумме, оканчивающейся членом $(-1)^r\Sigma_r$. Это количество не превышает δ_{q0} при нечетном r и не меньше δ_{q0} при четном r. [Подобная аргументация показывает, что обобщенный принцип из упр. 1.3.3–26 также обладает указанным свойством. *Источник*: C. Bonferroni, *Pubblicazioni del Reale Istituto Superiore di Scienze Economiche e Commerciali di Firenze* **8** (1936), 3–62.]

38. $z^{l+m-1}\binom{l+m-2}{m-1}_z = z^{l+m-1}(1-z^l)\ldots(1-z^{l+m-2})/((1-z)\ldots(1-z^{m-1}))$.

39. В соответствии с упр. 1.2.6–58 $[x^m](1+zx)(1+z^2x)\ldots(1+z^{l-1}x) = z^{m(m+1)/2}\binom{l-1}{m}_z$, что равно $(z-z^l)(z^2-z^l)\ldots(z^m-z^l)/((1-z)(1-z^2)\ldots(1-z^m))$. Ответ следует из теоремы C: замена $a_1\ldots a_m$ на $(a_1-m)\ldots(a_m-1)$ дает эквивалентное разбиение $n-m(m+1)/2$ на не более чем m частей, не превышающих $l-1-m$.

40. Пусть $\alpha = a_1\ldots a_m$ — разбиение не более чем на m частей и пусть $f(\alpha)=\infty$, если $a_1 \le l$, в противном случае $f(\alpha) = \min\{j \mid a_1 > l + a_{j+1}\}$. Пусть g_k — производящая функция для разбиений с $f(\alpha) > k$. Разбиения с $f(\alpha) = k < \infty$ характеризуются неравенствами

$$a_1 \ge a_2 \ge \cdots \ge a_k \ge a_1 - l > a_{k+1} \ge \cdots \ge a_{m+1} = 0.$$

Таким образом, $a_1a_2\ldots a_m = (b_k+l+1)(b_1+1)\ldots(b_{k-1}+1)b_{k+1}\ldots b_m$, где $f(b_1\ldots b_m) \ge k$; обратное также верно. Отсюда следует, что $g_k = g_{k-1} - z^{l+k}g_{k-1}$.

[См. *American J. Math.* **5** (1882), 254–257.]

41. См. G. Almkvist and G. E. Andrews, *J. Number Theory* **38** (1991), 135–144.

42. А. Вершик (A. Vershik) [*Functional Anal. Applic.* **30** (1996), 90–105, Theorem 4.7] привел формулу

$$\frac{1-e^{-c\varphi}}{1-e^{-c(\theta+\varphi)}}e^{-ck/\sqrt{n}} + \frac{1-e^{-c\theta}}{1-e^{-c(\theta+\varphi)}}e^{-ca_k/\sqrt{n}} \approx 1,$$

где константа c должна быть выбрана как функция от θ и φ так, чтобы площадь области была равна n. Эта константа c отрицательна, если $\theta\varphi < 2$, и положительна, если $\theta\varphi > 2$; при $\theta\varphi = 2$ область вырождается в прямую линию

$$\frac{k}{\theta\sqrt{n}} + \frac{a_k}{\varphi\sqrt{n}} \approx 1.$$

Если $\varphi = \infty$, получаем $c = \sqrt{\mathrm{Li}_2(t)}$, где t удовлетворяет соотношению $\theta = (\ln\frac{1}{1-t})/\sqrt{\mathrm{Li}_2(t)}$.

43. $p(n-k(k-1)/2)$. (Замените $a_1a_2\ldots a_k$ на $(a_1-k+1)(a_2-k+2)\ldots a_k$, чтобы получить эквивалентное разбиение $n-k(k-1)/2$.)

44. Считаем, что $n>0$. Количество разбиений с *разными* наименьшими частями (или с одной-единственной частью) равно $p(n+1)-p(n)$ — количеству разбиений $n+1$, которые не заканчиваются единицей, поскольку мы получаем первые из вторых путем изменения наименьшей части. Таким образом, окончательный ответ — $2p(n)-p(n+1)$. [См. R. J. Boscovich, *Giornale de' Letterati* (Rome, 1748), 15. Количество разбиений, в которых равны *три* наименьшие части, равно $3p(n)-p(n+1)-2p(n+2)+p(n+3)$; подобные формулы могут быть выведены и для других ограничений на меньшие части разбиений.]

45. В соответствии с формулой (37) имеем $p(n-j)/p(n)=1-Cjn^{-1/2}+(C^2j^2+2j)/(2n)-(8C^3j^3+60Cj^2+Cj+12C^{-1}j)/(48n^{3/2})+O(j^4n^{-2})$.

46. Если $n>1$, то $T_2'(n)=p(n-1)-p(n-2)\le p(n)-p(n-1)=T_2''(n)$, поскольку $p(n)-p(n-1)$ — количество разбиений n, не оканчивающихся единицей; каждое такое разбиение $n-1$ дает при увеличении наибольшей части разбиение для n. Однако эта разность достаточно мала: $(T_2''(n)-T_2'(n))/p(n)=C^2/n+O(n^{-3/2})$.

47. Тождество в указании получается путем дифференцирования (21); см. упр. 22. Вероятность получения количества частей $c_1\ldots c_n$ при $c_1+2c_2+\cdots+nc_n=n$ по индукции по n равна

$$\begin{aligned}\Pr(c_1\ldots c_n)&=\sum_{k=1}^{n}\sum_{j=1}^{c_k}\frac{kp(n-jk)}{np(n)}\Pr(c_1\ldots c_{k-1}(c_k-j)c_{k+1}\ldots c_n)\\&=\sum_{k=1}^{n}\sum_{j=1}^{c_k}\frac{k}{np(n)}=\frac{1}{p(n)}.\end{aligned}$$

[*Combinatorial Algorithms* (Academic Press, 1975), Chapter 10.]

48. Вероятность того, что j имеет определенное фиксированное значение на шаге N5, равна $6/(\pi^2j^2)+O(n^{-1/2})$, а среднее значение jk имеет порядок $\sqrt{n}$. Среднее время, затрачиваемое на шаге N4, равно $\Theta(n)$, так что среднее время работы алгоритма имеет порядок $n^{3/2}$. (Попробуйте самостоятельно провести более точный анализ.)

49. (а) Имеем $F(z)=\sum_{k=1}^{\infty}F_k(z)$, где $F_k(z)$ — производящая функция для всех разбиений, наименьшая часть которых $\ge k$, а именно $1/((1-z^k)(1-z^{k+1})\ldots)-1$.

(б) Пусть $f_k(n)=[z^n]\,F_k(z)/p(n)$. Тогда $f_1(n)=1$; $f_2(n)=1-p(n-1)/p(n)=Cn^{-1/2}+O(n^{-1})$; $f_3(n)=\bigl(p(n)-p(n-1)-p(n-2)+p(n-3)\bigr)/p(n)=2C^2n^{-1}+O(n^{-3/2})$; $f_4(n)=6C^3n^{-3/2}+O(n^{-2})$. (См. упр. 45.) Оказывается, что $f_{k+1}(n)=k!\,C^kn^{-k/2}+O(n^{-(k+1)/2})$; в частности, $f_5(n)=O(n^{-2})$. Следовательно $f_5(n)+\cdots+f_n(n)=O(n^{-1})$, поскольку $f_{k+1}(n)\le f_k(n)$.

Полученные результаты приводят к $[z^n]\,F(z)=p(n)\bigl(1+C/\sqrt{n}+O(n^{-1})\bigr)$.

50. (а) $c_m(m+k)=c_{m-1}(m-1+k)+c_m(k)=m-1-k+c(k)+1$ по индукции при $0\le k<m$.

(б) Потому что $\left|{m+k \atop m}\right|=p(k)$ для $0\le k\le m$.

(в) Когда $n=2m$, алгоритм H, по сути, генерирует разбиения m, и $j-1$ является второй наименьшей частью в разбиении, сопряженном к только что сгенерированному (исключение составляет случай $j-1=m$ сразу после разбиения $1\ldots1$, сопряженное к которому состоит из одной части).

(г) Если все части α превышают k, то пусть $\alpha k^{q+1}j$ соответствует $\alpha(k+1)$.

(д) Продолжим выполнять предыдущее упражнение. В соответствии с п. (г) производящая функция $G_k(z)$ для всех разбиений, вторая наименьшая часть которых $\ge k$, представляет собой $F_{k+1}(z)/(1-z)$. Следовательно, $C(z)=\bigl(F(z)-F_1(z)\bigr)/(1-z)+z/(1-z)^2$.

(е) Как и в предыдущем упражнении, можно показать, что $[z^n]\,G_k(n)/p(n) = O(n^{-k/2})$ при $k \le 5$; следовательно, $c(m)/p(m) = 1 + O(m^{-1/2})$. Отношения $(c(m)+1)/p(m)$, которые легко вычислить для малых m, достигают максимального значения 2.6 при $m = 7$, после чего монотонно уменьшаются. Таким образом, внимательное рассмотрение асимптотических границ ошибки завершает доказательство.

Примечание. Б. Фристедт (B. Fristedt) [*Trans. Amer. Math. Soc.* **337** (1993), 703–735] доказал среди прочего, что количество k в случайном разбиении n больше $Cx\sqrt{n}$ с асимптотической вероятностью e^{-x}.

52. В лексикографическом порядке $\binom{64+13}{13}$ разбиений 64 имеют $a_1 \le 13$; $\binom{50+10}{10}$ из них имеют $a_1 = 14$ и $a_2 \le 10$; и т. д. Таким образом, с учетом указания, разбиению $14\ 11\ 9\ 6\ 4\ 3\ 2\ 1^{15}$ предшествуют ровно $p(64) - 1000000$ разбиений в лексикографическом порядке, делая его миллионным в *обратном* лексикографическом порядке.

53. Как и в предыдущем ответе, $\binom{80}{12}$ разбиений 100 имеют $a_1 = 32$ и $a_2 \le 12$, и т. д.; так что лексикографически миллионное разбиение, в котором $a_1 = 32$, представляет собой $32\ 13\ 12\ 8\ 7\ 6\ 5\ 5\ 1^{12}$. Алгоритм H дает сопряженное к нему, а именно $20\ 8\ 8\ 8\ 8\ 6\ 5\ 4\ 3\ 3\ 3\ 3\ 2\ 1^{19}$.

54. (а) Очевидно, истинно. Этот вопрос — всего лишь разминка.

(б) Истинно, но не столь очевидно. Если $\alpha^T = a'_1 a'_2 \ldots$, то, рассматривая диаграмму Феррерса, получим

$$a_1 + \cdots + a_k + a'_1 + \cdots + a'_l \;\le\; n + kl \qquad \text{для } 1 \le k \le a'_l,$$

причем равенство достигается при $k = a'_l$. Таким образом, если $\alpha \succeq \beta$ и $a'_1 + \cdots + a'_l > b'_1 + \cdots + b'_l$ для некоторого l, при минимальном l получим $n + kl = b_1 + \cdots + b_k + b'_1 + \cdots + b'_l < a_1 + \cdots + a_k + a'_1 + \cdots + a'_l \le n + kl$, где $k = b'_l$, т. е. приходим к противоречию.

(в) Рекуррентное соотношение $c_k = \min(a_1 + \cdots + a_k,\ b_1 + \cdots + b_k) - (c_1 + \cdots + c_{k-1})$, очевидно, определяет наибольшую нижнюю границу, если $c_1 c_2 \ldots$ представляет собой разбиение. Это так, поскольку если $c_1 + \cdots + c_k = a_1 + \cdots + a_k$, то $0 \le \min(a_{k+1}, b_{k+1}) \le \min(a_{k+1}, b_{k+1} + b_1 + \cdots + b_k - a_1 - \cdots - a_k) = c_{k+1} \le a_{k+1} \le a_k = c_k + (c_1 + \cdots + c_{k-1}) - (a_1 + \cdots + a_{k-1}) \le c_k$.

(г) $\alpha \vee \beta = (\alpha^T \wedge \beta^T)^T$ (двойное сопряжение необходимо потому, что ориентированное на максимум рекуррентное соотношение, аналогичное соотношению из пункта (в), может не выполняться).

(д) $\alpha \wedge \beta$ имеет $\max(l, m)$ частей, а $\alpha \vee \beta$ — $\min(l, m)$ частей. (Рассмотрите первые компоненты их сопряжений.)

(е) Истинно для $\alpha \wedge \beta$ в соответствии с выводом в п. (в). Ложно для $\alpha \vee \beta$; например, $6321 \vee 543 = 633$ на рис. 52.

Источник: T. Brylawski, *Discrete Mathematics* **6** (1973), 201–219.

55. (а) Если $\alpha \rhd \beta$ и $\alpha \succeq \gamma \succeq \beta$, где $\gamma = c_1 c_2 \ldots$, то $a_1 + \cdots + a_k = c_1 + \cdots + c_k = b_1 + \cdots + b_k$ для всех k, за исключением $k = l$ и $k = l+1$; таким образом, α покрывает β. Следовательно, β^T покрывает α^T.

Обратно, если $\alpha \succeq \beta$ и $\alpha \ne \beta$, можно найти $\gamma \succeq \beta$, такое, что $\alpha \rhd \gamma$ или $\gamma^T \rhd \alpha^T$, следующим образом. Найдем наименьшее k, для которого $a_k > b_k$, наименьшее l, для которого $a_k > a_{l+1}$, и наименьшее m, для которого $a_k - 1 > a_{m+1}$. (Заметим, что $b_k > 0$.) Если $a_m > a_{m+1} + 1$, определим $\gamma = c_1 c_2 \ldots$ следующим образом: $c_k = a_k - [k = m] + [k = m+1]$. В противном случае положим $c_k = a_k - [k = l] + [k = m+1]$.

(б) Пусть α и β представляют собой строки из n нулей и единиц, как в (15). В этом случае $\alpha \rhd \beta$ тогда и только тогда, когда $\alpha \to \beta$, а $\beta^T \rhd \alpha^T$ тогда и только тогда, когда $\alpha \Rightarrow \beta$, где '$\to$' означает замену подстроки вида 011^q10 подстрокой 101^q01, а '$\Rightarrow$' означает замену подстроки вида 010^q10 подстрокой 100^q01 для некоторого $q \ge 0$.

(в) Разбиение покрывает не более $[a_1 > a_2] + \cdots + [a_{m-1} > a_m] + [a_m \ge 2]$ других разбиений. Разбиение $\alpha = (n_2+n_1-1)(n_2-2)(n_2-3)\ldots 21$ максимизирует эту величину в случае $a_m = 1$; случаи с $a_m \ge 2$ не дают никакого улучшения. (Сопряженное разбиение, а именно $(n_2-1)(n_2-2)\ldots 21^{n_1+1}$, обеспечивает ту же степень покрытия. Следовательно, и α, и α^T также *покрываются* максимальным количеством других разбиений.)

(г) Что то же самое, последовательные части μ отличаются не более чем на 1, и наименьшая часть равна 1; краевое представление не содержит единиц, следующих одна за другой.

(д) Используем краевое представление и заменим $\rhd$ отношением $\to$. Если $\alpha \to \alpha_1$ и $\alpha \to \alpha_1'$, можно легко показать наличие строки β, такой, что $\alpha_1 \to \beta$ и $\alpha_1' \to \beta$, например:

$$\begin{array}{ccccc} & \nearrow & 101^q0111^r10 & \searrow & \\ 011^q1011^r10 & & & & 101^q1011^r01. \\ & \searrow & 011^q1101^r01 & \nearrow & \end{array}$$

Пусть $\beta = \beta_2 \rhd \cdots \rhd \beta_m$, где β_m минимально. Тогда по индукции по $\max(k, k')$ получаем $k = m$ и $\alpha_k = \beta_m$; также $k' = m$ и $\alpha_{k'}' = \beta_m$.

(е) Установить $\beta \leftarrow \alpha^T$; затем устанавливать $\beta \leftarrow \beta'$ до тех пор, пока β не станет минимальным, используя любые подходящие разбиения β', такие, что $\beta \rhd \beta'$. Искомое разбиение — β^T.

Доказательство. Пусть $\mu(\alpha)$ — общее значение $\alpha_k = \alpha_{k'}'$ из п. (д); мы должны доказать, что из $\alpha \succeq \beta$ вытекает $\mu(\alpha) \succeq \mu(\beta)$. Существует последовательность $\alpha = \alpha_0$, $\ldots$, $\alpha_k = \beta$, где $\alpha_j \to \alpha_{j+1}$ или $\alpha_j \Rightarrow \alpha_{j+1}$ для $0 \le j < k$. Если $\alpha_0 \to \alpha_1$, имеем $\mu(\alpha) = \mu(\alpha_1)$; таким образом, этого достаточно, чтобы доказать, что из $\alpha \Rightarrow \beta$ и $\alpha \to \alpha'$ вытекает $\alpha' \succeq \mu(\beta)$. Однако мы имеем, например,

$$\begin{array}{ccccc} & \Nearrow & 100^q0111^r10 & \searrow & \\ 010^q1011^r10 & & & & 100^q1011^r01\,, \\ & \searrow & 010^q1101^r01 \to 010^{q-1}10011^r01 & \Nearrow & \end{array}$$

поскольку можем считать, что $q > 0$; прочие случаи аналогичны.

(ж) Частями λ_n являются $a_k = n_2 + [k \le n_1] - k$ для $1 \le k < n_2$; частями λ_n^T являются $b_k = n_2 - k + [n_2 - k < n_1]$ для $1 \le k \le n_2$. Алгоритм из (е) достигает λ_n^T из n^1 после $\binom{n_2+1}{3} - \binom{n_2-n_1}{2}$ шагов, поскольку каждый шаг увеличивает $\sum kb_k = \sum \binom{a_k+1}{2}$ на 1.

(з) Путь n, $(n-1)1$, $(n-2)2$, $(n-2)11$, $(n-3)21$, $\ldots$, 321^{n-5}, 31^{n-3}, 221^{n-4}, 21^{n-2}, 1^n длиной $2n-4$ при $n \ge 3$ является кратчайшим.

Можно показать, что самый длинный путь имеет $m = 2\binom{n_2}{3} + n_1(n_2-1)$ шагов. Один из таких путей имеет вид α_0, $\ldots$, α_k, $\ldots$, α_l, $\ldots$, α_m, где $\alpha_0 = n^1$; $\alpha_k = \lambda_n$; $\alpha_l = \lambda_n^T$; $\alpha_j \rhd \alpha_{j+1}$ для $0 \le j < l$; кроме того, $\alpha_{j+1}^T \rhd \alpha_j^T$ для $k \le j < m$.

Источник: C. Greene and D. J. Kleitman, *Europ. J. Combinatorics* **7** (1986), 1–10.

56. Положим $\lambda = u_1 \ldots u_m$ и $\mu = v_1 \ldots v_m$. В приведенном далее (неоптимизированном) алгоритме применяется теория из упр. 54 для генерации разбиений в солексном порядке, поддерживая $\alpha = a_1a_2\ldots a_m \preceq \mu$ и $\alpha^T = b_1b_2\ldots b_l \preceq \lambda^T$. Для поиска следующего за α элемента сначала нужно найти наибольшее j, такое, что b_j может быть увеличено. Тогда мы имеем $\beta = b_1\ldots b_{j-1}(b_j+1)1\ldots 1 \preceq \lambda^T$, следовательно, искомый следующий элемент — $\beta^T \wedge \mu$. Алгоритм поддерживает вспомогательные таблицы $r_j = b_j + \cdots + b_l$, $s_j = v_1 + \cdots + v_j$ и $t_j = w_j + w_{j+1} + \cdots$, где $\lambda^T = w_1w_2\ldots$.

M1. [Инициализация.] Установить $q \leftarrow 0$, $k \leftarrow u_1$. Для $j = 1$, $\ldots$, m, пока $u_{j+1} < k$, устанавливать $t_k \leftarrow q \leftarrow q + j$ и $k \leftarrow k - 1$. Затем вновь установить $q \leftarrow 0$ и для $j = 1$, $\ldots$, m установить $a_j \leftarrow v_j$, $s_j \leftarrow q \leftarrow q + a_j$. Затем еще раз установить

$q \leftarrow 0$ и $k \leftarrow l \leftarrow a_1$. Для $j = 1, \ldots, m$, пока $a_{j+1} < k$, устанавливать $b_k \leftarrow j$, $r_k \leftarrow q \leftarrow q + j$ и $k \leftarrow k - 1$. И наконец установить $t_1 \leftarrow 0$, $b_0 \leftarrow 0$, $b_{-1} \leftarrow -1$.

M2. [Посещение.] Посетить разбиение $a_1 \ldots a_m$ и/или сопряженное к нему разбиение $b_1 \ldots b_l$.

M3. [Поиск j.] Пусть j — наибольшее целое число $< l$, такое, что $r_{j+1} > t_{j+1}$ и $b_j \neq b_{j-1}$. Завершить работу алгоритма, если $j = 0$.

M4. [Увеличение b_j.] Установить $x \leftarrow r_{j+1} - 1$, $k \leftarrow b_j$, $b_j \leftarrow k + 1$ и $a_{k+1} \leftarrow j$. (Предыдущее значение a_{k+1} было равно $j - 1$. Теперь мы переходим к обновлению $a_1 \ldots a_k$ с использованием, по сути, метода из упр. 54, (в) для распределения x точек по столбцам $j + 1$, $j + 2$, $\ldots$.)

M5. [Мажоризация.] Установить $z \leftarrow 0$, а затем для $i = 1, \ldots, k$ выполнить следующее: установить $x \leftarrow x + j$, $y \leftarrow \min(x, s_i)$, $a_i \leftarrow y - z$, $z \leftarrow y$; если $i = 1$, установить $l \leftarrow p \leftarrow a_1$ и $q \leftarrow 0$; если $i > 1$, пока $p > a_i$, устанавливать $b_p \leftarrow i - 1$, $r_p \leftarrow q \leftarrow q + i - 1$, $p \leftarrow p - 1$. Наконец, пока $p > j$, устанавливать $b_p \leftarrow k$, $r_p \leftarrow q \leftarrow q + k$, $p \leftarrow p - 1$. Вернуться к шагу M2. ▌

57. Если $\lambda = \mu^T$, то, очевидно, существует только одна такая матрица, по сути, представляющая собой диаграмму Феррерса λ. Условие $\lambda \preceq \mu^T$ является необходимым, поскольку, если $\mu^T = b_1 b_2 \ldots$, мы имеем $b_1 + \cdots + b_k = \min(c_1, k) + \min(c_2, k) + \cdots$, и эта величина не должна быть меньше, чем количество единиц в первых k строках. Наконец, если существует матрица для λ и μ и если λ покрывает α, то можно легко построить матрицу для α и μ путем перемещения 1 из любой определенной строки в другую, имеющую меньшее количество единиц.

Примечания. Этот результат часто называют теоремой Гейла–Райзера по известным статьям Д. Гейла (D. Gale) [*Pacific J. Math.* **7** (1957), 1073–1082] и Г. Д. Райзера (H. J. Ryser) [*Canadian J. Math.* **9** (1957), 371–377]. Но количество 0–1 матриц с суммами строк λ и суммами столбцов μ представляет собой коэффициент при мономиальной симметричной функции $\sum x_{i_1}^{c_1} x_{i_2}^{c_2} \ldots$ в произведении элементарных симметричных функций $e_{r_1} e_{r_2} \ldots$, где

$$e_r \;=\; [z^r]\,(1 + x_1 z)(1 + x_2 z)(1 + x_3 z) \ldots .$$

В этом контексте полученный результат известен как минимум с 1930-х годов; см. формулу Д. Э. Литтлвуда (D. E. Littlewood) для $\prod_{m,n \geq 0}(1 + x_m y_n)$ в *Proc. London Math. Soc.* (2) **40** (1936), 49–70. [Гораздо ранее в *Philosophical Trans.* **147** (1857), 489–499, Кейли (Cayley) показал, что лексикографическое условие $\lambda \leq \mu^T$ является необходимым.] См. также алгоритм из упр. 7–108.

58. [Р. Ф. Мюрхед (R. F. Muirhead), *Proc. Edinburgh Math. Soc.* **21** (1903), 144–157.] Условие $\alpha \succeq \beta$ является необходимым, поскольку можно установить $x_1 = \cdots = x_k = x$ и $x_{k+1} = \cdots = x_n = 1$ и при этом устремить $x \to \infty$. Оно достаточное, поскольку мы должны доказать его, только когда α покрывает β. Тогда, если, скажем, части (a_1, a_2) становятся $(a_1 - 1, a_2 + 1)$, левая часть равна правой части плюс неотрицательная величина

$$\frac{1}{2m!} \sum x_{p_1}^{a_2} x_{p_2}^{a_2} \ldots x_{p_m}^{a_m} (x_{p_1}^{a_1 - a_2 - 1} - x_{p_2}^{a_1 - a_2 - 1})(x_{p_1} - x_{p_2}).$$

[*Историческая справка.* Статья Мюрхеда — наиболее раннее упоминание концепции, ныне известной как мажоризация; вскоре после него эквивалентное определение было дано М. О. Лоренцем (M. O. Lorenz), в *Quarterly Publ. Amer. Stat. Assoc.* **9** (1905), 209–219, где он интересовался исследованиями неравномерного распределения богатств. Еще одна эквивалентная концепция была сформулирована И. Шуром (I. Schur) в *Sitzungsberichte Berliner Math. Gesellschaft* **22** (1923), 9–20. Название “мажоризация” было введено Харди (Hardy), Литтлвудом (Littlewood) и Пойа (Pólya), которые описали ее наиболее важные

свойства в *Messenger of Math.* **58** (1929), 145–152; см. упр. 2.3.4.5–17. Полностью посвящена данной теме прекрасная книга А. У. Маршалла (A. W. Marshall) и И. Олкина (I. Olkin) *Inequalities* (Academic Press, 1979).]

59. Данной симметрией должны обладать единственные пути для $n = 0$, 1, 2, 3, 4 и 6. Имеется один такой путь для $n = 5$, а именно 11111, 2111, 221, 311, 32, 41, 5. Для $n = 7$ имеется четыре пути:

$$1111111, 211111, 22111, 2221, 322, 3211, 31111, 4111, 511, 421, 331, 43, 52, 61, 7;$$
$$1111111, 211111, 22111, 2221, 322, 421, 511, 4111, 31111, 3211, 331, 43, 52, 61, 7;$$
$$1111111, 211111, 31111, 22111, 2221, 322, 3211, 4111, 421, 331, 43, 52, 511, 61, 7;$$
$$1111111, 211111, 31111, 22111, 2221, 322, 421, 4111, 3211, 331, 43, 52, 511, 61, 7.$$

Других путей нет, поскольку для всех $n \geq 8$ имеется как минимум два самосопряженных разбиения (см. упр. 15).

60. Вместо $L(6,6)$ используйте (59); в противном случае везде используйте $L'(4,6)$ и $L'(3,5)$.

В $M(4,18)$, вставьте 444222, 4442211 между 443322 и 4432221.
В $M(5,11)$, вставьте 52211, 5222 между 62111 и 6221.
В $M(5,20)$, вставьте 5542211, 554222 между 5552111 и 555221.
В $M(6,13)$, вставьте 72211, 7222 между 62221 и 6322.
В $L(4,14)$, вставьте 44222, 442211 между 43322 и 432221.
В $L(5,15)$, вставьте 542211, 54222 между 552111 и 55221.
В $L(7,12)$, вставьте 62211, 6222 между 72111 и 7221.

62. Утверждение выполняется при $n = 7$, 8 и 9 за исключением двух случаев: $n = 8$, $m = 3$, $\alpha = 3221$; $n = 9$, $m = 4$, $\alpha = 432$.

64. Если $n = 2^k q$, где q нечетно, обозначим через ω_n разбиение $(2^k)^q$, т. е. q частей, равных 2^k. Рекурсивное правило

$$B(n) \;=\; B(n-1)^R 1,\ 2 \times B(n/2)$$

при $n > 0$, где $2 \times B(n/2)$ означает удвоение всех частей $B(n/2)$ (или пустую последовательность, если n нечетно), определяет симпатичный путь Грея, начинающийся с $\omega_{n-1}1$ и заканчивающийся ω_n (в предположении, что $B(0)$ — единственное разбиение 0). Таким образом,

$$B(1) = 1; \quad B(2) = 11, 2; \quad B(3) = 21, 111; \quad B(4) = 1111, 211, 22, 4.$$

Среди множества замечательных свойств этой последовательности имеется следующее: при четном n

$$B(n) = (2 \times B(0))1^n,\ (2 \times B(1))1^{n-2},\ (2 \times B(2))1^{n-4},\ \ldots,\ (2 \times B(n/2))1^0;$$

например,

$$B(8) = 11111111, 2111111, 221111, 41111, 4211, 22211, 2222, 422, 44, 8.$$

Приведенный далее алгоритм без использования циклов генерирует $B(n)$ для $n \geq 2$.

K1. [Инициализация.] Установить $c_0 \leftarrow p_0 \leftarrow 0$, $p_1 \leftarrow 1$. Если n четно, установить $c_1 \leftarrow n$, $t \leftarrow 1$; в противном случае пусть $n - 1 = 2^k q$, где q нечетно, и установить $c_1 \leftarrow 1$, $c_2 \leftarrow q$, $p_2 \leftarrow 2^k$, $t \leftarrow 2$.

K2. [Четное посещение.] Посетить разбиение $p_t^{c_t} \ldots p_1^{c_1}$. (Сейчас $c_t + \cdots + c_1$ четно.)

K3. [Изменение наибольшей части.] Если $c_t = 1$, разделить наибольшую часть: если $p_t \neq 2p_{t-1}$, установить $c_t \leftarrow 2$, $p_t \leftarrow p_t/2$, в противном случае установить $c_{t-1} \leftarrow$

$c_{t-1}+2$, $t \leftarrow t-1$. Но если $c_t > 1$, следует объединить две наибольшие части: если $c_t = 2$, установить $c_t \leftarrow 1$, $p_t \leftarrow 2p_t$, в противном случае установить $c_t \leftarrow c_t - 2$, $c_{t+1} \leftarrow 1$, $p_{t+1} \leftarrow 2p_t$, $t \leftarrow t+1$.

K4. [Нечетное посещение.] Посетить разбиение $p_t^{c_t} \ldots p_1^{c_1}$. (Сейчас $c_t + \cdots + c_1$ нечетно.)

K5. [Изменение части, следующей за наибольшей.] Сейчас мы намерены применить следующее преобразование: "временно удалить $c_t - [t$ четно$]$ наибольших частей, затем применить шаг K3, после чего восстановить удаленные части". Говоря более точно, имеется девять ситуаций. (1, а) Если c_t нечетно и $t = 1$, завершить работу алгоритма. (1, б1) Если c_t нечетно, $c_{t-1} = 1$ и $p_{t-1} = 2p_{t-2}$, установить $c_{t-2} \leftarrow c_{t-2} + 2$, $c_{t-1} \leftarrow c_t$, $p_{t-1} \leftarrow p_t$, $t \leftarrow t-1$. (1, б2) Если c_t нечетно, $c_{t-1} = 1$ и $p_{t-1} \neq 2p_{t-2}$, установить $c_{t-1} \leftarrow 2$, $p_{t-1} \leftarrow p_{t-1}/2$. (1, в1) Если c_t нечетно, $c_{t-1} = 2$ и $p_t = 2p_{t-1}$, установить $c_{t-1} \leftarrow c_t + 1$, $p_{t-1} \leftarrow p_t$, $t \leftarrow t-1$. (1, в2) Если c_t нечетно, $c_{t-1} = 2$ и $p_t \neq 2p_{t-1}$, установить $c_{t-1} \leftarrow 1$, $p_{t-1} \leftarrow 2p_{t-1}$. (1, г1) Если c_t нечетно, $c_{t-1} > 2$ и $p_t = 2p_{t-1}$, установить $c_{t-1} \leftarrow c_{t-1} - 2$, $c_t \leftarrow c_t + 1$. (1, г2) Если c_t нечетно, $c_{t-1} > 2$ и $p_t \neq 2p_{t-1}$, установить $c_{t+1} \leftarrow c_t$, $p_{t+1} \leftarrow p_t$, $c_t \leftarrow 1$, $p_t \leftarrow 2p_{t-1}$, $c_{t-1} \leftarrow c_{t-1} - 2$, $t \leftarrow t+1$. (2, а) Если c_t четно и $p_t = 2p_{t-1}$, set $c_t \leftarrow c_t - 1$, $c_{t-1} \leftarrow c_{t-1} + 2$. (2, б) Если c_t четно и $p_t \neq 2p_{t-1}$, установить $c_{t+1} \leftarrow c_t - 1$, $p_{t+1} \leftarrow p_t$, $c_t \leftarrow 2$, $p_t \leftarrow p_t/2$, $t \leftarrow t+1$. Вернуться к шагу K2. ▮

[Преобразования на шагах K3 и K5 аннулируют сами себя при повторном выполнении в строке. Эта конструкция разработана Т. Колтарстом (T. Colthurst) и М. Клебером (M. Kleber) в "A Gray path on binary partitions", `http://arxiv.org/abs/0907.3873`. Эйлер рассматривал количество таких разбиений в §50 своей статьи в 1750 году.]

65. Если $p_1^{e_1} \ldots p_r^{e_r}$ — разбиение m на простые множители, количество таких разложений равно $p(e_1) \ldots p(e_r)$, и мы можем положить $n = \max(e_1, \ldots, e_r)$. В самом деле, для каждого r-кортежа $(x_1, \ldots, x_r)$, где $0 \le x_k < p(e_k)$, можно положить $m_j = p_1^{a_{1j}} \ldots p_r^{a_{rj}}$, где $a_{k1} \ldots a_{kn}$ представляет собой (x_k+1)-е разбиение e_k. Таким образом, можно использовать рефлексивный код Грея для r-кортежей совместно с кодом Грея для разбиений.

66. Пусть $a_1 \ldots a_m$ — m-кортеж, удовлетворяющий указанным неравенствам. Этот кортеж можно отсортировать в невозрастающем порядке $a_{x_1} \ge \cdots \ge a_{x_m}$, где перестановка $x_1 \ldots x_m$ однозначно определяется при условии *устойчивой* сортировки; см. формулу 5–(2).

Если $j \prec k$, имеем $a_j \ge a_k$, следовательно, j появляется слева от k в перестановке $x_1 \ldots x_m$. Таким образом, $x_1 \ldots x_m$ представляет собой одну из перестановок, получающихся на выходе алгоритма 7.2.1.2V. Кроме того, j находится слева от k и тогда, когда $a_j = a_k$ и $j < k$, в соответствии со свойством устойчивости. Следовательно, a_{x_i} строго больше $a_{x_{i+1}}$, когда $x_i > x_{i+1}$ является "спуском".

Для генерации всех соответствующих разбиений n возьмем каждую из топологических перестановок $x_1 \ldots x_m$ и сгенерируем разбиения $y_1 \ldots y_m$ числа $n - t$, где t — *индекс* $x_1 \ldots x_m$ (см. раздел 5.1.1). Для $1 \le j \le m$ установим $a_{x_j} \leftarrow y_j + t_j$, где t_j — количество спусков справа от x_j в $x_1 \ldots x_m$.

Например, если $x_1 \ldots x_m = 314592687$, и мы хотим сгенерировать все случаи, для которых $a_3 > a_1 \ge a_4 \ge a_5 \ge a_9 > a_2 \ge a_6 \ge a_8 > a_7$. В этом случае $t = 1 + 5 + 8 = 14$, так что мы устанавливаем $a_1 \leftarrow y_2 + 2$, $a_2 \leftarrow y_6 + 1$, $a_3 \leftarrow y_1 + 3$, $a_4 \leftarrow y_3 + 2$, $a_5 \leftarrow y_4 + 2$, $a_6 \leftarrow y_7 + 1$, $a_7 \leftarrow y_9$, $a_8 \leftarrow y_8 + 1$ и $a_9 \leftarrow y_5 + 2$. Обобщенная производящая функция $\sum z_1^{a_1} \ldots z_9^{a_9}$ в смысле упр. 29 имеет вид

$$\frac{z_1^2 z_2 z_3^3 z_4^2 z_5^2 z_6 z_8 z_9^2}{(1-z_3)(1-z_3z_1)(1-z_3z_1z_4)(1-z_3z_1z_4z_5)\ldots(1-z_3z_1z_4z_5z_9z_2z_6z_8z_7)}.$$

Когда любым данным частичным упорядочением является $\prec$, обычная производящая функция для всех таких разбиений n имеет вид $\sum z^{\operatorname{ind}\alpha}/((1-z)(1-z^2)\ldots(1-z^m))$, где сумма берется по всем выходам α алгоритма 7.2.1.2V.

[См. расширения и приложения этих идей в работе R. P. Stanley, *Memoirs Amer. Math. Soc.* **119** (1972). Информация о "холмистых" разбиениях имеется в L. Carlitz, *Studies in Foundations and Combinatorics* (New York: Academic Press, 1978), 101–129.]

67. Если $n+1 = q_1 \ldots q_r$, где все множители $q_1, \ldots, q_r$ не меньше 2, мы получаем идеальное разбиение $\{(q_1-1)\cdot 1, (q_2-1)\cdot q_1, (q_3-1)\cdot q_1q_2, \ldots, (q_r-1)\cdot q_1\ldots q_{r-1}\}$, которое очевидным образом соответствует смешанной позиционной системе счисления (порядок множителей q_j имеет значение).

И наоборот, таким образом можно получить все идеальные разбиения. Предположим, что мультимножество $M = \{k_1\cdot p_1, \ldots, k_m\cdot p_m\}$ является идеальным разбиением, причем $p_1 < \cdots < p_m$; тогда должно выполняться условие $p_j = (k_1+1)\ldots(k_{j-1}+1)$ для $1 \le j \le m$, поскольку p_j представляет собой наименьшую сумму подмультимножества M, которое не является подмультимножеством $\{k_1\cdot p_1, \ldots, k_{j-1}\cdot p_{j-1}\}$.

Идеальное разбиение n с наименьшим числом элементов получается тогда и только тогда, когда все q_j — простые, поскольку $pq-1 > (p-1)+(q-1)$ для любых $p>1$ и $q>1$. Таким образом, например, минимальные идеальные разбиения 11 соответствуют упорядоченным разложениям $2\cdot 2\cdot 3$, $2\cdot 3\cdot 2$ и $3\cdot 2\cdot 2$. *Источник*: *Quarterly Journal of Mathematics* **21** (1886), 367–373.

68. (а) Если $a_i+1 \le a_j-1$ для некоторых i и j, можно заменить $\{a_i, a_j\}$ на $\{a_i+1, a_j-1\}$, увеличив таким образом произведение на $a_j - a_i - 1 > 0$. Следовательно, оптимум достигается только при оптимально сбалансированном разбиении из упр. 3. [L. Oettinger and J. Derbès, *Nouv. Ann. Math.* **18** (1859), 442; **19** (1860), 117–118.]

(б) Считаем, что $n>1$. Тогда ни одна из частей не равна 1; и если $a_j \ge 4$, можно заменить эту часть на $2+(a_j-2)$ без уменьшения произведения. Таким образом, можно считать, что все части равны 2 или 3. Можно улучшить результат, заменяя $2+2+2$ на $3+3$, следовательно, в разложении имеется не более двух 2. Оптимум, таким образом, равен $3^{n/3}$ при $n \bmod 3 = 0$; $4\cdot 3^{(n-4)/3} = 3^{(n-4)/3}\cdot 2\cdot 2 = (4/3^{4/3})3^{n/3}$ при $n \bmod 3 = 1$; $3^{(n-2)/3}\cdot 2 = (2/3^{2/3})3^{n/3}$ при $n \bmod 3 = 2$. [O. Mei?ner, *Mathematisch-naturwissenschaftliche Blätter* **4** (1907), 85.]

69. Все $n>2$ имеют решение $(n, 2, 1, \ldots, 1)$. Можно "отсеять" прочие случаи $\le N$, начиная с $s_2\ldots s_N \leftarrow 1\ldots 1$ и устанавливая затем $s_{ak-b} \leftarrow 0$ для всех $ak-b \le N$, где $a = x_1\ldots x_t - 1$, $b = x_1+\cdots+x_t-t-1$, $k \ge x_1 \ge \cdots \ge x_t$ и $a>1$, поскольку $k+x_1+\cdots+x_t+(ak-b-t-1) = kx_1\ldots x_t$. Последовательность $(x_1, \ldots, x_t)$ должна рассматриваться, только когда $(x_1\ldots x_t-1)x_1-(x_1+\cdots+x_t) < N-t$; можно также продолжать уменьшать N так, чтобы $s_N = 1$. Таким образом, следует испытать только $(32766, 1486539, 254887, 1511, 937, 478, 4)$ последовательностей $(x_1, \ldots, x_t)$ при начальном значении N, равном 2^{30}, и единственными "выжившими" останутся значения 2, 3, 4, 6, 24, 114, 174 и 444. [См. E. Trost, *Elemente der Math.* **11** (1956), 135; M. Misiurewicz, *Elemente der Math.* **21** (1966), 90.]

Примечания. Похоже, новых значений при $N\to\infty$ не появится, но для их исключения требуется новая идея. Простейшие последовательности $(x_1, \ldots, x_t) = (3)$ и $(2,2)$ исключают все $n>5$, для которых $n \bmod 6 \ne 0$; этот факт можно использовать для ускорения вычислений в 6 раз. Последовательности (6) и $(3,2)$ исключают 40% оставшихся значений (а именно все n вида $5k-4$ и $5k-2$); последовательности (8), $(4,2)$ и $(2,2,2)$ исключают 3/7 остатка; последовательности с $t=1$ требуют, чтобы $n-1$ было простым. Последовательности, в которых $x_1\ldots x_t = 2^r$, исключают около $p(r)$ вычетов $n \bmod (2^r-1)$. Последовательности, в которых $x_1\ldots x_t$ представляет собой произведение r различных простых чисел, исключают около ϖ_r вычетов $n \bmod (x_1\ldots x_t-1)$.

70. На каждом шаге одно разбиение n превращается в другое, так что в конечном счете мы должны достичь повторяющегося цикла. У многих разбиений просто выполняется циклический сдвиг по диагоналям, идущим сверху справа вниз влево на диаграмме Ферररса, превращая ее

$$\text{из}\quad \begin{array}{cccccc} x_1 & x_2 & x_4 & x_7 & x_{11} & x_{16}\ldots \\ x_3 & x_5 & x_8 & x_{12} & x_{17} & x_{23}\ldots \\ x_6 & x_9 & x_{13} & x_{18} & x_{24} & x_{31}\ldots \\ x_{10} & x_{14} & x_{19} & x_{25} & x_{32} & x_{40}\ldots \\ x_{15} & x_{20} & x_{26} & x_{33} & x_{41} & x_{50}\ldots \\ x_{21} & x_{27} & x_{34} & x_{42} & x_{51} & x_{61}\ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \qquad \text{в}\quad \begin{array}{cccccc} x_1 & x_3 & x_6 & x_{10} & x_{15} & x_{21}\ldots \\ x_2 & x_4 & x_7 & x_{11} & x_{16} & x_{22}\ldots \\ x_5 & x_8 & x_{12} & x_{17} & x_{23} & x_{30}\ldots \\ x_9 & x_{13} & x_{18} & x_{24} & x_{31} & x_{39}\ldots \\ x_{14} & x_{19} & x_{25} & x_{32} & x_{40} & x_{49}\ldots \\ x_{20} & x_{26} & x_{33} & x_{41} & x_{50} & x_{60}\ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}.$$

Другими словами, к ячейкам применяется перестановка $\rho = (1)(2\,3)(4\,5\,6)(7\,8\,9\,10)\ldots$. Исключения возникают только тогда, когда ρ попадает на пустую ячейку над точкой; например, x_{10} может быть пустой, а x_{11} — нет. Однако в таких случаях можно получить корректную новую диаграмму, перемещая верхнюю строку вниз и вставляя ее путем сортировки в правильное место после применения ρ. Такое перемещение всегда снижает количество занятых диагоналей, так что оно не может быть частью цикла. Таким образом, каждый цикл состоит только из перестановок ρ.

Если некоторый элемент на диагонали в циклическом разбиении пуст, то и все элементы следующей диагонали должны быть пусты. Если, скажем, x_5 пуст, повторное применение ρ сделает x_5 соседним с каждой из ячеек x_7, x_8, x_9, x_{10} следующей диагонали. Следовательно, если $n = \binom{n_2}{2} + \binom{n_1}{1}$ при $n_2 > n_1 \geq 0$, циклические состояния те же, что и при $n_2 - 1$ полностью заполненных диагоналях и n_1 точках в следующей. [Этот результат получен Ю. Брандтом (J. Brandt), *Proc. Amer. Math. Soc.* **85** (1982), 483–486. По утверждениям задача пришла из России через Болгарию и Швецию. См. также Martin Gardner, *The Last Recreations* (1997), Chapter 2.]

71. Когда $n = 1+\cdots+m > 1$, начальное разбиение $(m-1)(m-1)(m-2)\ldots 2\,1\,1$ находится на расстоянии $m(m-1)$ от циклического состояния, и это расстояние максимально. [K. Igusa, *Math. Magazine* **58** (1985), 259–271; G. Etienne, *J. Combin. Theory* **A58** (1991), 181–197.] Григгс (Griggs) и Хо (Ho) [*Advances in Appl. Math.* **21** (1998), 205–227] предположили, что в общем случае максимальное расстояние до цикла равно $\max(2n+2-n_1(n_2+1), n+n_2+1, n_1(n_2+1)) - 2n_2$ для всех $n > 1$; их гипотеза проверена для $n \leq 100$. Кроме того, наихудшее разбиение, похоже, единственное при $n_2 = 2n_1 + \{-1, 0, 2\}$.

72. Эквивалентно утверждению $a_1 < m - 1$ [B. Hopkins and J. A. Sellers, *Integers* **7**, 2 (2007), #A19].

73. (а) [Р. Стенли (R. Stanley), 1972 (неопубликовано).] Обменяем j-е появление k в разбиении $n = j \cdot k + \alpha$ с k-м появлением j в $k \cdot j + \alpha$ для каждого разбиения α числа $n - jk$. Например, при $n = 6$ обмены представляют собой

```
6,  51,  42,  411,  33,  321,  3111,  222,  2211,  21111,  111111.
a   b1   fg   c1g   hi   jk1   d1kh   n2i   m21n   e1mjf   1edcba
```

(б) $p(n-k) + p(n-2k) + p(n-3k) + \cdots$. [A. H. M. Hoare, *AMM* **93** (1986), 475–476.]

РАЗДЕЛ 7.2.1.5

1. Всякий раз, когда m устанавливается равным r на шаге H6, следует заменить его значение на $r - 1$.

2. **L1.** [Инициализация.] Установить $l_j \leftarrow j - 1$ и $a_j \leftarrow 0$ для $1 \leq j \leq n$. Установить также $h_1 \leftarrow n$, $t \leftarrow 1$ и установить l_0 равным любому устраивающему ненулевому значению.

L2. [Посещение.] Посетить t-блочное разбиение, представленное $l_1 \ldots l_n$ и $h_1 \ldots h_t$. (Ограниченно растущая строка, соответствующая этому разбиению, — $a_1 \ldots a_n$.)

L3. [Поиск j.] Установить $j \leftarrow n$; затем, пока $l_j = 0$, устанавливать $j \leftarrow j - 1$ и $t \leftarrow t - 1$.

L4. [Перемещение j в следующий блок.] Завершить работу алгоритма, если $j = 0$. В противном случае установить $k \leftarrow a_j + 1$, $h_k \leftarrow l_j$, $a_j \leftarrow k$. Если $k = t$, установить $t \leftarrow t+1$ и $l_j \leftarrow 0$; в противном случае установить $l_j \leftarrow h_{k+1}$. Наконец установить $h_{k+1} \leftarrow j$.

L5. [Перемещение $j+1, \ldots, n$ в блок 1.] Пока $j < n$, устанавливать $j \leftarrow j+1$, $l_j \leftarrow h_1$, $a_j \leftarrow 0$ и $h_1 \leftarrow j$. Вернуться к шагу L2. ▮

3. Пусть $\tau(k,n)$ — количество строк $a_1 \ldots a_n$, удовлетворяющих условию $0 \le a_j \le 1 + \max(k-1, a_1, \ldots, a_{j-1})$ для $1 \le j \le n$; так, $\tau(k,0) = 1$, $\tau(0,n) = \varpi_n$ и $\tau(k,n) = k\tau(k,n-1) + \tau(k+1,n-1)$. [С. Д. Вильямсон (S. G. Williamson) назвал $\tau(k,n)$ "хвостовым коэффициентом" (tail coefficient); см. *SICOMP* **5** (1976), 602–617.] Количество строк, которые генерируются алгоритмом H перед заданной ограниченно растущей строкой $a_1 \ldots a_n$, равно $\sum_{j=1}^n a_j \tau(b_j, n-j)$, где $b_j = 1 + \max(a_1, \ldots, a_{j-1})$. Выполняя расчеты с помощью предвычисленной таблицы хвостовых коэффициентов, находим, что приведенная формула дает 999999 при $a_1 \ldots a_{12} = 010220345041$.

4. Ниже приведены наиболее распространенные представители каждого типа (индекс указывает количество таких слов в GraphBase): $\texttt{zzzzz}_0$, $\texttt{ooooh}_0$, $\texttt{xxxix}_0$, $\texttt{xxxii}_0$, $\texttt{ooops}_0$, $\texttt{llull}_0$, $\texttt{llala}_0$, $\texttt{eeler}_0$, $\texttt{iitti}_0$, $\texttt{xxiii}_0$, $\texttt{ccxxv}_0$, $\texttt{eerie}_1$, $\texttt{llama}_1$, $\texttt{xxvii}_0$, $\texttt{oozed}_5$, $\texttt{uhuuu}_0$, $\texttt{mamma}_1$, $\texttt{puppy}_{28}$, $\texttt{anana}_0$, $\texttt{hehee}_0$, $\texttt{vivid}_{15}$, $\texttt{rarer}_3$, $\texttt{etext}_1$, $\texttt{amass}_2$, $\texttt{again}_{137}$, $\texttt{ahhaa}_0$, $\texttt{esses}_1$, $\texttt{teeth}_{25}$, $\texttt{yaaay}_0$, $\texttt{ahhhh}_2$, $\texttt{pssst}_2$, $\texttt{seems}_7$, $\texttt{added}_6$, $\texttt{lxxii}_0$, $\texttt{books}_{184}$, $\texttt{swiss}_3$, $\texttt{sense}_{10}$, $\texttt{ended}_3$, $\texttt{check}_{160}$, $\texttt{level}_{18}$, $\texttt{tepee}_4$, $\texttt{slyly}_5$, $\texttt{never}_{154}$, $\texttt{sells}_6$, $\texttt{motto}_{21}$, $\texttt{whooo}_2$, $\texttt{trees}_{384}$, $\texttt{going}_{307}$, $\texttt{which}_{151}$, $\texttt{there}_{174}$, $\texttt{three}_{100}$, $\texttt{their}_{3834}$. (См. S. Golomb, *Math. Mag.* **53** (1980), 219–221. Слова только с двумя различными буквами, разумеется, редки. Восемнадцать представителей, приведенных здесь с индексом 0, можно найти в бóльших словарях или на англоязычных страницах в Интернете.)

5. (а) $112 = \rho(0225)$. Это последовательность $r(0)$, $r(1)$, $r(4)$, $r(9)$, $r(16)$, $\ldots$, где $r(n)$ получается путем записи n в виде десятичного числа (с одним или несколькими ведущими нулями), применения функции ρ из упр. 4 и последующего удаления ведущих нулей. Обратите внимание, что $n/9 \le r(n) \le n$.

(б) $1012 = r(45^2)$. Последовательность та же, что и в п. (а), но отсортированная и с удаленными дубликатами. (Кто же думал, что $88^2 = 7744$, $212^2 = 44944$, а $264^2 = 69696$?)

6. Воспользуемся подходом с применением топологической сортировки из алгоритма 7.2.1.2V с соответствующим частичным упорядочением: включаем c_j цепочек длиной j с упорядоченными наименьшими элементами. Например, если $n = 20$, $c_2 = 3$ и $c_3 = c_4 = 2$, мы используем упомянутый алгоритм для поиска всех перестановок $a_1 \ldots a_{20}$ множества $\{1, \ldots, 20\}$, таких, что $1 \prec 2$, $3 \prec 4$, $5 \prec 6$, $1 \prec 3 \prec 5$, $7 \prec 8 \prec 9$, $10 \prec 11 \prec 12$, $7 \prec 10$, $13 \prec 14 \prec 15 \prec 16$, $17 \prec 18 \prec 19 \prec 20$, $13 \prec 17$, образуя ограниченно растущие строки $\rho(f(a_1) \ldots f(a_{20}))$, где функция ρ определена в упр. 4, а $(f(1), \ldots, f(20)) = (1,1,2,2,3,3,4,4,4,5,5,5,6,6,6,6,7,7,7,7)$. Общее количество выходных данных, конечно же, определяется формулой (48).

7. Ровно ϖ_n. Это перестановки, которые получаются путем обращения порядка блоков слева направо в (2) и опускания символов '|': 1234, 4123, 3124, 3412, $\ldots$, 4321. [См. A. Claesson, *European J. Combinatorics* **22** (2001), 961–971. С. В. Китаев в своей работе *Discrete Math.* **298** (2005), 212–229, открыл далеко идущее обобщение. Пусть π — перестановка $\{0, \ldots, r\}$, g_n — количество перестановок $a_1 \ldots a_n$ множества $\{1, \ldots, n\}$, таких, что из

$a_{k-0\pi} > a_{k-1\pi} > \cdots > a_{k-r\pi} > a_j$ вытекает $j > k$, и пусть f_n — количество перестановок $a_1 \ldots a_n$, для которых шаблон $a_{k-0\pi} > a_{k-1\pi} > \cdots > a_{k-r\pi}$ полностью отсутствует при $r < k \le n$. Тогда $\sum_{n\ge 0} g_n z^n/n! = \exp(\sum_{n\ge 1} f_{n-1} z^n/n!)$.]

8. Для каждого разбиения множества $\{1, \ldots, n\}$ на m блоков расположим блоки в порядке убывания их наименьших элементов и будем переставлять все элементы блоков, кроме наименьших, всеми возможными способами. Например, при $n = 9$ и $m = 3$ разбиение $126|38|4579$ дает нам 457938126 и одиннадцать прочих случаев, полученных из данного путем перестановок множеств $\{5, 7, 9\}$ и $\{2, 6\}$ в рамках этих множеств. (По сути, тот же метод генерирует все перестановки, имеющие ровно k циклов; см. подраздел "Замечательное соответствие" в разделе 1.3.3.)

9. Среди перестановок мультимножества $\{k_0 \cdot 0, k_1 \cdot 1, \ldots, k_{n-1} \cdot (n-1)\}$ ограниченным ростом обладают ровно

$$\binom{k_0 + k_1 + \cdots + k_{n-1}}{k_0,\, k_1,\, \ldots,\, k_{n-1}} \frac{k_0}{(k_0 + k_1 + \cdots + k_{n-1})} \frac{k_1}{(k_1 + \cdots + k_{n-1})} \cdots \frac{k_{n-1}}{k_{n-1}}$$

из них, поскольку $k_j/(k_j + \cdots + k_{n-1})$ представляет собой вероятность того, что j предшествует значениям $\{j + 1, \ldots, n - 1\}$.

Среднее количество нулей при $n > 0$ равно $1 + (n-1)\varpi_{n-1}/\varpi_n = \Theta(\log n)$, поскольку общее количество нулей во всех ϖ_n случаях равно $\sum_{k=1}^{n} k\binom{n-1}{k-1}\varpi_{n-k} = \varpi_n + (n-1)\varpi_{n-1}$.

10. Для данного разбиения множества $\{1, \ldots, n\}$ построим ориентированное дерево на множестве $\{0, 1, \ldots, n\}$ так, чтобы $j-1$ являлся родительским узлом по отношению ко всем членам блока, наименьший член которого — j. Затем пометим листья, сохраняя порядок, и удалим остальные метки. Например, 15 разбиений в (2) соответствуют следующим деревьям.

Чтобы обратить процесс, возьмем полупомеченное дерево и присвоим новые номера его узлам, рассматривая сначала узлы, встречающиеся на пути от корня к наименьшему листу, затем ко второму по величине листу и т. д. Количество листьев равно $n + 1$ минус количество блоков. [Эта конструкция тесно связана с упр. 2.3.4.4–18 и многими перечислениями в соответствующем разделе. См. P. L. Erdős and L. A. Székely, *Advances in Applied Math.* **10** (1989), 488–496.]

11. Чистые буквометики получаются в 900 из 64 855 разбиений множества не более чем на 10 блоков, для которых $\rho(a_1 \ldots a_{13}) = \rho(a_5 \ldots a_8 a_1 \ldots a_4 a_9 \ldots a_{13})$, и в 563 527 из 13 788 536 разбиений, для которых $\rho(a_1 \ldots a_{13}) < \rho(a_5 \ldots a_8 a_1 \ldots a_4 a_9 \ldots a_{13})$. Примерами первого типа являются `aaaa` + `aaaa` = `baaac`, `aaaa` + `aaaa` = `bbbbc` и `aaaa` + `aaab` = `baaac`; примерами последнего типа — `abcd` + `efgd` = `dceab` (`goat` + `newt` = `tango`) и `abcd` + `efgd` = `dceaf` (`clad`+`nerd` = `dance`). [Идея применения генератора разбиений для решения буквометиков принадлежит Алану Сатклиффу (Alan Sutcliffe).]

12. (а) Строим $\rho((a_1 a_1') \ldots (a_n a_n'))$, где ρ определено в упр. 4, поскольку $x \equiv y$ (по модулю $\Pi \vee \Pi'$) тогда и только тогда, когда $x \equiv y$ (по модулю Π) и $x \equiv y$ (по модулю Π').

(б) Представим Π с помощью связей, как это было сделано в упр. 2; Π' представим, как в алгоритме 2.3.3E, и используем этот алгоритм для того, чтобы сделать $j \equiv l_j$ при $l_j \ne 0$ (для повышения эффективности можно считать, что Π имеет как минимум столько же блоков, сколько и Π'.)

(в) Когда один блок Π разбивается на две части, т. е. когда два блока Π' были слиты.

(г) $\binom{t}{2}$; (д) $(2^{s_1-1} - 1) + \cdots + (2^{s_t-1} - 1)$.

(е) Истинно. Пусть $\Pi \vee \Pi'$ содержит блоки $B_1|B_2|\cdots|B_t$, где $\Pi = B_1B_2|B_3|\cdots|B_t$. Тогда Π', по сути, является разбиением множества $\{B_1, \ldots, B_t\}$ таким, что $B_1 \not\equiv B_2$, а $\Pi \wedge \Pi'$ получается путем слияния блока Π', содержащего B_1, с блоком, содержащим B_2. [Конечная решетка, удовлетворяющая этому условию, называется *нижней полумодулярной* (*lower semimodular*); см. G. Birkhoff, *Lattice Theory* (1940), §I.8. Решетка мажоризации из упр. 7.2.1.4–54 этим свойством не обладает; например, $\alpha = 4111$, а $\alpha' = 331$.]

(ж) Ложно. Пусть, например, $\Pi = 0011$, $\Pi' = 0101$.

(з) Блоки Π и Π' представляют собой объединения блоков $\Pi \vee \Pi'$, так что можно считать, что $\Pi \vee \Pi' = \{1, \ldots, t\}$. Как и в п. (б), объединим j с l_j для получения за r шагов Π, имеющего $t-r$ блоков. Будучи примененными к Π', каждое из этих объединений снижает количество блоков на 0 или 1. Следовательно, $b(\Pi') - b(\Pi \wedge \Pi') \le r = b(\Pi \vee \Pi') - b(\Pi)$.

[В *Algebra Universalis* **10** (1980), 74–95, П. Пудлак (P. Pudlák) и И. Тума (J. Tůma) доказали, что *каждая* конечная решетка является подрешеткой решетки разбиений множества $\{1, \ldots, n\}$ для соответствующего большого значения n.]

13. [См. *Advances in Math.* **26** (1977), 290–305.] Если j наибольших элементов t-блочного разбиения находятся в отдельных блоках, состоящих из одного элемента, а следующие $n-j$ элементов — нет, то будем говорить, что разбиение имеет порядок $t-j$. Определим "строку Стирлинга" Σ_{nt} как последовательность порядков t-блочных разбиений Π_1, Π_2, $\ldots$; например, $\Sigma_{43} = 122333$. Тогда $\Sigma_{tt} = 0$, и мы получаем $\Sigma_{(n+1)t}$ из Σ_{nt} путем замены каждой цифры d строкой $d^d(d{+}1)^{d+1}\ldots t^t$ длиной $\binom{t+1}{2} - \binom{d}{2}$; например,

$$\Sigma_{53} = 1223332\underset{\wedge}{2}3333\underset{\wedge}{2}333333\underset{\wedge}{3}3333\underset{\wedge}{3}333.$$

Основная идея состоит в рассмотрении процесса лексикографической генерации алгоритма H. Предположим, что $\Pi = a_1 \ldots a_n$ представляет собой t-блочное разбиение порядка j; тогда оно является лексикографически наименьшим t-блочным разбиением, ограниченно растущая строка которого начинается с $a_1 \ldots a_{n-t+j}$. Разбиениями, покрываемыми разбиением Π, являются (в лексикографическом порядке) Π_{12}, Π_{13}, Π_{23}, Π_{14}, Π_{24}, Π_{34}, $\ldots$, $\Pi_{(t-1)t}$, где Π_{rs} означает "объединение блоков r и s разбиения Π" (т. е. "замена всех появлений $s{-}1$ на $r{-}1$ с последующим применением функции ρ для получения ограниченно растущей строки"). Если Π' представляет собой одно из последних $\binom{t}{2} - \binom{j}{2}$ из указанных разбиений от $\Pi_{1(j+1)}$ и далее, то Π — наименьшее t-блочное разбиение, за которым следует Π'. Например, если $\Pi = 001012034$, то $n = 9$, $t = 5$, $j = 3$, а соответствующие разбиения Π' представляют собой $\rho(001012004)$, $\rho(001012014)$, $\rho(001012024)$, $\rho(001012030)$, $\rho(001012031)$, $\rho(001012032)$, $\rho(001012033)$.

Следовательно, $f_{nt}(N) = f_{nt}(N-1) + \binom{t}{2} - \binom{j}{2}$, где j — N-я цифра Σ_{nt}.

14. **E1.** [Инициализация.] Установить $a_j \leftarrow 0$ и $b_j \leftarrow d_j \leftarrow 1$ для $1 \le j \le n$.

E2. [Посещение.] Посетить ограниченно растущую строку $a_1 \ldots a_n$.

E3. [Поиск j.] Установить $j \leftarrow n$; затем, пока $a_j = d_j$, устанавливать $d_j \leftarrow 1 - d_j$ и $j \leftarrow j - 1$.

E4. [Выполнено?] Завершить работу алгоритма, если $j = 1$. В противном случае перейти к шагу E6, если $d_j = 0$.

E5. [Перемещение вниз.] Если $a_j = 0$, установить $a_j \leftarrow b_j$, $m \leftarrow a_j + 1$ и перейти к шагу E7. В противном случае при $a_j = b_j$ установить $a_j \leftarrow b_j - 1$, $m \leftarrow b_j$ и перейти к шагу E7. В противном случае установить $a_j \leftarrow a_j - 1$ и вернуться к шагу E2.

E6. [Перемещение вверх.] Если $a_j = b_j - 1$, установить $a_j \leftarrow b_j$, $m \leftarrow a_j + 1$, и перейти к шагу E7. В противном случае при $a_j = b_j$ установить $a_j \leftarrow 0$, $m \leftarrow b_j$ и перейти к шагу E7. В противном случае установить $a_j \leftarrow a_j + 1$ и вернуться к шагу E2.

E7. [Коррекция $b_{j+1}\ldots b_n$.] Установить $b_k \leftarrow m$ для $k = j+1, \ldots, n$. Вернуться к шагу E2. ▌

[Этот алгоритм можно существенно оптимизировать, поскольку, как и в алгоритме H, j почти всегда равно n.]

15. Это разбиение соответствует первым n цифрам бесконечной бинарной строки $01011011011\ldots$, поскольку ϖ_{n-1} четно тогда и только тогда, когда $n \bmod 3 = 0$ (см. упр. 23).

16. 00012, 01012, 01112, 00112, 00102, 01102, 01002, 01202, 01212, 01222, 01022, 01122, 00122, 00121, 01121, 01021, 01221, 01211, 01201, 01200, 01210, 01220, 01020, 01120, 00120.

17. В приведенном решении используются две взаимно рекурсивные процедуры, $f(\mu,\nu,\sigma)$ и $b(\mu,\nu,\sigma)$, для "прямой" и "обратной" генераций $A_{\mu\nu}$ при $\sigma = 0$ и $A'_{\mu\nu}$ при $\sigma = 1$. Для начала процесса в предположении $1 < m < n$ сначала нужно установить $a_j \leftarrow 0$ для $1 \le j \le n-m$ и $a_{n-m+j} \leftarrow j-1$ для $1 \le j \le m$, а затем вызвать $f(m,n,0)$.

Процедура $f(\mu,\nu,\sigma)$. Если $\mu = 2$, посетить $a_1\ldots a_n$; в противном случае вызвать $f(\mu-1,\nu-1,(\mu+\sigma) \bmod 2)$. Затем, если $\nu = \mu+1$, выполнить следующее: изменить a_μ с 0 на $\mu-1$ и посетить $a_1\ldots a_n$; затем установить $a_\nu \leftarrow a_\nu - 1$ и посетить $a_1\ldots a_n$, выполняя эти действия до тех пор, пока не будет выполнено условие $a_\nu = 0$. Но если $\nu > \mu+1$, изменить $a_{\nu-1}$ (если $\mu+\sigma$ нечетно) или a_μ (если $\mu+\sigma$ четно) с 0 на $\mu-1$; затем вызвать $b(\mu,\nu-1,0)$, если $a_\nu+\sigma$ нечетно, $f(\mu,\nu-1,0)$, если $a_\nu+\sigma$ четно; и пока $a_\nu > 0$, устанавливать $a_\nu \leftarrow a_\nu - 1$ и вызывать $b(\mu,\nu-1,0)$ или $f(\mu,\nu-1,0)$, как описано выше, повторяя эти действия до тех пор, пока не будет выполнено условие $a_\nu = 0$.

Процедура $b(\mu,\nu,\sigma)$. Если $\nu = \mu+1$, сначала выполнить следующее: посетить $a_1\ldots a_n$ и установить $a_\nu \leftarrow a_\nu + 1$, повторяя эти действия до тех пор, пока не будет выполнено условие $a_\nu = \mu-1$; затем посетить $a_1\ldots a_n$ и изменить a_μ с $\mu-1$ на 0. Но если $\nu > \mu+1$, вызвать $f(\mu,\nu-1,0)$, если $a_\nu+\sigma$ нечетно, или $b(\mu,\nu-1,0)$, если $a_\nu+\sigma$ четно; затем, пока $a_\nu < \mu-1$, устанавливать $a_\nu \leftarrow a_\nu + 1$ и вызывать $f(\mu,\nu-1,0)$ или $b(\mu,\nu-1,0)$, как описано выше, повторяя эти действия до тех пор, пока не будет выполнено условие $a_\nu = \mu-1$; наконец изменить $a_{\nu-1}$ (если $\mu+\sigma$ нечетно) или a_μ (если $\mu+\sigma$ четно) с $\mu-1$ на 0. В конце в обоих случаях, если $\mu = 2$, посетить $a_1\ldots a_n$, в противном случае вызвать $b(\mu-1,\nu-1,(\mu+\sigma) \bmod 2)$.

Бóльшая часть времени тратится на обработку случая $\mu = 2$; в этом случае можно применить более быстрые подпрограммы, основанные на бинарном коде Грея (и отличающиеся от последовательностей Раски). Более эффективная процедура может использоваться и при $\mu = \nu-1$.

18. Соответствующая последовательность должна начинаться (или заканчиваться) элементом $01\ldots(n-1)$. В соответствии с упр. 32 указанный код Грея не существует, если $0 \ne \delta_n \ne (1)^{0+1+\cdots+(n-1)}$, т. е. когда $n \bmod 12$ равно 4, 6, 7 или 9.

Случаи $n = 1, 2, 3$ легко разрешимы; при $n = 5$ имеется 1 927 683 326 решений. Таким образом, вероятно, имеется несметное количество решений при всех $n \ge 8$, за исключением указанных выше значений n. На самом деле, вероятно, можно найти такой путь Грея по всем ϖ_{nk} рассматриваемым в ответе к упр. 28, (д) строкам, за исключением случаев $n \equiv 2k + (2,4,5,7)$ (по модулю 1)2.

Примечание. Обобщенное число Стирлинга $\left\{{n \atop m}\right\}_{-1}$ из упр. 30 больше 1 при $2 < m < n$, так что такого кода Грея для разбиений множества $\{1,\ldots,n\}$ на m блоков может не быть.

19. (а) Необходимо заменить (6) шаблоном 0, 2, …, m, …, 3, 1 или обратным к нему, как в случае чун-упорядоченной (endo-order) последовательности (7.2.1.3–(45)).

(б) Можно обобщить (8) и (9) для получения последовательностей $A_{mn\alpha}$ и A'_{mna}, которые начинаются с $0^{n-m}01\ldots(m-1)$ и заканчиваются $01\ldots(m-1)\alpha$ и $0^{n-m-1}01\ldots(m-1)a$

соответственно, где $0 \le a \le m-2$, а α — любая строка $a_1 \ldots a_{n-m}$, такая, что $0 \le a_j \le m-2$. При $2 < m < n$ новые рекурсивные соотношения выглядят следующим образом:

$$A_{m(n+1)(\alpha a)} = \begin{cases} A_{(m-1)n(b\beta)}x_1, A^R_{mn\beta}x_1, A_{mn\alpha}x_2, \ldots, A_{mn\alpha}x_m, & \text{если } m \text{ четно;} \\ A'_{(m-1)nb}x_1, A_{mn\alpha}x_1, A^R_{mn\alpha}x_2, \ldots, A_{mn\alpha}x_m, & \text{если } m \text{ нечетно;} \end{cases}$$

$$A'_{m(n+1)a} = \begin{cases} A'_{(m-1)nb}x_1, A_{mn\beta}x_1, A^R_{mn\beta}x_2, \ldots, A^R_{mn\beta}x_m, & \text{если } m \text{ четно;} \\ A_{(m-1)n(b\beta)}x_1, A^R_{mn\beta}x_1, A_{mn\beta}x_2, \ldots, A^R_{mn\beta}x_m, & \text{если } m \text{ нечетно.} \end{cases}$$

Здесь $b = m-3$, $\beta = b^{n-m}$, а $(x_1, \ldots, x_m)$ — путь от $x_1 = m-1$ до $x_m = a$.

20. 012323212122; в общем случае $(a_1 \ldots a_n)^T = \rho(a_n \ldots a_1)$ при использовании обозначений из упр. 4.

21. Числа $\langle s_0, s_1, s_2, \ldots\rangle = \langle 1, 1, 2, 3, 7, 12, 31, 59, 164, 339, 999, \ldots\rangle$ удовлетворяют рекуррентным соотношениям $s_{2n+1} = \sum_k \binom{n}{k} s_{2n-2k}$, $s_{2n+2} = \sum_k \binom{n}{k}(2^k+1)s_{2n-2k}$ из-за связи средних элементов с другими. Поэтому $s_{2n} = n!\,[z^n]\,\exp\bigl((e^{2z}-1)/2 + e^z - 1\bigr)$ и $s_{2n+1} = n!\,[z^n]\,\exp\bigl((e^{2z}-1)/2 + e^z + z - 1\bigr)$. Рассматривая разбиения множества в первой половине, мы также имеем $s_{2n} = \sum_k \left\{ {n \atop k} \right\} x_k$ и $s_{2n+1} = \sum_k \left\{ {n+1 \atop k} \right\} x_{k-1}$, где $x_n = 2x_{n-1} + (n-1)x_{n-2} = n!\,[z^n]\,\exp(2z + z^2/2)$. [Т. С. Моцкин (T. S. Motzkin) рассматривал последовательность $\langle s_{2n}\rangle$ в *Proc. Symp. Pure Math.* **19** (1971), 173.]

22. (а) $\sum_{k=0}^\infty k^n \Pr(X=k) = e^{-1}\sum_{k=0}^\infty k^n/k! = \varpi_n$ в соответствии с (16).

(б) $\sum_{k=0}^\infty k^n \Pr(X=k) = \sum_{k=0}^\infty k^n \sum_{j=0}^m \binom{j}{k}(-1)^{j-k}/j!$; внутреннюю сумму можно расширить до $j=\infty$, поскольку $\sum_k \binom{j}{k}(-1)^k k^n = 0$ при $j > n$. Таким образом, получаем $\sum_{k=0}^\infty (k^n/k!)\sum_{l=0}^\infty (-1)^l/l! = \varpi_n$. [См. J. O. Irwin, *J. Royal Stat. Soc.* **A118** (1955), 389–404; J. Pitman, *AMM* **104** (1997), 201–209.]

23. (а) В соответствии с (14) формула справедлива для $f(x) = x^n$, так что она выполняется и в общем случае. (С учетом (16) мы также получаем $\sum_{k=0}^\infty f(k)/k! = ef(\varpi)$.)

(б) Предположим, что мы доказали данное отношение для k, и пусть $h(x) = (x-1)^{\underline{k}} f(x)$, $g(x) = f(x+1)$. Тогда $f(\varpi+k+1) = g(\varpi+k) = \varpi^{\underline{k}} g(\varpi) = h(\varpi+1) = \varpi h(\varpi) = \varpi^{\underline{k+1}} f(\varpi)$. [См. J. Touchard, *Ann. Soc. Sci. Bruxelles* **53** (1933), 21–31. Такое символьное "теневое исчисление", разработанное Джоном Блиссардом (John Blissard) в *Quart. J. Pure and Applied Math.* **4** (1861), 279–305, оказывается весьма полезным; однако пользоваться им следует с осторожностью, поскольку из $f(\varpi) = g(\varpi)$ не следует, что $f(\varpi)h(\varpi) = g(\varpi)h(\varpi)$.]

(в) Указание представляет собой частный случай упр. 4.6.2–16(с). Применение $f(x) = x^n$ и $k = p$ в (б) приводит к $\varpi_n \equiv \varpi_{p+n} - \varpi_{1+n}$.

(г) Полином $x^N - 1$ делится на $g(x) = x^p - x - 1$ по модулю p, поскольку $x^{p^k} \equiv x + k$ и $x^N \equiv x^{\bar p} \equiv x^{\underline{p}} \equiv x^p - x \equiv 1$ (по модулю $g(x)$ и p). Таким образом, если $h(x) = (x^N - 1)x^n/g(x)$, то мы получаем $h(\varpi) \equiv h(\varpi+p) = \varpi^{\underline{p}} h(\varpi) \equiv (\varpi^p - \varpi)h(\varpi)$; и $0 \equiv g(\varpi)h(\varpi) = \varpi^{N+n} - \varpi^n$ (по модулю p).

24. Указание следует из индукции по e, поскольку $x^{\underline{p^e}} = \prod_{k=0}^{p-1}(x - kp^{e-1})^{\underline{p^{e-1}}}$. Можно также доказать по индукции по n, что из $x^n \equiv r_n(x)$ (по модулю $g_1(x)$ и p) вытекает

$$x^{p^{e-1}n} \equiv r_n(x)^{p^{e-1}} \text{ (по модулю } g_e(x),\ pg_{e-1}(x),\ \ldots,\ p^{e-1}g_1(x) \text{ и } p^e).$$

Следовательно, $x^{p^{e-1}N} = 1 + h_0(x)g_e(x) + ph_1(x)g_{e-1}(x) + \cdots + p^{e-1}h_{e-1}(x)g_1(x) + p^e h_e(x)$ для некоторых полиномов $h_k(x)$ с целыми коэффициентами. Мы имеем $h_0(\varpi)\varpi^n \equiv h_0(\varpi + p^e)(\varpi + p^e)^n = \varpi^{\underline{p^e}} h_0(\varpi)\varpi^n \equiv \bigl(g_e(\varpi)+1\bigr)h_0(\varpi)\varpi^n$ по модулю p^e; следовательно,

$$\varpi^{p^{e-1}N+n} \;=\; \varpi^n + h_0(\varpi)g_e(\varpi)\varpi^n + ph_1(\varpi)g_{e-1}(\varpi)\varpi^n + \cdots \;\equiv\; \varpi^n.$$

[Аналогичный вывод примени́м и тогда, когда $p = 2$, но при этом следует положить $g_{j+1}(x) = g_j(x)^2 + 2[j = 2]$, и мы получим $\varpi_n \equiv \varpi_{n+3\cdot 2^e}$ (по модулю 2^e). Эти результаты получены Маршаллом Холлом (Marshall Hall); см. *Bull. Amer. Math. Soc.* **40** (1934), 387; *Amer. J. Math.* **70** (1948), 387–388. Дополнительную информацию можно найти в W. F. Lunnon, P. A. B. Pleasants, and N. M. Stephens, *Acta Arith.* **35** (1979), 1–16.]

25. Первое неравенство следует из применения к дереву ограниченно растущих строк более общего принципа: в любом дереве, для всех некорневых узлов p которого выполняется условие $\deg(p) \geq \deg(\mathrm{parent}(p))$, имеем $w_k/w_{k-1} \leq w_{k+1}/w_k$, где w_k — общее количество узлов на уровне k. Если $m = w_{k-1}$ узлов на уровне $k-1$ имеют соответственно $a_1, \ldots, a_m$ дочерних узлов, то они имеют как минимум $a_1^2 + \cdots + a_m^2$ "внучатых" узлов; следовательно, $w_{k-1}w_{k+1} \geq m(a_1^2 + \cdots + a_m^2) \geq (a_1 + \cdots + a_m)^2 = w_k^2$.

Рассматривая второе неравенство, заметим, что $\varpi_{n+1} - \varpi_n = \sum_{k=0}^{n} \bigl(\binom{n}{k} - \binom{n-1}{k-1}\bigr)\varpi_{n-k}$; таким образом,

$$\frac{\varpi_{n+1}}{\varpi_n} - 1 \;=\; \sum_{k=0}^{n-1} \binom{n-1}{k} \frac{\varpi_{n-k}}{\varpi_n} \;\leq\; \sum_{k=0}^{n-1} \binom{n-1}{k} \frac{\varpi_{n-k-1}}{\varpi_{n-1}} \;=\; \frac{\varpi_n}{\varpi_{n-1}},$$

поскольку, например, $\varpi_{n-3}/\varpi_n = (\varpi_{n-3}/\varpi_{n-2})(\varpi_{n-2}/\varpi_{n-1})(\varpi_{n-1}/\varpi_n)$ меньше или равно $(\varpi_{n-4}/\varpi_{n-3})(\varpi_{n-3}/\varpi_{n-2})(\varpi_{n-2}/\varpi_{n-1}) = \varpi_{n-4}/\varpi_{n-1}$.

26. Имеется $\binom{n-1}{n-t}$ направленных вправо путей от $n1$ до tt; можно представить их строками из нулей и единиц, где 0 означает "вправо", 1 — "вверх", а позиции единиц говорят нам, какие $n-t$ элементов находятся в блоке с единицей. Следующий шаг при $t > 1$ представляет собой дальний переход влево; так что мы можем продолжить путь, который определяет разбиение оставшихся $t-1$ элементов. Например, разбиение 14|2|3 соответствует при использовании данных соглашений пути 0010, где соответствующие биты означают, что $1 \not\equiv 2$, $1 \not\equiv 3$, $1 \equiv 4$, $2 \not\equiv 3$. [Возможны и многие другие интерпретации. Предложенное здесь соглашение показывает, что ϖ_{nk} подсчитывает разбиения с $1 \not\equiv 2, \ldots, 1 \not\equiv k$ — комбинаторное свойство, открытое Г. В. Беккером (H. W. Becker); см. *AMM* **51** (1944), 47, и *Mathematics Magazine* **22** (1948), 23–26.]

27. (а) В общем случае $\lambda_0 = \lambda_1 = \lambda_{2n-1} = \lambda_{2n} = 0$. В приведенном далее списке показаны также ограниченно растущие строки, соответствующие каждому циклу при использовании алгоритма из п. (б).

0,0,0,0,0,0,0,0,0	0123	0,0,1,0,0,0,0,0,0	0012	0,0,1,1,1,0,0,0,0	0102
0,0,0,0,0,0,1,0,0	0122	0,0,1,0,0,0,1,0,0	0011	0,0,1,1,1,0,1,0,0	0100
0,0,0,0,1,0,0,0,0	0112	0,0,1,0,1,0,0,0,0	0001	0,0,1,1,1,1,1,0,0	0120
0,0,0,0,1,0,1,0,0	0111	0,0,1,0,1,0,1,0,0	0000	0,0,1,1,11,1,1,0,0	0101
0,0,0,0,1,1,1,0,0	0121	0,0,1,0,1,1,1,0,0	0010	0,0,1,1,2,1,1,0,0	0110

(б) Название "диаграмма" вызвано связью с разделом 5.1.4 и разработанной в нем теорией, приводящей к интересному взаимно однозначному соответствию. Можно представить разбиения множества на треугольной шахматной доске, помещая ладью в столбец l_j строки $n+1-j$, если $l_j \neq 0$ в представлении со связанным списком из упр. 2 (см. ответ к упр. 5.1.3–19). Например, на приведенном рисунке показано ладейное представление 135|27|489|6. Ненулевые связи могут также быть эквивалентно определены с помощью двустрочных массивов наподобие $\binom{1\,2\,3\,4\,8}{3\,7\,5\,8\,9}$; см. 5.1.4–(11).

Рассмотрим путь длиной $2n$, который начинается в нижнем левом углу этой треугольной диаграммы и следует по ребрам правой границы, заканчиваясь в верхнем правом углу. Точками этого

пути являются $z_k = (\lfloor k/2 \rfloor, \lceil k/2 \rceil)$ для $0 \le k \le 2n$. Более того, прямоугольник вверху и слева от z_k содержит те ладьи, которые добавляют пары координат $\begin{smallmatrix} i \\ j \end{smallmatrix}$ в двустрочный массив, при этом $i \le \lfloor k/2 \rfloor$ и $j > \lceil k/2 \rceil$; в нашем примере для $9 \le k \le 12$ имеется ровно две такие ладьи, а именно $\binom{2\,4}{7\,8}$. Теорема 5.1.4A гласит, что такие двустрочные массивы эквивалентны диаграмме (P_k, Q_k), в которой элементы P_k берутся из нижней строки, а элементы Q_k — из верхней, причем и P_k, и Q_k имеют одну и ту же форму. В диаграмме P выгодно использовать элементы в порядке уменьшения, а в Q — в порядке увеличения, так что в нашем примере они представляют собой следующее.

k	P_k	Q_k	k	P_k	Q_k	k	P_k	Q_k
2	3	1	7	7 5	2 3	12	8 7	2 4
3	3	1	8	8 5 7	2 3 4	13	8	4
4	7 3	1 2	9	8 7	2 4	14	8	4
5	7	2	10	8 7	2 4	15	·	·
6	7 5	2 3	11	8 7	2 4	16	9	8

P_k и Q_k пустые при $k = 0$, 1, 17 и 18.

Таким способом каждое разбиение множества приводит к циклу колеблющейся диаграммы $\lambda_0, \lambda_1, \ldots, \lambda_{2n}$, если положить, что λ_k — разбиение целого числа, определяющее общую форму P_k и Q_k. (В нашем примере это цикл 0, 0, 1, 1, 11, 1, 2, 2, 21, 11, 11, 11, 11, 1, 1, 0, 1, 0, 0.) Кроме того, $t_{2k-1} = 0$ тогда и только тогда, когда строка $n + 1 - k$ не содержит ни одной ладьи, тогда и только тогда, когда k — наименьшее значение в своем блоке.

И обратно: элементы P_k и Q_k могут быть однозначно восстановлены из последовательности форм λ_k. А именно $Q_k = Q_{k-1}$ если $t_k = 0$. В противном случае при k четном Q_k представляет собой Q_{k-1} с числом $k/2$, помещенным в новую ячейку справа в строке t_k; при k нечетном Q_k получается из Q_{k-1} с использованием алгоритма 5.1.4D для удаления крайнего справа элемента строки t_k. Аналогичная процедура определяет P_k на основе значений P_{k+1} и t_{k+1}, так что можно работать в обратном направлении, от P_{2n} к P_0. Таким образом, последовательности форм λ_k достаточно для того, чтобы указать, где именно следует размещать ладьи.

Циклы колеблющихся диаграмм были разработаны в статье W. Y. C. Chen, E. Y. P. Deng, R. R. X. Du, R. P. Stanley, and C. H. Yan [*Transactions of the Amer. Math. Soc.* **359** (2007), 1555–1575], в которой было показано, что эта конструкция приводит к важным (и удивительным) следствиям. Например, если разбиение множества Π соответствует циклу колеблющейся диаграммы $\lambda_0, \lambda_1, \ldots, \lambda_{2n}$, назовем *дуальным* к нему разбиение Π^D, которое соответствует последовательности транспонированных форм $\lambda_0^T, \lambda_1^T, \ldots, \lambda_{2n}^T$. Тогда в соответствии с упр. 5.1.4–7 Π содержит "k-пересечение в l", т. е. последовательность индексов такую, что $i_1 < \cdots < i_k \le l < j_1 < \cdots < j_k$ и $i_1 \equiv j_1, \ldots, i_k \equiv j_k$ (по модулю Π), тогда и только тогда, когда Π^D содержит "k-вложение в l", которое представляет собой последовательность индексов, такую, что $i'_1 < \cdots < i'_k \le l < j'_k < \cdots < j'_1$ и $i'_1 \equiv j'_1, \ldots, i'_k \equiv j'_k$ (по модулю Π^D). Заметим также, что инволюция, по сути, представляет собой разбиение множества, в котором все блоки имеют размер 1 или 2; дуальной к инволюции является инволюция, имеющая те же множества с одним элементом. В частности, ду-

альным совершенному паросочетанию (когда отсутствуют множества с одним элементом) является совершенное паросочетание.

Кроме того, аналогичная конструкция применима для размещения ладей в *любой* диаграмме Феррерса, а не только в диаграмме ступенчатой формы, соответствующей разбиениям множества. Для заданной диаграммы Феррерса, которая имеет не более m частей, ни одна из которых не превышает n, мы просто рассматриваем путь $z_0 = (0,0)$, $z_1, \ldots, z_{m+n} = (n,m)$, который идет по правому ребру диаграммы и обусловливает, что $\lambda_k = \lambda_{k-1} + e_{t_k}$ при $z_k = z_{k-1} + (1,0)$ и $\lambda_k = \lambda_{k-1} - e_{t_k}$ при $z_k = z_{k-1} + (0,1)$. Доказательство, приведенное для ступенчатых диаграмм, показывает также, что любое размещение ладей на диаграмме Феррерса, когда в каждой строке и каждом столбце имеется не более одной ладьи, соответствует единственному такого рода циклу диаграммы.

[Истинно между тем гораздо большее! См. A. Berele, *J. Combinatorial Theory* **A43** (1986), 320–328; S. Fomin, *J. Combinatorial Theory* **A72** (1995), 277–292; M. van Leeuwen, *Electronic J. Combinatorics* **3**, 2 (1996), paper #R15.]

28. (а) Определим взаимно однозначное соответствие между расстановкой ладей путем обмена позиций ладей в строках j и $j+1$ тогда и только тогда, когда в выступающей части более длинной строки имеется ладья:

(б) Если выполнить транспонирование всех ладей, это соотношение становится очевидным из определения полинома.

(в) Предположим, что $a_1 \ge a_2 \ge \cdots$ и $a_k > a_{k+1}$. Тогда

$$R(a_1, a_2, \ldots) = xR(a_1-1, \ldots, a_{k-1}-1, a_{k+1}, \ldots) + yR(a_1, \ldots, a_{k-1}, a_k-1, a_{k+1}, \ldots)$$

поскольку первый член учитывает случаи, когда ладья находится в строке k и столбце a_k. Кроме того, исходя из расстановки пустых элементов, $R(0) = 1$. Далее из этих рекуррентных соотношений находим

$$R(1) = x + y;\quad R(2) = R(1,1) = x + xy + y^2;\quad R(3) = R(1,1,1) = x + xy + xy^2 + y^3;$$
$$R(2,1) = x^2 + 2xy + xy^2 + y^3;$$
$$R(3,1) = R(2,2) = R(2,1,1) = x^2 + x^2y + xy + 2xy^2 + xy^3 + y^4;$$
$$R(3,1,1) = R(3,2) = R(2,2,1) = x^2 + 2x^2y + x^2y^2 + 2xy^2 + 2xy^3 + xy^4 + y^5;$$
$$R(3,2,1) = x^3 + 3x^2y + 3x^2y^2 + x^2y^3 + 3xy^3 + 2xy^4 + xy^5 + y^6.$$

(г) Например, формула $\varpi_{73}(x,y) = x\varpi_{63}(x,y) + y\varpi_{74}(x,y)$ эквивалентна формуле $R(5,4,4,3,2,1) = xR(4,3,3,2,1) + yR(5,4,3,3,2,1)$, частному случаю (в); очевидно, что $\varpi_{nn}(x,y) = R(n-2,\ldots,0)$ эквивалентно $\varpi_{(n-1)1}(x,y) = R(n-2,\ldots,1)$.

(д) В действительности $y^{k-1}\varpi_{nk}(x,y)$ — указанная сумма по всем ограниченно растущим строкам $a_1 \ldots a_n$, для которых $a_2 > 0, \ldots, a_k > 0$.

29. (а) Если ладьи находятся в столбцах $(c_1, \ldots, c_n)$, то количество свободных ячеек равно количеству инверсий перестановок $(n+1-c_1)\ldots(n+1-c_n)$. [Поверните правую часть примера на рис. 56 на 180° и сравните результат с иллюстрацией после соотношения 5.1.1–(5).]

(б) Каждая конфигурация $r \times r$ может быть размещена, скажем, в строках $i_1 < \cdots < i_r$ и столбцах $j_1 < \cdots < j_r$, что дает $(m-r)(n-r)$ свободных ячеек в невыбранных строках и столбцах; $(i_2-i_1+1) + 2(i_3-i_2-1) + \cdots + (r-1)(i_r-i_{r-1}-1) + r(m-i_r)$ — в невыбранных

строках и выбранных столбцах и аналогичное количество в выбранных строках и невыбранных столбцах. Кроме того, сумма

$$\sum_{1\le i_1<\cdots<i_r\le m} y^{(i_2-i_1+1)+2(i_3-i_2-1)+\cdots+(r-1)(i_r-i_{r-1}-1)+r(m-i_r)}$$

может рассматриваться как сумма $y^{a_1+a_2+\cdots+a_{m-r}}$ по всем разбиениям $r \ge a_1 \ge a_2 \ge \cdots \ge a_{m-r} \ge 0$, так что согласно теореме C она равна $\binom{m}{r}_y$. В соответствии с п. (а) полином $r!_y$ генерирует свободные ячейки для выбранных строк и столбцов. Следовательно, искомый ответ — $y^{(m-r)(n-r)}\binom{m}{r}_y\binom{n}{r}_y r!_y = y^{(m-r)(n-r)} m!_y n!_y/((m-r)!_y(n-r)!_y r!_y)$.

(в) Левая часть формулы представляет собой производящую функцию $R_m(t+a_1,\ldots,t+a_m)$ для диаграммы Феррерса с t дополнительными столбцами высотой m. Поскольку существует $t+a_m$ способов размещения ладьи в строке m, это дает нам $1+y+\cdots+y^{t+a_m-1} = (1-y^{t+a_m})/(1-y)$ свободных ячеек, связанных с этими выборами; тогда в строке $m-1$ имеется $t+a_{m-1}-1$ доступных ячеек и т. д.

Аналогично правая часть равна $R_m(t+a_1,\ldots,t+a_m)$. Если $m-k$ ладей расставляются в столбцах $> t$, мы должны поместить k ладей в столбцах $\le t$ в k неиспользованных строк; но мы видели, что производящая функция для свободных ячеек при размещении k ладей на доске размером $k\times t$ представляет собой $t!_y/(t-k)!_y$.

Примечания. Доказанная здесь формула может рассматриваться как полиномиальное тождество переменных y и y^t; следовательно, она корректна при произвольном t, хотя наше доказательство и предполагает, что t — неотрицательное целое число. Этот результат был открыт для $y=1$ в работе J. Goldman, J. Joichi, and D. White, *Proc. Amer. Math. Soc.* **52** (1975), 485–492. Справедливость для общего случая установлена в статье A. M. Garsia and J. B. Remmel, *J. Combinatorial Theory* **A41** (1986), 246–275, в которой подобная аргументация использовалась для доказательства дополнительной формулы

$$\sum_{t=0}^{\infty} z^t \prod_{j=1}^{m} \frac{1-y^{a_j+m-j+t}}{1-y} = \sum_{k=0}^{n} k!_y \left(\frac{z}{1-yz}\right)\cdots\left(\frac{z}{1-y^k z}\right) R_{m-k}(a_1,\ldots,a_m).$$

(г) Это утверждение, непосредственно следующее из п. (в), влечет за собой также то, что $R(a_1,\ldots,a_m) = R(a'_1,\ldots,a'_m)$ тогда и только тогда, когда равенство выполняется для всех x и для любого ненулевого значения y. Полином Пирса $\varpi_{nk}(x,y)$ из упр. 28, (г) представляет собой ладейный полином для $\binom{n-1}{k-1}$ различных диаграмм Феррерса; например, $\varpi_{63}(x,y)$ перечисляет расстановки ладей для форм 43321, 44221, 44311, 4432, 53221, 53311, 5332, 54211, 5422 и 5431.

30. (а) Имеем $\varpi_n(x,y) = \sum_m x^{n-m}A_{mn}$, где $A_{mn} = R_{n-m}(n-1,\ldots,1)$ подчиняется простому закону: если мы не помещаем ладью в строку 1 формы $(n-1,\ldots,1)$, то эта строка содержит $m-1$ свободных ячеек, поскольку $n-m$ ладей находятся в других строках. Но если мы разместим здесь ладью, то оставим 0 или 1, или $\cdots$, или $m-1$ ее ячеек свободными. Следовательно, $A_{mn} = y^{m-1}A_{(m-1)(n-1)} + (1+y+\cdots+y^{m-1})A_{m(n-1)}$, и по индукции $A_{mn} = y^{m(m-1)/2}\left\{{n \atop m}\right\}_y$.

(б) Формула $\varpi_{n+1}(x,y) = \sum_k \binom{n}{k} x^{n-k}y^k\varpi_k(x,y)$ дает

$$A_{m(n+1)} = \sum_k \binom{n}{k} y^k A_{(m-1)k}.$$

(в) Из пп. (а) и (б) получаем

$$\frac{z^n}{(1-z)(1-(1+q)z)\ldots(1-(1+q+\cdots+q^{n-1})z)} = \sum_k \left\{ {k \atop n} \right\}_q z^k;$$

$$\sum_k \binom{n}{k}_q (-1)^k q^{\binom{k}{2}} e^{(1+q+\cdots+q^{n-k-1})z} = q^{\binom{n}{2}}\, n!_q \sum_k \left\{ {k \atop n} \right\}_q \frac{z^k}{k!}.$$

[Вторая формула доказывается по индукции по n, поскольку обе части формулы удовлетворяют дифференциальному уравнению $G'_{n+1}(z) = (1+q+\cdots+q^n)e^z G_n(qz)$; в упр. 1.2.6–58 равенство доказывается для $z = 0$.]

Историческая справка. Леонард Карлиц (Leonard Carlitz) ввел q-числа Стирлинга в *Transactions of the Amer. Math. Soc.* **33** (1933), 127–129. Затем в *Duke Math. J.* **15** (1948), 987–1000, он вывел (помимо прочего) соответствующее обобщение формулы 1.2.6–(45):

$$(1+q+\cdots+q^{m-1})^n \;=\; \sum_k \left\{ {n \atop k} \right\}_q q^{\binom{k}{2}} \frac{m!_q}{(m-k)!_q}.$$

31. $\exp(e^{w+z} + w - 1)$; таким образом, $\varpi_{nk} = (\varpi+1)^{n-k}\varpi^{k-1} = \varpi^{n+1-k}(\varpi-1)^{k-1}$ в обозначениях упр. 23. [L. Moser and M. Wyman, *Trans. Royal Soc. Canada* (3) **43** (1954), Section 3, 31–37.] Фактически число $\varpi_{nk}(x,1)$ из упр. 28, (г) генерируется с помощью функции $\exp((e^{xw+xz}-1)/x + xw)$.

32. Имеем $\delta_n = \varpi_n(1,-1)$, а в обобщенном треугольнике Пирса (упр. 28, (г)) при $x = 1$ и $y = -1$ легко обнаружить простую схему: $|\varpi_{nk}(1,-1)| \le 1$ и $\varpi_{n(k+1)}(1,-1) \equiv \varpi_{nk}(1,-1) + (-1)^n$ (по модулю 3) для $1 \le k < n$. [В *JACM* **20** (1973), 512–513, Гидеон Эрлих (Gideon Ehrlich) дал комбинаторное доказательство эквивалентного результата.]

33. Представление разбиения множества с помощью расстановки ладей, как в ответе к упр. 27, приводит к ответу ϖ_{nk} (путем установки $x = y = 1$ в упр. 28, (г). [Случай $k = n$ был рассмотрен в работе H. Prodinger, *Fibonacci Quarterly* **19** (1981), 463–465.]

34. (а) *Sonetti* Гвиттоне включают 149 сонетов со схемой 01010101232323, 64 сонета со схемой 01010101234234, два — со схемой 01010101234342, семь — со схемами, использованными по одному разу (наподобие 01100110234432), и 29 стихов, которые не могут считаться сонетами, так как в них нет 14 строк.

(б) *Canzoniere* Петрарки включает 115 сонетов со схемой 01100110234234, 109 — со схемой 01100110232323, 66 — со схемой 01100110234324, 7 — со схемой 01100110232232 и 20 других со схемами наподобие 01010101232323, причем ни одна из них не повторяется более четырех раз.

(в) В *Amoretti* Спенсера в 88 из 89 сонетов использована схема 01011212232344; в исключении — сонете номер 8 — использована шекспировская схема.

(г) Во всех 154 сонетах Шекспира используется очень простая схема 01012323454566, за исключением двух (99 и 126), которые не состоят из 14 строк.

(д) Все 44 сонета Браунинг из *Sonnets From the Portuguese* подчиняются схеме Петрарки 01100110232323.

Иногда строки рифмуются (случайно?), даже когда они не обязаны рифмоваться; например, последний сонет Браунинг на самом деле имеет схему 01100110121212.

Кстати, в длинной песни из *Божественной комедии* Данте используется замкнутая схема рифм, в которой $1 \equiv 3$ и $3n-1 \equiv 3n+1 \equiv 3n+3$ для $n = 1, 2, \ldots$.

35. Каждая неполная n-строчная схема Π соответствует разбиению $\{1,\ldots,n+1\}$ без частей с одним элементом, в котором $(n+1)$ группируется со всеми одноэлементными частями Π. [Г. В. Беккер (H. W. Becker) привел алгебраическое доказательство в *AMM* **48** (1941), 702. Заметим также, что согласно принципу включения и исключения $\varpi'_n =$

$\sum_k \binom{n}{k}(-1)^{n-k}\varpi_k$, а $\varpi_n = \sum_k \binom{n}{k}\varpi'_k$; фактически с использованием обозначений из упр. 23 можно записать $\varpi' = \varpi - 1$. Д. А. Шаллит (J. O. Shallit) предложил расширение треугольника Пирса путем установки $\varpi_{n(n+1)} = \varpi'_n$; см. упр. 33 и 38, (д). Фактически ϖ_{nk} представляет собой количество разбиений $\{1, \ldots, n\}$, обладающих тем свойством, что $1, \ldots, k-1$ не являются одноэлементными частями; см. H. W. Becker, *Bull. Amer. Math. Soc.* **58** (1952), 63.]

36. $\exp(e^z - 1 - z)$. (В общем случае, если ϑ_n — количество разбиений $\{1, \ldots, n\}$ на подмножества допустимых размеров $s_1 < s_2 < \cdots$, экспоненциальной производящей функцией $\sum_n \vartheta_n z^n/n!$ является $\exp(z^{s_1}/s_1! + z^{s_2}/s_2! + \cdots)$, поскольку $(z^{s_1}/s_1! + z^{s_2}/s_2! + \cdots)^k$ представляет собой экспоненциальную производящую функцию для разбиения ровно на k частей.)

37. Всего имеется $\sum_k \binom{n}{k}\varpi'_k\varpi'_{n-k}$ возможных вариантов длиной n, так что для $n = 14$ имеется 784 071 966 схем. (Но превзойти схему Пушкина очень трудно.)

38. (а) Представим, что мы начинаем с $x_1x_2\ldots x_n = 0\,1\ldots(n-1)$, затем последовательно удаляем некоторый элемент b_j и помещаем его слева, выполняя эти действия для $j = 1, 2, \ldots, n$. Тогда x_k представляет собой k-й перемещенный (считая от последнего) элемент, для $1 \le k \le |\{b_1, \ldots, b_n\}|$; см. упр. 5.2.3–36. Следовательно, массив $x_1\ldots x_n$ вернется в первоначальное состояние тогда и только тогда, когда $b_n\ldots b_1$ является ограниченно растущей строкой. [Robbins and Bolker, *Æquat. Math.* **22** (1981), 281–282.]

Другими словами, пусть $a_1\ldots a_n$ — ограниченно растущая строка. Установим $b_{-j} \leftarrow j$ и $b_{j+1} \leftarrow a_{n-j}$ для $0 \le j < n$. Затем для $1 \le j \le n$ определим k_j с помощью правила, согласно которому b_j есть k_j-й отличный от других элемент последовательности b_{j-1}, $b_{j-2}, \ldots$. Например, строка $a_1\ldots a_{16} = 0123032303456745$ соответствует таким способом σ-циклу 6688448628232384.

(б) Такие пути соответствуют ограниченно растущим строкам с $\max(a_1, \ldots, a_n) \le m$, так что искомый ответ — $\left\{{n \atop 0}\right\} + \left\{{n \atop 1}\right\} + \cdots + \left\{{n \atop m}\right\}$.

(в) Можно считать, что $i = 1$, поскольку последовательность $k_2\ldots k_nk_1$ является σ-циклом, если таковым является последовательность $k_1k_2\ldots k_n$. Таким образом, ответ представляет собой количество ограниченно растущих строк с $a_n = j-1$, а именно $\left\{{n-1 \atop j-1}\right\} + \left\{{n-1 \atop j}\right\} + \left\{{n-1 \atop j+1}\right\} + \cdots$.

(г) Если ответ — f_n, то должно выполняться соотношение $\sum_k \binom{n}{k} f_k = \varpi_n$, поскольку σ_1 представляет собой тождественную перестановку. Следовательно, $f_n = \varpi'_n$, количеству разбиений без одноэлементных частей (упр. 35).

(д) Вновь ответ ϖ'_n, в соответствии с пп. (а) и (г). [Следовательно, $\varpi'_p \bmod p = 1$, когда p простое.]

39. Замена $u = t^{p+1}$ дает $\frac{1}{p+1}\int_0^\infty e^{-u}u^{(q-p)/(p+1)}\,du = \frac{1}{p+1}\Gamma\left(\frac{q+1}{p+1}\right)$.

40. Имеем $g(z) = cz - n\ln z$, так что седловая точка находится в n/c. Теперь углы прямоугольного пути располагаются в точках $\pm n/c \pm mi/c$, а $\exp g(n/c + it) = (e^nc^n/n^n) \times \exp(-t^2c^2/(2n) + it^3c^3/(3n^2) + \cdots)$. Окончательный результат — $e^n(c/n)^{n-1}/\sqrt{2\pi n}$, умноженное на $1 + n/12 + O(n^{-2})$.

(Конечно, этот результат можно получить и быстрее, положив в интеграле $w = cz$. Но приведенный здесь ответ просто механически применяет метод седловой точки, не пытаясь прояснять суть дела.)

41. Конечный результат опять представляет собой простое умножение (21) на c^{n-1}; однако в данном случае более существенно *левое* ребро прямоугольного пути, а не правое. (Кстати, при $c = -1$ мы не можем вывести аналог (22) с использованием контура Ганкеля, когда x действительно и положительно, поскольку интеграл по такому пути расходится. Но при обычном определении z^x подходящий путь интегрирования дает при $n = x > 0$ формулу $-(\cos\pi x)/\Gamma(x)$.)

42. При четном n мы имеем $\oint e^{z^2}\,dz/z^n = 0$. В противном случае как левое, так и правое ребра прямоугольника с вершинами $\pm\sqrt{n/2} \pm in$ при больших n дают приближенно вклад

$$\frac{e^{n/2}}{2\pi(n/2)^{n/2}} \int_{-\infty}^{\infty} \exp\Bigl(-2t^2 - \frac{(-it)^3}{3}\frac{2^{3/2}}{n^{1/2}} + \frac{(it)^4}{n} - \cdots\Bigr)\,dt.$$

Мы можем ограничить $|t| \le n^\epsilon$, чтобы показать, что этот интеграл равен $I_0 + (I_4 - \frac{4}{9}I_6)/n$ с относительной ошибкой $O(n^{9\epsilon-3/2})$, где $I_k = \int_{-\infty}^{\infty} e^{-2t^2} t^k\,dt$. Как и ранее, в действительности относительная ошибка равна $O(n^{-2})$; и мы выводим окончательный ответ:

$$\frac{1}{((n-1)/2)!} = \frac{e^{n/2}}{\sqrt{2\pi}(n/2)^{n/2}}\Bigl(1 + \frac{1}{12n} + O\Bigl(\frac{1}{n^2}\Bigr)\Bigr), \qquad \text{где } n \text{ нечетно.}$$

(Аналогом (22) является $(\sin\frac{\pi x}{2})^2/\Gamma((x-1)/2)$ при $n = x > 0$.)

43. Пусть $f(z) = e^{e^z}/z^n$. Когда $z = -n + it$, мы имеем $|f(z)| < en^{-n}$; когда $z = t + 2\pi in + i\pi/2$, мы имеем $|f(z)| = |z|^{-n} < (2\pi n)^{-n}$. Так что интегралом можно пренебречь везде, кроме пути $z = \xi + it$; на этом пути $|f|$ уменьшается с возрастанием $|t|$ от 0 до π. Мы уже имеем $|f(z)|/f(\xi) = O(\exp(-n^{2\epsilon}/(\log n)^2))$ при $t = n^{\epsilon-1/2}$. Когда же $|t| > \pi$, то $|f(z)|/f(\xi) < 1/|1 + i\pi/\xi|^n = \exp(-\frac{n}{2}\ln(1 + \pi^2/\xi^2))$.

44. Положим в (25) $u = na_2t^2$, чтобы получить $\Re\int_0^\infty e^{-u}\exp(n^{-1/2}c_3(-u)^{3/2} + n^{-1}c_4(-u)^2 + n^{-3/2}c_5(-u)^{5/2} + \cdots)\,du/\sqrt{na_2u}$, где $c_k = (2/(\xi+1))^{k/2}(\xi^{k-1} + (-1)^k(k-1)!)/k! = a_k/a_2^{k/2}$. Это выражение приводит к сумме по всем разбиениям $2l$

$$b_l = \sum_{\substack{k_1+2k_2+3k_3+\cdots=2l\\ k_1+k_2+k_3+\cdots=m\\ k_1,k_2,k_3,\ldots\ge 0}} \Bigl(-\frac{1}{2}\Bigr)^{\underline{l+m}} \frac{c_3^{k_1}}{k_1!}\frac{c_4^{k_2}}{k_2!}\frac{c_5^{k_3}}{k_3!}\cdots.$$

Например, $b_1 = \frac{3}{4}c_4 - \frac{15}{16}c_3^2$.

45. Чтобы получить $\varpi_n/n!$, заменим $g(z)$ на $e^z - (n+1)\ln z$ в выводе (26). Это изменение приводит к умножению подынтегрального выражения в предыдущем ответе на $1/(1+it/\xi)$, которое представляет собой $1/(1 - n^{-1/2}a(-u)^{1/2})$, где $a = -\sqrt{2/(\xi+1)}$. Таким образом, получаем сумму

$$b_l' = \sum_{\substack{k+k_1+2k_2+3k_3+\cdots=2l\\ k_1+k_2+k_3+\cdots=m\\ k,k_1,k_2,k_3,\ldots\ge 0}} \Bigl(-\frac{1}{2}\Bigr)^{\underline{l+m}} a^k \frac{c_3^{k_1}}{k_1!}\frac{c_4^{k_2}}{k_2!}\frac{c_5^{k_3}}{k_3!}\cdots$$

из $p(2l) + p(2l-1) + \cdots + p(0)$ членов; $b_1' = \frac{3}{4}c_4 - \frac{15}{16}c_3^2 + \frac{3}{4}ac_3 - \frac{1}{2}a^2$. [Коэффициент b_1' был получен другим способом в работе L. Moser and M. Wyman, *Trans. Royal Soc. Canada* (3) **49**, Section 3 (1955), 49–54, авторы которой впервые вывели асимптотический ряд для ϖ_n. Их приближение немного менее точное, чем (26) с n, замененным на $n+1$, поскольку оно не проходит через седловую точку. Формула (26) получена в I. J. Good, *Iranian J. Science and Tech.* **4** (1975), 77–83.]

46. Из (13) и (31) видно, что для фиксированного значения k при $n \to \infty$ выполняется соотношение $\varpi_{nk} = (1 - \xi/n)^k\varpi_n(1 + O(n^{-1}))$. Это приближение справедливо и при $k = n$, но с относительной ошибкой $O((\log n)^2/n)$.

47. Шаги (H1, H2, ..., H6) выполняются $(1, \varpi_n, \varpi_n - \varpi_{n-1}, \varpi_{n-1}, \varpi_{n-1}, \varpi_{n-1} - 1)$ раз соответственно. Общее количество установок $j \leftarrow j - 1$ в цикле на шаге H4 составляет $\varpi_{n-2} + \varpi_{n-3} + \cdots + \varpi_1$ раз; общее количество установок $b_j \leftarrow m$ в цикле на шаге H6

составляет $(\varpi_{n-2}-1)+\cdots+(\varpi_1-1)$ раз. Отношение ϖ_{n-1}/ϖ_n приближенно равно $(\ln n)/n$, и $(\varpi_{n-2}+\cdots+\varpi_1)/\varpi_n \approx (\ln n)^2/n^2$.

48. Можно легко проверить взаимозамену суммирования и интегрирования в

$$\frac{e\varpi_x}{\Gamma(x+1)} = \frac{1}{2\pi i}\oint \frac{e^{e^z}}{z^{x+1}}\,dz = \frac{1}{2\pi i}\oint \sum_{k=0}^{\infty} \frac{e^{kx}}{k!\,z^{x+1}}\,dz$$
$$= \sum_{k=0}^{\infty} \frac{1}{k!}\frac{1}{2\pi i}\oint \frac{e^{kz}}{z^{x+1}}\,dz = \sum_{k=0}^{\infty} \frac{1}{k!}\frac{k^x}{\Gamma(x+1)}.$$

49. Если $\xi = \ln n - \ln\ln n + x$, то мы получаем $\beta = 1 - e^{-x} - \alpha x$. Следовательно, по формуле обращения Лагранжа (упр. 4.7–8)

$$x \;=\; \sum_{k=1}^{\infty} \frac{\beta^k}{k}\,[t^{k-1}]\left(\frac{f(t)}{1-\alpha f(t)}\right)^k \;=\; \sum_{k=1}^{\infty}\sum_{j=0}^{\infty} \frac{\beta^k}{k}\alpha^j \binom{k+j-1}{j}\,[t^{k-1}]\,f(t)^{j+k},$$

где $f(t) = t/(1-e^{-t})$. Так что окончательный результат следует из известного тождества

$$\left(\frac{z}{1-e^{-z}}\right)^m \;=\; \sum_{n=0}^{\infty} \begin{bmatrix} m \\ m-n \end{bmatrix} \frac{z^n}{(m-1)(m-2)\ldots(m-n)}.$$

(С интерпретацией этого тождества следует быть осторожным при $n \ge m$; коэффициент при z^n представляет собой полином от m степени n, как поясняется в уравнении (7.59) в *CMath*.)

Формула из этого упражнения открыта Л. Комте (L. Comtet), *Comptes Rendus Acad. Sci.* (A) **270** (Paris, 1970), 1085–1088, который распознал коэффициенты, ранее вычисленные в работе N. G. de Bruijn, *Asymptotic Methods in Analysis* (1958), 25–28. Сходимость при $n \ge e$ была показана Джеффри (Jeffrey), Корлессом (Corless), Харе (Hare) и Кнутом (Knuth), *Comptes Rendus Acad. Sci.* (I) **320** (1995), 1449–1452, которые также вывели несколько более быстро сходящуюся формулу.

(Уравнение $\xi e^{\xi} = n$ имеет и комплексные корни. Мы можем получить их все, используя $\ln n + 2\pi i m$ вместо $\ln n$ в формуле из этого упражнения; при $m \ne 0$ сумма быстро сходится. См. Corless, Gonnet, Hare, Jeffrey, and Knuth, *Advances in Computational Math.* **5** (1996), 347–350.)

50. Пусть $\xi = \xi(n)$. Тогда $\xi'(n) = \xi/((\xi+1)n)$, и можно показать, что ряд Тейлора

$$\xi(n+k) \;=\; \xi + k\xi'(n) + \frac{k^2}{2}\xi''(n) + \cdots$$

сходится при $|k| < n + 1/e$.

На самом деле справедливо гораздо большее, поскольку функция $\xi(n) = -T(-n)$ получается из функции дерева $T(z)$ путем аналитического продолжения на отрицательные значения действительной оси. (Функция дерева имеет квадратичную сингулярность в $z = e^{-1}$; после обхода этой сингулярности мы встретимся с логарифмической сингулярностью в $z = 0$, представляющей собой часть интересной многоуровневой поверхности Римана, на которой квадратичная сингулярность появляется только на уровне 0.) Производные функции дерева удовлетворяют соотношению $z^k T^{(k)}(z) = R(z)^k p_k(R(z))$, где $R(z) = T(z)/(1-T(z))$, а $p_k(x)$ — полином степени $k-1$, определяемый соотношением $p_{k+1}(x) = (1+x)^2 p_k'(x) + k(2+x)p_k(x)$. Например,

$$p_1(x) = 1,\;\; p_2(x) = 2 + x,\;\; p_3(x) = 9 + 10x + 3x^2,\;\; p_4(x) = 64 + 113x + 70x^2 + 15x^3.$$

(Коэффициенты $p_k(x)$, кстати, перечисляют некоторые филогенетические деревья, именуемые деревьями Грега: $[x^j]\,p_k(x)$ представляет собой количество ориентированных деревьев

с j непомеченными узлами и k помеченными узлами, причем все листья должны быть помечены, а непомеченные узлы должны иметь не менее двух дочерних. См. J. Felsenstein, *Systematic Zoology* **27** (1978), 27–33; L. R. Foulds and R. W. Robinson, *Lecture Notes in Math.* **829** (1980), 110–126; C. Flight, *Manuscripta* **34** (1990), 122–128.) Если $q_k(x) = p_k(-x)$, то по индукции можно доказать, что $(-1)^m q_k^{(m)}(x) \ge 0$ при $0 \le x \le 1$. Таким образом, $q_k(x)$ монотонно убывает от k^{k-1} до $(k-1)!$ при x, возрастающем от 0 до 1, для всех $k, m \ge 1$. Отсюда следует, что

$$\xi(n+k) = \xi + \frac{kx}{n} - \left(\frac{kx}{n}\right)^2 \frac{q_2(x)}{2!} + \left(\frac{kx}{n}\right)^3 \frac{q_3(x)}{3!} - \cdots, \qquad x = \frac{\xi}{\xi+1},$$

где частичные суммы при $k > 0$ оказываются поочередно больше и меньше точного значения.

51. Имеются две седловые точки, $\sigma = \sqrt{n+5/4} - 1/2$ и $\sigma' = -1 - \sigma$. Интегрирование по прямоугольному пути с вершинами $\sigma \pm im$ и $\sigma' \pm im$ показывает, что при $n \to \infty$ можно рассматривать только σ (хотя σ' вносит относительную ошибку порядка $e^{-\sqrt{n}}$, которая может оказаться существенной при малых n). Рассуждая почти так же, как и в (25), но с $g(z) = z + z^2/2 - (n+1)\ln z$, мы найдем, что хорошим приближением для t_n является

$$\frac{n!}{2\pi}\int_{-n^\epsilon}^{n^\epsilon} e^{g(\sigma) - a_2 t^2 + a_3 i t^3 + \cdots + a_l (-it)^l + O(n^{(l+1)\epsilon - (l-1)/2})}\,dt, \qquad a_k = \frac{\sigma+1}{k\sigma^{k-1}} + \frac{[k=2]}{2}.$$

Интеграл раскладывается в ряд, как в упр. 44:

$$\frac{n!\,e^{(n+\sigma)/2}}{2\sigma^{n+1}\sqrt{\pi a_2}}\bigl(1 + b_1 + b_2 + \cdots + b_m + O(n^{-m-1})\bigr).$$

Теперь $c_k = (\sigma+1)\sigma^{1-k}(1+1/(2\sigma))^{-k/2}/k$ для $k \ge 3$, следовательно, $(2\sigma+1)^{3k}\sigma^k b_k$ является полиномом от σ степени $2k$; например:

$$b_1 \;=\; \frac{3}{4}c_4 - \frac{15}{16}c_3^2 \;=\; \frac{8\sigma^2 + 7\sigma - 1}{12\sigma(2\sigma+1)^3}.$$

В частности, приближение Стирлинга и член b_1 после подстановки σ дают

$$t_n = \frac{1}{\sqrt{2}} n^{n/2} e^{-n/2+\sqrt{n}-1/4}\left(1 + \frac{7}{24}n^{-1/2} - \frac{119}{1152}n^{-1} - \frac{7933}{414720}n^{-3/2} + O(n^{-2})\right),$$

результат существенно более точный, чем формула 5.1.4–(53), причем полученный ценой значительно меньших усилий.

52. Пусть $G(z) = \sum_k \Pr(X = k) z^k$, так что j-й кумулянт κ_j равен $j!\,[t^j]\,\ln G(e^t)$. В случае (а) имеем $G(z) = e^{e^{\xi z} - e^\xi}$; следовательно,

$$\ln G(e^t) = e^{\xi e^t} - e^\xi = e^\xi(e^{\xi(e^t-1)} - 1) = e^\xi \sum_{k=1}^\infty (e^t - 1)^k \frac{\xi^k}{k!}, \quad \kappa_j = e^\xi \sum_k \left\{ {k \atop j} \right\} \xi^k [j \ne 0].$$

Случай (б) представляет собой разновидность дуальной ситуации: здесь $\kappa = j = \varpi_j\,[j \ne 0]$, поскольку

$$G(z) = e^{e^{-1}-1} \sum_{j,k} \left\{ {k \atop j} \right\} e^{-j} \frac{z^k}{k!} = e^{e^{-1}-1} \sum_j \frac{(e^{z-1} - e^{-1})^j}{j!} = e^{e^{z-1}-1}.$$

[Если $\xi e^\xi = 1$, в случае (а) мы имеем $\kappa_j = e\varpi\,[j \ne 0]$. Но если $\xi e^\xi = n$, то среднее равно $\kappa_1 = n$, а дисперсия σ^2 равна $(\xi+1)n$. Таким образом, формула в упр. 45 гласит, что среднее значение n встречается с вероятностью, приближенно равной $1/\sqrt{2\pi\sigma}$ с относительной

ошибкой $O(1/n)$. Данное наблюдение приводит к другому способу доказательства этой формулы.]

53. Можно записать $\ln G(e^t) = \mu t + \sigma^2 t^2/2 + \kappa_3 t^3/3! + \cdots$, как в формуле 1.2.10–(23); и существует такая положительная константа δ, что $\sum_{j=3}^{\infty} |\kappa_j| t^j/j! < \sigma^2 t^2/6$ при $|t| \le \delta$. Следовательно, если $0 < \epsilon < 1/2$, можно доказать, что

$$\begin{aligned}[z^{\mu n+r}]\, G(z)^n &= \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{G(e^{it})^n\, dt}{e^{it(\mu n+r)}} \\ &= \frac{1}{2\pi}\int_{-n^{\epsilon-1/2}}^{n^{\epsilon-1/2}} \exp\Bigl(-irt - \frac{\sigma^2 t^2 n}{2} + O(n^{3\epsilon-1/2})\Bigr)\, dt + O(e^{-cn^{2\epsilon}})\end{aligned}$$

при $n \to \infty$ для некоторой константы $c > 0$: подынтегральная функция при $n^{\epsilon-1/2} \le |t| \le \delta$ ограничена по абсолютному значению величиной $\exp(-\sigma^2 n^{2\epsilon}/3)$, а при $\delta \le |t| \le \pi$ ее величина не превышает α^n, где $\alpha = \max|G(e^{it})|$ меньше 1, поскольку отдельные члены $p_k e^{kit}$ в силу нашего предположения не лежат на прямой линии. Таким образом,

$$\begin{aligned}[z^{\mu n+r}]\, G(z)^n &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\Bigl(-irt - \frac{\sigma^2 t^2 n}{2} + O(n^{3\epsilon-1/2})\Bigr)\, dt + O(e^{-cn^{2\epsilon}}) \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\Bigl(-\frac{\sigma^2 n}{2}\Bigl(t + \frac{ir}{\sigma^2 n}\Bigr)^2 - \frac{r^2}{2\sigma^2 n} + O(n^{3\epsilon-1/2})\Bigr)\, dt + O(e^{-cn^{2\epsilon}}) \\ &= \frac{e^{-r^2/(2\sigma^2 n)}}{\sigma\sqrt{2\pi n}} + O(n^{3\epsilon-1}).\end{aligned}$$

Аналогично, учитывая κ_3, κ_4, $\ldots$, можно улучшить оценку до $O(n^{-m})$ для произвольно большого m; таким образом, результат верен также при $\epsilon = 0$. [В действительности такое уточнение приводит к "разложению Эджворта", в соответствии с которым $[z^{\mu n+r}]\, G(z)^n$ асимптотически стремится к

$$\frac{e^{-r^2/(2\sigma^2 n)}}{\sigma\sqrt{2\pi n}} \sum_{\substack{k_1+2k_2+3k_3+\cdots=m \\ k_1+k_2+k_3+\cdots=l \\ k_1,k_2,k_3,\ldots\ge 0 \\ 0\le s\le l+m/2}} \frac{(-1)^s(2l+m)^{\underline{2s}}}{\sigma^{4l+2m-2s}2^s s!}\, \frac{r^{2l+m-2s}}{n^{l+m-s}}\, \frac{1}{k_1!\,k_2!\,\ldots}\Bigl(\frac{\kappa_3}{3!}\Bigr)^{k_1}\Bigl(\frac{\kappa_4}{4!}\Bigr)^{k_2}\cdots.$$

Абсолютная ошибка равна $O(n^{-p/2})$, где скрытая в O константа зависит только от p и G, но не от r или n, если мы ограничимся случаями $m < p-1$. Например, при $p = 3$ мы получим

$$[z^{\mu n+r}]\, G(z)^n = \frac{e^{-r^2/(2\sigma^2 n)}}{\sigma\sqrt{2\pi n}}\Bigl(1 - \frac{\kappa_3}{2\sigma^4}\Bigl(\frac{r}{n}\Bigr) + \frac{\kappa_3}{6\sigma^6}\Bigl(\frac{r^3}{n^2}\Bigr)\Bigr) + O\Bigl(\frac{1}{n^{3/2}}\Bigr)$$

и еще семь членов при $p = 4$. [См. П. Л. Чебышев, *Записки Имп. Акад. Наук* **55** (1887), № 6, 1–16; *Acta Math.* **14** (1890), 305–315; F. Y. Edgeworth, *Trans. Cambridge Phil. Soc.* **20** (1905), 36–65, 113–141; H.Cramér, *Skandinavisk Aktuarietidsskrift* **11** (1928), 13–74, 141–180.]

54. Формулы (40) эквивалентны записи $\alpha = s\coth s + s$, $\beta = s\coth s - s$.

55. Пусть $c = \alpha e^{-\alpha}$. Итерации метода Ньютона $\beta_0 = c$, $\beta_{k+1} = (1-\beta_k)ce^{\beta_k}/(1-ce^{-\beta_k})$ быстро сходятся к точному значению, за исключением случая, когда α очень близко к 1. Например, при $\alpha = \ln 4$ величина β_7 отличается от $\ln 2$ менее чем на 10^{-75}.

56. (a) По индукции по n, $g^{(n+1)}(z) = (-1)^n\Bigl(\dfrac{\sum_{k=0}^{n}\binom{n}{k}e^{(n-k)z}}{\alpha(e^z-1)^{n+1}} - \dfrac{n!}{z^{n+1}}\Bigr)$.

(б) $\sum_{k=0}^n \left\langle {n \atop k} \right\rangle e^{k\sigma}/n! = \int_0^1 \ldots \int_0^1 \exp(\lfloor u_1 + \cdots + u_n \rfloor \sigma)\, du_1 \ldots du_n$
$< \int_0^1 \ldots \int_0^1 \exp((u_1 + \cdots + u_n)\sigma)\, du_1 \ldots du_n = (e^\sigma - 1)^n/\sigma^n.$

Нижняя граница выводится аналогично, поскольку $\lfloor u_1 + \cdots + u_n \rfloor > u_1 + \cdots + u_n - 1$.

(в) Таким образом, $n!\,(1-\beta/\alpha) < (-\sigma)^n g^{(n+1)}(\sigma) < 0$, и мы должны только убедиться, что $1 - \beta/\alpha < 2(1-\beta)$, а именно что $2\alpha\beta < \alpha + \beta$. Но в соответствии с упр. 54 $\alpha\beta < 1$ и $\alpha + \beta > 2$.

57. (а) $n + 1 - m = (n+1)(1 - 1/\alpha) < (n+1)(1 - \beta/\alpha) = (n+1)\sigma/\alpha \le 2N$, как в ответе к упр. 56, (в). (б) Величина $\alpha + \alpha\beta$ возрастает с ростом α, поскольку ее производная по α равна $1 + \beta + \beta(1-\alpha)/(1-\beta) = (1-\alpha\beta)/(1-\beta) + \beta > 0$. Следовательно, $1 - \beta < 2(1 - 1/\alpha)$.

58. (а) Производная $|e^{\sigma+it} - 1|^2/|\sigma + it|^2 = (e^{\sigma+it} - 1)(e^{\sigma-it} - 1)/(\sigma^2 + t^2)$ по t равна $(\sigma^2 + t^2)\sin t - t(2\sin\frac{t}{2})^2 - (2\sinh\frac{\sigma}{2})^2 t$, умноженному на положительную функцию. Эта производная всегда отрицательна при $0 < t \le 2\pi$, поскольку она меньше, чем $t^2 \sin t - t(2\sin\frac{t}{2})^2 = 8u\sin u\cos u(u - \tan u)$, где $t = 2u$.

Пусть $s = 2\sinh\frac{\sigma}{2}$. При $\sigma \ge \pi$ и $2\pi \le t \le 4\pi$ производная остается отрицательной, поскольку $t \le 4\pi \le s^2 - \sigma^2/(2\pi) \le s^2 - \sigma^2/t$. Аналогично, когда $\sigma \ge 2\pi$, производная остается отрицательной при $4\pi \le t \le 168\pi$; доказательство становится все легче и легче.

(б) Пусть $t = u\sigma/\sqrt{N}$. Тогда (41) и (42) доказывают, что

$$\int_{-\tau}^{\tau} e^{(n+1)g(\sigma+it)}\, dt =$$
$$\frac{(e^\sigma - 1)^m}{\sigma^n\sqrt{N}} \int_{-N^\epsilon}^{N^\epsilon} \exp\Bigl(-\frac{u^2}{2} + \frac{(-iu)^3 a_3}{N^{1/2}} + \cdots + \frac{(-iu)^l a_l}{N^{l/2-1}} + O(N^{(l+1)\epsilon-(l-1)/2})\Bigr)\, du,$$

где $(1-\beta)a_k$ — полином степени $k-1$ от α и β, причем $0 \le a_k \le 2/k$. (Например, $6a_3 = (2 - \beta(\alpha+\beta))/(1-\beta)$ и $24a_4 = (6 - \beta(\alpha^2 + 4\alpha\beta + \beta^2))/(1-\beta)$.) Монотонность подынтегральной функции показывает, что интегралом по остальному диапазону можно пренебречь. Теперь применим метод Лапласа и распространим интеграл на $-\infty < u < \infty$, а затем воспользуемся формулой из ответа к упр. 44, полагая $c_k = 2^{k/2} a_k$ для определения $b_1, b_2, \ldots$.

(в) Докажем, что $|e^z - 1|^m \sigma^{n+1}/((e^\sigma - 1)^m |z|^{n+1})$ экспоненциально мало́ на этих трех путях. Если $\sigma \le 1$, эта величина меньше, чем $1/(2\pi)^{n+1}$ (поскольку, например, $e^\sigma - 1 > \sigma$). Если $\sigma > 1$, имеем $\sigma < 2|z|$ и $|e^z - 1| \le e^\sigma - 1$.

59. В этом предельном случае $\alpha = 1 + n^{-1}$ и $\beta = 1 - n^{-1} + \frac{2}{3}n^{-2} + O(n^{-3})$; следовательно, $N = 1 + \frac{1}{3}n^{-1} + O(n^{-2})$. Старший член $\beta^{-n}/\sqrt{2\pi N}$ равен $e/\sqrt{2\pi}$, умноженному на $1 - \frac{1}{3}n^{-1} + O(n^{-2})$ (заметим, что $e/\sqrt{2\pi} \approx 1.0844$). Значение a_k в ответе к упр. 58, (б) оказывается равным $1/k + O(n^{-1})$. Так что уточняющие члены в первом приближении равны

$$\frac{b_j}{N^j} = [z^j]\, \exp\Bigl(-\sum_{k=1}^{\infty} \frac{B_{2k} z^{2k-1}}{2k(2k-1)}\Bigr) + O\Bigl(\frac{1}{n}\Bigr),$$

т. е. члены в (расходящемся) ряде соответствуют приближению Стирлинга

$$\frac{1}{1!} \sim \frac{e}{\sqrt{2\pi}}\Bigl(1 - \frac{1}{12} + \frac{1}{288} + \frac{139}{51840} - \frac{571}{2488320} - \cdots\Bigr).$$

60. (а) Количество m-арных строк длиной n, в которых встречаются все m цифр, равно $m!\left\{ {n \atop m} \right\}$, и принцип включения и исключения приводит к выражению этой величины в виде $\binom{m}{0} m^n - \binom{m}{1}(m-1)^n + \cdots$. Теперь обратитесь к упр. 7.2.1.4–37.

(б) Имеем $(m-1)^n/(m-1)! = (m^n/m!)m\exp(n\ln(1 - 1/m))$, а $\ln(1 - 1/m)$ меньше, чем $-n^{\epsilon-1}$.

(в) В этом случае $\alpha > n^\epsilon$ и $\beta = \alpha e^{-\alpha} e^\beta < \alpha e^{1-\alpha}$. Таким образом, $1 < (1-\beta/\alpha)^{m-n} < \exp(nO(e^{-\alpha}))$; а также $1 > e^{-\beta m} = e^{-(n+1)\beta/\alpha} > \exp(-nO(e^{-\alpha}))$. Так (45) превращается в $(m^n/m!)(1 + O(n^{-1}) + O(ne^{-n^\epsilon}))$.

61. Сейчас $\alpha = 1 + \frac{r}{n} + O(n^{2\epsilon-2})$ и $\beta = 1 - \frac{r}{n} + O(n^{2\epsilon-2})$. Таким образом, $N = r + O(n^{2\epsilon-1})$, и случай $l = 0$ в (43) сводится к

$$n^{\underline{r}} \left(\frac{n}{2}\right)^r \frac{e^r}{r^r\sqrt{2\pi r}} \left(1 + O(n^{2\epsilon-1}) + O\left(\frac{1}{r}\right)\right).$$

(Это приближение хорошо соответствует таким тождествам, как $\left\{{n \atop n-1}\right\} = \binom{n}{2}$ и $\left\{{n \atop n-2}\right\} = 2\binom{n}{4} + \binom{n+1}{4}$; на самом деле мы имеем

$$\left\{{n \atop n-r}\right\} = \frac{n^{2r}}{2^r r!}\left(1 + O\left(\frac{1}{n}\right)\right) \qquad \text{при } n \to \infty$$

где r — константа, в соответствии с формулами (6.42) и (6.43) из *CMath*.)

62. Утверждение истинно при $1 \le n \le 10000$ (причем $m = \lfloor e^\xi - 1 \rfloor$ в 5648 случаях). Э. Р. Канфилд (E. R. Canfield) и К. Померанц (C. Pomerance) в статье, содержащей прекрасный обзор ранних работ по связанным темам, показали, что указанное утверждение выполняется для всех достаточно больших n, и что максимум достигается в *обоих* случаях, только если $e^\xi \bmod 1$ очень близко к $\frac{1}{2}$. [*Integers* **2** (2002), A1, 1–13.]

63. (а) Результат выполняется при $p_1 = \cdots = p_n = p$, поскольку $a_{k-1}/a_k = \big(k/(n+1-k)\big) \times ((n-\mu)/\mu) \le (n-\mu)/(n+1-\mu) < 1$. Он также верен по индукции и при $p_n = 0$ или 1. В общем случае рассмотрим минимум $a_k - a_{k-1}$ по всем вариантам выбора $(p_1, \ldots, p_n)$, таким, что $p_1 + \cdots + p_n = \mu$. Если $0 < p_1 < p_2 < 1$, пусть $p_1' = p_1 - \delta$ и $p_2' = p_2 + \delta$, и заметим, что $a_k' - a_{k-1}' = a_k - a_{k-1} + \delta(p_1 - p_2 - \delta)\alpha$ для некоторого α, зависящего только от $p_3, \ldots, p_n$. В точке минимума должно быть $\alpha = 0$; таким образом, можно выбрать δ так, чтобы либо $p_1' = 0$, либо $p_2' = 1$. Следовательно, минимума можно достичь, когда все p_j имеют одно из трех значений $\{0, 1, p\}$. Но мы должны доказать, что в таких случаях $a_k - a_{k-1} > 0$.

(б) Изменение каждого p_j на $1 - p_j$ изменяет μ на $n - \mu$, а a_k на a_{n-k}.

(в) У $f(x)$ нет положительных корней. Следовательно, $f(z)/f(1)$ имеет вид, представленный в пп. (а) и (б).

(г) Пусть $C(f)$ — количество изменений знака в последовательности коэффициентов f; мы хотим показать, что $C((1-x)^2 f) = 2$. В действительности $C((1-x)^m f) = m$ для всех $m \ge 0$. $C((1-x)^m) = m$ и $C((a+bx)f) \le C(f)$ при положительных a и b; следовательно, $C((1-x)^m f) \le m$. Если $f(x)$ — любой ненулевой полином, $C((1-x)f) > C(f)$; следовательно, $C((1-x)^m f) \ge m$.

(д) Поскольку $\sum_k \left[{n \atop k}\right] x^k = x(x+1)\ldots(x+n-1)$, п. (в) применим непосредственно с $\mu = H_n$. Для полиномов $f_n(x) = \sum_k \left\{{n \atop k}\right\} x^k$ можно воспользоваться п. (в) с $\mu = \varpi_{n+1}/\varpi_n - 1$, если $f_n(x)$ имеет n действительных корней. Последнее утверждение следует по индукции, поскольку $f_{n+1}(x) = x(f_n(x) + f_n'(x))$: если $a > 0$ и если $f(x)$ имеет n действительных корней, то столько же их и у функции $g(x) = e^{ax} f(x)$. Но $g(x) \to 0$ при $x \to -\infty$; следовательно, $g'(x) = e^{ax}(af(x) + f'(x))$ также имеет n действительных корней (один далеко слева и $n-1$ — между корнями $g(x)$).

[См. E. Laguerre, *J. de Math.* (3) **9** (1883), 99–146; W. Hoeffding, *Annals Math. Stat.* **27** (1956), 713–721; J. N. Darroch, *Annals Math. Stat.* **35** (1964), 1317–1321; J. Pitman, *J. Combinatorial Theory* **A77** (1997), 297–303.]

64. Нужно применить компьютерную алгебру для вычитания $\ln \varpi_n$ из $\ln \varpi_{n-k}$.

65. Она равна ϖ_n^{-1}, умноженному на количество встречающихся k-блоков плюс количество встречающихся упорядоченных пар k-блоков в списке всех разбиений множества,

а именно $\left(\binom{n}{k}\varpi_{n-k} + \binom{n}{k}\binom{n-k}{k}\varpi_{n-2k}\right)/\varpi_n$, минус квадрат от (49). Асимптотически это значение представляет собой $(\xi^k/k!)(1+O(n^{4\epsilon-1}))$.

66. (Максимум (48) при $n = 100$ достигается только для трех разбиений $7^1 6^2 5^4 4^6 3^7 2^6 1^4$, $7^1 6^2 5^4 4^6 3^8 2^5 1^3$ и $7^1 6^2 5^4 4^7 3^6 2^6 1^3$.)

67. Математическое ожидание значения M^k равно ϖ_{n+k}/ϖ_n. Таким образом, согласно (50) среднее значение равно $\varpi_{n+1}/\varpi_n = n/\xi + \xi/\bigl(2(\xi+1)^2\bigr) + O(n^{-1})$, а дисперсия равна

$$\frac{\varpi_{n+2}}{\varpi_n} - \frac{\varpi_{n+1}^2}{\varpi_n^2} = \left(\frac{n}{\xi}\right)^2\left(1 + \frac{\xi(2\xi+1)}{(\xi+1)^2 n} - 1 - \frac{\xi^2}{(\xi+1)^2 n} + O\left(\frac{1}{n^2}\right)\right) = \frac{n}{\xi(\xi+1)} + O(1).$$

68. Максимальное количество ненулевых компонентов во всех частях разбиения равно $n = n_1 + \cdots + n_m$; оно встречается тогда и только тогда, когда все части компонентов равны 0 или 1. Тогда значения $l + 1 = n$ и $b = mn_1 + (m-1)n_2 + \cdots + n_m$ достигают максимумов. [Вот почему лучше выбирать имена элементов мультимножества так, чтобы $n_1 \le n_2 \le \cdots \le n_m$.]

69. В начале шага M3, если $k > b$ и $l = r - 1$, перейти к шагу M5. На шаге M5, если $j = a$ и $(v_j - 1)(r - l) < u_j$, перейти к шагу M6 вместо уменьшения v_j.

70. (а) $\left|{n-1 \atop r-1}\right| + \left|{n-2 \atop r-1}\right| + \cdots + \left|{r-1 \atop r-1}\right|$, поскольку $\left|{n-k \atop r-1}\right|$ включает блок $\{0, \ldots, 0, 1\}$ с k нулями. Общее количество, известное также как $p(n-1, 1)$, равно $p(n-1) + \cdots + p(1) + p(0)$.

(б) Ровно $N = \left\{{n-1 \atop r}\right\} + \left\{{n-2 \atop r-2}\right\}$ из r-блочных разбиений $\{1, \ldots, n-1, n\}$ останутся теми же при взаимообмене $n - 1 \leftrightarrow n$. Таким образом, искомый ответ равен $N + \frac{1}{2}\left(\left\{{n \atop r}\right\} - N\right) = \frac{1}{2}\left(\left\{{n \atop r}\right\} + N\right)$, что также равно числу ограниченно растущих строк $a_1 \ldots a_n$, у которых $\max(a_1, \ldots, a_n) = r - 1$ и $a_{n-1} \le a_n$. Общее количество равно $\frac{1}{2}(\varpi_n + \varpi_{n-1} + \varpi_{n-2})$.

71. $\lfloor \frac{1}{2}(n_1+1)\ldots(n_m+1) - \frac{1}{2} \rfloor$, поскольку имеется $(n_1+1)\ldots(n_m+1) - 2$ *композиций* из двух частей и половина из них находится не в лексикографическом порядке, если только все n_j не являются четными. (См. упр. 7.2.1.4–31. Формулы вплоть до пяти частей были выведены в работе E. M. Wright, *Proc. London Math. Soc.* (3) **11** (1961), 499–510.)

72. Да. Описанный далее алгоритм вычисляет $a_{jk} = p(j, k)$ для $0 \le j, k \le n$ за $\Theta(n^4)$ шагов. Начнем с установки $a_{jk} \leftarrow 1$ для всех j и k. Затем для $l = 0, 1, \ldots, n$ и $m = 0, 1, \ldots, n$ (в любом порядке), если $l + m > 1$, устанавливаем $a_{jk} \leftarrow a_{jk} + a_{(j-l)(k-m)}$ для $j = l, \ldots, n$ и $k = m, \ldots, n$ (в возрастающем порядке).

(См. табл. A.1. Аналогичный метод вычисляет $p(n_1, \ldots, n_m)$ за $O(n_1 \ldots n_m)^2$ шагов. В упомянутой ранее статье Чима (Cheema) и Моцкин (Motzkin) вывели рекуррентное соотношение

$$n_1 p(n_1, \ldots, n_m) = \sum_{l=1}^{\infty} \sum_{k_1, \ldots, k_m \ge 0} k_1 p(n_1 - k_1 l, \ldots, n_m - k_m l),$$

но эта интересная формула облегчает вычисления только в отдельных случаях.)

Таблица A.1

Количества мультиразбиений

n	0	1	2	3	4	5	6
$p(0,n)$	1	1	2	3	5	7	11
$p(1,n)$	1	2	4	7	12	19	30
$p(2,n)$	2	4	9	16	29	47	77
$p(3,n)$	3	7	16	31	57	97	162
$p(4,n)$	5	12	29	57	109	189	323
$p(5,n)$	7	19	47	97	189	339	589

n	0	1	2	3	4	5
$P(0,n)$	1	2	9	66	712	10457
$P(1,n)$	1	4	26	249	3274	56135
$P(2,n)$	2	11	92	1075	16601	325269
$P(3,n)$	5	36	371	5133	91226	2014321
$P(4,n)$	15	135	1663	26683	537813	13241402
$P(5,n)$	52	566	8155	149410	3376696	91914202

73. Да. Пусть $P(m,n) = p(1,\ldots,1,2,\ldots,2)$, где всего имеется m единиц и n двоек; тогда $P(m,0) = \varpi_m$, и можно воспользоваться рекуррентным соотношением

$$2P(m,n+1) \;=\; P(m+2,n) + P(m+1,n) + \sum_k \binom{n}{k} P(m,k).$$

Это рекуррентное соотношение можно доказать, рассмотрев, что произойдет, если мы заменим два x в мультимножестве для $P(m,n+1)$ двумя различными элементами x и x'. Мы получим $2P(m,n+1)$ разбиений, представляющих разбиения $P(m+2,n)$, за исключением $P(m+1,n)$ случаев, когда x и x' принадлежат одному и тому же блоку, и в $\binom{n}{k}P(m,n-k)$ случаях, когда блоки, содержащие x и x', идентичны и имеют k дополнительных элементов.

Примечания. См. табл. A.1. Вот еще одно рекуррентное соотношение, менее удобное для вычислений:

$$P(m+1,n) \;=\; \sum_{j,k} \binom{n}{k} \binom{n-k+m}{j} P(j,k).$$

Последовательность $P(0,n)$ была впервые изучена в работах E. K. Lloyd, *Proc. Cambridge Philos. Soc.* **103** (1988), 277–284, и G. Labelle, *Discrete Math.* **217** (2000), 237–248, где вычисления выполнялись совершенно иным образом. В упр. 70, (б) показано, что $P(m,1) = (\varpi_m+\varpi_{m+1}+\varpi_{m+2})/2$; в общем случае $P(m,n)$ можно записать с использованием обозначений из упр. 23 как $\varpi^m q_n(\varpi)$, где $q_n(x)$ — полином степени $2n$, определяемый производящей функцией $\sum_{n=0}^\infty q_n(x)z^n/n! = \exp\bigl((e^z + (x+x^2)z - 1)/2\bigr)$. Таким образом, в соответствии с упр. 31

$$\sum_{n=0}^\infty P(m,n)\frac{z^n}{n!} \;=\; e^{(e^z-1)/2} \sum_{k=0}^\infty \frac{\varpi_{(2k+m+1)(k+m+1)}}{2^k}\frac{z^k}{k!}.$$

Лабель (Labelle) доказал как частный случай гораздо более общего результата, что количество разбиений $\{1,1,\ldots,n,n\}$ ровно на r блоков равно

$$n!\,[x^r z^n]\, e^{-x+x^2(e^z-1)/2} \sum_{k=0}^\infty e^{zk(k+1)/2}\frac{x^k}{k!}.$$

75. Метод седловой точки дает $Ce^{An^{2/3}+Bn^{1/3}}/n^{55/36}$, где $A = 3\zeta(3)^{1/3}$, $B = \pi^2\zeta(3)^{-1/3}/2$ и $C = \zeta(3)^{19/36}(2\pi)^{-5/6}3^{-1/2}\exp\bigl(1/3 + B^2/4 + \zeta'(2)/(2\pi^2) - \gamma/12\bigr)$. [F. C. Auluck, *Proc. Cambridge Philos. Soc.* **49** (1953), 72–83; E. M. Wright, *American J. Math.* **80** (1958), 643–658.]

76. Воспользовавшись тем фактом, что $p(n_1,n_2,n_3,\ldots) \ge p(n_1+n_2,n_3,\ldots)$, а следовательно, $P(m+2,n) \ge P(m,n+1)$, по индукции можно доказать, что $P(m,n+1) \ge (m+n+1)P(m,n)$. Таким образом,

$$2P(m,n) \;\le\; P(m+2,n-1) + P(m+1,n-1) + eP(m,n-1).$$

Итерируя это неравенство, получаем, что $2^nP(0,n) = (\varpi^2+\varpi)^n + O(n(\varpi^2+\varpi)^{n-1}) = (n\varpi_{2n-1} + \varpi_{2n})(1 + O((\log n)^3/n))$. (Более точная асимптотическая формула может быть получена из производящей функции в ответе к упр. 73.)

78. 3 3 3 3 2 1 0 0 0
1 0 0 0 2 2 3 2 0 (Поскольку все кодированные разбиения
2 2 1 0 0 2 1 0 2 должны быть (000000000).)
2 1 0 2 2 0 0 1 3

79. Имеется 432 таких цикла. Однако они дают только 304 различных цикла разбиений множества, поскольку различные циклы могут описывать одну и ту же последовательность разбиений. Например, (000012022332321) и (000012022112123) в этом смысле эквивалентны.

80. [См. F. Chung, P. Diaconis and R. Graham, *Discrete Mathematics* **110** (1992), 52–55.] Построим ориентированный граф с ϖ_{n-1} вершинами и ϖ_n дугами; каждая ограниченно растущая строка $a_1 \ldots a_n$ определяет дугу от вершины $a_1 \ldots a_{n-1}$ к вершине $\rho(a_2 \ldots a_n)$, где ρ — функция из упр. 4. (Например, дуга 01001213 идет от 0100121 к 0110203.) Каждый универсальный цикл определяет в этом ориентированном графе цепь Эйлера; и обратно: каждая цепь Эйлера может использоваться для определения одного или нескольких ограниченно растущих универсальных последовательностей на элементах множества $\{0, 1, \ldots, n-1\}$.

Цепь Эйлера существует согласно методу из раздела 2.3.4.2, если положить, что последний выход из каждой ненулевой вершины $a_1 \ldots a_{n-1}$ идет по дуге $a_1 \ldots a_{n-1} a_{n-1}$. Однако последовательность может не быть циклической. Например, не существует универсального цикла при n, меньшем 4; при n, равном 4, универсальная последовательность 000012030110100222 определяет цикл разбиений множества, который не соответствует ни одному универсальному циклу.

Существование цикла может быть доказано для $n \geq 6$, если начать с цепи Эйлера, которая начинается с $0^n xyx^{n-3} u(uv)^{\lfloor (n-2)/2 \rfloor} u^{[n \text{ нечетно}]}$ для некоторых различных элементов $\{u, v, x, y\}$. Такой шаблон возможен, если заменить последний выход из $0^k 121^{n-3-k}$ с $0^{k-1} 121^{n-2-k}$ на $0^{k-1} 121^{n-3-k} 2$ для $2 \leq k \leq n-4$, а последними выходами из 0121^{n-4} и $01^{n-3}2$ будут соответственно $010^{n-4}1$ и $0^{n-3}10$. Теперь, если мы будем выбирать числа цикла *в обратном порядке*, таким образом определяя u и v, то можем положить x и y наименьшими элементами, отличными от $\{0, u, v\}$.

Можно найти, что количество универсальных циклов такого специального типа огромно — как минимум

$$\Bigl(\prod_{k=2}^{n-1} \bigl(k!\,(n-k)\bigr)^{\left\{{n-1 \atop k}\right\}}\Bigr) \Big/ \bigl((n-1)!\,(n-2)^3 3^{2n-5} 2^2\bigr) \qquad \text{при } n \geq 6.$$

Пока что среди них неизвестен ни один, который был бы легко декодируем. Случай $n = 5$ рассматривается ниже.

81. Заметив, что $\varpi_5 = 52$, мы используем универсальный цикл для $\{1, 2, 3, 4, 5\}$, элементами которого являются 13 треф, 13 бубен, 13 червей, 12 пик и джокер. Один такой цикл, найденный методом проб и ошибок с использованием цепи Эйлера, описанной в предыдущем упражнении, представляет собой

(♠♠♠♠♣♣♢♡J♣♡♢♢♡♠♣♣♢♡♣♢♣♡♢♢♢♣♡♣♡♣♣♣♣♣♣♢♢♢♣♢♠♠♢♡♡♡♠♡♡♡♠♠♢♢).

(В действительности всего имеется 114 056 таких циклов, если прибегать к расстановке $a_k = a_{k-1}$ в последнюю очередь и вводить джокер в колоду, как только это становится возможным.) Если считать джокер обычной картой пик, фокус получается с вероятностью $\frac{47}{52}$.

82. Имеется 13 644 решения, хотя это количество снижается до 1 981, если считать

Наименьшая сумма равна 5/2, наибольшая — 25/2; замечательное решение

представляет собой один из двух разных способов получить сумму 118/15. [Эта задача была предложена Б. А. Кордемским в его книге *Математическая смекалка* (1954) (задача 78 как в оригинальном издании, так и в английском переводе *The Moscow Puzzles* (1972)).]

РАЗДЕЛ 7.2.1.6

1. Он может "видеть" левую скобку слева от каждого внутреннего узла, а правую — снизу от каждого внутреннего узла. Можно также связать правые скобки со встречающимися *внешними* узлами, кроме последнего □; см. упр. 20.

2. **Z1.** [Инициализация.] Установить $z_k \leftarrow 2k-1$ для $0 \le k \le n$ (считаем, что $n \ge 2$.)

Z2. [Посещение.] Посетить сочетание $z_1 z_2 \ldots z_n$.

Z3. [Простой случай?] Если $z_{n-1} < z_n - 1$, установить $z_n \leftarrow z_n - 1$ и вернуться к шагу Z2.

Z4. [Поиск j.] Установить $j \leftarrow n-1$ и $z_n \leftarrow 2n-1$. Пока $z_{j-1} = z_j - 1$, устанавливать $z_j \leftarrow 2j-1$ и $j \leftarrow j-1$.

Z5. [Уменьшение z_j]. Завершить работу алгоритма, если $j = 1$. В противном случае установить $z_j \leftarrow z_j - 1$ и вернуться к шагу Z2. ▮

3. Пометим узлы леса в прямом порядке обхода. Первые $z_k - 1$ элементов $a_1 \ldots a_{2n}$ содержат $k-1$ левых скобок и $z_k - k$ правых скобок. Таким образом, имеется избыток $2k-1-z_k$ левых скобок по сравнению с правыми, когда "червяк" впервые достигает узла k; а $2k-1-z_k$ представляет собой уровень (или глубину) этого узла.

Пусть $q_1 \ldots q_n$ — инверсия $p_1 \ldots p_n$, так что узел k представляет собой q_k-й узел в обратном порядке обхода. Поскольку k находится слева от j в $p_1 \ldots p_n$ тогда и только тогда, когда $q_k < q_j$, мы видим, что c_k — количество узлов j, предшествующих k в прямом порядке обхода, но следующих за ним в обратном порядке обхода, т. е. количество истинных предков k; это вновь дает нам уровень k.

Альтернативное доказательство. Можно также показать, что обе последовательности — $z_1 \ldots z_n$ и $c_1 \ldots c_n$ — обладают, по сути, одной и той же рекурсивной структурой (5): $Z_{pq} = (Z_{p(q-1)} + 1^p),\ 1(Z_{(p-1)q} + 1^{p-1})$ при $0 \le p \le q$; а $C_{pq} = C_{p(q-1)},\ (q-p)C_{(p-1)q}$. (Рассмотрите пары к последней, предпоследней и так далее левым скобкам.)

Кстати, формула '$c_{k+1} + d_k = c_k + 1$' эквивалентна (11).

4. Почти истинно; просто строки $d_1 \ldots d_n$ и $z_1 \ldots z_n$ находятся в *убывающем* порядке, в то время как $p_1 \ldots p_n$ и $c_1 \ldots c_n$ — в возрастающем. (Это лексикографическое свойство для последовательности перестановок $p_1 \ldots p_n$ автоматически из лексикографического порядка соответствующих таблиц инверсий $c_1 \ldots c_n$ не наследуется; этот результат выполняется для данного конкретного класса $p_1 \ldots p_n$.)

5. $d_1 \ldots d_{15} = 0\,2\,0\,0\,2\,0\,0\,1\,0\,3\,2\,0\,1\,0\,4$; $z_1 \ldots z_{15} = 1\,2\,5\,6\,7\,10\,11\,12\,14\,15\,19\,22\,23\,25\,26$; $p_1 \ldots p_{15} = 2\,1\,5\,4\,8\,10\,9\,7\,11\,6\,13\,15\,14\,12\,3$; $c_1 \ldots c_{15} = 0\,1\,0\,1\,2\,1\,2\,3\,3\,4\,2\,1\,2\,2\,3$.

6. Скобки связаны одна с другой, как обычно; мы просто искривляем строку и замыкаем ее в кольцо так, что a_{2n} становится соседним с a_1, и замечаем, что разница между левыми и правыми скобками может быть восстановлена из контекста. Если a_1 соответствует низу окружности, как в табл. 1, то мы получаем приведенную диаграмму. [A. Errera, *Mémoires de la Classe Sci. 8°, Acad. Royale de Belgique* (2) **11**, 6 (1931), 26 pp.]

7. (а) Она равна `))()...()`; установка $a_1 \leftarrow$ '`(`' восстановит начальное состояние строки. (б) Будет восстановлено исходное бинарное дерево (из шага B1), за исключением того, что $l_n = n+1$.

8. $l_1 \ldots l_{15} = 2\,0\,4\,5\,0\,7\,8\,0\,10\,0\,0\,13\,0\,15\,0$; $r_1 \ldots r_{15} = 3\,0\,0\,6\,0\,12\,11\,9\,0\,0\,0\,0\,14\,0\,0$; $e_1 \ldots e_{15} = 1\,0\,3\,1\,0\,2\,2\,0\,1\,0\,0\,2\,0\,1\,0$; $s_1 \ldots s_{15} = 1\,0\,12\,1\,0\,5\,3\,0\,1\,0\,0\,3\,0\,1\,0$.

9. Узел j является (истинным) предком узла k тогда и только тогда, когда $j < k$ и $s_j + j \ge k$. (Как следствие получаем $c_1 + \cdots + c_n = s_1 + \cdots + s_n$.)

10. Если j — индекс z_k k-й левой скобки, то $w_j = c_k + 1$ и $w_{j'} = c_k$, где j' — индекс соответствующей правой скобки.

11. Поменяйте местами левые и правые скобки в $a_{2n} \ldots a_1$ для получения зеркального образа $a_1 \ldots a_{2n}$.

12. Зеркальное отображение (4) соответствует лесу

;

однако суть транспонирования станет более очевидной, если изобразить связи с правым братом и левым потомком горизонтально и вертикально, а затем выполнить транспонирование в стиле транспонирования матрицы.

13. (а) По индукции по количеству узлов можно получить*

$$\text{preorder}(F^R) = \text{postorder}(F)^R \qquad \text{и} \qquad \text{postorder}(F^R) = \text{preorder}(F)^R.$$

(б) Пусть F соответствует бинарному дереву B; тогда $\text{preorder}(F) = \text{preorder}(B)$ и $\text{postorder}(F) = \text{inorder}(B)$, как упоминалось после 2.3.2–(6). А потому $\text{preorder}(F^T) = \text{preorder}(B^R) = \text{postorder}(B)^R$ не имеет простой связи с $\text{preorder}(F)$ или $\text{postorder}(F)$. Но $\text{postorder}(F^T) = \text{inorder}(B^R) = \text{inorder}(B)^R = \text{postorder}(F)^R$.

14. В соответствии с ответом к упр. 13 $\text{postorder}(F^{RT}) = \text{preorder}(F) = \text{preorder}(B)$, где F естественным путем соответствует B; в свою очередь, $\text{postorder}(F^{TR}) = \text{preorder}(F^T)^R = \text{postorder}(B)$. Таким образом, $F^{RT} = F^{TR}$ тогда и только тогда, когда F имеет не более одного узла.

15. Если F^R естественным образом соответствует бинарному дереву B', то корнем B' является корень крайнего справа дерева в F. Левая связь узла x в B' представляет собой связь с крайним слева дочерним узлом x в F^R, который является крайним справа дочерним узлом x в F; аналогично правая связь представляет собой связь с левым братом x в F.

Примечание: поскольку B естественным образом соответствует F^{RT}, в ответе к упр. 13 говорится о том, что $\text{inorder}(B) = \text{postorder}(F^{RT}) = \text{postorder}(F^R)^R = \text{preorder}(F)$.

16. Лес $F \mid G$ получается путем размещения деревьев F под первым в обратном порядке узлом G. Ассоциативность оператора следует из того, что $F \mid (G \mid H) = (H^T G^T F^T)^T =$

* Напомним, что “preorder” означает “прямой порядок обхода”, “postorder” — “обратный”, а “inorder” — “симметричный порядок обхода”. — *Примеч. ред.*

$(F|G)|H$. Заметим, кстати, что $\mathrm{postorder}(F|G) = \mathrm{postorder}(F)\,\mathrm{postorder}(G)$ и что $F|(GH) = (F\,|\,G)H$, если G — непустой лес.

17. Любой непустой лес может быть записан как $F = (G\,|\cdot)H$, где $\cdot$ означает одноузловой лес; тогда $F^R = H^R(G^R|\cdot)$ и $F^T = (H^T|\cdot)G^T$. В частности, равенство $F^R = F^T$ невозможно, если только H не является пустым лесом Λ, поскольку первое дерево H^R не может быть $H^T\,|\cdot$; G также должно быть Λ. Кроме того, $F = F^T$ тогда и только тогда, когда $G = H^T$. В этом случае невозможно также выполнение $F^R = F^{RT}$, если только не выполняется условие $G = \Lambda$; в противном случае первое дерево G^{TR} должно иметь больше узлов, чем само G.

Похоже на то, что условие $F^{RT} = F^{TR}$ не выполняется, если только не выполняется условие $F = F^R$. При таком предположении $F^{RT} = F^{TR}$ тогда и только тогда, когда и F, и F^T являются самосопряженными. Девид Кэллан (David Callan) открыл два бесконечных семейства таких лесов с параметрами $i, j, k \geq 0$:

(В этих примерах $i = 2$, $j = 3$ и $k = 5$.) Существуют ли другие возможности?

18. Всего имеется $C_{15} = 9\,694\,845$ лесов, разбитых на $20\,982$ класса. Наибольший класс, одним из элементов которого является `(((()()))()()((()()()))()()()`, представляет собой цикл длиной $58\,968$. Самыми короткими являются шесть двухэлементных классов (в соответствии с упр. 17), состоящие из строк

```
()()()()()()()()()()()()()()(),      ()()()()(((()()()()()))()()()(),
()()()((((((()()()))))))()()(),      ()()((((((((((()()))))))))))()(),
()(((()))(((()()))))(()()))(),        ()((((((((((((()))))))))))))()
```

и их транспозиций. Несколько странноватые строки `(((((((()))))))))()()()()()()()()`, `()()()()()()()((((((((()))))))))` и `((((((((()()()()()()()))))))))` имеют клинообразные бинарные деревья и образуют единый класс размером 3. Элемент (2) содержится в пути, проходящем от `()((()(()()))(())(((()))()))()` до `(((())))(()()))(()))(()()))(()()))` и содержащем 3120 элементов. В соответствии с гипотезой из ответа к упр. 19 кратчайший возможный цикл имеет длину 6; при $n = 15$ имеется 66 таких циклов. (Следующий в порядке возрастания размера цикл имеет длину 10 и включает в себя строку `()(()()())()((((()))()))((()))).`)

19. Преобразование F_j в F_{j+1} алгоритмом P можно перефразировать следующим образом: “найти последний узел в прямом порядке обхода, скажем, x, который имеет левого брата, скажем, y. Удалить x из его семейства и сделать его новым крайним справа дочерним узлом y. И если $x < n$, заменить всех потомков x $(x + 1, \ldots, n)$ тривиальными одноузловыми деревьями”.

Таким образом, преобразование F_j^R в F_{j+1}^R может быть описано следующим образом, если вспомнить, что k-й узел F_j в прямом порядке обхода представляет собой k-й с конца

узел F_j^R в обратном порядке обхода: "найти первый узел в обратном порядке обхода, скажем, x, который имеет правого брата, скажем, y. Удалить x из его семейства и сделать его новым крайним слева дочерним узлом y. И если $x > 1$, заменить всех потомков x ($x-1$, ..., 1) тривиальными одноузловыми деревьями".

Аналогично можно перефразировать преобразование G_j в G_{j+1}, выполняемое алгоритмом B: "найти j, корень крайнего слева нетривиального дерева; затем найти k, его крайний справа дочерний узел. Удалить k и его потомки из семейства j и вставить их между j и правым братом j. Наконец, если $j > 1$, сделать j и его правые братья дочерними по отношению к $j-1$, $j-1$ — дочерним по отношению к $j-2$ и т. д.".

Когда это преобразование изменяет представление с левым братом и правым сыном с G_j^{RT} на G_{j+1}^{RT} (см. упр. 15), оно оказывается идентичным преобразованию F_j^R в F_{j+1}^R в представлении с левым сыном и правым братом. Таким образом, $G_j^{RT} = F_j^R$, поскольку выполнение этого тождества при $j = 1$ очевидно.

(Отсюда следует, что последовательность таблиц $e_1 \ldots e_{n-1}$ для бинарных деревьев, сгенерированных алгоритмом B, совпадает с последовательностью таблиц $d_{n-1} \ldots d_1$ для строк скобок, сгенерированных алгоритмом P; это явление продемонстрировано в табл. 1 и 2.)

Ряд симметрий списков лесов был изучен М. Ч. Эром (M. C. Er) в *Comp. J.* **32** (1989), 76–85.

20. (а) Это утверждение, обобщающее лемму 2.3.1P, легко доказать по индукции.

(б) Приведенная ниже процедура практически идентична алгоритму P.

T1. [Инициализация.] Установить $b_{3k-2} \leftarrow 3$ и $b_{3k-1} \leftarrow b_{3k} \leftarrow 0$ для $1 \le k \le n$; установить также $b_0 \leftarrow b_N \leftarrow 0$ и $m \leftarrow N-3$, где $N = 3n+1$.

T2. [Посещение.] Посетить $b_1 \ldots b_N$. (Сейчас $b_m = 3$ и $b_{m+1} \ldots b_N = 0 \ldots 0$.)

T3. [Простой случай?] Установить $b_m \leftarrow 0$. Если $b_{m-1} = 0$, установить $b_{m-1} \leftarrow 3$, $m \leftarrow m-1$ и перейти к шагу T2.

T4. [Поиск j.] Установить $j \leftarrow m-1$ и $k \leftarrow N-3$. Пока $b_j = 3$, устанавливать $b_j \leftarrow 0$, $b_k \leftarrow 3$, $j \leftarrow j-1$ и $k \leftarrow k-3$.

T5. [Увеличение b_j.] Завершить работу алгоритма, если $j = 0$. В противном случае установить $b_j \leftarrow 3$, $m \leftarrow N-3$ и вернуться к шагу T2. ▌

[См. Ш. Закс (S. Zaks), *Theoretical Comp. Sci.* **10** (1980), 63–82. В этой статье Закс указывает, что еще проще сгенерировать последовательность $z_1 \ldots z_n$ индексов j, таких, что $b_j = 3$, с использованием алгоритма, почти идентичного алгоритму из ответа к упр. 2, поскольку сочетание $z_1 \ldots z_n$ для корректного тернарного дерева характеризуется неравенствами $z_{k-1} < z_k \le 3k-2$.]

21. Для решения этой задачи можно, по сути, скомбинировать алгоритм P с алгоритмом 7.2.1.2L. Для удобства будем считать, что $n_t > 0$ и $n_1 + \cdots + n_t > 1$.

G1. [Инициализация.] Установить $l \leftarrow N$. Затем для $j = t, \ldots, 2, 1$ (в указанном порядке) выполнить n_j раз следующие операции: установить $b_{l-j} \leftarrow j$, $b_{l-j+1} \leftarrow \cdots \leftarrow b_{l-1} \leftarrow 0$ и $l \leftarrow l-j$. Наконец установить $b_0 \leftarrow b_N \leftarrow c_0 \leftarrow 0$ и $m \leftarrow N-t$.

G2. [Посещение.] Посетить $b_1 \ldots b_N$. (В этот момент $b_m > 0$ и $b_{m+1} = \cdots = b_N = 0$.)

G3. [Простой случай?] Если $b_{m-1} = 0$, установить $b_{m-1} \leftarrow b_m$, $b_m \leftarrow 0$, $m \leftarrow m-1$ и вернуться к шагу G2.

G4. [Поиск j.] Установить $c_1 \leftarrow b_m$, $b_m \leftarrow 0$, $j \leftarrow m-1$ и $k \leftarrow 1$. Пока $b_j \ge c_k$, устанавливать $k \leftarrow k+1$, $c_k \leftarrow b_j$, $b_j \leftarrow 0$ и $j \leftarrow j-1$.

G5. [Увеличение b_j.] Если $b_j > 0$, найти наименьшее $l \ge 1$, такое, что $b_j < c_l$, и выполнить обмен $b_j \leftrightarrow c_l$. В противном случае при $j > 0$, установить $b_j \leftarrow c_1$ и $c_1 \leftarrow 0$. В противном случае завершить работу алгоритма.

G6. [Обращение и развертывание.] Установить $j \leftarrow k$ и $l \leftarrow N$. Пока $c_j > 0$, устанавливать $b_{l-c_j} \leftarrow c_j$, $l \leftarrow l - c_j$ и $j \leftarrow j - 1$. Затем установить $m \leftarrow N - c_k$ и вернуться к шагу G2. ▮

В этом алгоритме предполагается, что $N > n_1 + 2n_2 + \cdots + tn_t$. [См. *SICOMP* **8** (1979), 73–81.]

22. Вначале заметим, что значение d_1 может увеличиваться тогда и только тогда, когда в связанном представлении $r_1 = 0$. В противном случае следующий за $d_1 \ldots d_{n-1}$ элемент получается путем поиска наименьшего j, такого, что $d_j > 0$, и установки $d_j \leftarrow 0$, $d_{j+1} \leftarrow d_{j+1} + 1$. Можно считать, что $n > 2$.

K1. [Инициализация.] Установить $l_k \leftarrow k + 1$ и $r_k \leftarrow 0$ для $1 \le k < n$; установить также $l_n \leftarrow r_n \leftarrow 0$.

K2. [Посещение.] Посетить бинарное дерево, представленное связями $l_1 l_2 \ldots l_n$ и $r_1 r_2 \ldots r_n$.

K3. [Простой случай?] Установить $y \leftarrow r_1$. Если $y = 0$, установить $r_1 \leftarrow 2$, $l_1 \leftarrow 0$ и вернуться к шагу K2. В противном случае при $l_1 = 0$ установить $l_1 \leftarrow 2$, $r_1 \leftarrow r_2$, $r_2 \leftarrow l_2$, $l_2 \leftarrow 0$ и вернуться к шагу K2. В противном случае установить $j \leftarrow 2$ и $k \leftarrow 1$.

K4. [Поиск j и k.] Если $r_j > 0$, установить $k \leftarrow j$ и $y \leftarrow r_j$. Затем, если $j \ne y - 1$, установить $j \leftarrow j + 1$ и повторить данный шаг.

K5. [Перетасовка поддеревьев.] Установить $l_j \leftarrow y$, $r_j \leftarrow r_y$, $r_y \leftarrow l_y$ и $l_y \leftarrow 0$. Если $j = k$, перейти к шагу K2.

K6. [Сдвиг поддеревьев.] Завершить работу алгоритма, если $y = n$. В противном случае, пока $k > 1$, устанавливать $k \leftarrow k - 1$, $j \leftarrow j - 1$ и $r_j \leftarrow r_k$. Затем, пока $j > 1$, устанавливать $j \leftarrow j - 1$ и $r_j \leftarrow 0$. Вернуться к шагу K2. ▮

(См. анализ алгоритма в упр. 45. Корш [*Comp. J.* **48** (2005), 488–497; **49** (2006), 351–357; **54** (2011), 776–785] показал, что данный алгоритм, а также алгоритмы P и B можно интересным способом расширить для t-арных деревьев.)

23. (а) Поскольку переменная z_n начинает со значения $2n - 1$ и проходит назад и вперед C_{n-1} раз, в конце ее значение оказывается равным $2n - 1 - (C_{n-1} \bmod 2)$, где $n > 1$. Кроме того, последнее значение z_j представляет собой константу для всех $n \ge j$. Таким образом, последняя строка $z_1 z_2 \ldots$ имеет вид 1 2 5 6 9 11 13 14 17 19 ... и содержит все нечетные числа $< 2n$ за исключением 3, 7, 15, 31, ...

(б) Аналогично перестановка в прямом порядке обхода, характеризующая последнее дерево, представляет собой $2^k\ 2^{k-1}\ \ldots\ 1\ 3\ 5\ 6\ 7\ 9\ 10\ \ldots$, где $k = \lfloor \lg n \rfloor$. В лесу узел 2^j является родительским по отношению к 2^{j-1} узлам $\{2^{j-1}, 2^{j-1}+1, \ldots, 2^j - 1\}$, где $1 < j \le k$, а деревья $\{2^k + 1, \ldots, n\}$ тривиальны.

Примечание. Если алгоритм N после его завершения перезапустить с шага N2, он будет генерировать ту же последовательность, но в обратном порядке. Тем же свойством обладает и алгоритм L.

24. $l_0 l_1 \ldots l_{15} = 2\,0\,1\,0\,3\,0\,0\,6\,5\,0\,8\,0\,0\,12\,11\,4$; $r_1 \ldots r_{15} = 0\,15\,0\,10\,7\,0\,0\,9\,0\,14\,13\,0\,0\,0\,0$; $k_1 \ldots k_{15} = 0\,0\,2\,2\,4\,5\,5\,4\,8\,4\,10\,11\,11\,10\,2$; $q_1 \ldots q_{15} = 2\,1\,15\,4\,3\,10\,8\,5\,7\,6\,9\,14\,11\,13\,12$ и $u_1 \ldots u_{15} = 12\,3\,1\,0\,0\,5\,0\,3\,1\,0\,0\,1\,0\,1\,0$. (Если узлы леса F пронумерованы в обратном порядке обхода, k_j является левым братом j; или, если j — крайний слева дочерний узел узла p, то $k_j = k_p$. Иначе говоря, k_j — родитель j в лесу F^{TR}. И k_j также равно $j - 1 - u_{n+1-j}$, количеству элементов слева от j в строке $q_1 \ldots q_n$, которые меньше, чем j.)

25. Используя подсказки из алгоритмов N и L, мы хотим расширить каждое $(n-1)$-узловое дерево до списка из двух или большего количества n-узловых деревьев. Идея в этом случае заключается в том, чтобы сделать n дочерним узлом $n-1$ в бинарном дереве в начале и в конце каждого такого списка. В приведенном далее алгоритме используются

дополнительные поля связи p_j и s_j, где p_j указывает на родителя j в лесу, а s_j указывает на левого брата j или на крайнего правого брата j, если узел j — крайний слева в своем семействе. (Эти указатели p_j и s_j, конечно же, не являются перестановками $p_1 \ldots p_n$ из табл. 1 или $s_1 \ldots s_n$ из табл. 2. Фактически $s_1 \ldots s_n$ представляет собой перестановку λ из упр. 33.)

M1. [Инициализация.] Установить $l_j \leftarrow j+1$, $r_j \leftarrow 0$, $s_j \leftarrow j$, $p_j \leftarrow j-1$ и $o_j \leftarrow -1$ для $1 \le j \le n$, за исключением $l_n \leftarrow 0$.

M2. [Посещение.] Посетить $l_1 \ldots l_n$ и $r_1 \ldots r_n$. Затем установить $j \leftarrow n$.

M3. [Поиск j.] Если $o_j > 0$, установить $k \leftarrow p_j$ и перейти к шагу M5, если $k \ne j-1$. Если $o_j < 0$, установить $k \leftarrow s_j$ и перейти к шагу M4, если $k \ne j-1$. Если $k = j-1$, в любом случае установить $o_j \leftarrow -o_j$, $j \leftarrow j-1$ и повторить этот шаг.

M4. [Передача вниз.] (В этот момент k является левым братом j или крайним справа членом семейства j.) Если $k \ge j$, завершить работу алгоритма при $j = 1$, в противном случае установить $x \leftarrow p_j$, $l_x \leftarrow 0$, $z \leftarrow k$ и $k \leftarrow 0$ (тем самым отсоединив узел j от его родителя и выведя его на верхний уровень). Но если $k < j$, установить $x \leftarrow p_j + 1$, $z \leftarrow s_x$, $r_k \leftarrow 0$ и $s_x \leftarrow k$ (тем самым отсоединив узел j от k и перенеся его на уровень вниз). Затем установить $x \leftarrow k+1$, $y \leftarrow s_x$, $s_x \leftarrow z$, $s_j \leftarrow y$, $r_y \leftarrow j$ и $x \leftarrow j$. Пока $x \ne 0$, устанавливать $p_x \leftarrow k$ и $x \leftarrow r_x$. Вернуться к шагу M2.

M5. [Передача вверх.] (В этот момент k является родителем j.) Установить $x \leftarrow k+1$, $y \leftarrow s_j$, $z \leftarrow s_x$, $s_x \leftarrow y$ и $r_y \leftarrow 0$. Если $k \ne 0$, установить $y \leftarrow p_k$, $r_k \leftarrow j$, $s_j \leftarrow k$, $s_{y+1} \leftarrow z$ и $x \leftarrow j$; в противном случае установить $y \leftarrow j-1$, $l_y \leftarrow j$, $s_j \leftarrow z$ и $x \leftarrow j$. Пока $x \ne 0$, устанавливать $p_x \leftarrow y$ и $x \leftarrow r_x$. Вернуться к шагу M2. ▮

Примечания о времени работы алгоритма. Можно доказать, как в упр. 44, что стоимость шага M3 составляет $2C_n + 3(C_{n-1} + \cdots + C_1)$ обращений к памяти и что шаги M4 и M5 вместе стоят $8C_n - 2(C_{n-1} + \cdots + C_1)$ плюс удвоенное количество присваиваний $x \leftarrow r_x$. Последнюю величину трудно точно проанализировать; например, при $n = 15$ и $j = 6$ алгоритм устанавливает $x \leftarrow r_x$ ровно $(1, 2, 3, 4, 5, 6)$ раз в $(45, 23, 7, 9, 2, 4)$ случаях соответственно. Однако эвристически среднее количество присваиваний $x \leftarrow r_x$ должно составлять примерно $2 - 2^{j-n}$ для заданного j, т. е. всего около $(2C_n - (C_n - C_{n-1}) - (C_{n-1} - C_{n-2})/2 - (C_{n-2} - C_{n-3})/4 - \cdots)/C_n \approx 8/7$. Эмпирические тесты подтверждают предсказанное поведение, показывая, что общая стоимость в пересчете на одно дерево стремится к $265/21 \approx 12.6$ обращений к памяти при $n \to \infty$.

26. (а) Необходимость условия очевидна. А если оно выполняется, можно построить F единственным образом: узел 1 и его братья являются корнями леса, а их потомки индуктивно определяются пересекающимися разбиениями. (Фактически можно вычислить координаты глубины $c_1 \ldots c_n$ непосредственно из ограниченно растущей строки Π — $a_1 \ldots a_n$ следующим образом: установить $c_1 \leftarrow 0$ и $i_0 \leftarrow 0$. Для $2 \le j \le n$, если $a_j > \max(a_1, \ldots, a_{j-1})$, установить $c_j \leftarrow c_{j-1} + 1$ и $i_{a_j} \leftarrow c_j$, в противном случае установить $c_j \leftarrow i_{a_j}$.)

(б) Если Π и Π' удовлетворяют условию непересекаемости, то их наибольшим общим утончением является $\Pi \vee \Pi'$, так что можно поступить так, как в упр. 7.2.1.5–12, (а).

(в) Пусть $x_1, \ldots, x_m$ — дочерние узлы некоторого узла в F и пусть $1 \le j < k \le m$. Образуем F' путем удаления $x_{j+1}, \ldots, x_k$ из их семейства и присоединения их в качестве дочерних узлов $x_{j+1} - 1$, крайнего правого потомка x_j.

(г) Очевидно из ответа к п. (в). Таким образом, леса ранжируются снизу вверх по количеству содержащихся в них внутренних узлов (которое на единицу меньше количества блоков в Π).

(д) Ровно $\sum_{k=0}^{n} e_k(e_k - 1)/2$, где $e_0 = n - e_1 - \cdots - e_n$ представляет собой количество корней.

(е) Дуализация подобна операции транспонирования в упр. 12, но вместо левого сына и правого брата мы используем левого брата и правого сына и выполняем транспонирование относительно *младшей* диагонали.

("Правые" связи теперь указывают вниз. Обратите внимание, что j является крайним справа дочерним узлом k в лесу F тогда и только тогда, когда j является левым братом k в F^D. Прямой порядок обхода F^D представляет собой обращение прямого порядка обхода F, так же как обратный порядок обхода F^T представляет собой обращение обратного порядка обхода F.)

(ж) Из (е) видно, что F' покрывает F тогда и только тогда, когда F^D покрывает F'^D. (Следовательно, F^D имеет $n+1-k$ листьев, если у F их k.)

(з) $F \barwedge F' = (F^D \veebar F'^D)^D$.

(и) Нет. Если бы это было так, то в соответствии с дуальностью должно было бы выполняться равенство. Но, например, $0101 \barwedge 0121 = 0000$ и $0101 \veebar 0121 = 0123$, в то время как $\text{leaves}(0101) + \text{leaves}(0121) \neq \text{leaves}(0000) + \text{leaves}(0123)$*.

[Непересекающиеся разбиения впервые были рассмотрены Г. В. Беккером (H. W. Becker) в *Math. Mag.* **22** (1948), 23–26. В 1971 году Г. Креверас (G. Kreweras) доказал, что они образуют решетку; см. ссылки в ответе к упр. 2.3.4.6–3.]

27. (а) Это утверждение эквивалентно упр. 2.3.3–19.

(б) Если представить лес с использованием связей с правым сыном и левым братом, прямой порядок обхода будет соответствовать симметричному обходу бинарного дерева (см. упр. 2.3.2–5), а s_j будет равно размеру правого поддерева узла j. Поворот влево вокруг любого внутреннего узла этого бинарного дерева уменьшает ровно одну из координат s, и величина уменьшения мала настолько, насколько это возможно при условии корректности таблицы $s_1 \ldots s_n$. Следовательно, F' покрывает F тогда и только тогда, когда F получается из F' с помощью такого поворота. (Поворот в представлении с левым сыном и правым братом аналогичен, но по отношению к обратному порядку обхода.)

(в) Дуализация сохраняет отношение покрытия, но заменяет "лево" на "право" и наоборот.

(г) $F \top F' = (F^D \perp F'^D)^D$. Таким же образом, как указывается в упр. 6.2.3–32, можно независимо минимизировать размеры левых поддеревьев.

(д) Покрывающее преобразование в ответе 26, (в) очевидным образом делает $s_j \le s'_j$ для всех j.

(е) Истинно, поскольку $F \barwedge F' \ltimes F \dashv F \perp F'$ и $F \barwedge F' \ltimes F' \dashv F \perp F'$.

(ж) Ложно; например, $0121 \veebar 0122 = 0123$ и $0121 \top 0122 = 0122$. (Но, применив дуализацию к выражению из ответа (е), получим $F \top F' \dashv F \veebar F'$.)

(з) Длиннейший путь (длиной $\binom{n}{2}$) получается путем многократного уменьшения крайнего справа ненулевого значения s_j на 1. Кратчайший путь длиной $n-1$ получается путем многократного присваивания крайнему слева ненулевому s_j значения 0.

---

* leaves() означает количество листьев у аргумента этой функции. — *Примеч. пер.*

В ответе к упр. 6.2.3–32 содержится много ссылок на литературу о решетках Тамари.

28. (а) Нужно просто вычислить $\min(c_1, c_1') \ldots \min(c_n, c_n')$ и $\max(c_1, c_1') \ldots \max(c_n, c_n')$, поскольку $c_1 \ldots c_n$ является корректной последовательностью глубин тогда и только тогда, когда $c_1 = 0$ и $c_j \le c_{j-1} + 1$ для $1 < j \le n$.

(б) Это очевидно вытекает из ответа (а). *Примечание*: элементы любой дистрибутивной решетки могут быть представлены как порядковые идеалы некоторого частичного упорядочения. Для случая, приведенного на рис. 62, это частичное упорядочение показано справа; аналогичная треугольная сетка со сторонами длиной $n-2$ дает решетку Стенли порядка n.

(в) Возьмем узел k леса F, у которого имеется левый брат j. Удалим k из его семейства и поместим его в качестве нового правого дочернего узла j, за которым разместим его бывшие дочерние узлы в качестве новых дочерних узлов j; бывшие дочерние узлы k сохраняют своих потомков. (Эта операция соответствует замене '`)(`' на '`()`' в строке вложенных скобок. Таким образом, "идеальный" код Грея для скобок соответствует гамильтонову пути в покрывающем графе решетки Стенли. Для $n = 4$ имеется 38 таких путей, а именно $(8, 6, 6, 8, 4, 6)$ путей от 0123 до $(0001, 0010, 0012, 0100, 0111, 0120)$ соответственно.)

(г) Истинно, поскольку отношение покрытия из ответа (в) обладает лево-правой симметрией. ($F \subseteq F'$ тогда и только тогда, когда $w_j \le w_j'$ для $0 \le j \le 2n$, где глубины червяка w_j определены в упр. 10. Если $w_0 \ldots w_{2n}$ представляет собой обход червяком леса F, то обратная последовательность $w_{2n} \ldots w_0$ представляет собой обход червяком леса F^R. Обратите внимание, что отношение покрытия изменяет только одну координату w_j. Вычислить $F \cap F'$ и $F \cup F'$ можно, если искать минимумы и максимумы для w, а не для c.)

(д) См. упр. 9. (Таким образом, $F \perp F' \subseteq F \cap F'$, и т. д., как в упр. 27, (е).)

Примечания: Стенли разработал свою решетку в *Fibonacci Quarterly* **13** (1975), 222–223. Поскольку на одних и тех же элементах определены три важные решетки, требуются три различные обозначения для соответствующих упорядочений; здесь использованы символы $\mathrel{|\!<}$, $\dashv$ и $\subseteq$, напоминающие первые буквы фамилий Креверас, Тамари и Стенли.*

29. Если "склеить" шесть правильных пятиугольников, то получатся 14 вершин, координаты которых после соответствующих поворотов и масштабирования будут следующими:

$$p_{1010} = p_{0000}^{-} = p_{3000}^{*} = p_{2100}^{*-} = (-1, \sqrt{3}, 2/\phi);$$
$$p_{0010} = p_{3100}^{*} = (\phi^{-2}, \sqrt{3}\,\phi, 0); \quad p_{3010} = p_{0100}^{-} = (0, 0, 2); \quad p_{3210} = p_{0200}^{-} = (2, 0, 2/\phi);$$
$$p_{0210} = p_{3200}^{*} = (\sqrt{5}, \sqrt{3}, 0); \quad p_{1000} = p_{2000}^{*} = (-\phi^2, \sqrt{3}/\phi, 0);$$

где $(x, y, z)^*$ означает $(x, -y, z)$, а $(x, y, z)^-$ означает $(x, y, -z)$. Но тогда три четырехугольные "грани" не являются квадратами; более того, их вершины даже не лежат в одной плоскости.

(Можно получить похожее тело с правильными квадратами и неправильными пятиугольниками, соединив два подходящих тетраэдра и обрезав три склеенных угла. Альтернативные множества координат для ассоциаэдра, которые представляют определенный математический интерес, но не обладают внешней привлекательностью, рассматриваются Гюнтером Циглером (Günter Ziegler) в *Lectures on Polytopes* (New York: Springer, 1995), пример 9.11.)

30. (а) $\bar{f}_{n-1} \ldots \bar{f}_1\, 0$, поскольку внутренний узел j в симметричном порядке обхода имеет непустое правое поддерево тогда и только тогда, когда внутренний узел $j+1$ в симметричном порядке обхода имеет пустое левое поддерево.

(б) В общем случае, если след имеет вид $1^{p_1} 0^{q_1+1} 1^{p_2+1} 0^{q_2+1} \ldots 1^{p_k+1} 0^{q_k+1}$, нам нужно подсчитать все бинарные деревья, узлы которых в симметричном порядке обхода имеют

* В оригинале указано, что имеется в виду написание фамилии Стенли по-русски. — *Примеч. пер.*

спецификацию $R^{p_1}NL^{q_1}BR^{p_2}NL^{q_2}B\ldots R^{p_k}NL^{q_k}$, где B означает "оба поддерева не пусты", R означает "правое поддерево не пустое, левое — пустое", L означает "левое поддерево не пустое, правое — пустое", а N означает "оба поддерева пусты". Это количество в общем случае равно

$$\binom{p_1+q_1}{p_1}\binom{p_2+q_2}{p_2}\ldots\binom{p_k+q_k}{p_k}C_{k-1}$$

и в конкретном указанном в условии случае составляет $\binom{1+0}{1}\binom{0+0}{0}\binom{1+0}{1}\binom{5+3}{5}\binom{0+0}{0}\binom{0+0}{0}\times\binom{0+2}{0}\binom{0+0}{0}\binom{1+2}{1}C_8 = 240\,240$.

(в) В соответствии с упр. 3 $d_j = 0$ тогда и только тогда, когда $c_{j+1} > c_j$.

(г) В соответствии с упр. 27, (а) в общем случае след $F \perp F'$ равен $f_1\ldots f_n \wedge f'_1\ldots f'_n$; в соответствии с упр. 27, (г) след $F \top F'$ равен $f_1\ldots f_n \vee f'_1\ldots f'_n$.

[Тот факт, что в решетке Тамари всегда имеется комплементарный элемент, доказан Г. Лаксером (H. Lakser); см. G. Grätzer, *General Lattice Theory* (1978), упр. I.6.30.]

31. (а) 2^{n-1}; см. упр. 6.2.2–5.

(б) $c_1 \le \cdots \le c_n$; $d_1, \ldots, d_{n-1} \le 1$; из $e_j > 0$ следует $e_j + \cdots + e_n = n - j$; $k_{j+1} \le k_j + 1$; $p_1 \le \cdots \le p_j \ge \cdots \ge p_n$ для некоторого j; из $s_j > 0$ следует $s_j = n - j$; $u_1 \ge \cdots \ge u_n$; $z_{j+1} \le z_j + 2$. (Прочие ограничения, применимые в общем случае, снижают количество возможных кортежей с данными свойствами до 2^{n-1} в каждом из случаев.)

(в) Истинно только в n случаях из 2^{n-1}. (Но F^T *является* вырожденным.)

(г) Вырожденный лес со следом $f_1\ldots f_n$ обладает тем свойством, что $c_{j+1} = c_j + f_j$. Элементы $j < k$ являются братьями тогда и только тогда, когда $f_j = f_{j+1} = \cdots = f_{k-1} = 0$. Таким образом, если F'' является вырожденным лесом со следом $f_1\ldots f_n \wedge f'_1\ldots f'_n$, то $F'' \ltimes F$ и $F'' \ltimes F'$; следовательно, $F'' \ltimes F \barwedge F' \dashv F \perp F'$. Согласно ответу (б) мы также имеем $F \perp F' \dashv F''$. Аналогично доказывается и то, что $F \veebar F' = F \top F'$ представляет собой вырожденный лес со следом $f_1\ldots f_n \vee f'_1\ldots f'_n$.

Таким образом, когда решетки Кравераса и Тамари ограничены вырожденными лесами, они становятся идентичными булевой решетке подмножеств множества $\{1, \ldots, n-1\}$. [Этот результат в случае решетки Тамари открыт Джорджем Марковски (George Markowsky), *Order* **9** (1992), 265–290, в статье которого содержится также описание многих других свойств решеток Тамари.]

32. Предположим, что s-координатами F и F' являются соответственно $s_1\ldots s_n$ и $s'_1\ldots s'_n$. Назовем индекс j *замороженным*, если $s_j < s'_j$ или $j = 0$. Мы хотим определить значения замороженных координат и максимизировать остальные координаты. Пусть $s_0 = n$ и пусть для $0 \le k \le n$

$$s''_k = s_j - k + j, \quad \text{где } j = \max\{i \mid 0 \le i \le k,\ i \text{ заморожено, а } i + s_i \ge k\}.$$

Поскольку $s_k \le s_j - (k-j)$ при $0 \le k-j \le s_j$, имеем $s''_k \ge s_k$, причем равенство достигается, когда k — замороженный индекс.

Значения $s''_0 s''_1\ldots s''_n$ соответствуют корректному лесу в соответствии с условием из упр. 27, (а). Если $k \ge 0$ и $0 \le l \le s''_k = s_j - k + j$ и $s''_{k+l} = s_{j'} - k - l + j'$, то мы имеем $s''_{k+l} + l \le s''_k$ для $0 \le j' - j \le s_j$, потому что в этом случае $s_{j'} + j' - j \le s_j$. Поскольку $j + s_j \ge k + l \ge j'$, не может быть $j > j'$ или $j' > j + s_j$.

Пусть F''' представляет собой лес с координатами, удовлетворяющими условию $s_k \le s'''_k \le s''_k$. Тогда $\min(s'_k, s'''_k) = s_k$, поскольку $s_k = s''_k$ при замороженном k, в противном случае $s_k = s'_k$.

И наоборот, если F''' — лес, такой, что $F' \perp F''' = F$, должно выполняться $s_k \le s'''_k \le s''_k$. Условие $s'''_k < s_k$ влечет за собой $s'''_k < s'_k$. А если k — минимальное значение, для которого $s'''_k > s''_k$, то мы имеем $s''_k = s_j - k + j$ для некоторого замороженного j, где $0 \le j \le k$ и $j + s_j \ge k$. Тогда из $s'''_j \ge s_j$ вытекает $k - j \le s'''_j$, следовательно, $s'''_k + k - j \le s'''_j$.

Если $j < k$, мы имеем $s_j''' \le s_j'' = s_j$, т. е. получаем противоречие. Но из $j = k$ вытекает $\min(s_k''', s_k') > s_k$.

Чтобы получить первый полудистрибутивный закон, применим этот принцип, заменив F на $F \perp G$, а F' на F. Тогда из гипотез $F \dashv G \dashv F''$ и $F \dashv H \dashv F''$ следует, что $F \dashv G \top H \dashv F''$. Второй полудистрибутивный закон получается путем применения дуальности к первому.

(Ральф Фриз (Ralph Freese) предложил называть F'' *псевдодополнением* (*pseudo-complement*) F' над F.)

33. (а) Пусть $k\lambda = \texttt{LLINK}[k]$, если $\texttt{LLINK}[k] \ne 0$, в противном случае пусть оно принимает значение $\texttt{RLINK}[k-1]$, если $k \ne 1$, а если это не так, то пусть $k\lambda$ равно корню бинарного дерева. Это правило определяет перестановку, поскольку $k\lambda = j$ тогда и только тогда, когда $k = \mathrm{parent}(j) + [j$ — правый дочерний узел$]$ или $k = 1$ и j является корнем. Кроме того, $k\lambda \ge k$, когда $\texttt{LLINK}[k] = 0$, и $k\sigma\lambda \le k$, когда $\texttt{RLINK}[k] = 0$. [Обобщение на t-арные деревья можно найти в работе P. H. Edelman, *Discrete Math.* **40** (1982), 171–179.]

(б) Используя представление (2) из ответа к упр. 26, (е), мы видим, что в этом случае $\lambda(F)$ равно $(3\,1)(2)(12\,6\,4)(5)(11\,7)(14\,13)(9\,8)(15)(10)$. В общем случае циклы представляют собой семейства лесов, расположенные в уменьшающемся порядке в пределах каждого цикла; узлы пронумерованы в прямом порядке обхода. [См. Dershowitz and Zaks, *Discrete Math.* **62** (1986), 215–218.]

(в) $\lambda(F^D) = \rho\sigma\lambda(F)\rho$, где ρ — "зеркальная" перестановка $(1\,n)(2\,n{-}1)\ldots$, поскольку в дуальном лесу выполняются обмен $\texttt{LLINK} \leftrightarrow \texttt{RLINK}$ и зеркальное отражение нумерации в прямом порядке обхода.

(г) Разбиение цикла $(x_j\,x_k)(x_1\ldots x_m) = (x_1\ldots x_j x_{k+1}\ldots x_m)(x_{j+1}\ldots x_k)$ соответствует ответу к упр. 26, (в).

(д) Согласно ответу к п. (г), каждый покрывающий путь соответствует разложению $(n\ldots 2\,1)$. Пусть q_n обозначает количество таких разложений. Тогда мы получаем рекуррентное соотношение $q_1 = 1$ и $q_n = \sum_{l=1}^{n-1}(n{-}l)\binom{n-2}{l-1}q_l q_{n-l}$, поскольку имеется $n{-}l$ выборов, где $k{-}j = l$, при которых первая транспозиция разбивает цикл на части размерами l и $n{-}l$, после чего имеется $\binom{n-2}{l-1}$ способов чередования последующих разложений. Решением рекуррентного соотношения является $q_n = n^{n-2}$, поскольку

$$\sum_{l=1}^{n-1}\binom{n-1}{l}l^{l-1}(y-l)^{n-1-l} = \lim_{x\to 0}\sum_{l=1}^{n-1}\binom{n-1}{l}(x+l)^{l-1}(y-l)^{n-1-l}$$
$$= \lim_{x\to 0}\frac{(x+y)^{n-1}-y^{n-1}}{x} = (n-1)y^{n-1}.$$

[См. J. Dénes, *Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* **4** (1959), 63–70. Естественным представляется поиск соответствия между факторизацией и помеченными свободными деревьями, поскольку количество и тех, и других равно n^{n-2}. Вероятно, простейшее из них для данного $(1\,2\ldots n) = (x_1\,y_1)\ldots(x_{n-1}\,y_{n-1})$, где $x_j < y_j$, следующее. Предположим, что цикл, содержащий x_j и y_j в $(x_j\,y_j)\ldots(x_{n-1}\,y_{n-1})$, представляет собой $(z_1\ldots z_m)$, где $z_1 < \cdots < z_m$. Если $y_j = z_m$, пусть $a_j = z_1$, в противном случае пусть $a_j = \min\{z_i \mid z_i > x_j\}$. Можно показать, что $a_1\ldots a_{n-1}$ — последовательность парковки $n-1$ машин, а в упр. 6.4–31 показана ее связь со свободными деревьями.]

34. Каждый покрывающий путь снизу вверх эквивалентен диаграмме Юнга формы $(n-1, n-2, \ldots, 1)$, так что можно применить теорему 5.1.4H (см. упр. 5.3.4–38).

[Подсчет таких путей в решетке Тамари остается загадкой: соответствующая последовательность имеет вид 1, 1, 2, 9, 98, 2981, 340549, ...]

35. Умножьте на $n+1$ и обратитесь к *AMM* **97** (1990), 626–630.

36. Путем очевидных поправок к шагам T1–T5 можно обобщить результат для t-арных деревьев, $t \ge 1$. Пусть $C_n^{(t)}$ — количество t-арных деревьев с n внутренними узлами; таким образом, $C_n = C_n^{(2)}$ и $C_n^{(t)} = ((t-1)n+1)^{-1}\binom{tn}{n}$. Если между посещениями изменяется h степеней b_j, то $h \ge x$ в $C_{n-x}^{(t)}$ случаях. Так что простой случай осуществляется с вероятностью $1 - C_{n-1}^{(t)}/C_n^{(t)} \approx 1-(t-1)^{t-1}/t^t$, и среднее количество установок $b_j \leftarrow 0$ на шаге T4 составляет $(C_{n-1}^{(t)} + \cdots + C_1^{(t)})/C_n^{(t)} \approx (t-1)^{t-1}/(t^t - (t-1)^{t-1})$, или 4/23 при $t = 3$. Можно также изучить t-арную рекурсивную структуру $A_{pq}^{(t)} = 0\,A_{p(q-1)}^{(t)},\ t\,A_{(p-1)q}^{(t)}$, $0 \le (t-1)p \le q \ne 0$, обобщающую (5). Количество таких последовательностей степеней $C_{pq}^{(t)}$ удовлетворяет рекуррентному соотношению (21) с тем исключением, что $C_{pq}^{(t)} = 0$ при $p < 0$ или $(t-1)p > q$. Общее решение имеет вид

$$C_{pq}^{(t)} = \frac{q-(t-1)p+1}{q+1}\binom{p+q}{p} = \binom{p+q}{p} - (t-1)\binom{p+q}{p-1},$$

и $C_n^{(t)} = C_{n((t-1)n)}^{(t)}$. Треугольник для $t = 3$ начинается так, как показано справа.

1				
1				
1	1			
1	2			
1	3	3		
1	4	7		
1	5	12	12	
1	6	18	30	
1	7	25	55	55
1	8	33	88	143

[Числа Фусса–Каталана $C_n^{(t)}$ впервые были рассмотрены в работе N. Fuss, *Nova acta acad. scient. imp. Pet.* **9** (1791), 243–251.]

37. Базовое лексикографическое рекуррентное соотношение для всех таких лесов имеет вид

$$A(n_0, n_1, \ldots, n_t) = 0\,A(n_0-1, n_1, \ldots, n_t),\ 1\,A(n_0, n_1-1, \ldots, n_t),\ \ldots,\ t\,A(n_0, n_1, \ldots, n_t-1),$$

где $n_0 > n_2 + 2n_3 + \cdots + (t-1)n_t$ и $n_1, \ldots, n_t \ge 0$; в противном случае $A(n_0, n_1, \ldots, n_t)$ — пустые последовательности, за исключением последовательности $A(0, \ldots, 0) = \epsilon$, состоящей из одной пустой строки. На шаге G1 вычисляется первый элемент $A(n_0, \ldots, n_t)$. Мы хотим проанализировать пять величин:

C, количество выполнений шага G2 (общее количество лесов);
E, количество переходов от шага G3 к шагу G2 (количество простых случаев);
K, количество перемещений некоторых b_i в список c на шаге G4;
L, количество сравнений b_j с некоторым c_i на шаге G5;
Z, количество установок $c_1 \leftarrow 0$ на шаге G5.

Тогда всего присваивание $b_{l-c_j} \leftarrow c_j$ в цикле на шаге G6 выполняется $K - Z - n_1 - \cdots - n_t$ раз.

Пусть n — вектор $(n_0, n_1, \ldots, n_t)$ и пусть e_j — единичный вектор с 1 в позиции j. Пусть $|n| = n_0 + n_1 + \cdots + n_t$ и $\|n\| = n_1 + 2n_2 + \cdots + tn_t$. Используя эти обозначения, можно записать приведенное выше базовое рекуррентное соотношение в более удобном виде:

$$A(n) = 0\,A(n-e_0),\ 1\,A(n-e_1),\ \ldots,\ t\,A(n-e_t) \qquad \text{при } |n| > \|n\|.$$

Рассмотрим общее рекуррентное соотношение

$$F(n) = f(n) + \Bigl(\sum_{j=0}^{t} F(n-e_j)\Bigr)\bigl[|n| > \|n\|\bigr],$$

где $F(n) = 0$, если вектор n имеет отрицательный компонент. Если $f(n) = [|n| = 0]$, то $F(n) = C(n)$ — общее количество лесов. В соответствии с ответом к упр. 2.3.4.4–32

$$C(n) = \frac{(|n|-1)!\,(|n|-\|n\|)}{n_0!\,n_1!\ldots n_t!} = \sum_{j=0}^{t}(1-j)\binom{|n|-1}{n_0, \ldots, n_{j-1}, n_j-1, n_{j+1}, \ldots, n_t},$$

что является обобщением формулы для $C_{pq}^{(t)}$ из ответа к упр. 36 (которая представляет собой случай $n_0 = (t-1)q+1$ и $n_t = p$). Аналогично при выборе других функций ядра $f(n)$ мы получаем рекуррентные соотношения для остальных величин $E(n)$, $K(n)$, $L(n)$ и $Z(n)$, необходимых для нашего анализа:

$$\begin{array}{lll}
f(n) = \bigl[|n| = n_0 + 1 \text{ и } n_0 > \|n\|\bigr] & \text{дает} & F(n) = E(n); \\
f(n) = \bigl[|n| > n_0\bigr] & \text{дает} & F(n) = E(n) + K(n); \\
f(n) = \bigl[|n| = \|n\| + 1\bigr] & \text{дает} & F(n) = C(n) + K(n) - Z(n); \\
f(n) = \sum_{1\le j<k\le t} n_j[n_k > 0] & \text{дает} & F(n) = L(n).
\end{array}$$

Символьные методы из упр. 2.3.4.4–32, похоже, не в состоянии привести к быстрому решению этих более общих рекуррентных соотношений, но можно легко установить значение $C - E$, заметив, что $b_m + m < N$ на шаге G2 тогда и только тогда, когда предыдущим шагом был G3. Следовательно,

$$C(n) - E(n) = \sum_{j=1}^{t} C(n - f_j), \qquad \text{где } f_j = e_j - (j-1)e_0;$$

эта сумма перечисляет все подлеса, в которых $n_1 + \cdots + n_t$ — количество внутренних узлов (не являющихся листьями) — уменьшено на 1. Аналогично через

$$C^{(x)}(n) = \sum \{C(n - i_1 f_1 - \cdots - i_t f_t) \mid i_1 + \cdots + i_t = x\}$$

можно обозначить количество подлесов, имеющих $n_1 + \cdots + n_t - x$ внутренних узлов. Тогда мы имеем

$$K(n) - Z(n) = \sum_{x=1}^{|n|} C^{(x)}(n).$$

Эта формула аналогична (20), поскольку $k - [b_j = 0] \ge x \ge 1$ на шаге G5 тогда и только тогда, когда $b_{m-x} > 0$ и $b_{m-x+1} \ge \cdots \ge b_m$. Такие строки степеней в прямом порядке обхода взаимно однозначно соответствуют лесам из $C^{(x)}(n)$, если удалить из строки $b_1 \ldots b_N$ подстроку $b_{m-x+1} \ldots b_m$ и соответствующее количество завершающих нулей.

Из этих формул можно заключить, что алгоритм Закса–Ричардса при $n_1 = n_2 + \cdots + n_t + O(1)$ требует только $O(1)$ операций на одно посещение леса, поскольку $C(n - f_j)/C(n) = n_j n_0^{\underline{j-1}}/(|n|-1)^{\underline{j}} \le 1/4 + O(|n|^{-1})$ при $j > 1$. На самом деле значение K достаточно мало́ почти во всех случаях, представляющих практический интерес. Однако при больших значениях n_1 алгоритм может оказаться медленным. Например, если $t = 1$, $n_0 = m + r + 1$ и $n_1 = m$, алгоритм, по сути, вычисляет все r-сочетания $m + r$ элементов; тогда $C(n) = \binom{m+r}{r}$ и $K(n) - Z(n) = \binom{m+r}{r+1} = \Omega(mC(n))$ при фиксированном r. [Чтобы обеспечить эффективность во всех случаях, можно отслеживать завершающие единицы; см. Ruskey and R?lants van Baronaigien, *Congressus Numerantium* **41** (1984), 53–62.]

Точные формулы для K, Z и (в особенности) L, похоже, слишком сложны, но эти величины можно вычислить следующим образом. Назовем "активным блоком" леса крайнюю справа подстроку ненулевых степеней; например, активным блоком 302102021230000000 является 2123. Все перестановки активного блока встречаются одинаково часто. Действительно, обозначим через $D(n)$ сумму величин "`завершающие_нули`$(\beta) - 1$", взятую по всем строкам степеней в прямом порядке обхода β для лесов со спецификацией n. Тогда блок с n'_j вхождениями j для $1 \le j \le t$ является активным ровно в $D(n - n'_1 f_1 - \cdots - n'_t f_t) + [n'_1 + \cdots + n'_t = n_1 + \cdots + n_t]$ случаях. Например, для получения леса с активным блоком 2123 можно вставить подстроку 21230000 в три места заданной строки 3021020000. Вклады

в K и L при выравнивании активного блока влево (когда перед ним нет ни одного нуля) могут быть вычислены, как в упр. 7.2.1.2–6, а именно

$$k(n) = w(e_{n_1}(z)\dots e_{n_t}(z)), \qquad l(n) = w\Big(e_{n_1}(z)\dots e_{n_t}(z)\sum_{1\le i<j\le t}(n_i - zr_i(z))r_j(z)\Big)$$

с использованием обозначений из упомянутого упражнения. Аналогичные вклады получаются и в общем случае; следовательно,

$$K(n) = k(n) + \sum D(n-n')k(n'),\ L(n) = l(n) + \sum D(n-n')l(n'),\ Z(n) = \sum D(n-n'),$$

где суммирование выполняется по всем векторам n', таким, что $n'_j \le n_j$ для $1 \le j \le t$ и $|n'| - \|n'\| = |n| - \|n\|$ и $n'_1 + \cdots + n'_t \le n_1 + \cdots + n_t - 2$.

Остается определить $D(n)$. Пусть $C(n;j)$ означает количество лесов со спецификацией $n = (n_0, \dots, n_t)$, в которых последний внутренний узел в прямом порядке обхода имеет степень j. Тогда

$$C(n) = \sum_{j=1}^{t} C(n;j) \text{ и } C(n+e_1;1) = C(n+e_2;2) = \cdots = C(n+e_t;t) = C(n) + D(n).$$

Из этой бесконечной системы линейных уравнений можно вывести, что $C(n) + D(n)$ равно

$$\sum_{i_2=0}^{n_2}\cdots\sum_{i_t=0}^{n_t}(-1)^{i_2+\cdots+i_t}\binom{i_2+\cdots+i_t}{i_2,\dots,i_t}C(n+(1+i_2+\cdots+i_t)e_1 - i_2f_2 - \cdots - i_tf_t).$$

Разумеется, было бы желательно получить более простое выражение, если, конечно, оно существует.

38. Очевидно, что на шаге L1 выполняется $4n+2$ обращений к памяти. Переход от шага L3 к шагу L4 или L5 выполняется ровно $C_j - C_{j-1}$ раз для конкретного значения j; следовательно, его стоимость составляет $2C_n + 3\sum_{j=0}^{n}(n-j)(C_j - C_{j-1}) = 2C_n + 3(C_{n-1} + \cdots + C_1 + C_0)$ обращений к памяти. Шаги L4 и L5 имеют суммарную стоимость $6C_n - 6$. Таким образом, весь процесс требует $9 + O(n^{-1/2})$ обращений к памяти на одно посещение.

39. Диаграмма Юнга формы (q,p) и записи y_{ij} соответствуют элементу A_{pq}, который имеет левые скобки в позициях $p+q+1-y_{21}, \dots, p+q+1-y_{2p}$, а правые — в позициях $p+q+1-y_{11}, \dots, p+q+1-y_{1q}$. Длины уголков равны $\{q+1, q, \dots, 1, p, p-1, \dots, 1\}\setminus\{q-p+1\}$; так что $C_{pq} = (p+q)!(q-p+1)/(p!(q+1)!)$ согласно теореме 5.1.4H.

40. (а) $C_{pq} = \binom{p+q}{p} - \binom{p+q}{p-1} \equiv \binom{p+q}{p} + \binom{p+q}{p-1} = \binom{p+q+1}{p}$ (по модулю 2); теперь воспользуйтесь упр. 1.2.6–11. (б) Из 7.1.3–(36) мы знаем, что $\nu(n \,\&\, (n+1)) = \nu(n+1) - 1$.

41. Она равна $C(wz)/(1 - zC(wz)) = 1/(1 - z - wzC(wz)) = (1 - wC(wz))/(1 - w - z)$, где $C(z)$ — производящая функция для чисел Каталана (18). Легко видеть, что первая из этих формул, $C(wz) + zC(wz)^2 + z^2C(wz)^3 + \cdots$, эквивалентна (24). [См. P. A. MacMahon, *Combinatory Analysis* **1** (Cambridge Univ. Press, 1915), 128–130.]

42. (а) Элементы $a_1 \dots a_n$ определяют полностью самосопряженную строку вложенных скобок $a_1 \dots a_{2n}$, причем имеется $C_{q(n-q)}$ вариантов $a_1 \dots a_n$ с q правыми скобками. Таким образом, окончательный ответ имеет вид

$$\sum_{q=0}^{\lfloor n/2\rfloor} C_{q(n-q)} = \sum_{q=0}^{\lfloor n/2\rfloor}\left(\binom{n}{q} - \binom{n}{q-1}\right) = \binom{n}{\lfloor n/2\rfloor}.$$

(б) Ровно $C_{(n-1)/2}$ [n нечетно], поскольку самотранспонированное бинарное дерево определяется его левым поддеревом. (в) Ответ тот же, поскольку F является самодуальным тогда и только тогда, когда F^R является самотранспонированным.

43. По индукции по $q-p$ получаем $C_{pq} = C_q - \binom{q-p-1}{1}C_{q-1} + \cdots = \sum_{r=0}^{q-p}(-1)^r\binom{q-p-r}{r}C_{q-r}$.

44. Количество обращений к памяти между посещениями составляет $3j-2$ на шаге B3, $h+1$ на шаге B4 и 4 на шаге B5, где h — количество присваиваний $y \leftarrow r_y$. Количество бинарных деревьев, у которых $h \geq x$, для заданных j и x равно $[z^{n-j-x-1}]\,C(z)^{x+3}$ при $j<n$, поскольку такие деревья получаются путем присоединения $x+3$ поддеревьев ниже $j+x+1$ внутренних узлов. Присваивая $x=0$ и используя формулу (24) и упр. 43, мы находим, что данное значение j встречается $C_{(n-j-1)(n-j+1)} = C_{n+1-j} - C_{n-j}$ раз. Таким образом, сумма $\sum j$ по всем бинарным деревьям равна $n+\sum_{j=1}^{n}(C_{n+1-j}-C_{n-j})j = C_n + C_{n-1} + \cdots + C_1$. Аналогично сумма $\sum(h+1)$ равна $\sum_{j=1}^{n-1}\sum_{x=0}^{n-j-1} C_{(n-j-x-1)(n-j+1)} = \sum_{j=1}^{n-1} C_{(n-j-1)(n-j+2)} = \sum_{j=1}^{n}(C_{n-j+2} - 2C_{n-j+1}) = C_{n+1} - (C_n + C_{n-1} + \cdots + C_0)$. Таким образом, общая стоимость алгоритма составляет $C_{n+1} + 4C_n + 2(C_{n-1} + \cdots + C_1) + O(n) = \bigl(26/3 - 10/(3n) + O(n^{-2})\bigr)C_n$ обращений к памяти.

45. Каждый из простых случаев на шаге K3 встречается C_{n-1} раз, так что общая стоимость этого шага составляет $3C_{n-1}+8C_{n-1}+2(C_n-2C_{n-1})$ обращений к памяти. Выборка r_i на шаге K4 выполняется $[z^{n-i-1}]\,C(z)^{i+2} = C_{(n-i-1)n}$ раз; суммирование по $i \geq 2$ дает общее количество обращений к памяти в этом цикле, равное $C_{(n-3)(n+1)} = C_{n+1} - 3C_n + C_{n-1}$. Стоимость шага K5 равна $6C_n - 12C_{n-1}$; шаг K6 немного более сложен, но можно показать, что операция $r_j \leftarrow r_k$ выполняется $C_n - 3C_{n-1} + 1$ раз при $n > 2$, в то время как операция $r_j \leftarrow 0$ выполняется $C_{n-1} - n + 1$ раз. Таким образом, общее количество обращений к памяти равно $C_{n+1} + 7C_n - 9C_{n-1} + n + 3 = \bigl(8.75 - 9.375/n + O(n^{-2})\bigr)C_n$.

Хотя это общее количество асимптотически хуже, чем у алгоритма B в ответе к упр. 44, большой отрицательный коэффициент при n^{-1} означает, что в действительности алгоритм B выигрывает только при $n \geq 58$; а такие большие значения n на практике не встречаются.

Однако Скарбек (Skarbek) улучшил алгоритм B до приведенного далее алгоритма B*, который генерирует деревья в обратном порядке с помощью вспомогательной таблицы $c_1 \ldots c_n$.

B1*. [Инициализация.] Установить $l_k \leftarrow c_k \leftarrow 0$ и $r_k \leftarrow k+1$ для $1 \leq k < n$; установить также $l_n \leftarrow r_n \leftarrow 0$ и установить $r_{n+1} \leftarrow 1$ (для удобства работы на шаге B3*).

B2*. [Посещение.] Посетить бинарное дерево, представленное массивами связей $l_1 l_2 \ldots l_n$ и $r_1 r_2 \ldots r_n$.

B3*. [Поиск j.] Установить $j \leftarrow 1$. Пока $r_j = 0$, устанавливать $l_j \leftarrow c_j \leftarrow 0$, $r_j \leftarrow j+1$ и $j \leftarrow j+1$. Затем завершить работу алгоритма, если $j > n$.

B4*. [Понижение r_j.] Установить $x \leftarrow r_j$, $r_j \leftarrow r_x$, $r_x \leftarrow 0$, $z \leftarrow c_j$, $c_j \leftarrow x$. Если $z > 0$, установить $r_z \leftarrow x$; в противном случае установить $l_j \leftarrow x$. Вернуться к шагу B2*. ▮

Если хранить значения r_1 и c_1 в регистрах, данному алгоритму потребуется только $4C_n + C_{n-1} + 4(C_{n-1} + C_{n-2} + \cdots + C_0) + 3n - 6 = \bigl(67/12 + 73/(24n) + O(n^{-2})\bigr)C_n$ обращений к памяти для генерации всех C_n деревьев. [См. W. Skarbek, *Fundamenta Informatic?* **75** (2007), 505–536.]

46. (а) Движение влево от $\textcircled{pq}$ увеличивает площадь на $q-p$.

(б) Шаги влево на пути от $\textcircled{nn}$ до $\textcircled{00}$ соответствуют левым скобкам в $a_1 \ldots a_{2n}$, и на таком k-м шаге мы имеем $q-p = c_k$.

(в) Эквивалентно $C_{n+1}(x) = \sum_{k=0}^{n} x^k C_k(x) C_{n-k}(x)$. Это рекуррентное соотношение выполняется, поскольку $(n+1)$-узловой лес F состоит из корня крайнего слева дерева с k-узловым лесом F_l (потомками этого корня) и $(n-k)$-узлового леса F_r (остальные деревья), и поскольку

$$\begin{aligned}\text{длина внутреннего пути}(F) \;=\; & k + \text{длина внутреннего пути}(F_l)\\ & + \text{длина внутреннего пути}(F_r).\end{aligned}$$

(г) Строки $A_{p(p+r)}$ имеют вид $\alpha_0)\alpha_1)\ldots\alpha_{r-1})\alpha_r$, где каждое α_j — подстрока корректно вложенных скобок. Площадь такой строки равна сумме по всем j площадей α_j плюс $r-j$, умноженное на количество левых скобок в α_j.

Примечания. Полиномы $C_{pq}(x)$ введены Л. Карлицем (L. Carlitz) и Д. Риорданом (J. Riordan) в *Duke Math. J.* **31** (1964), 371–388; тождество из п. (г) эквивалентно их формуле (10.12). Они также доказали, что

$$C_{pq}(x) \;=\; \sum_r (-1)^r x^{r(r-1)-\binom{q-p}{2}} \binom{q-p-r}{r}_x C_{q-r}(x),$$

что обобщает результат упр. 43. Из п. (в) получается бесконечная цепная дробь $C(x,z) = 1/(1-z/(1-xz/(1-x^2z/(1-\cdots))))$, для которой Д. Н. Ватсон (G. N. Watson) доказал ее равенство $F(x,z)/F(x,z/x)$, где

$$F(x,z) = \sum_{n=0}^{\infty} \frac{(-1)^n\, x^{n^2} z^n}{(1-x)(1-x^2)\ldots(1-x^n)};$$

см. *J. London Math. Soc.* **4** (1929), 39–48. Мы уже встречались с этой производящей функцией в несколько измененном виде в упр. 5.2.1–15.

Длина внутреннего пути леса представляет собой "длину левого пути" соответствующего бинарного дерева, а именно сумму количества левых ветвлений на пути от корня по всем внутренним узлам. Более общий полином

$$C_n(x,y) = \sum x^{\text{длина левого пути}(T)} y^{\text{длина правого пути}(T)},$$

где сумма берется по всем n-узловым бинарным деревьям T, похоже, не имеет простого аддитивного рекуррентного соотношения наподобие соотношения для $C_{nn}(x) = C_n(x,1)$, рассмотренного в этом упражнении; однако мы имеем $C_{n+1}(x,y) = \sum_k x^k C_k(x,y)\, y^{n-k} \times C_{n-k}(x,y)$. Следовательно, сверхпроизводящая функция $C(x,y,z) = \sum_n C_n(x,y) z^n$ удовлетворяет функциональному уравнению $C(x,y,z) = 1 + zC(x,y,xz)C(x,y,yz)$. (Случай $x=y$ был рассмотрен в упр. 2.3.4.5–5.)

47. $C_n(x) = \sum_q x^{\binom{q-p}{2}} C_{pq}(x) C_{(n-q)(n-1-p)}(x)$ для $0 \le p < n$.

48. Пусть, с использованием обозначений из упр. 46, $\bar{C}(z) = C(-1,z)$ и пусть $\bar{C}(z)\bar{C}(-z) = F(z^2)$. Тогда $\bar{C}(z) = 1 + zF(z^2)$ и $\bar{C}(-z) = 1 - zF(z^2)$; так что $F(z) = 1 - zF(z)^2$ и $F(z) = C(-z)$. Отсюда следует, что $C_{pq}(-1) = [z^p]\, C(-z^2)^{\lceil (q-p)/2 \rceil} (1 + zC(-z^2))^{[q-p \text{ четно}]}$, что при четном q равно $(-1)^{(p/2)} C_{(p/2)(q/2-1)}[p \text{ четно}]$, а при нечетном q равно $(-1)^{\lfloor p/2 \rfloor} C_{\lfloor p/2 \rfloor \lfloor q/2 \rfloor}$. Идеальный код Грея для строк A_{pq} может существовать, только если $|C_{pq}(-1)| \le 1$, поскольку соответствующий граф — двудольный (см. рис. 62); $|C_{pq}(-1)|$ представляет собой разность между размерами частей, поскольку каждый идеальный переход изменяет $c_1 + \cdots + c_n$ на ± 1.

49. Алгоритм U при $n=15$ и $N=10^6$ дает строку `()(()())((((()))))(((((()))))`.

50. Внесите следующие изменения в алгоритм U. На шаге U1 добавьте присваивание $r \leftarrow 0$. На шаге U3 замените проверку $N \le c'$ проверкой $a_m =$ '`)`'. На шаге U4 замените присваивание $N \leftarrow N - c'$ присваиванием $r \leftarrow r + c'$. Кроме того, на шагах U3 и U4 опустите присваивание a_m.

Строка из (1), оказывается, имеет ранг 3 141 592. (Кто бы мог подумать?)

51. По теореме 7.2.1.3L $N = \binom{\bar{z}_1}{n} + \binom{\bar{z}_2}{n-1} + \cdots + \binom{\bar{z}_n}{1}$; следовательно, $\kappa_n N = \binom{\bar{z}_1}{n-1} + \binom{\bar{z}_2}{n-2} + \cdots + \binom{\bar{z}_n}{0}$, поскольку $\bar{z}_n \ge 1$. Теперь заметим, что $N - \kappa_n N$ представляет собой ранг $z_1 z_2 \ldots z_n$ согласно формуле (23) и упр. 50. (Пусть, например, $z_1 \ldots z_4 = 1256$ с рангом 6 в табл. 1. Тогда $\bar{z}_1 \ldots \bar{z}_4 = 7632$, $N = 60$ и $\kappa_4 60 = 54$. Заметим, что N достаточно велико, поскольку $\bar{z}_1 = 2n-1$; из рис. 47 видно, что $\kappa_n N$ обычно *превышает* N, когда N мало́.)

52. Количество завершающих правых скобок имеет то же распределение, что и количество ведущих левых скобок, а последовательность вложенных строк, начинающаяся с '$(^k)$', представляет собой $(^k)A_{(n-k)(n-1)}$. Следовательно, вероятность того, что $d_n = k$, равна $C_{(n-k)(n-1)}/C_n$. Используя 1.2.6–(25), можно найти, что

$$\sum_{k=0}^{n}\binom{k}{t}C_{(n-k)(n-1)} = \sum_{k=0}^{n}\left(\binom{2n-1-k}{n-1} - \binom{2n-1-k}{n}\right)\binom{k}{t}$$
$$= \binom{2n}{n+t} - \binom{2n}{n+t+1} = C_{(n-t)(n+t)},$$

откуда следует, что среднее значение и дисперсия равны соответственно $3n/(n+2) = 3 - 6/(n+2)$ и $2n(2n^2 - n - 1)/((n+2)^2(n+3)) = 4 + O(n^{-1})$. [Моменты этого распределения впервые были вычислены Р. Кемпом (R. Kemp) в *Acta Informatica* **35** (1998), 17–89, теорема 9. Заметим, что $c_n = d_n - 1$ обладает, по сути, тем же поведением.]

53. (а) $3n/(n+2)$ в соответствии с упр. 52.

(б) H_n в соответствии с упр. 6.2.2–7.

(в) $2 - 2^{-n}$ (по индукции).

(г) Любая (фиксированная) последовательность левых и правых ветвлений имеет одно и то же распределение шагов до того, как встретится лист (другими словами, вероятность встретить узел с бинарным обозначением Дьюи 01101 та же, что и вероятность встретить узел 00000). Таким образом, если $X = k$ с вероятностью p_k, каждый из 2^k потенциальных узлов на уровне k является внешним с вероятностью p_k. Математическое ожидание значения $\sum_k 2^k p_k$, таким образом, представляет собой математическое ожидание количества внешних узлов, а именно $n+1$ во всех трех случаях. (Этот результат можно проверить непосредственно, с учетом того, что $p_k = C_{(n-k)(n-1)}/C_n$ в случае (а), $p_k = \binom{n}{k}/n!$ в случае (б) и $p_k = 2^{-k+[k=n]}$ в случае (в).)

Примечания. Средняя длина пути в указанных трех случаях равна $\Theta(\sqrt{n})$, $\Theta(\log n)$ и $\Theta(n)$ соответственно; таким образом, этот путь оказывается более длинным при более коротком среднем пути к листу. Это поясняется тем, что вездесущие "дыры" вблизи корня приводят к удлинению других путей. Случай (а) имеет интересное обобщение для t-арных деревьев, для которых (с использованием обозначений из упр. 36) $p_k = C^{(t)}_{(n-k)((t-1)n-1)}/C^{(t)}_n$. Таким образом, среднее расстояние до листа равно $(t+1)n/((t-1)n+2)$, и очень поучительно доказать с использованием телескопических рядов, что $\sum_k t^k C^{(t)}_{(n-k)((t-1)n-1)} = \binom{tn}{n}$.

54. Дифференцируя по x, получаем

$$C'(x,z) = zC'(x,z)C(x,xz) + zC(x,z)(C'(x,xz) + zC_l(x,xz)),$$

где $C_l(x,z)$ обозначает производную $C(x,z)$ по z. Таким образом, $C'(1,z) = 2zC'(1,z)C(z) + z^2C(z)C'(z)$; а поскольку $C'(z) = C(z)^2 + 2zC(z)C'(z)$, можно найти решение для $C'(1,z)$, получив $z^2C(z)^3/(1-2zC(z))^2$. Следовательно, $\sum(c_1 + \cdots + c_n) = [z^n]\,C'(1,z) = 2^{2n-1} - \frac{1}{2}(3n+1)C_n$, что согласуется с упр. 2.3.4.5–5. Аналогично находим

$$\sum(c_1 + \cdots + c_n)^{\underline{2}} = [z^n]\,C''(1,z) = \left(\frac{5n^2+19n+6}{6}\right)\binom{2n}{n} - \left(1 + \frac{3n}{2}\right)4^n.$$

Таким образом, искомые математическое ожидание и дисперсия равны соответственно $\frac{1}{2}\sqrt{\pi}n^{3/2} + O(n)$ и $(\frac{5}{6} - \frac{\pi}{4})n^{3/2} + O(n)$.

55. Дифференцируя, как это делалось в ответе к упр. 54, и используя формулы из упр. 46, (г) и 5.2.1–14 вместе с $[z^n]\,C(z)^r/(1-4z) = 2^{2n+r} - \sum_{j=1}^{r} 2^{r-j}\binom{2n+j}{n}$, получаем

$$\begin{aligned} C'_{p(p+r)}(1) &= [z^p]\left((r+1)\frac{z^2C(z)^{r+3}}{1-4z} + \binom{r+1}{2}\frac{zC(z)^{r+2}}{\sqrt{1-4z}}\right) \\ &= [z^p]\left((r+1)\frac{C(z)^{r+1}-2C(z)^r+C(z)^{r-1}}{1-4z} + \binom{r+1}{2}\frac{C(z)^{r+1}-C(z)^r}{\sqrt{1-4z}}\right) \\ &= (r+1)\left(2^{2p+r-1} - \binom{2p+r+1}{p} - \sum_{j=1}^{r-1} 2^{r-1-j}\binom{2p+j}{p}\right) + \binom{r+1}{2}\binom{2p+r}{p-1}. \end{aligned}$$

56. Используйте упр. 1.2.6–53, (b). [См. *BIT* **30** (1990), 67–68.]

57. $2S_0(a,b) = \binom{2a}{a}\binom{2b}{b} + \binom{2a+2b}{a+b}$ согласно 1.2.6–(21). Упр. 1.2.6–53 гласит, что

$$\sum_{k=a-m}^{a}\binom{2a}{a-k}\binom{2b}{b-k}k = (m+1)(a+b-m)\binom{2a}{m+1}\binom{2b}{a+b-m};$$

следовательно, $2S_1(a,b) = \binom{2a}{a}\binom{2b}{b}\frac{ab}{a+b}$. А поскольку $b^2S_p(a,b) - S_{p+2}(a,b) = S_p(a,b-1)$, находим, что $2S_2(a,b) = \binom{2a+2b}{a+b}\frac{ab}{2a+2b-1}$; $2S_3(a,b) = \binom{2a}{a}\binom{2b}{b}a^2b^2/(a+b)^2$. Формула (30) получается путем подстановки $a=m$, $b=n-m$ и $C_{(x-k)(x+k)} = \binom{2x}{x-k} - \binom{2x}{x-k-1}$. Аналогично среднее значение w_{2m-1} равно $\sum_{k\ge 0}(2k-1)C_{(m-k)(m+k-1)}C_{(n-m-k+1)(n-m+k)}$, деленному на C_n, или

$$\frac{2S_3(m,n+1-m) - S_2(m,n+1-m)}{m(n+1-m)C_n} = \frac{m(n+1-m)}{n}\binom{2m}{m}\binom{2n+2-2m}{n+1-m}\Big/\binom{2n}{n} - 1.$$

[R. Kemp, *BIT* **20** (1980), 157–163; H. Prodinger, *Soochow J. Math.* **9** (1983), 193–196.]

58. Суммируя по всем случаям, когда левое поддерево содержит k внутренних узлов, получаем

$$t_{lmn} = [l=m=n=0] + \sum_{k=0}^{m-1} C_k t_{(l-1)(m-k-1)(n-k-1)} + \sum_{k=m}^{n-1} C_{n-1-k}t_{(l-1)mk}.$$

Таким образом, тройная производящая функция $t(v,w,z) = \sum_{l,m,n} t_{lmn}v^lw^mz^n$ удовлетворяет уравнению

$$t(v,w,z) = 1 + vwzC(wz)t(v,w,z) + vzC(z)t(v,w,z).$$

Аналогичное линейное соотношение получается и для $t(w,z) = \partial t(v,w,z)/\partial v\,|_{v=1}$, поскольку $t(1,w,z) = \sum_{n=0}^{\infty}\sum_{m=0}^{n} C_nw^mz^n = (C(z) - wC(wz))/(1-w)$ и $zC(z)^2 = C(z) - 1$. Алгебраические преобразования дают

$$t(w,z) = \frac{C(z) + wC(wz) - (1+w)}{(1-w)^2z} - \frac{2wC(z)C(wz)}{(1-w)^2} - \frac{C(z) - wC(wz)}{1-w},$$

и мы получаем формулу $t_{mn} = (m+1)C_{n+1} - 2\sum_{k=0}^{m}(m-k)C_kC_{n-k} - C_n$. Теперь доказать, что

$$\sum_{k=0}^{m-1}(k+1)C_kC_{n-1-k} = \frac{m}{2n}\binom{2m}{m}\binom{2n-2m}{n-m},$$

можно так же легко, как в упр. 56; отсюда следует, что

$$t_{mn} = 2\binom{2m}{m}\binom{2n-2m}{n-m}\frac{(2m+1)(2n-2m+1)}{(n+1)(n+2)} - C_n \qquad \text{для } 0 \le m \le n.$$

[P. Kirschenhofer, *J. Combinatorics, Information and System Sciences* **8** (1983), 44–60. Информация о более высоких моментах и обобщениях представлена в работах W. J. Gutjahr, *Random Structures & Algorithms* **3** (1992), 361–374; A. Panholzer and H. Prodinger, *J. Statistical Planning and Inference* **101** (2002), 267–279. Обратите внимание, что производящая функция $t(v,w,z)$ дает

$$t_{lmn} = \sum_k \binom{l}{k} C_{(m-k)(m-1)} C_{(n-m-l+k)(n-m-1)}.$$

Используя тот факт, что $\sum_k \binom{k}{r} C_{(n-k)(m-1)} = C_{(n-r)(m+r)}$ при $m \ge 1$, мы получаем формулу $t_{mn} + C_n = \sum_k (k+1) C_{(m-k)(m-1)} C_{(n-m)(n-m+k+1)}$, сумма в которой может быть записана в аналитическом виде.]

59. $$T(w,z) = \frac{w\bigl(C(z)-C(wz)\bigr)}{(1-w)} - wzC(z)C(wz) + zC(z)T(w,z) + wzC(wz)T(wz)$$
$$= \frac{w\bigl(\bigl(C(z)+C(wz)-2\bigr)/z - (1+w)C(z)C(wz) - (1-w)\bigl(C(z)-C(wz)\bigr)\bigr)}{(1-w)^2}.$$

Следовательно, $T_{mn} = t_{mn} - \sum_{k=m}^{n} C_k C_{n-k}$. [Является ли данное доказательство комбинаторным?] Итак,

$$T_{mn} = \binom{2m}{m}\binom{2n+2-2m}{n+1-m}\frac{4m(n+1-m)+n+1}{2(n+1)(n+2)} - \frac{1}{2}C_{n+1} - C_n \quad \text{при } 1 \le m \le n.$$

60. (а) Это количество правых скобок в соатомах. (Следовательно, это также количество k, для которых $w_{2k-1} < 0$ в соответствующем "обходе червяка".)

(б) Для удобства положим $d(\text{`('}) = +1$ и $d(\text{`)'}) = -1$.

A1. [Инициализация.] Установить $i \leftarrow j \leftarrow 1$ и $k \leftarrow 2n$.

A2. [Выполнено?] Завершить работу алгоритма, если $j > k$. В противном случае установить $a_j \leftarrow$ '(', $j \leftarrow j+1$.

A3. [Атом?] Если $b_i =$ ')', установить $s \leftarrow -1$, $i \leftarrow i+1$ и перейти к шагу A4. В противном случае установить $s \leftarrow 1$, $i \leftarrow i+1$, и, пока $s > 0$, устанавливать $a_j \leftarrow b_i$, $j \leftarrow j+1$, $s \leftarrow s + d(b_i)$, $i \leftarrow i+1$. Вернуться к шагу A2.

A4. [Соатом.] Установить $s \leftarrow s + d(b_i)$. Тогда, если $s < 0$, установить $a_k \leftarrow b_i$, $k \leftarrow k-1$, $i \leftarrow i+1$ и повторить шаг A4. В противном случае установить $a_k \leftarrow$ ')', $k \leftarrow k-1$, $i \leftarrow i+1$ и вернуться к шагу A2. ▮

(в) Строка с дефектом 11, отображающаяся на строку (1), выглядит следующим образом: `(()))(((()))))))(()((())(()))))(((`. В общем случае мы находим такую строку, определяя индекс m непосредственно перед l-й с конца правой скобкой, и индексы (u_0, v_0), $\ldots$, (u_{s-1}, v_{s-1}) соответствующих одна другой скобок, таких, что $u_j \le m < v_j$.

I1. [Инициализация.] Установить $c \leftarrow j \leftarrow s \leftarrow 0$, $k \leftarrow m \leftarrow 2n$ и $u_0 \leftarrow 2n+1$.

I2. [Сканирование справа налево.] Если $k = 0$, перейти к шагу I5; если $a_k =$ ')', перейти к шагу I3; если $a_k =$ '(', перейти к шагу I4.

I3. [Обработка ')'.] Установить $r_j \leftarrow k$, $j \leftarrow j+1$, $c \leftarrow c+1$. Если $c = l$, установить $m \leftarrow k-1$, $s \leftarrow j$ и $u_s \leftarrow k$. Затем уменьшить k на 1 и вернуться к шагу I2.

I4. [Обработка '('.] (В этот момент левая скобка a_k соответствует правой скобке $a_{r_{j-1}}$.) Установить $j \leftarrow j-1$. Если $r_j > m$, установить $u_j \leftarrow k$ и $v_j \leftarrow r_j$. Затем уменьшить k на 1 и вернуться к шагу I2.

I5. [Подготовка к перестановке.] Установить $i \leftarrow j \leftarrow 1$, $k \leftarrow 2n$ и $c \leftarrow 0$.

I6. [Перестановка.] Пока $j \neq u_c$, устанавливать $b_i \leftarrow a_j$, $i \leftarrow i+1$, $j \leftarrow j+1$. Затем завершить работу алгоритма, если $c = s$; в противном случае установить $b_i \leftarrow$ ‘)’, $i \leftarrow i+1$, $j \leftarrow j+1$. Пока $k \neq v_c$, устанавливать $b_i \leftarrow a_k$, $i \leftarrow i+1$, $k \leftarrow k-1$. Затем установить $b_i \leftarrow$ ‘(’, $i \leftarrow i+1$, $k \leftarrow k-1$, $c \leftarrow c+1$ и повторить шаг I6. ▌

Примечания. Тот факт, что дефект l ($0 \le l \le n$) имеют ровно C_n сбалансированных строк длиной $2n$, был открыт П. А. Мак-Мэганом (P. A. MacMahon) [*Philosophical Transactions* **209** (1909), 153–175, §20], а затем повторно открыт К. Л. Чангом (K. L. Chung) и В. Феллером (W. Feller) [*Proc. Nat. Acad. Sci.* **35** (1949), 605–608] с применением производящих функций. Простое комбинаторное пояснение было найдено позже Д. Л. Ходжесом-мл. (J. L. Hodges, Jr.) [*Biometrika* **42** (1955), 261–262], который заметил, что если $\beta_1 \ldots \beta_r$ имеет дефект $l > 0$ и если $\beta_k = \alpha_k^R$ — ее крайний справа соатом, то сбалансированная строка $\beta_1 \ldots \beta_{k-1}(\beta_{k+1} \ldots \beta_r)\alpha_k'^R$ имеет дефект $l-1$ (и это преобразование обратимо). Эффективное отображение в данном упражнении похоже на конструкцию М. Д. Аткинсона (M. D. Atkinson) и Й.-Р. Сака (J.-R. Sack) [*Information Processing Letters* **41** (1992), 21–23].

61. (а) Пусть $c_j = 1 - b_j$; тогда $c_j \le 1$, $c_1 + \cdots + c_N = f$, и мы должны доказать, что утверждение

$$c_1 + c_2 + \cdots + c_k \;<\; f \qquad \text{тогда и только тогда, когда} \qquad k < N$$

выполняется ровно для f циклических сдвигов. Можно определить c_j для всех целых чисел j, полагая $c_{j\pm N} = c_j$. Определим также Σ_j для всех j, полагая $\Sigma_0 = 0$ и $\Sigma_j = \Sigma_{j-1} + c_j$; тогда $\Sigma_{j+Nt} = \Sigma_j + ft$ и $\Sigma_{j+1} \le \Sigma_j + 1$. Отсюда следует, что для каждого целого x существует наименьшее целое $j = j(x)$, такое, что $\Sigma_j = x$. Кроме того, $j(x) < j(x+1)$ и $j(x+f) = j(x) + N$. Таким образом, интересующее нас условие выполняется тогда и только тогда, когда мы выполняем сдвиг на $j(x) \bmod N$ для $x = 1, 2, \ldots$ или f. (История этой важной леммы рассматривается в ответе к упр. 2.3.4.4–32.)

(б) Начнем с $l \leftarrow m \leftarrow s \leftarrow 0$. Затем для $k = 1, 2, \ldots, N$ (в указанном порядке) выполним следующие действия: установим $s \leftarrow s + 1 - b_k$ и, если $s > m$, установим $m \leftarrow s$, $j_l \leftarrow k$ и $l \leftarrow (l+1) \bmod f$. В соответствии с доказательством из п. (а) ответами будут $j_0, \ldots, j_{f-1}$.

(в) Начнем с произвольной строки $b_1 b_2 \ldots b_N$, содержащей n_j значений j, $0 \le j \le t$. Применим к этой строке случайную перестановку, а затем — алгоритм из п. (б). Затем случайным образом выберем значение из $(j_0, \ldots, j_{f-1})$ и используем результат циклического сдвига в качестве определяющей лес последовательности в прямом порядке обхода.

[См. L. Alonso, J. L. Rémy, and R. Schott, *Algorithmica* **17** (1997), 162–182, где приведен еще более общий алгоритм.]

62. Битовые строки $(l_1 \ldots l_n, r_1 \ldots r_n)$ корректны тогда и только тогда, когда строка $b_1 \ldots b_n$, где $b_j = l_j + r_j$, корректна в упр. 20. Следовательно, можно воспользоваться упр. 61. [См. J. F. Korsh, *Information Processing Letters* **45** (1993), 291–294.]

63.

64. $X = 2k + b$, где $(k, b) = (0,1), (2,1), (0,0), (5,1), (6,0), (1,1)$; в конце $L_0 L_1 \ldots L_{12} = 5$ 11 3 4 0 7 9 8 1 6 10 12 2.

65. См. A. Panholzer and H. Prodinger, *Discrete Mathematics* **250** (2002), 181–195; M. Luczak and P. Winkler, *Random Structures & Algorithms* **24** (2004), 420–443.

66. (а) “Сожмем” белые ребра, объединив соединяемые ими узлы. Например, одиннадцати изображенным деревьям Шрёдера для $n = 3$ будут соответствовать обычные деревья

При таком соответствии левая связь означает "это дочерний узел", белая правая связь означает "здесь есть и другие дочерние узлы", а правая черная связь означает "это последний дочерний узел".

(б) Имитируем алгоритм L, но между поворотами используем обычный код Грея для прохода по всем цветовым шаблонам имеющихся правых связей (в этом упражнении проиллюстрирована рассматриваемая последовательность для случая $n = 3$).

Заметим, что деревья Шрёдера также соответствуют последовательно-параллельным графам, как в дереве из (53). Однако при этом налагается определенный порядок на ребра и/или сверхребра, соединенные параллельно; так что более точно говорить о соответствии последовательно-параллельным графам, *уложенным на плоскости* (с непомеченными ребрами и вершинами, за исключением s и t).

67. $S(z) = 1 + zS(z)\bigl(1 + 2(S(z) - 1)\bigr)$, поскольку $1 + 2(S(z) - 1)$ подсчитывает правые поддеревья; следовательно, искомая функция $S(z) = (1 + z - \sqrt{1 - 6z + z^2})/(4z)$.

Примечания. Мы уже встречались с числами Шрёдера в упр. 2.3.4.4–31, где $G(z) = zS(z)$; а также в упр. 2.2.1–11, где $b_n = 2S_{n-1}$ при $n \ge 2$ и где было найдено рекуррентное соотношение $(n-1)S_n = (6n-3)S_{n-1} - (n-2)S_{n-2}$. Асимптотический рост чисел Шрёдера исследован в упр. 2.2.1–12. Треугольный массив чисел S_{pq}, аналогичный (22), можно использовать для генерации случайных деревьев Шрёдера. Эти числа удовлетворяют соотношению

$$\begin{aligned} S_{pq} &= S_{p(q-1)} + S_{(p-1)q} + S_{(p-2)q} + \cdots + S_{0q} = S_{p(q-1)} + 2S_{(p-1)q} - S_{(p-1)(q-1)} \\ &= \frac{q-p+1}{q+1}\sum_{k=0}^{p}\binom{q+1}{p-k}\binom{p-1}{k}2^k = \sum_{k=0}^{p}\biggl(\binom{q}{p-k}\binom{p-1}{k} - \binom{q}{p-k-1}\binom{p-1}{k-1}\biggr)2^k \\ &= [w^p z^q]\, S(wz)/(1 - zS(wz)); \end{aligned}$$

двойная производящая функция в последней строке открыта Эмериком Дойчем (Emeric Deutsch). Многие другие свойства деревьев Шрёдера рассматриваются в книге Ричарда Стенли (Richard Stanley) *Enumerative Combinatorics* **2** (1999), упр. 6.39.

68. Единственная строка, содержащая пустую битовую строку ϵ. (Общее правило (36) для перехода от порядка $n-1$ к порядку n превращает эту строку в '0 1', шаблон порядка 1.)

69. Первые $\binom{6}{3} = 20$ строк представляют собой шаблон рождественской елки порядка 6, если игнорировать '10' в начале каждой строки. Увидеть шаблон порядка 7 немного труднее; но имеется $\binom{7}{3} = 35$ строк, в которых крайняя слева запись начинается с 0. Игнорируйте во всех таких строках крайнюю справа битовую строку, а также 0 в начале каждой остающейся битовой строки. (Возможны и другие ответы на данный вопрос.)

70. Если σ находится в столбце k шаблона рождественской елки, то σ' представляет собой строку, находящуюся в столбце $n - k$ той же строки. (Если вместо битов рассматривать скобки, то это правило согласно (39) дает зеркальное отображение свободных скобок в смысле ответа к упр. 11.)

71. M_{tn} представляет собой сумму t наибольших биномиальных коэффициентов $\binom{n}{k}$, поскольку в каждой строке шаблона рождественской елки может содержаться не более t элементов множества S и поскольку мы получаем такое множество S путем выбора всех строк σ, у которых $(n - t)/2 \le \nu(\sigma) \le (n + t - 1)/2$. (Формула

$$M_{tn} = \sum_{n-t\le 2k\le n+t-1}\binom{n}{k}$$

проста, насколько это возможно; однако для малых t имеются более простые формулы, наподобие $M_{(2)n} = M_{n+1}$, а кроме того, $M_{tn} = 2^n$ для $t > n$.)

72. Мы получим M_{sn}, то же число, что и в предыдущем упражнении. В действительности по индукции можно доказать, что существует ровно $\binom{n}{n-k} - \binom{n}{k-s}$ строк длиной $s + n - 2k \geq 0$.

73. 011001001000000000100101001100, 111001011011111111101101011100; см. (38).

74. В соответствии с лексикографическим порядком нужно подсчитать количество строк, крайние справа элементы которых имеют вид $0{*}^{29}$, $10{*}^{28}$, $110{*}^{27}$, $111000{*}^{24}$, $11100100{*}^{22}$, $111001010{*}^{21}$, $11100101100{*}^{19}$, $111001011010{*}^{18}$, $1110010110110{*}^{17}$, ..., а именно все 30-битовые строки, предшествующие строке $\tau = 111001011011111111101101011100$.

Если θ имеет на p больше единиц, чем нулей, то количество строк шаблона рождественской елки, заканчивающихся битовой строкой $\theta{*}^n$, равно количеству строк, завершающихся $1^p{*}^n$; и в соответствии с упр. 71 равно $M_{(p+1)n}$, так как все такие строки являются потомками начальной строки '$0^p\ 0^{p-1}1\ \ldots\ 1^p$' на n-м шаге.

Следовательно, ответ на поставленный вопрос $M_{0(29)} + M_{1(28)} + M_{2(27)} + M_{1(24)} + \cdots + M_{(12)3} + M_{(13)2} = \sum_{k=1}^{21} M_{(2k-1-z_k)(n-z_k)} = 0 + \binom{28}{14} + \binom{27}{14} + \binom{27}{13} + \binom{24}{12} + \cdots + 8 + 4 = 84867708$, где $(z_1, \ldots, z_{21}) = (1, 2, 3, 6, \ldots, 27, 28)$ представляет собой последовательность позиций, в которых в τ встречаются единицы.

75. Имеем $r_1^{(n)} = M_{n-2}$, поскольку строка $r_1^{(n)}$ представляет собой нижний потомок первой строки в (33). Кроме того, $r_{j+1}^{(n)} - r_j^{(n)} = M_{j(n-1-j)} - M_{(j-1)(n-2-j)} = M_{(j+1)(n-2-j)}$ в соответствии с формулой из ответа к упр. 74, поскольку соответствующая последовательность $z_1 \ldots z_{n-1}$ для строки $r_j^{(n)}$ представляет собой $1^j 0 1^{n-1-j}$. Следовательно, поскольку $M_{jn}/M_n \to j$ для фиксированного значения j при $n \to \infty$, получим

$$\lim_{n\to\infty} \frac{r_j^{(n)}}{M_n} = \sum_{k=1}^{j} \frac{k}{2^{k+1}} = 1 - \frac{j+2}{2^{j+1}}.$$

Мы также неявно доказали, что $\sum_{k=0}^{n} M_{k(n-k)} = M_{n+1} - 1$.

76. Первые $\binom{2n}{n}$ элементов бесконечной последовательности

$$Q = 1313351313351335355713133513133513353557131335131335355713353557357357 75 \ldots$$

представляют собой размеры строк в шаблоне порядка $2n$; эта последовательность $Q = q_1 q_2 q_3 \ldots$ является единственной фиксированной точкой преобразования, отображающего $1 \mapsto 13$ и $n \mapsto (n-2)\,n\,n(n+2)$ для нечетных $n > 1$ и представляющего два шага из (36).

Пусть $f(x) = \limsup_{n\to\infty} s(\lceil xM_n \rceil)/n$ для $0 < x \leq 1$. Эта функция, по-видимому, почти везде стремится к нулю; однако в соответствии с упр. 72 она равна 1, когда x имеет вид $(q_1 + \cdots + q_j)/2^n$. С другой стороны, если определить $g(x) = \lim_{n\to\infty} s(\lceil xM_n \rceil)/\sqrt{n}$, то функция $g(x)$ является измеримой, с $\int_0^1 g(x)\,dx = \sqrt{\pi}$, хотя $g(x)$ бесконечна, когда $f(x) > 0$. (Строгое доказательство или опровержение этих гипотез пока отсутствует.)

77. Указание следует из (39) при рассмотрении обхода червяка; так что можно действовать следующим образом.

X1. [Инициализация.] Установить $a_j \leftarrow 0$ для $0 \leq j \leq n$; установить также $x \leftarrow 1$. (На последующих шагах $x = 1 + 2(a_1 + \cdots + a_n)$.)

X2. [Коррекция хвоста.] Пока $x \leq n$, устанавливать $a_x \leftarrow 1$ и $x \leftarrow x + 2$.

X3. [Посещение.] Посетить битовую строку $a_1 \ldots a_n$.

X4. [Простой случай?] Если $a_n = 0$, установить $a_n \leftarrow 1$, $x \leftarrow x + 2$ и вернуться к шагу X3.

X5. [Поиск и продвижение a_j.] Установить $a_n \leftarrow 0$ и $j \leftarrow n - 1$. Затем, пока $a_j = 1$, устанавливать $a_j \leftarrow 0$, $x \leftarrow x-2$ и $j \leftarrow j-1$. Завершить работу, если $j = 0$; в противном случае установить $a_j \leftarrow 1$ и вернуться к шагу X2. ▮

78. Истинно согласно (39) и упр. 11.

79. (а) Перечислите индексы нулей, затем индексы единиц; например, битовая строка в упр. 73 соответствует перестановке 1 4 5 7 8 10 11 12 13 20 23 25 29 30 2 3 6 9 14 15 16 17 18 19 21 22 24 26 27 28.

(б) В верхней строке диаграммы P содержатся индексы левых скобок и свободных скобок (пользуясь соглашениями из (39)); остальные индексы находятся во второй строке. Таким образом, из (38) получаем

$$P = \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 1 & 2 & 3 & 6 & 8 & 9 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 21 & 22 & 24 & 26 & 27 & 28 \\ \hline 4 & 5 & 7 & 10 & 20 & 23 & 25 & 29 & 30 & & & & & & & & & & & & \\ \cline{1-9}\end{array}.$$

[См. обобщение для цепочек подмультимножеств в K.-P. Vo, *SIAM J. Algebraic and Discrete Methods* **2** (1981), 324–332.]

80. Этот любопытный факт является следствием упр. 79 совместно с теоремой 6 из статьи автора о диаграммах; см. *Pacific J. Math.* **34** (1970), 709–727.

81. Предположим, что σ и σ' принадлежат соответственно цепочкам длиной s и s' в шаблонах рождественской елки порядка n и n'. В биклаттере могут находиться не более $\min(s, s')$ из ss' возможных пар строк из этих цепочек. Кроме того, с учетом (39) эти ss' пар строк в действительности при конкатенации образуют ровно $\min(s, s')$ цепочек в шаблоне рождественской елки порядка $n + n'$. Следовательно, сумма $\min(s, s')$ по всем парам цепочек равна $M_{n+n'}$, что и дает искомый результат. Кстати, при этом мы доказали неочевидное тождество

$$\sum_{j,\,k} \min(m + 1 - 2j, n + 1 - 2k)\, C_{j(m-j)}\, C_{k(n-k)} \;=\; M_{m+n}.$$

Примечания. Это расширение теоремы Спернера было доказано независимо Г. Катона (G. Katona) [*Studia Sci. Math. Hungar.* **1** (1966), 59–63] и Д. Кляйтманом (D. Kleitman) [*Math. Zeitschrift* **90** (1965), 251–259]. Доказательство из этого упражнения и некоторые дополнительные результаты приведены в Greene and Kleitman, *J. Combinatorial Theory* **A20** (1976), 80–88.

82. (а) Имеется как минимум одно вычисление в каждой строке m; два их тогда и только тогда, когда $s(m) > 1$ и первое вычисление дает 0. Таким образом, если f тождественно 1, мы получаем минимум, M_n; если f тождественно 0, мы получаем максимум, $M_n + \sum_m [s(m) > 1] = M_{n+1}$.

(б) Пусть $f(\chi(m, n/2)) = 0$ в $C_{n/2}$ случаях, когда $s(m) = 1$; в противном случае пусть $f(\chi(m, a)) = 1$, где a определяется алгоритмом. Когда n нечетно, из этого правила вытекает, что $f(\sigma)$ всегда равно 1; но когда n четно, $f(\sigma) = 0$ тогда и только тогда, когда σ — первая битовая строка в своей строке. (Чтобы понять, почему это так, воспользуйтесь тем фактом, что строка, содержащая σ'_j в (41), всегда имеет размер $s - 2$.) Эта функция f в действительности монотонна; если $\sigma \le \tau$ и если σ имеет свободные левые скобки, то то же справедливо и для τ. Например, в случае $n = 8$ имеем

$$f(x_1, \ldots, x_8) = x_8 \vee x_6x_7 \vee x_4x_5(x_6 \vee x_7) \vee x_2x_3(x_4(x_5 \vee x_6 \vee x_7) \vee x_5(x_6 \vee x_7)).$$

(в) В этих обстоятельствах (45) является решением для всех n.

83. На шаге H4 возможно не более трех исходов; на самом деле при $s(m) = 1$ — не более двух. [См. более строгие границы в упр. 5.3.4–31; в обозначениях из этого упражнения имеется ровно $\delta_n + 2$ монотонных булевых функций от n булевых переменных.]

84. Для решения задачи разобъем 2^n битовых строк вместо цепочек на M_n блоков, где строки $\{\sigma_1, \ldots, \sigma_s\}$ каждого блока удовлетворяют условию $\|A\sigma_i^T - A\sigma_j^T\| \ge 1$ при $i \ne j$;

тогда условию $\|A\sigma^T - b\| < \frac{1}{2}$ могут удовлетворять не более одной битовой строки в каждом блоке.

Пусть A' обозначает первые $n-1$ столбцов A, а v — n-й столбец. Предположим, что $\{\sigma_1, \ldots, \sigma_s\}$ — блок A', при этом количество индексов таково, что $v^T A' \sigma_1^T$ минимально среди всех $v^T A' \sigma_j^T$. Тогда правило (36) определяет соответствующие блоки для A, поскольку $\|A(\sigma_i 0)^T - A(\sigma_j 0)^T\| = \|A(\sigma_i 1)^T - A(\sigma_j 1)^T\| = \|A'\sigma_i^T - A'\sigma_j^T\|$ и

$$\begin{aligned}\|A(\sigma_j 1)^T - A(\sigma_1 0)^T\|^2 &= \|A'\sigma_j^T + v - A'\sigma_1^T\|^2 \\ &= \|A'(\sigma_j - \sigma_1)^T\|^2 + \|v\|^2 + 2v^T A'(\sigma_j - \sigma_1)^T \ge \|v\|^2 \ge 1.\end{aligned}$$

[Справедливо и многое другое; см. *Advances in Math.* **5** (1970), 155–157. Этот результат является расширением теоремы Д. Э. Литтлвуда (J. E. Littlewood) и А. С. Оффорда (A. C. Offord), *Mat. Sbornik* **54** (1943), 277–285, которые рассмотрели случай $m = 2$.]

85. Если V имеет размерность $n-m$, можно перенумеровать координаты так, что

$$\begin{array}{cccccc}(1, & 0, & \ldots, & 0, & x_{11}, & \ldots, & x_{1m}) \\ (0, & 1, & \ldots, & 0, & x_{21}, & \ldots, & x_{2m}) \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ (0, & 0, & \ldots, & 1, & x_{(n-m)1}, & \ldots, & x_{(n-m)m})\end{array}$$

представляет собой базис, в котором ни один вектор-строка $v_j = (x_{j1}, \ldots, x_{jm})$ не состоит из нулей. Пусть $v_{n-m+1} = (-1, 0, \ldots, 0)$, $\ldots$, $v_n = (0, 0, \ldots, -1)$. Тогда количество 0–1 векторов в V равно количеству 0–1 решений $Ax = 0$, где A представляет собой матрицу размером $m \times n$ со столбцами v_1, $\ldots$, v_n. Но эта величина не превосходит количество решений $\|Ax\| < \frac{1}{2}\min(\|v_1\|, \ldots, \|v_n\|)$, которое согласно упр. 84 не превосходит M_n.

И обратно: базис с $m = 1$ и $x_{j1} = (-1)^{j-1}$ дает M_n решений. [Этот результат применяется в электронном голосовании; см. диссертацию Голля (Golle) (Stanford, 2004).]

86. Сначала переупорядочим 4-узловые поддеревья так, чтобы их коды уровней представляли собой 0121 (плюс константа); затем будем сортировать все бóльшие и бóльшие поддеревья, пока все они не станут каноническими. В результате мы получим коды уровней 0 1 2 3 4 3 2 1 2 3 2 1 2 0 1, а соответствующими указателями на родительские узлы будут 0 1 2 3 4 3 2 1 8 9 8 1 12 0 14.

87. (а) Условие выполняется тогда и только тогда, когда $c_1 < \cdots < c_k \ge c_{k+1} \ge \cdots \ge c_n$ для некоторого k, так что общее количество случаев равно $\sum_k \binom{n-1}{n-k} = 2^{n-1}$.

(б) Заметим, что $c_1 \ldots c_k = c'_1 \ldots c'_k$ тогда и только тогда, когда $p_1 \ldots p_k = p'_1 \ldots p'_k$; в таких случаях $c_{k+1} < c'_{k+1}$ тогда и только тогда, когда $p_{k+1} < p'_{k+1}$.

88. Посещается ровно A_{n+1} лесов, и у A_k из них $p_k = \cdots = p_n = 0$. Следовательно, шаг O4 выполняется A_n раз; p_k на шаге O5 изменяется $A_{k+1} - 1$ раз при $1 \le k < n$. Шаг O5 также изменяет p_n всего $A_n - 1$ раз. Среднее количество обращений к памяти на одно посещение, таким образом, составляет $2 + 3/(\alpha - 1) + O(1/n) \approx 3.534$, если хранить p_n в регистре. [См. E. Kubicka, *Combinatorics, Probability and Computing* **5** (1996), 403–417.]

89. Если на шаге O5 присваивание $p_n \leftarrow p_j$ выполняется ровно Q_n раз, то присваивание $p_k \leftarrow p_j$ выполняется ровно $Q_k + A_{k+1} - A_k$ раз для $1 < k < n$, поскольку каждый префикс канонического $p_1 \ldots p_n$ является каноническим. Итак, получаем $(Q_1, Q_2, \ldots) = (0, 0, 1, 2, 5, 9, 22, 48, 118, 288, \ldots)$; и можно показать, что элементы этой последовательности имеют вид $Q_n = \sum_{d \ge 1} \sum_{1 \le c < n/d-1} a_{(n-cd)(n-cd-d)}$, где a_{nk} — количество канонических родительских последовательностей $p_1 \ldots p_n$ с $p_n = k$. Однако сами значения a_{nk} пока что остаются загадкой.

90. (а) Это свойство эквивалентно 2.3.4.4–(7); вершина 0 является центроидом.

(б) Пусть $m = \lfloor n/2 \rfloor$. Требуется внести следующие изменения. В конце шага O1 установить $p_{m+1} \leftarrow 0$, а также $p_{2m+1} \leftarrow 0$, если n нечетно. В конце шага O4 установить $i \leftarrow j$ и, пока $p_i \neq 0$, устанавливать $i \leftarrow p_i$. (Тогда i является корнем дерева, содержащего j и k.) В начале шага O5, если $k = i + m$ и $i < j$, установить $j \leftarrow i$ и $d \leftarrow m$.

(в) Если n четно, бицентроидных деревьев с $n+1$ вершинами не существует. В противном случае найдем все пары $(p'_1 \ldots p'_m, p''_1 \ldots p''_m)$ канонических лесов из $m = \lfloor n/2 \rfloor$ узлов, где $p'_1 \ldots p'_m \geq p''_1 \ldots p''_m$; пусть $p_1 = 0$, $p_{j+1} = p'_j + 1$ и $p_{m+j+1} = (p''_j + m + 1)[p''_j > 0]$ для $1 \leq j \leq m$. (Две инкарнации алгоритма O будут генерировать все такие последовательности. Данный алгоритм для свободных деревьев разработан Ф. Раски (F. Ruskey) и Г. Ли (G. Li); см. *SODA* **10** (1999), S939–S940.)

91. Воспользуйтесь следующей рекурсивной процедурой $W(n)$. Если $n \leq 2$, процедура возвращает единственное n-узловое ориентированное дерево. В противном случае выбираются положительные целые числа j и d так, что данная пара (j, d) получается с вероятностью $dA_dA_{n-jd}/((n-1)A_n)$. Вычисляются случайные ориентированные деревья $T' \leftarrow W(n - jd)$ и $T'' \leftarrow W(d)$. Возвращается дерево T, полученное путем связи j клонов T'' с корнем дерева T'. [*Combinatorial Algorithms* (Academic Press, 1975), глава 25.]

92. Не всегда. [Р. Л. Камминс (R. L. Cummins) в *IEEE Trans.* **CT-13** (1966), 82–90, доказал, что граф $S(G)$ всегда содержит цикл; см. также C. A. Holzmann and F. Harary, *SIAM J. Applied Math.* **22** (1972), 187–193. Однако их построение непригодно для эффективного вычисления, поскольку требует предварительной информации о четности размеров промежуточных результатов.]

93. Да. Шаг S7 аннулирует шаг S3; шаг S9 отменяет удаление на шаге S8.

94. Например, можно использовать поиск вглубь с использованием вспомогательной таблицы $b_1 \ldots b_n$.

i) Установить $b_1 \ldots b_n \leftarrow 0 \ldots 0$, затем установить $v \leftarrow 1$, $w \leftarrow 1$, $b_1 \leftarrow 1$ и $k \leftarrow n - 1$.

ii) Установить $e \leftarrow n_{v-1}$. Пока $t_e \neq 0$, выполнять следующие подшаги.

  а) Установить $u \leftarrow t_e$. Если $b_u \neq 0$, перейти к подшагу (в).

  б) Установить $b_u \leftarrow w$, $w \leftarrow u$, $a_k \leftarrow e$, $k \leftarrow k - 1$. Завершить работу алгоритма, если $k = 0$.

  в) Установить $e \leftarrow n_e$.

iii) Если $w \neq 1$, установить $v \leftarrow w$, $w \leftarrow b_w$ и вернуться к шагу (ii). В противном случае сообщить об ошибке — данный граф не является связным.

В действительности работу алгоритма можно завершать, как только подшаг (б) сделает k равным 1, поскольку алгоритм S никогда не смотрит на начальное значение a_1. Однако мы можем продолжить работу и выполнить проверку связности графа.

95. Описанные далее шаги выполняют поиск в ширину от u, выясняя, достижима ли вершина v без использования ребра e. Используется вспомогательный массив $b_1 \ldots b_n$ указателей на дуги, который должен быть инициализирован значениями $0 \ldots 0$ в конце шага S1; мы снова сбросим его значения в $0 \ldots 0$.

i) Установить $w \leftarrow u$ и $b_w \leftarrow v$.

ii) Установить $f \leftarrow n_{u-1}$. Пока $t_f \neq 0$, выполнять следующие подшаги.

  а) Установить $v' \leftarrow t_f$. Если $b_{v'} \neq 0$, перейти к подшагу (г).

  б) Если $v' \neq v$, установить $b_{v'} \leftarrow v$, $b_w \leftarrow v'$, $w \leftarrow v'$ и перейти к подшагу (г).

  в) Если $f \neq e \oplus 1$, перейти к шагу (v).

  г) Установить $f \leftarrow n_f$.

iii) Установить $u \leftarrow b_u$. Если $u \neq v$, вернуться к шагу (ii).

iv) Установить $u \leftarrow t_e$. Пока $u \neq v$, устанавливать $w \leftarrow b_u$, $b_u \leftarrow 0$, $u \leftarrow w$. Перейти к шагу S9 (e является мостом).

v) Установить $u \leftarrow t_e$. Пока $u \neq v$, устанавливать $w \leftarrow b_u$, $b_u \leftarrow 0$, $u \leftarrow w$. Затем вновь установить $u \leftarrow t_e$ и продолжить выполнение шага S8 (e — не мост).

Перед началом описанных вычислений можно воспользоваться двумя быстрыми эвристиками. Если $d_u = 1$, то очевидно, что e является мостом, а если $l_{l_e} \neq 0$, то очевидно, что e мостом не является (поскольку существует другое ребро между u и v). Эти частные случаи быстро обнаруживаются при поиске в ширину. Эксперименты автора показывают, что их использование, определенно, сто́ит затрачиваемых усилий. Например, проверка l_{l_e} обычно позволяет сэкономить 3% общего времени работы алгоритма.

96. (а) Пусть e_k — дуга $k-1 \to k$. Шаги, описанные в упр. 94, устанавливают $a_k \leftarrow e_{n+1-k}$ для $n > k \geq 1$. Тогда на уровне k мы выполняем сжатие e_{n-k}, $1 \leq k < n-1$. После посещения (единственного) остовного дерева $e_{n-1} \ldots e_2 e_n$ мы восстанавливаем e_{n-k} и быстро обнаруживаем, что для $n-1 > k \geq 1$ это мост. Таким образом, время работы линейно зависит от n; в авторской реализации алгоритма выполняется ровно $102n - 226$ обращений к памяти при $n \geq 3$.

Однако этот результат весьма существенно зависит от порядка ребер в начальном остовном дереве. Если на шаге S1 получается "порядок органных труб" наподобие

$$e_{n/2+1}\, e_{n/2}\, e_{n/2+2}\, e_{n/2-1} \ldots e_{n-1}\, e_2$$

в позициях $a_2 \ldots a_{n-1}$ при четном n, то время работы должно составлять $\Omega(n^2)$, поскольку $\Omega(n)$ проверок мостов требуют по $\Omega(n)$ шагов.

(б) Здесь a_k изначально представляет собой e_{n-k} для $n > k \geq 1$, где e_1 — дуга $n \to 1$. Посещенные остовные деревья при $n \geq 4$ представляют собой соответственно $e_{n-2} \ldots e_1 e_n$, $e_{n-2} \ldots e_1 e_{n-1}$, $e_{n-2} \ldots e_2 e_{n-1} e_n$, $e_{n-2} \ldots e_3 e_{n-1} e_n e_1$, $\ldots$, $e_{n-1} e_n e_1 \ldots e_{n-3}$. Вслед за деревом $e_{n-2} \ldots e_{k+2} e_{n-1} e_n e_1 \ldots e_k$ вычисления опускаются на уровень $n-k-3$ и вновь поднимаются для $0 \leq k \leq n-4$; все проверки мостов эффективны. Таким образом общее время работы квадратично (в авторской версии — $35.5n^2 + 7.5n - 145$ обращений к памяти при $n \geq 5$).

Кстати, P_n в обозначениях Stanford GraphBase представляет собой $board(n, 0, 0, 0, 1, 0, 0)$, а C_n — $board(n, 0, 0, 0, 1, 1, 0)$; вершины в Stanford GraphBase нумеруются от 0 до $n-1$.

97. Да, когда $\{s, t\}$ равно $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, $\{2, 4\}$ или $\{3, 4\}$, но не $\{1, 4\}$.

98. $A' =$

; это так называемый "дуальный планарный граф" к планарному графу A.
(Близкие деревья A' комплементарны остовным деревьям A, и наоборот.)

99. Указанный метод работает (в чем можно убедиться по индукции по размеру дерева), по сути, по тем же причинам, по которым он работает для n-кортежей в разделе 7.2.1.1, но с дополнительным условием: мы должны последовательно обозначать каждый дочерний узел непростого узла.

Листья всегда пассивны, и они не являются ни простыми, ни непростыми; поэтому мы считаем, что внутренние узлы пронумерованы от 1 до m в прямом порядке обхода. Пусть $f_p = p$ для всех внутренних узлов, за исключением тех случаев, когда p — пассивный непростой узел, ближайший к которому справа непростой узел является активным; в этом случае f_p должен указывать на ближайший активный непростой узел слева. (Для такого определения мы представляем наличие искусственных узлов 0 и $m+1$ слева и справа; оба эти узла непростые и активные.)

F1. [Инициализация.] Установить $f_p \leftarrow p$ для $0 \leq p \leq m$; установить также $t_0 \leftarrow 1$, $v_0 \leftarrow 0$ и установить каждое z_p так, чтобы $r_{z_p} = d_p$.

F2. [Выбор узла p.] Установить $q \leftarrow m$; затем, пока $t_q = v_q$, устанавливать $q \leftarrow q - 1$. Установить $p \leftarrow f_q$ и $f_q \leftarrow q$; завершить работу алгоритма, если $p = 0$.

F3. [Изменение d_p.] Установить $s \leftarrow d_p$, $s' \leftarrow r_s$, $k \leftarrow v_p$ и $d_p \leftarrow s'$. (Сейчас $k = v_s \neq v_{s'}$.)

F4. [Обновление значений.] Установить $q \leftarrow s$ и $v_q \leftarrow k \oplus 1$. Пока $d_q \neq 0$, устанавливать $q \leftarrow d_q$ и $v_q \leftarrow k \oplus 1$. (Сейчас q является листом, который входит в конфигурацию, если $k = 0$, и покидает ее, если $k = 1$.) Аналогично установить $q \leftarrow s'$ и $v_q \leftarrow k$. Пока $d_q \neq 0$, устанавливать $q \leftarrow d_q$ и $v_q \leftarrow k$. (Сейчас q является листом, который покидает конфигурацию, если $k = 0$, и входит в нее, если $k = 1$.)

F5. [Посещение.] Посещение текущей конфигурации, представленной всеми значениями листьев.

F6. [Пассивизация p?] (Все непростые узлы справа от p в настоящий момент активны.) Если $d_p \neq z_p$, вернуться к шагу F2. В противном случае установить $z_p \leftarrow s$, $q \leftarrow p - 1$; пока $t_q = v_q$, устанавливать $q \leftarrow q - 1$. (Сейчас q является первым непростым узлом слева от p; мы сделаем p пассивным.) Установить $f_p \leftarrow f_q$, $f_q \leftarrow q$ и вернуться к шагу F2. ▌

Хотя шаг F4 может преобразовывать непростые узлы в простые и обратно, обновлять фокусные указатели не требуется, поскольку они остаются корректно установленными.

100. Завершенную программу под названием GRAYSPSPAN можно найти на веб-сайте автора. Ее асимптотическая эффективность может быть доказана с привлечением результата, полученного в упр. 110.

102. Если это так, то обычные остовные деревья могут быть перечислены в *строгом порядке двери-вертушки*, при котором ребра, на каждом шаге входящие в дерево и покидающие его, являются смежными.

Интересные алгоритмы для генерации всех ориентированных остовных деревьев с заданным корнем приведены в работах Harold N. Gabow and Eugene W. Myers, *SICOMP* **7** (1978), 280–287, и S. Kapoor and H. Ramesh, *Algorithmica* **27** (2000), 120–130.

103. (а) Рассыпание лексикографически увеличивает $(x_0, x_1, \ldots, x_n)$, но не изменяет $x_0 + \cdots + x_n$. В каком бы порядке мы ни рассыпали V_i и V_j, результат в любом случае окажется один и тот же.

(б) Добавление песчинки изменяет 16 устойчивых состояний следующим образом.

```
Дано: 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
+ 0001 0001 0010 0011 0001 0101 0110 0111 0101 1001 1010 1011 1001 1101 1110 1111 1101
+ 0010 0010 0011 0001 0010 0110 0111 0101 0110 1010 1011 1001 1010 1110 1111 1101 1110
+ 0100 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111 0100 0101 0110 0111
+ 1000 1000 1001 1010 1011 1100 1101 1110 1111 0100 0101 0110 0111 1000 1001 1010 1011
```

Рекуррентными состояниями являются девять случаев с $x_1 + x_2 > 0$ и $x_3 + x_4 > 0$. Заметим, что многократное добавление 0001 приводит к бесконечному циклу $0000 \to 0001 \to 0010 \to 0011 \to 0001 \to 0010 \to \cdots$; однако состояния 0001, 0010 и 0011 *не* являются рекуррентными.

(в) Если $x = \sigma(x + t)$, то также $x = \sigma(x + kt)$ для всех $k \geq 0$. Все компоненты t положительны; таким образом, $x = \sigma(x + \max(d_1, \ldots, d_n)t)$ — рекуррентное состояние. И наоборот, предположим, что $x = \sigma(d + y)$, где все $y_i \geq 0$; тогда $d + y + t$ рассыпается до состояния $x + t$, а оно рассыпается до $\sigma(d) + y + t = d + y$. Следовательно, $\sigma(x + t) = \sigma(d + y) = x$.

(г) Имеется $N = \det(a_{ij})$ классов, поскольку элементарные операции со строками (упр. 4.6.1–19) приводят матрицу к треугольному виду при сохранении конгруэнтности.

(д) Имеются неотрицательные целые числа $m_1, \ldots, m_n, m'_1, \ldots, m'_n$, такие, что

$$x + m_1 a_1 + \cdots + m_n a_n \;=\; x' + m'_1 a_1 + \cdots + m'_n a_n \;\text{ и равно, скажем, } y.$$

Для достаточно большого k вектор $y + kt$ рассыпается за $m_1 + \cdots + m_n$ шагов до $x + kt$, а за $m'_1 + \cdots + m'_n$ шагов — до $x' + kt$. Следовательно, $x = \sigma(x + kt) = \sigma(x' + kt) = x'$.

(е) Приведение к треугольному виду в ответе (г) показывает, что $x \equiv x + Ny$ для произвольных векторов y. Рассыпание сохраняет конгруэнтность; следовательно, каждый класс содержит рекуррентное состояние.

(ж) Поскольку в сбалансированном ориентированном графе $a = a_1 + \cdots + a_n$, мы получаем $x \equiv x + a$. Если x — рекуррентное состояние, мы видим, что фактически каждая вершина при сокращении $x + a$ до x рассыпается ровно один раз, поскольку векторы $\{a_1, \ldots, a_n\}$ линейно независимы.

И наоборот: если $\sigma(x + a) = x$, мы должны доказать, что состояние x рекуррентно. Пусть $z_m = \sigma(ma)$; должны существовать некоторые положительные k и m, для которых $z_{m+k} = z_m$. Тогда каждая вершина рассыпается k раз, пока $z_m + ka$ сокращается до z_m; следовательно, существуют векторы $y_j = (y_{j1}, \ldots, y_{jn})$ с $y_{jj} \geq d_j$, такие, что $(m + k)a$ рассыпается до y_j. Отсюда следует, что $x + n(m + k)a$ рассыпается до $x + y_1 + \cdots + y_n$ и $\sigma(x + y_1 + \cdots + y_n) = \sigma(x + n(m + k)a) = x$.

(з) При циклическом рассмотрении индексов остовные деревья с дугами $V_j \to V_0$ для $j = i_1, \ldots, i_k$ имеют $n - k$ других дуг: $V_j \to V_{j-1}$ для $i_l < j \leq i_l + q_l$ и $V_j \to V_{j+1}$ для $i_l + q_l < j < i_{l+1}$. Аналогично рекуррентные состояния имеют $x_j = 2$ для $j = i_1, \ldots, i_k$ и $x_j = 1$ для $i_l < j < i_{l+1}$, за исключением $x_j = 0$ при $j = i_l + q_l$ и $q_l > 0$.

(и) В этом случае состояние $x = (x_1, \ldots, x_n)$ является рекуррентным тогда и только тогда, когда $(n - x_1, \ldots, n - x_n)$ является решением задачи о парковке, упомянутой в указании, поскольку $t = (1, \ldots, 1)$, а последовательность неприпаркованных машин оставляет "дыру", которая останавливает рассыпание $x + t$ в x.

Примечания. Эта модель барханов, разработанная Дипаком Дхаром (Deepak Dhar) [*Phys. Review Letters* **64** (1990), 1613–1616], привела к появлению массы статей в физических журналах. Дхар заметил, что если внести случайным образом M песчинок, то при $M \to \infty$ все рекуррентные состояния становятся равновероятными. Данное упражнение основано на работе R. Cori and D. Rossin, *European J. Combinatorics* **21** (2000), 447–459.

Теория барханов доказывает, что каждый ориентированный граф D порождает абелеву группу, элементы которой некоторым образом соответствуют ориентированным остовным деревьям графа D с корнем V_0. В частности, это истинно в случае, когда D — обычный граф с дугами $u \to v$ и $v \to u$ для смежных узлов u и v. Таким образом, например, можно "сложить" два остовных дерева, а некоторое остовное дерево может рассматриваться как "нуль". Элегантное соответствие между остовными деревьями и рекуррентными состояниями в частном случае, когда D представляет собой обычный граф, найдено Р. Кори (R. Cori) и И. Ле Борном (Y. Le Borgne), *Advances in Applied Math.* **30** (2003), 44–52. Однако для ориентированного графа D в общем случае простое соответствие неизвестно. Предположим, например, что $n = 2$ и $(e_{10}, e_{12}, e_{20}, e_{21}) = (p, q, r, s)$; тогда имеется $pr + ps + qr$ ориентированных деревьев, и рекуррентные состояния соответствуют обобщенным двумерным торам, как в упр. 7–137. Но даже в "сбалансированном" случае, когда $p + q \geq s$ и $r + s \geq q$, по-видимому, нет простого отображения остовных деревьев на рекуррентные состояния.

104. (а) Если $\det(\alpha I - C) = 0$, существует вектор $x = (x_1, \ldots, x_n)^T$, такой, что $Cx = \alpha x$ и $\max(x_1, \ldots, x_n) = x_m = 1$ для некоторого m. Тогда $\alpha = \alpha x_m = c_{mm} - \sum_{j \neq m} e_{mj} x_j \geq c_{mm} - \sum_{j \neq m} e_{mj} = 0$. (Кстати, действительная симметричная матрица, собственные значения которой неотрицательны, называется *положительно полуопределенной*. Наше доказательство устанавливает хорошо известный факт, что этим свойством обладает любая

действительная симметричная матрица, у которой $c_{mm} \geq \left|\sum_{j\neq m} c_{mj}\right|$ для $1 \leq m \leq n$.) Таким образом, $\alpha_0 \geq 0$; а так как $C(1,\ldots,1)^T = (0,\ldots,0)^T$, то $\alpha_0 = 0$.

(б) $\det(xI - C(G)) = x(x-\alpha_1)\ldots(x-\alpha_{n-1})$; согласно теореме о матрице, соответствующей дереву, коэффициент при x равен $(-1)^{n-1}n$, умноженному на количество остовных деревьев.

(в) $\det(\alpha I - C(K_n)) = \det((\alpha-n)I + J) = (\alpha-n)^{n-1}\alpha$ в соответствии с упр. 1.2.3–36; здесь J — матрица, все элементы которой равны 1. Таким образом, сторонами полного графа являются 0, n, $\ldots$, n.

105. (а) Если $e_{ij} = a + be'_{ij}$, то $C(G) = naI - aJ + bC(G')$. Если C — произвольная матрица, суммы элементов строк которой равны нулю, то можно доказать тождество

$$\det(xI + yJ - zC) = \frac{x+ny}{x}\, z^n \det((x/z)I - C),$$

прибавляя столбцы от 2-го до n-го к первому, вынося за скобки $(x+ny)/x$, вычитая y/x раз первый столбец из столбцов со 2-го по n-й, а затем вычитая столбцы со 2-го по n-й из первого. Следовательно, принимая $x = \alpha - na$, $y = a$, $z = b$, $a = 1$ и $b = -1$, мы находим, что G имеет стороны 0, $n - \alpha_{n-1}$, $\ldots$, $n - \alpha_1$. (В частности, этот результат согласуется с упр. 104, (в), если G' — пустой граф $\overline{K_n}$.)

(б) Отсортируйте $\{\alpha'_0, \ldots, \alpha'_{n'-1}, \alpha''_0, \ldots, \alpha''_{n''-1}\}$. (Для разнообразия — очень простой случай.)

(в) Здесь $\overline{G} = \overline{G'} \oplus \overline{G''}$, так что в соответствии с (а) и (б) сторонами G являются $\{0, n'+n'', n''+\alpha'_1, \ldots, n''+\alpha'_{n'-1}, n'+\alpha''_1, \ldots, n'+\alpha''_{n''-1}\}$. (В частности, G представляет собой $K_{m,n}$, если $G' = \overline{K_m}$ и $G'' = \overline{K_n}$, следовательно, сторонами $K_{m,n}$ являются $\{0$, $(n-1)\cdot m$, $(m-1)\cdot n$, $m+n\}$.)

(г) $C(G) = I_{n'} \otimes C(G'') + C(G') \otimes I_{n''}$, где I_n обозначает единичную матрицу размером $n \times n$, а символ $\otimes$ — прямое произведение матриц (произведение Кронекера). Сторонами $C(G)$ являются $\{\alpha'_j + \alpha''_k \mid 0 \leq j < n', 0 \leq k < n''\}$; если A и B — произвольные матрицы, собственными значениями которых являются $\{\lambda_1, \ldots, \lambda_m\}$ и $\{\mu_1, \ldots, \mu_n\}$ соответственно, то собственные значения $A \otimes I_n + I_m \otimes B$ представляют собой mn сумм $\lambda_j + \mu_k$.

Доказательство. Выберем S и T так, что $S^{-}AS$ и $T^{-}BT$ — треугольные матрицы. Затем воспользуемся матричным тождеством $(A\otimes B)(C\otimes D) = AC \otimes BD$, чтобы показать, что $(S\otimes T)^{-}(A\otimes I_n + I_m \otimes B)(S\otimes T) = (S^{-}AS)\otimes I_n + I_m \otimes (T^{-}BT)$. (В частности, многократное применение этой формулы показывает, что сторонами n-мерного куба являются $\{\binom{n}{0}\cdot 0$, $\binom{n}{1}\cdot 2$, $\ldots$, $\binom{n}{n}\cdot 2n\}$, а формула (57) следует из упр. 104, (б).)

(д) Если G — однородный граф степени d, его сторонами являются $\alpha_j = d - \lambda_{j+1}$, где $\lambda_1 \geq \cdots \geq \lambda_n$ — собственные значения матрицы смежности $A = (a_{ij})$. Матрицей смежности графа G' является $A' = B^TB - d'I_{n'}$, где $B = (b_{ij})$ — матрица инциденций размером $n \times n'$ с элементами $b_{ij} = [$ребро i соединено с вершиной $j]$, а $n = n'd'/2$ — количество ребер. Матрицей смежности графа G является $A = BB^T - 2I_n$. Мы имеем

$$x^n \det(xI_{n'} - B^TB) \;=\; x^{n'} \det(xI_n - BB^T);$$

это тождество следует из того факта, что коэффициенты $\det(xI - A)$ можно выразить через $\operatorname{trace}(A^k)$ для $k = 1, 2, \ldots$ с использованием тождества Ньютона (упр. 1.2.9–10). Так что сторонами G являются стороны G' плюс $n - n'$ сторон, равных $2d'$. [Этот результат получен в работе Е. Б. Ваховский, *Сибирский Мат. журнал* **6** (1965), 44–49; см. также H. Sachs, *Wissenschaftliche Zeitschrift der Technischen Hochschule Ilmenau* **13** (1967), 405–412.]

(е) $A = A' \otimes A''$, так что сторонами являются $\{d''\alpha'_j + d'\alpha''_k - \alpha'_j\alpha''_k \mid 0 \leq j < n', 0 \leq k < n''\}$.

(ж) $A(G) = I_{n'} \otimes A'' + A' \otimes I_{n''} + A' \otimes A'' = (I_{n'} + A') \otimes (I_{n''} + A'') - I_n$ дает стороны $\{(d''+1)\alpha'_j + (d'+1)\alpha''_k - \alpha'_j\alpha''_k \mid 0 \leq j < n', 0 \leq k < n''\}$.

(з) Если G' — однородный граф, можно сделать $S^- A' S$ диагональной матрицей с элементами $d' - \alpha'_j$, при этом одновременно $S^- J_{n'} S$ является диагональной матрицей с элементами $(n', 0, \ldots, 0)$, поскольку $(1, \ldots, 1)^T$ представляет собой собственный вектор как для A', так и для $J_{n'}$. Таким образом, в соответствии с формулой из ответа к упр. 7–96, (в), стороны равны $\{d + (d' - \alpha'_j n'[j=0])(d'' - \alpha''_k) + (d' - \alpha'_j)(d'' - \alpha''_k - n''[k=0]) \mid 0 \le j < n', 0 \le k < n''\}$, где $d = d'(n'' - d'') + (n' - d')d''$.

(и) Аналогичные рассуждения приводят к масштабированным сторонам $\{n''\alpha'_j \mid 0 \le j < n'\}$ графа G', совместно с n' копиями смещенных сторон $\{d'n'' + \alpha''_k \mid 1 \le k < n''\}$ графа G''.

106. (а) Если α — сторона пути P_n, то имеется ненулевое решение $(x_0, x_1, \ldots, x_{n+1})$ уравнений $\alpha x_k = 2x_k - x_{k-1} - x_{k+1}$ для $1 \le k \le n$, причем $x_0 = x_1$ и $x_n = x_{n+1}$. Если положить $x_k = \cos(2k-1)\theta$, то мы получим $x_0 = x_1$ и $2x_k - x_{k-1} - x_{k+1} = 2x_k - (2\cos 2\theta)x_k$; следовательно, $2 - 2\cos 2\theta = 4\sin^2\theta$ будет стороной, если выбрать θ так, чтобы $x_n = x_{n+1}$ и чтобы все x не были нулями. Таким образом, сторонами P_n являются $\sigma_{0n}, \ldots, \sigma_{(n-1)n}$.

Поскольку $c(P_n) = 1$, из упр. 104, (б) должно получаться $\alpha_1 \ldots \alpha_{n-1} = n$; следовательно,

$$c(P_m \,\square\, P_n) \;=\; \prod_{j=1}^{m-1} \prod_{k=1}^{n-1} (\sigma_{jm} + \sigma_{kn}).$$

(б), (в) Аналогично, если α является стороной цикла C_n, имеется ненулевое решение указанных уравнений, такое, что $x_n = x_0$. В этом случае мы испытываем $x_k = \cos 2k\theta$ и находим решения при $\theta = j\pi/n$ для $0 \le j < \lceil n/2 \rceil$. В то же время $x_k = \sin k\theta$ дает остальные линейно независимые решения для $\lceil n/2 \rceil \le j < n$. Следовательно, сторонами C_n являются $\sigma_{0n}, \sigma_{2n}, \ldots, \sigma_{(2n-2)n}$; и мы получаем

$$c(P_m \,\square\, C_n) = n \prod_{j=1}^{m-1} \prod_{k=1}^{n-1} (\sigma_{jm} + \sigma_{(2k)n}), \quad c(C_m \,\square\, C_n) = mn \prod_{j=1}^{m-1} \prod_{k=1}^{n-1} (\sigma_{(2j)m} + \sigma_{(2k)n}).$$

Пусть $f_n(x) = (x + \sigma_{1n}) \ldots (x + \sigma_{(n-1)n})$ и $g_n(x) = (x + \sigma_{2n}) \ldots (x + \sigma_{(2n-2)n})$. Эти полиномы имеют целые коэффициенты; в самом деле, $f_n(x) = U_{n-1}(x/2 + 1)$ и $g_n(x) = 2(T_n(x/2 + 1) - 1)/x$, где $T_n(x)$ и $U_n(x)$ представляют собой полиномы Чебышева, определяемые как $T_n(\cos\theta) = \cos n\theta$ и $U_n(\cos\theta) = (\sin(n+1)\theta)/\sin\theta$. Вычисление $c(P_m \,\square\, P_n)$ можно свести к вычислению детерминанта размером $m \times m$, поскольку он является результантом $f_m(x)$ и $f_n(-x)$; см. упр. 4.6.1–12. Аналогично $\frac{1}{n} c(P_m \,\square\, C_n)$ и $\frac{1}{mn} c(C_m \,\square\, C_n)$ представляют собой результанты $f_m(x)$ и $g_n(-x)$, а также $g_m(x)$ и $g_n(-x)$ соответственно.

Пусть $\alpha_n(x) = \prod_{d \backslash n} f_d(x)^{\mu(n/d)}$; таким образом, $\alpha_1(x) = 1$, $\alpha_2(x) = x + 2$, $\alpha_3(x) = (x+3)(x+1)$, $\alpha_4(x) = x^2 + 4x + 2$, $\alpha_5(x) = (x^2 + 5x + 5)(x^2 + 3x + 1)$, $\alpha_6(x) = x^2 + 4x + 1$ и т. д. Рассматривая так называемые характеристические полиномы, можно показать, что полином $\alpha_n(x)$ является неприводимым над целыми числами при четном n, а в противном случае представляет собой произведение двух неприводимых полиномов одной и той же степени. Аналогично если $\beta_n(x) = \prod_{d \backslash n} g_d(x)^{\mu(n/d)}$, то $\beta_n(x)$ является квадратом неприводимого полинома при $n \ge 3$. Эти факты поясняют наличие достаточно малых простых множителей в результатах. Например, наибольший простой множитель в $c(P_m \,\square\, P_n)$ при $m \le n \le 10$ равен 1009; он встречается только в результанте $\alpha_6(x)$ с $\alpha_9(-x)$, который равен $662\,913 = 3^2 \cdot 73 \cdot 1009$.

107. Для $n = (1, \ldots, 5)$ имеется $(1, 1, 2, 6, 21)$ неизоморфных графов; однако в силу упр. 105, (а) нам нужно рассмотреть только случаи с $\le \frac{1}{2}\binom{n}{2}$ ребрами. Оставшиеся случаи при $n = 4$ представляют собой свободные деревья: звезда является дополнением к $K_1 \oplus K_3$ со сторонами 0, 1, 1, 4; а путь P_4 согласно упр. 106 имеет стороны $0,\ 2 - \sqrt{2},\ 2,\ 2 + \sqrt{2}$. При $n = 5$ имеется три свободных дерева: звезда со сторонами 0, 1, 1, 1, 5; путь P_5, стороны

которого — 0, $2-\phi$, $3-\phi$, $1+\phi$, $2+\phi$; а также со сторонами 0, r_1, 1, r_2, r_3, где $(r_1, r_2, r_3) \approx (0.52, 2.31, 4.17)$ являются корнями уравнения $x^3 - 7x^2 + 13x - 5 = 0$.

Наконец имеется пять случаев с одним циклом: представляет собой K_1—$(K_2 \oplus \overline{K_2})$, так что его сторонами являются 0, 1, 1, 3, 5; C_5 имеет стороны 0, $3-\phi$, $3-\phi$, $2+\phi$, $2+\phi$; имеет стороны 0, r_1, r_2, 3, r_3; сторонами его дополнения являются 0, $5-r_3$, 2, $5-r_2$, $5-r_1$; а стороны равны 0, $(5-\sqrt{13})/2$, $3-\phi$, $2+\phi$, $(5+\sqrt{13})/2$.

108. Пусть дан ориентированный граф D с вершинами $\{V_1, \ldots, V_n\}$ и пусть e_{ij} — количество дуг от V_i к V_j. Определим $C(D)$ и его стороны, как и ранее. Поскольку матрица $C(D)$ не обязательно симметрична, действительность сторон больше не гарантирована. Но если α представляет собой сторону, то стороной является и комплексно сопряженное значение $\bar{\alpha}$; и если мы упорядочим стороны в соответствии с их действительными частями, то, как и ранее, получим $\alpha_0 = 0$. Формула $c(D) = \alpha_1 \ldots \alpha_{n-1}/n$ остается корректной, если рассматривать $c(D)$ как *среднее* количество *ориентированных* остовных деревьев, взятое по всем n возможным корням V_j. Очевидно, что стороны транзитивного турнира $\vec{K_n}$, дугами которого являются $V_i \to V_j$ при $1 \le i < j \le n$, равны 0, 1, $\ldots$, $n-1$; очевидны также и стороны его подграфов.

Выводы в пп. (а)–(г) ответа к упр. 105 остаются без изменений. Например, рассмотрим K_1—$\vec{K_3}$, сторонами которого являются 0, 2, 3, 4; этот ориентированный граф D имеет $(2, 4, 6, 12)$ ориентированных остовных деревьев с четырьмя возможными корнями, и $c(D)$ действительно равно $2 \cdot 3 \cdot 4/4$. Обратите также внимание, что ориентированный граф является своим собственным дополнением и имеет те же стороны, что и $\vec{K_3}$.

Ориентированные графы допускают также другое семейство интересных операций: если D' и D'' являются ориентированными графами на непересекающихся множествах вершин V' и V'', рассмотрим добавление a дуг $v' \to v''$ и b дуг $v'' \to v'$, где $v' \in V'$ и $v'' \in V''$. Работая с определителями так же, как и в упр. 105, (а), можно показать, что полученный в результате ориентированный граф имеет стороны $\{0, an'' + bn', an'' + \alpha'_1, \ldots, an'' + \alpha'_{n'-1}, bn' + \alpha''_1, \ldots, bn' + \alpha''_{n''-1}\}$. В частном случае $a = 1$ и $b = 0$ можно ввести удобное обозначение для полученного ориентированного графа: $D' \to D''$; так, например, $\vec{K_n} = K_1 \to \vec{K_{n-1}}$. Ориентированный граф $K_{n_1} \to K_{n_2} \to \cdots \to K_{n_m}$ на $n_1 + n_2 + \cdots + n_m$ вершинах имеет стороны $\{0, n_m \cdot s_m, \ldots, n_2 \cdot s_2, (n_1 - 1) \cdot s_1\}$, где $s_k = n_k + \cdots + n_m$.

Сторонами ориентированного пути $\vec{P_n}$ от V_1 до V_n, очевидно, являются 0, 1, $\ldots$, 1. Ориентированный цикл $\vec{C_n}$ имеет стороны $\{0, 1-\omega, \ldots, 1-\omega^{n-1}\}$, где $\omega = e^{2\pi i/n}$.

Имеется также красивый результат для ориентированного графа, в котором дуги соответствуют вершинам исходного графа: стороны графа D^* получаются из сторон графа D простым добавлением $\tau_k - 1$ копий числа σ_k $(1 \le k \le n)$, где τ_k — входящая степень вершины V_k, а σ_k — ее исходящая степень. (Если $\tau_k = 0$, мы *удаляем* одну сторону, равную σ_k.) Соответствующее доказательство похоже (хотя и проще) на доказательство из упр. 2.3.4.2–21.

Историческая справка. Результаты упр. 104, (б) и 105, (а) основаны на работах А. К. Кельманса, *Автоматика и Телемеханика* **26** (1965), 2194–2204; **27**, 2 (февраль 1966), 56–65 (переведена на английский язык в *Automation and Remote Control* **26** (1965), 2118–2129; **27** (1966), 233–241). Мирославу Фидлеру (Miroslav Fiedler) [*Czech. Math. J.* **23** (1973), 298–305] принадлежит авторство упр. 105, (г); им же доказаны интересные результаты о стороне α_1, которую он назвал алгебраической связностью графа G. Герман Креверас (Germain Kreweras) в *J. Combinatorial Theory* **B24** (1978), 202–212, пересчитал остовные деревья решеток, цилиндров и торов, а также ориентированные остовные деревья ориентированных торов, таких, как $\vec{C_m} \square \vec{C_n}$. Отличный обзор сторон графов опубликован Бояном Мохаром (Bojan Mohar) в *Graph Theory, Combinatorics and Applications* (Wiley, 1991), 871–898; *Discrete Math.* **109** (1992), 171–183. Исчерпывающее обсуждение важных семейств собственных значений графов и их свойств, включая полную библиографию,

можно найти в книге D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of Graphs* by third edition (1995).

109. Вероятно, пояснение связано с барханами из упр. 103.

110. Доказательство по индукции. Предположим, что имеется $k \geq 1$ параллельных ребер между u и v. Тогда $c(G) = kc(G_1) + c(G_2)$, где G_1 — это граф G с отождествленными вершинами u и v, а G_2 — граф G, из которого удалены эти k ребер. Пусть $d_u = k + a$ и $d_v = k + b$.

Случай 1. G_2 — связный граф. Тогда $ab > 0$, так что можно записать $a = x + 1$ и $b = y + 1$. Мы имеем $c(G_1) > \alpha\sqrt{x + y + 1}$ и $c(G_2) > \alpha\sqrt{xy}$, где α — произведение по всем остальным $n - 2$ вершинам; легко убедиться, что

$$k\sqrt{x + y + 1} + \sqrt{xy} \;\geq\; \sqrt{(x + k)(y + k)}.$$

Случай 2. Не существует таких вершин u и v, для которых G_2 — связный граф. Тогда каждое мультиребро графа G представляет собой мост; другими словами, G — свободное дерево, за исключением наличия параллельных ребер. В этом случае результат тривиален, если существует вершина степени 1. В противном случае предположим, что u — конечная точка и ее степень равна $d_u = k$ ребер u — v. Если $d_v > k + 1$, то $c(G) = kc(G_1) > \alpha k\sqrt{x}$, где $d_v = k + 1 + x$, и легко проверить, что при $x > 0$ справедливо неравенство $k\sqrt{x} > \sqrt{(k - 1)(k + x)}$. Если $d_v = k$, мы получаем $c(G) = k > \sqrt{(k - 1)^2}$. Наконец, если $d_v = k + 1$, примем $v_0 = u$, $v_1 = v$ и рассмотрим единственный путь v_1 — v_2 — $\cdots$ — v_r, где $r > 1$ и v_r имеет степень, бóльшую 2; каждая пара вершин v_j и v_{j+1} $(1 \leq j < r)$ соединяется только одним ребром. Гипотеза индукции выполняется.

[Другие нижние границы количества остовных деревьев получены в работе A. V. Kostochka, *Random Structures & Algorithms* **6** (1995), 269–274.]

111. `2 1 5 4 11 7 9 8 6 10 15 12 14 13 3`.

112. Либо p находится на четном уровне и является предком q, либо q находится на нечетном уровне и является предком p.

113. prepostorder(F^R)=postpreorder$(F)^R$ и postpreorder(F^R)=prepostorder$(F)^R$.

114. Наиболее элегантный подход, учитывающий, что лес может быть пустым, заключается в такой инициализации, что `CHILD(Λ)` указывает на корень крайнего слева дерева, если таковое имеется. Затем первое посещение обеспечивается установками `Q` $\leftarrow \Lambda$, `L` $\leftarrow -1$ с последующим переходом к шагу Q6.

115. Предположим, что имеется n_e узлов на четных уровнях и n_o узлов на нечетных уровнях и что n'_e узлов на четных уровнях не являются листьями. Тогда шаги (Q1, ..., Q7) выполняются соответственно $(n_e + n_o,\ n_o,\ n'_e,\ n_e,\ n'_e,\ n_o + 1,\ n_e)$ раз, включая одно выполнение шага Q6 в соответствии с ответом к упр. 114.

116. (а) Этот результат следует из алгоритма Q.

(б) В действительности все узлы, не являющиеся обычными, строго чередуются между везучими и невезучими, начиная и заканчивая везучими узлами. *Доказательство*: рассмотрите лес F', полученный из леса n путем удаления крайнего слева листа, и воспользуйтесь индукцией по n.

117. Такие леса — это те, которые в представлении с левым сыном и правым братом дают вырожденное бинарное дерево (упр. 31). Таким образом, окончательный ответ — 2^{n-1}.

118. (а) t^{k-2} для $k > 1$; везение встречается только возле крайних листьев.

(б) Получающееся интересное рекуррентное соотношение приводит к решению $(F_k + 1 - (k + 1) \bmod 3)/2$.

119. Пометим каждый узел x значением $v(x) = \sum\{\, 2^k \mid k$ — метка дуги на пути от корня к $x\}$. Тогда значения узлов в прямообратном порядке обхода представляют собой бинарный код Грея Γ_n, поскольку в упр. 113 показано, что они удовлетворяют рекуррентному соотношению 7.2.1.1–(5).

(Если применить тот же способ назначения меток к обычному биномиальному дереву T_n и обойти его в прямом порядке, то мы получим просто целые числа 0, 1, ..., $2^n - 1$.)

120. Ложно. Только четыре из "полых" вершин на приведенной иллюстрации могут следовать за двумя "квадратными" вершинами в гамильтоновом цикле; следовательно, одной полой паре не повезло. [См. H. Fleischner and H. V. Kronk, *Monatshefte für Mathematik* **76** (1972), 112–117.]

121. Более того, гамильтонов путь от u до v в T^2 существует тогда и только тогда, когда выполняются подобные условия, описанные далее. Мы оставляем u и/или v в $T^{(\prime)}$, если они имеют степень 1, и требуем, чтобы путь в условии (i) находился в пути от u к v (исключая сами u и v). Условие (ii) также усиливается посредством замены 'вершины степени 4' на 'опасные вершины', где вершина $T^{(\prime)}$ называется опасной, если она либо имеет степень 4, либо имеет степень 2 и равна u или v. Наименьший невозможный случай — $T = P_4$, где u и v выбраны так, чтобы они не являлись конечными точками. [*Casopis pro Pěstováni Matematiky* **89** (1964), 323–338.]

Следовательно, T^2 содержит гамильтонов цикл тогда и только тогда, когда T является *гусеницей* (*caterpillar*), т. е. свободным деревом, производная которого представляет собой путь. [См. Frank Harary and A. J. Schwenk, *Mathematika* **18** (1971), 138–140.]

122. (а) Можно представить выражение в виде бинарного дерева, в котором операторы представлены внутренними узлами, а цифры — внешними. Если бинарное дерево реализовано так, как в алгоритме B, то главное ограничение, налагаемое грамматикой, состоит в том, что если $r_j = k > 0$, то оператором в узле j будет $+$ или $-$ тогда и только тогда, когда оператором в узле k будет $\times$ или $/$. Следовательно, общее количество возможных деревьев с n листьями равно $2^n S_{n-1}$, где S_n — число Шрёдера; при $n = 9$ это значение равно 10 646 016. (См. упр. 66, только замените в нем "лево" на "право" и наоборот.) Можно быстро сгенерировать их все и убедиться, что всего имеется 1641 решение. Без умножения обходится только одно выражение, а именно $1 + 2/((3-4)/(5+6) - (7-8)/9)$; двадцать выражений требуют пять пар скобок, как в $(((1-2)/((3/4) \times 5 - 6)) \times 7 + 8) \times 9$; совсем без скобок обходятся только пятнадцать выражений.

(б) В этом случае существует $1 + \sum_{k=1}^{8} \binom{8}{k} 2^{k+1} S_k = 23\,463\,169$ возможных деревьев и 3366 решений. Самое короткое, длиной 12, было найдено Дьюдени [*The Weekly Dispatch* (18 June 1899)] и имеет вид $123 - 45 - 67 + 89$; однако сам Дьюдени в то время не был уверен, что оно наилучшее. Самое длинное решение имеет длину 27; таких решений, как упоминалось выше, двадцать.

(в) При этом количество возможных случаев резко возрастает до $2 + \sum_{k=1}^{8} \binom{8}{k} 4^{k+1} S_k = 8\,157\,017\,474$, и теперь имеется 97 221 решение. Самое длинное среди них (оно единственное) имеет длину 40: $((((.1/(.2 + .3))/.4)/.5)/(.6 - .7))/(.8 - .9)$. Есть пять занимательных примеров наподобие $.1 + (2 + 3 + 4 + 5) \times 6 + 7 + 8 + .9$ с семью плюсами, а также десять решений наподобие $(1 - .2 - .3 - 4 - .5 - 6) \times (7 - 8 - 9)$ с семью минусами.

Есть только одно небольшое правило и ни одного точного способа показать, что нами получено наилучшее возможное решение.

— ГЕНРИ Э. ДЬЮДЕНИ (HENRY E. DUDENEY) (1899)

Примечания. В первом издании *Illustriertes Spielbuch für Mädchen* (1864) Мари Леске (Marie Leske) содержится первое известное упоминание этой задачи; в одиннадцатом издании (1889) приведено решение $100 = 1+2+3+4+5+6+7+8\times 9$. См. также ссылку из упр. 7.2.1.1–111.

Ричард Беллман (Richard Bellman) пояснил в *AMM* **69** (1962), 640–643, как решается частный случай (а), когда операторы ограничены сложением и умножением и нельзя использовать скобки. Его метод динамического программирования можно применять и при решении более общей задачи для снижения количества рассматриваемых случаев. Идея состоит в определении действительных чисел, получаемых для каждого подынтервала цифр $\{1,\ldots,n\}$ для данного оператора в качестве корня дерева. Можно также сэкономить на вычислениях, отбрасывая случаи, которые для подынтервалов $\{1,\ldots,8\}$ и $\{2,\ldots,9\}$ не могут привести к целым решениям. Таким образом, количество существенно различных деревьев, которые требуется рассмотреть, уменьшается до (а) 2 735 136; (б) 6 813 760; (в) 739 361 319.

Арифметика с плавающей точкой для решения данной задачи ненадежна, но можно воспользоваться подпрограммами для работы с рациональными числами из раздела 4.5.1; при этом никогда не приходится работать с целыми числами, по абсолютному значению превышающими 10^9.

123. (а) 2284; но $2284 = (1+2\times 3)\times(4+5\times 67)-89$. (б) 6964; но $6964 = (1/.2)\times 34+5+6789$. (в) 14 786; но $14\,786 = -1+2\times(.3+4+5)\times(6+789)$. [Если допустить использование знака “минус” слева от выражения, как это делал Дьюдени, то получится 1362, 2759 и 85 597 дополнительных решений упр. 122, (а)–(в), включая девятнадцать более длинных выражений в задаче (а), таких как $-(1-2)\times((3+4)\times(5-(6-7)\times 8)+9)$. При таком расширении соглашений наименьшими непредставимыми числами являются (а) 3802, (б) 8312 и (в) 17 722.] Общее количество представимых целых чисел (положительных, отрицательных и нуля) оказывается равным (а) 27 666; (б) 136 607; (в) 200 765.

124. Числа Хортона–Страхлера впервые возникли при изучении течения рек в работах R. E. Horton, *Bull. Geol. Soc. Amer.* **56** (1945), 275–370; A. N. Strahler, *Bull. Geol. Soc. Amer.* **63** (1952), 1117–1142. Многие идеи по рисованию деревьев можно найти в классической статье Viennot, Eyrolles, Janey, and Arquès, *Computer Graphics* **23**, 3 (July 1989), 31–40.

РАЗДЕЛ 7.2.1.7

1. Вероятно, оно связано с гексаграммой 21 (“решение числовых задач большого объема” (“crunching”)), (䷔); однако древние комментаторы в большей степени связывают эту гексаграмму с обеспечением законности, чем со взаимодействием электронов*.

2. (а) Пусть первый нуклеотид в кодоне (T, C, A, G) будет представлен соответственно элементами (䷁, ䷗, ䷀, ䷖); второй нуклеотид аналогично представлен соответственно элементами (䷲, ䷳, ䷢, ䷣); а третий — элементами (䷊, ䷉, ䷋, ䷈). Наложим одно на другое все три представления. Таким образом, например, гексаграмма 34 представляет собой ䷡ = ䷁ + ䷲ + ䷊ и, таким образом, является кодоном TTC, который отображается на аминокислоту F. При использовании такого соответствия гексаграммы 34–54 отображаются соответственно на значения (F, G, L, Q, W, D, S, −, P, Y, K, A, I, T, N, H, M, R, V, E, C). Кроме того, имеется три гексаграммы, отображающиеся на ‘−’, — это гексаграммы с номерами 1, 9 и 41, а именно ䷀, ䷈ и ䷨, которые в *I Ching* имеют названия соответственно “созидание”, “укрощение” и “устранение излишнего”; все они на удивление подходят для обозначения завершения белка.

---

* По всей видимости, автор подразумевает еще одно значение слова “crunching” — “раздавливать с хрустом”. — *Примеч. пер.*

(б) Рассмотрим $\binom{64}{6,6,6,4,4,4,4,4,3,3,2,2,2,2,2,2,2,2,2,1,1} \approx 2.3 \times 10^{69}$ способов перестановки элементов массива генетического кода размером $4 \times 4 \times 4$. Ровно

$$240288040217578979000399368196455132845166871875018555392000000 0 \approx 2.4 \times 10^{63}$$

из них содержат как минимум один отрезок, состоящий из 21 различного последовательного элемента. [Использование принципа включения и исключения позволяет показать, что любое мультимножество $\{(n_1{+}1)\cdot x_1, \ldots, (n_r{+}1)\cdot x_r\}$ с r различными элементами и $n_r = 0$ имеет ровно

$$(n+1)\binom{n}{n_1,\ldots,n_r} r! - \sum_{k=1}^{r} (n+1-k)k!(r-k)!\, a_k \sum_{\substack{0\le d_1,\ldots,d_r\le 1\\ d_1+\cdots+d_r=k}} \binom{n-k}{n_1-d_1,\ldots,n_r-d_r}$$

таких перестановок, где $n = n_1 + \cdots + n_r$, а a_k — количество неприводимых перестановок с k элементами (упр. 7.2.1.2–100).] Таким образом, только одна из каждого миллиона перестановок обладает указанным свойством.

Однако имеется $4!^3\binom{6}{2,2,2} = 1244160$ способов представления кодонов так, как это сделано в п. (а), и большинство из них соответствуют различным перестановкам аминокислот (за исключением взаимообмена представлений `T` и `C` в третьей позиции).

И действительно, эмпирически около 31% всех перестановок 64 гексаграмм дают соответствующее отображение кодонов. Таким образом, построение в п. (а) не дает оснований подозревать авторов *I Ching* в каком-либо участии в разработке генетического кода.

3. Поскольку $F_{31} - 10^6 = F_{28} + F_{22} + F_{20} + F_{18} + F_{16} + F_{14} + F_9$, миллионной строфой является

◡◡◡◡◡◡◡—◡◡◡—————◡◡◡◡—◡◡.

Ответ на второй вопрос еще проще: $F_{31} - (F_5 + F_8 + F_{10} + F_{16} + F_{18} + F_{27} + F_{30}) = 314\,159$.

4. Одно из двух появлений **כמדיך** в строке 4 должно быть **כמידך**; аналогично одно из **ירדמנ** в строке 8 должно быть **ירמדנ**. В шести случаях крайние справа буквы **כך** повторяются дважды в строках 3 и 4, в то время как перестановки с крайними справа буквами **כך** отсутствуют. Эти недочеты, вероятно, вызваны типографскими опечатками и/или ошибками переписчика, но не являются ошибками самого Донноло.

5. Последней перестановкой должна быть , а не .

6. В списке Мерсенна n-е значение m_n равно $n!$ только при $1 \le n \le 13$ и $15 \le n \le 38$. Мерсенн знал, что $14! = 87178291200 \ne m_{14} = 8778291200$, поскольку он вставил отсутствующую единицу в личную копию книги (в настоящее время являющуюся собственностью Bibliothèque Nationale; ее факсимиле было опубликовано в 1963 году). Однако остальные ошибки в его книге не просто типографские, поскольку они распространяются на последующие значения, за исключением случая m_{50}, а именно: $m_{39} = 39! + 10^{26} - 10^{10}$; $m_{40} = 40m_{39}$; $m_{41} = 41m_{40} - 4\cdot 10^{25} - 14\cdot 10^{11}$; $m_n = nm_{n-1}$ для $n = 42$, 43, 44, 46, 47, 48, 49, 55, 60 и 62; $m_{50} = 50m_{49} + 10^{66}$; $m_{51} = 51\cdot 50\cdot m_{49}$. Похоже, при вычислении $m_{45} = 9\cdot 45\cdot m_{44} - 10^{40} + 10^{29}$ Мерсенн хотел упростить вычисления, поскольку легче выполнить умножение на 5 и на 9, чем на 45. Но он ошибся и выполнил умножение на 9 *дважды*. Большинство ошибок указывают на ненадежность метода умножения, который может зависеть от того, какие счеты использованы для умножения: $m_{52} = 52m_{51} + 5\cdot 10^{56} - 2\cdot 10^{47} + 10^{34}$; $m_{53} = 53m_{52} - 4\cdot 10^{29}$; $m_{54} = 54m_{53} + 10^{16}$; $m_{57} = 57m_{56} + 10^{33} + 10^{24}$; $m_{58} = 58m_{57} + 10^{67} - 10^{35} + 10^{32} + 11\cdot 10^{26}$; $m_{59} = 59m_{58} + 10^{66} + 10^{49} - 10^{28}$; $m_{61} = 61m_{60} - 5\cdot 10^{81}$; $m_{63} = 63m_{62} + 10^{82} - 10^{74}$; $m_{64} = 64m_{63} + 3\cdot 10^{81} + 10^{67} + 2\cdot 10^{38} - 2\cdot 10^{33} - 10^{23}$.

Оставшийся случай, $m_{56} \approx 10.912m_{55}$, непостижим. Это значение $\equiv 56m_{55}$ (по модулю 10^{17}), но эти цифры также ничего не говорят ни уму, ни сердцу. Можно ли их как-то правдоподобно пояснить?

Примечания. Анастасиус Кирхер (Athanasius Kircher) явно копировал результаты Мерсенна, когда табулировал $n!$ для $1 \le n \le 50$ на с. 157 своей книги *Ars Magna Sciendi* (1669), поскольку он воспроизвел все ошибки Мерсенна. Кирхер, впрочем, использовал значения $10m_{14}$, $m_{45}/10$ и $10m_{49}$ вместо m_{14}, m_{45} и m_{49}; вероятно, он пытался сделать последовательность растущей более равномерно. Неясно, кто первым вычислил точное значение 39!; в упр. 1.2.5–4 описана история числа 1000!.

7. Базовыми перестановками являются 12345, 13254, 14523, 15432, 12453, 14235, 15324, 13542, 12534, 15243, 13425, 14352. Но после этого мы находим, что все 60 четных перестановок одновременно и живы, и мертвы, поскольку (9) отличается от (8) на четную перестановку. (Кроме того, если бы как-то удалось исправить ситуацию при $n = 5$, половина живых перестановок при $n = 6$ стали бы нечетными.)

8. Например, можно заменить (9) на

$$a_n a_3 \ldots a_{n-1} a_2 a_1,\ a_1 a_4 \ldots a_n a_3 a_2,\ \ldots,\ a_{n-1} a_2 \ldots a_{n-2} a_1 a_n,$$

где для перестановок из (8) выполняются обмен местами двух конечных элементов слева и справа и циклический сдвиг последовательности влево на один элемент. Такое изменение работает правильно, поскольку все перестановки имеют корректную четность, а живые и мертвые перестановки содержат a_1 во всех возможных положениях. (По сути, мы получаем дуальную таблицу Симса для чередующейся группы, как в 7.2.1.2–(32); однако вместо $(0, 1, \ldots, n-1)$ наши элементы именованы как $(n, n-1, \ldots, 1)$.)

Более простой способ генерации перестановок с корректными знаками был опубликован Э. Безу (É. Bézout) [*Mémoires Acad. Royale des Sciences* (Paris, 1764), 292]. Каждая перестановка $\pm a_1 \ldots a_{n-1}$ множества $\{1, \ldots, n-1\}$ дает n других, $\pm a_1 \ldots a_{n-1} a_n \mp a_1 \ldots a_{n-2} a_n a_{n-1} \pm \cdots$.

9. (٠, ١, ٢, ٣, ٤, ٥, ٦, ٧, ٨, ٩); или, пожалуй, мы должны сказать (٩, ٨, ٧, ٦, ٥, ٤, ٣, ٢, ١, ٠).

Примечания. Для порядковых номеров в уравнениях использовалась иная система; например, 'ر' означало 200. Кроме того, следует заметить, что (11) на самом деле представляет собой перевод работы аль-Самаваля на *современный* арабский; Ахмад и Рашед основывались на копии XIV века, в которой использовалось похожее, но более старое начертание цифр: (٥, ١, ٢, ٣, , , ٦, ٧, ٨, ٩). Сам аль-Самаваль мог пользоваться еще более древним начертанием.

10. Если все 56 вариантов равновероятны, ответ равен $56H_{56} \approx 258.2$, как в задаче о собирании купонов (упр. 3.3.2–8). Но $(6, 30, 20)$ случаев имеют вероятности соответственно $(1/216, 1/72, 1/36)$; так что верным будет ответ

$$\int_0^\infty \bigl(1 - (1 - e^{-t/216})^6 (1 - e^{-t/72})^{30} (1 - e^{-t/36})^{20}\bigr)\, dt \approx 546.6,$$

что составляет около 42% от верхней границы $216H_{216}$. [См. P. Flajolet, D. Gardy, and L. Thimonier, *Discrete Applied Math.* **39** (1992), 207–229.]

11. В таблице приведены $\binom{6}{3} = 20$ сочетаний (b, c, d, B, C, D) по три; кроме того, они находятся в лексикографическом порядке, если считать b < c < d < B < C < D. Буква t (𝔱) означает "переход от нижнего регистра к верхнему". [См. A. Bonner, *Selected Works of Ramon Llull* (Princeton: 1985), 596–597.] На рисунке есть две опечатки: в начале строки 6 'd' должно быть 'b'; в конце строки 18 'c' должно быть 'd'. Строка 1 была бы более согласованной с остальными, если бы имела вид

;

но в этой строке переключение регистра, конечно же, не требуется.

12. Умножьте цикл Пуансо на 5 и прибавьте 2 (по модулю 7).

13. При наличии n различных букв лучше писать только n различных строк.

$$\frac{a.\ aa.\ aaa}{b.\ ab.\ aab.\ aaab.\ bb.\ abb.\ aabb.\ aaabb}$$

Тогда присваивание весов $(a, b) = (1, 4)$ дает числа от 1 до 11, как в (18). (Первая строка (16) также должна быть опущена.) Аналогично для $\{a, a, a, b, b, c\}$ можно неявно присвоить c вес 12 и добавить дополнительную строку

$$c.\ ac.\ aac.\ aaac.\ bc.\ abc.\ aabc.\ aaabc.\ bbc.\ abbc.\ aabbc.\ aaabbc.$$

[Я. Бернулли (J. Bernoulli) *почти* сделал это в *Ars Conjectandi*, часть 2, глава 6.]

14. ABC ABD ABE ACD ACE ACB ADE ADB ADC AEB AEC AED BCD BCE BCA BDE BDA BDC BEA BEC BED BAC BAD BAE CDE CDA CDB CEA CEB CED CAB CAD CAE CBD CBE CBA DEA DEB DEC DAB DAC DAE DBC DBE DBA DCE DCA DCB EAB EAC EAD EBC EBD EBA ECD ECA ECB EDA EDB EDC. Это облексное упорядочение (см. алгоритм 7.2.1.3R), циклически проходящее по еще неиспользованным буквам.

[Аналогичное упорядочение использовалось при построении всех 120 перестановок пяти букв в каббалистической работе *Sha'ari Tzedeq*, приписываемой Натану бен Саадьяху Харару (Natan ben Sa'adyah Har'ar) из Мессины, Сицилия, жившему в XIII веке; см. *Le Porte della Giustizia* (Milan: Adelphi, 2001).]

15. После j мы помещаем $(n-1)$-сочетания с повторениями множества $\{j, \ldots, m\}$, так что ответ — $\binom{(m+1-j)+(n-1)-1}{n-1} = \binom{m+n-j-1}{n-1}$. [Жан Боррель (Jean Borrel), известный также как Бутеонис (Buteonis), указал этот результат в своей книге *Logistica* (Lyon: 1560) на с. 305–309. Он протабулировал все выбрасывания n костей для $1 \le n \le 4$, а затем использовал суммирование по j, чтобы получить, что есть всего $56+35+20+10+4+1 = 252$ различных результата выпадения $n = 5$ костей и 462 для $n = 6$.]

16. **N1.** [Инициализация.] Установить $r \leftarrow n$, $t \leftarrow 0$ и $a_0 \leftarrow 0$.

N2. [Продвижение.] Пока $r \ge q$, устанавливать $t \leftarrow t + 1$, $a_t \leftarrow q$ и $r \leftarrow r - q$. Затем, если $r > 0$, установить $t \leftarrow t + 1$ и $a_t \leftarrow r$.

N3. [Посещение.] Посетить композицию $a_1 \ldots a_t$.

N4. [Поиск j.] Устанавливать $j \leftarrow t$, $t - 1$, ... до тех пор, пока не будет выполнено условие $a_j \ne 1$. Завершить работу алгоритма, если $j = 0$.

N5. [Уменьшение a_j.] Установить $a_j \leftarrow a_j - 1$, $r \leftarrow t - j + 1$, $t \leftarrow j$; вернуться к шагу N2. ▌

Например, композициями для $n = 7$ и $q = 3$ являются 331, 322, 3211, 313, 3121, 3112, 31111, 232, 2311, 223, 2221, 2212, 22111, 2131, 2122, 21211, 2113, 21121, 21112, 211111, 133, 1321, 1312, 13111, 1231, 1222, 12211, 1213, 12121, 12112, 121111, 1132, 11311, 1123, 11221, 11212, 112111, 11131, 11122, 111211, 11113, 111121, 111112, 1111111.

Сутры 79 и 80 Нараяны (Nārāyaṇa), по сути, описывают приведенную процедуру, но с обращенными строками (133, 223, 1123, ...), поскольку он предпочитал уменьшающийся солексный порядок. [Ранее Сарнгадева (Śārṅgadeva) в *Saṅgītaratnākara* §5.316–375, детально разработал теорию для множества всех композиций (ритмов), которые могут быть образованы базовыми частями $\{1, 2, 4, 6\}$.]

17. Количество V_n посещений равно $F^{(q)}_{n+q-1} = \Theta(\alpha^n_q)$; см. упр. 5.4.2–7. Количество X_n выполнений проверки $a_j = 1$ на шаге N4 удовлетворяет уравнению $X_n = X_{n-1} + \cdots + X_{n-q} + 1$, и мы находим $X_n = V_0 + \cdots + V_n = (qV_n + (q-1)V_{n-1} + \cdots + V_{n-q+1} - 1)/(q-1) = \Theta(V_n)$. Количество Y_n установок $a_t \leftarrow q$ на шаге N2 удовлетворяет тому же рекуррентному соотношению, и мы находим, что $Y_n = X_{n-q}$. И наконец условие $r = 0$ на шаге N2 выполняется V_{n-q} раз.

18. Если воспользоваться римской записью чисел, то этот год записывается как MDCLXVI, а M > D > C > L > X > V > I.

19. В строках 329 и 1022 (всего среди 1022 строф Путеануса данным свойством обладают 139).

20. Если ‘tria’ предшествует ‘lumina’, то имеется $5! \times 2! \times (11, 12, 12, 16)$ способов получить дактиль в $(1, 2, 3, 4)$-й стопе соответственно; если ‘lumina’ предшествует ‘tria’, таких способов $5! \times 2! \times (16, 12, 12, 11)$. Таким образом, общее количество перестановок — 24480. [Лейбниц (Leibniz) рассматривал эту задачу в конце своей *Dissertatio de Arte Combinatoria* и получил ответ, равный 45870; однако его рассуждения переполнены ошибками.]

21. (а) Ясно, что производящая функция $1/\bigl((1 - zu - yu^2)(1 - zv - yv^2)(1 - zw - yw^2)\bigr)$ равна $\sum_{p,q,r,s,t\ge 0} f(p, q, r; s, t)\, u^p v^q w^r z^s y^t$.

(б) Если ‘tibi’ соответствует $\smile\smile$, а ‘Virgo’ соответствует $--$, то общее количество равно $3!\,3!$, умноженному на $\sum_{k=0}^{3}\bigl(f(2k+1, 6-2k, 2; 3, 3) + f(2k, 6-2k, 2; 2, 3)\bigr)$, а именно $36\bigl((7{+}7) + (9{+}5) + (10{+}5) + (14{+}7)\bigr) = 2304$. В противном случае ‘tibi’ соответствует $\smile-$, ‘Virgo’ представляет собой $-\smile$, а общее количество равно $2!\,3!$, умноженному на $\sum_{k=0}^{3}\bigl(f(2k, 5{-}2k, 2; 3, 2) + f(2k, 6{-}2k, 1; 3, 2)\bigr)$, т. е. общее количество гекзаметров составляет $12\bigl((7{+}6) + (5{+}4) + (4{+}4) + (0{+}6)\bigr) = 432$.

(в) Пятая стопа начинается со второго слога ‘c?lo’, ‘dotes’ или ‘Virgo’. Следовательно, дополнительное количество равно $3!\,3!\sum_{k=0}^{2} f(2k, 5{-}2k, 2; 3, 2) = 36(7{+}5{+}4) = 576$, и общее количество составляет $2304 + 432 + 576 = 3312$.

22. Пусть $\alpha \in \{\text{quot}, \text{sunt}, \text{tot}\}$, $\beta \in \{\text{c?lo}, \text{dotes}, \text{Virgo}\}$, $\sigma = \text{sidera}$ и $\tau = \text{tibi}$. Анализ Престе, по сути, эквивалентен анализу Бернулли, но он забыл включить 36 случаев $\alpha\alpha\alpha\tau\beta\beta\sigma\beta$. (В его защиту можно сказать, что эти случаи бесплодны с точки зрения поэзии; Путеанус не нашел им применения.) В издании 1675 года книги Престе опущены также все перестановки, заканчивающиеся на $\tau\beta$.

Уоллис разделил все возможности на 23 типа: $T_1 \cup T_2 \cup \cdots \cup T_{23}$. Он заявил, что типы 6 и 7 дают по 324 строфы; но в действительности $|T_6| = |T_7| = 252$, поскольку его переменная i должна быть 7, а не 9. Кроме того, многие решения подсчитаны им дважды: $|T_3 \cap T_5| = 72$, $|T_2 \cap T_7| = |T_5 \cap T_7| = |T_3 \cap T_6| = |T_6 \cap T_{10}| = 36$ и $|T_{11} \cap T_{12}| = |T_{12} \cap T_{13}| = |T_{14} \cap T_{15}| = 12$. Он пропустил 36 вариантов $\alpha\beta\beta\alpha\sigma\alpha\tau\beta$ (19 из которых использовал Путеанус). Он также пропустил все перестановки из упр. 21, (в); Путеанус использовал 250 из их общего количества 576. В латинском издании книги Уоллиса, опубликованном в 1693 году, был исправлен ряд типографских ошибок, но не ошибки математические.

Уитворт и Хартли пропустили все случаи, когда ‘tibi’ = $\smile-$ (см. упр. 19), возможно, потому, что количество людей, знающих классический гекзаметр, постоянно уменьшается.

[Кстати об ошибках: на самом деле Путеанус опубликовал только 1020 различных перестановок, поскольку строки 592 и 593 в его списке эквивалентны строкам 601 и 602. Однако у него не должно было возникнуть проблем с поиском еще двух гекзаметров — их можно получить, например, заменой ‘tot sunt’ на ‘sunt tot’ в строках 252, 345, 511, 548, 659, 663, 678, 693 и 797.]

23. Читая каждую диаграмму слева направо, так что $12\,|\,345 \leftrightarrow$, получаем следующее.

24. Вот его правило: для $k = 0, 1, \ldots, n-1$ и для каждого сочетания $0 < j_1 < \cdots < j_k < n$ из $n-1$ элементов по k посещаем все разбиения $\{1, \ldots, n-1\} \setminus \{j_1, \ldots, j_k\}$ вместе с блоком $\{j_1, \ldots, j_k, n\}$. Для $n = 5$ получается следующий порядок.

Но, строго говоря, ответ к данному упражнению — "нет", поскольку правило Хонды неполное, пока не указан порядок сочетаний. Он генерировал сочетания в *солексном* порядке (лексикографическом порядке $j_t \ldots j_1$). Лексикографический порядок $j_1 \ldots j_t$ также согласуется с приведенным списком для $n = 4$, но при этом располагается перед . *Источник*: T. Hayashi, *Tôhoku Math. J.* **33** (1931), 332–337.

25. Нет; в (28) отсутствует схема $14|235$ (которая представляет собой зеркальное отражение второй схемы).

26. Пусть a_n — количество неприводимых разбиений $\{1, \ldots, n\}$ и пусть a'_n — количество одновременно неприводимых и полных разбиений. Эти последовательности начинаются следующим образом: $\langle a_1, a_2, \ldots\rangle = \langle 1, 1, 2, 6, 22, 92, 426, \ldots\rangle$, $\langle a'_1, a'_2, \ldots\rangle = \langle 0, 1, 1, 3, 9, 33, 135, \ldots\rangle$; и ответом к данному упражнению является $a'_n - 1$ для $n \geq 2$. Также оказывается, что a_n — количество симметричных полиномов степени n от некоммутирующих переменных. [См. M. C. Wolf, *Duke Math. J.* **2** (1936), 626–637; автором данной работы также протабулированы неприводимые разбиения на k частей.]

Если $A(z) = \sum_n a_n z^n$ и если $B(z) = \sum_n \varpi_n z^n$ представляет собой неэкспоненциальную производящую функцию для чисел Белла, то $A(z)B(z) = B(z) - 1$, следовательно, $A(z) = 1 - 1/B(z)$. Из результата упр. 7.2.1.5–35 вытекает, что $\sum_n a'_n z^n = zA(z)/(1 + z - A(z)) = z(B(z) - 1)/(1 + zB(z))$. К сожалению, $B(z)$ не имеет красивого аналитического вида, хотя и удовлетворяет интересному функциональному соотношению $1 + zB(z) = B(z/(1 + z))$. Заметим, что неприводимые разбиения множества с $n > 1$ соответствуют циклам колеблющейся диаграммы, в которых не имеется трех последовательных λ, равных нулю (см. упр. 7.2.1.5–27).

27. Задача поставлена неоднозначно, так как нет точного определения диаграмм генджико. Давайте потребуем, чтобы все вертикальные линии одного блока имели одинаковую длину; при таком ограничении, например, разбиение $145|236$ не имеет диаграммы генджико с одним пересечением, так как диаграмма не разрешена.

Количество разбиений без пересечений равно C_n (см. упр. 7.2.1.6–26). Для наличия одного пересечения элементы пересекающихся блоков должны находиться в ограниченно растущей строке $x^i y x^j y^k$ либо $x^i y^{j+1} x y^k$, либо $x^i y^j x y^k x^l$, где $i, j, k, l > 0$.

Предположим, что мы рассматриваем шаблон $x^i y x^j y^k$. Имеется $t = i + j + k + 2$ "слотов" между $i + 1 + j + k$ элементами данного шаблона, и количество способов заполнения их непересекающимися разбиениями равно $\sum_{i_1 + \cdots + i_t = n-i-j-k-1} C_{i_1} \ldots C_{i_t}$. Это число можно выразить как

$$[z^{n-i-j-k-1}]\, C(z)^{i+j+k+2} = C_{(n-i-j-k-1)n}$$

согласно 7.2.1.6–(24). Суммирование по k дает $C_{(n-i-j-2)(n+1)}$; последующее суммирование по j и i дает $C_{(n-4)(n+3)}$.

Аналогично два других шаблона дают вклады $C_{(n-5)(n+3)}$ и $C_{(n-5)(n+4)}$. Общее количество разбиений с одним пересечением, таким образом, равно $C_{(n-5)(n+3)} + C_{(n-4)(n+4)}$.

28. Упорядочим делители $cbbaaa$ по количеству их простых сомножителей, а затем — в солексикографическом порядке: $1 \prec a \prec b \prec c \prec aa \prec ba \prec ca \prec bb \prec cb \prec aaa \prec baa \prec caa \prec bba \prec cba \prec cbb \prec baaa \prec caaa \prec bbaa \prec cbaa \prec cbba \prec bbaaa \prec cbaaa \prec cbbaa \prec cbbaaa$. Для каждого такого делителя d в уменьшающемся порядке считаем d первым сомножителем и рекурсивно добавляем все разложения числа $cbbaaa/d$, первый множитель которых $\preceq d$.

Если делители упорядочены лексикографически ($1 < a < aa < aaa < b < ba < \cdots < cbbaa < cbbaaa$), алгоритм Уоллиса становится эквивалентным алгоритму 7.2.1.5M с $(n_1, n_2, n_3) = (1, 2, 3)$. Вероятно, он выбрал этот более сложный порядок делителей в связи с тем, что он более близок к обычному порядку чисел при $a \approx b \approx c$; например, при $(a, b, c) = (7, 11, 13)$ получается точный числовой порядок. Генерируя делители в соответствии с этой более сложной схемой, Уоллис, по сути, генерировал сочетания мультимножества, которые, как отмечалось в разделе 7.2.1.3, эквивалентны ограниченным композициям. [*Источник*: *A Discourse of Combinations* (1685), 126–128, с двумя исправленными типографскими ошибками.]

29. Разложения $edcba$, $edcb \cdot a$, $edca \cdot b$, $\ldots$, $e \cdot d \cdot c \cdot b \cdot a$ соответствуют следующему.

30. Коэффициент равен нулю, кроме случаев $i_1 + 2i_2 + \cdots = n$; тогда он равен $\binom{m}{k} a_0^{m-k} \binom{k}{i_1, i_2, \ldots}$, где $k = i_1 + i_2 + \cdots$. (Рассмотрите $(a_0 z)^m$, умноженное на $(1 + (a_1/a_0)z + (a_2/a_0)z^2 + \cdots)^m$.)

31. Порядок, генерируемый этим алгоритмом, — уменьшающийся лексикографический, обратный разбиению (31), если считать, что для разбиений $a_1 \ldots a_k$ выполняется соотношение $a_1 \geq \cdots \geq a_k$; порядок де Муавра — возрастающий *солексикографический*.

32. $20 \cdot 1 = 7 + 13 \cdot 1 = 2 \cdot 7 + 6 \cdot 1 = 10 + 10 \cdot 1 = 10 + 7 + 3 \cdot 1 = 2 \cdot 10$. В общем случае Бошкович предложил начинать с $n \cdot 1$ и вычислять следующий элемент $a \cdot 10 + b \cdot 7 + c \cdot 1$ следующим образом: если $c \geq 7$, следующий элемент представляет собой $a \cdot 10 + (b + 1) \cdot 7 + (c - 7) \cdot 1$; в противном случае при $c + 7b \geq 10$ следующий элемент представляет собой $(a + 1) \cdot 10 + (c + 7b - 10) \cdot 1$; в противном случае завершаем работу.

“Я могу, — сказал Пуаро совершенно уверенным тоном, — и ошибаться”.

— АГАТА КРИСТИ (AGATHA CHRISTIE),
После похорон (*After the Funeral*) (1953)

— HARPO MARX, *The Cocoanuts* (1925)

— MARCEL MARCEAU, *Baptiste* (1946)

ПРИЛОЖЕНИЕ А

ТАБЛИЦЫ ЗНАЧЕНИЙ НЕКОТОРЫХ КОНСТАНТ

Таблица 1

Величины, часто используемые в стандартных подпрограммах и при анализе компьютерных программ (40 десятичных знаков)

$\sqrt{2} =$	1.41421 35623 73095 04880 16887 24209 69807 85697−
$\sqrt{3} =$	1.73205 08075 68877 29352 74463 41505 87236 69428+
$\sqrt{5} =$	2.23606 79774 99789 69640 91736 68731 27623 54406+
$\sqrt{10} =$	3.16227 76601 68379 33199 88935 44432 71853 37196−
$\sqrt[3]{2} =$	1.25992 10498 94873 16476 72106 07278 22835 05703−
$\sqrt[3]{3} =$	1.44224 95703 07408 38232 16383 10780 10958 83919−
$\sqrt[4]{2} =$	1.18920 71150 02721 06671 74999 70560 47591 52930−
$\ln 2 =$	0.69314 71805 59945 30941 72321 21458 17656 80755+
$\ln 3 =$	1.09861 22886 68109 69139 52452 36922 52570 46475−
$\ln 10 =$	2.30258 50929 94045 68401 79914 54684 36420 76011+
$1/\ln 2 =$	1.44269 50408 88963 40735 99246 81001 89213 74266+
$1/\ln 10 =$	0.43429 44819 03251 82765 11289 18916 60508 22944−
$\pi =$	3.14159 26535 89793 23846 26433 83279 50288 41972−
$1^\circ = \pi/180 =$	0.01745 32925 19943 29576 92369 07684 88612 71344+
$1/\pi =$	0.31830 98861 83790 67153 77675 26745 02872 40689+
$\pi^2 =$	9.86960 44010 89358 61883 44909 99876 15113 53137−
$\sqrt{\pi} = \Gamma(1/2) =$	1.77245 38509 05516 02729 81674 83341 14518 27975+
$\Gamma(1/3) =$	2.67893 85347 07747 63365 56929 40974 67764 41287−
$\Gamma(2/3) =$	1.35411 79394 26400 41694 52880 28154 51378 55193+
$e =$	2.71828 18284 59045 23536 02874 71352 66249 77572+
$1/e =$	0.36787 94411 71442 32159 55237 70161 46086 74458+
$e^2 =$	7.38905 60989 30650 22723 04274 60575 00781 31803+
$\gamma =$	0.57721 56649 01532 86060 65120 90082 40243 10422−
$\ln \pi =$	1.14472 98858 49400 17414 34273 51353 05871 16473−
$\phi =$	1.61803 39887 49894 84820 45868 34365 63811 77203+
$e^\gamma =$	1.78107 24179 90197 98523 65041 03107 17954 91696+
$e^{\pi/4} =$	2.19328 00507 38015 45655 97696 59278 73822 34616+
$\sin 1 =$	0.84147 09848 07896 50665 25023 21630 29899 96226−
$\cos 1 =$	0.54030 23058 68139 71740 09366 07442 97660 37323+
$-\zeta'(2) =$	0.93754 82543 15843 75370 25740 94567 86497 78979−
$\zeta(3) =$	1.20205 69031 59594 28539 97381 61511 44999 07650−
$\ln \phi =$	0.48121 18250 59603 44749 77589 13424 36842 31352−
$1/\ln \phi =$	2.07808 69212 35027 53760 13226 06117 79576 77422−
$-\ln \ln 2 =$	0.36651 29205 81664 32701 24391 58232 66946 94543−

Таблица 2

Величины, часто используемые в стандартных подпрограммах
и при анализе компьютерных программ (40 шестнадцатеричных знаков)

Слева от знака равенства используется десятичная запись.

$0.1 =$	0.1999 9999 9999 9999 9999 9999 9999 9999 9999 999A−
$0.01 =$	0.028F 5C28 F5C2 8F5C 28F5 C28F 5C28 F5C2 8F5C 28F6−
$0.001 =$	0.0041 8937 4BC6 A7EF 9DB2 2D0E 5604 1893 74BC 6A7F−
$0.0001 =$	0.0006 8DB8 BAC7 10CB 295E 9E1B 089A 0275 2546 0AA6+
$0.00001 =$	0.0000 A7C5 AC47 1B47 8423 0FCF 80DC 3372 1D53 CDDD+
$0.000001 =$	0.0000 10C6 F7A0 B5ED 8D36 B4C7 F349 3858 3621 FAFD−
$0.0000001 =$	0.0000 01AD 7F29 ABCA F485 787A 6520 EC08 D236 9919+
$0.00000001 =$	0.0000 002A F31D C461 1873 BF3F 7083 4ACD AE9F 0F4F+
$0.000000001 =$	0.0000 0004 4B82 FA09 B5A5 2CB9 8B40 5447 C4A9 8188−
$0.0000000001 =$	0.0000 0000 6DF3 7F67 5EF6 EADF 5AB9 A207 2D44 268E−
$\sqrt{2} =$	1.6A09 E667 F3BC C908 B2FB 1366 EA95 7D3E 3ADE C175+
$\sqrt{3} =$	1.BB67 AE85 84CA A73B 2574 2D70 78B8 3B89 25D8 34CC+
$\sqrt{5} =$	2.3C6E F372 FE94 F82B E739 80C0 B9DB 9068 2104 4ED8−
$\sqrt{10} =$	3.298B 075B 4B6A 5240 9457 9061 9B37 FD4A B4E0 ABB0−
$\sqrt[3]{2} =$	1.428A 2F98 D728 AE22 3DDA B715 BE25 0D0C 288F 1029+
$\sqrt[3]{3} =$	1.7137 4491 23EF 65CD DE7F 16C5 6E32 67C0 A189 4C2B−
$\sqrt[4]{2} =$	1.306F E0A3 1B71 52DE 8D5A 4630 5C85 EDEC BC27 3436+
$\ln 2 =$	0.B172 17F7 D1CF 79AB C9E3 B398 03F2 F6AF 40F3 4326+
$\ln 3 =$	1.193E A7AA D030 A976 A419 8D55 053B 7CB5 BE14 42DA−
$\ln 10 =$	2.4D76 3776 AAA2 B05B A95B 58AE 0B4C 28A3 8A3F B3E7+
$1/\ln 2 =$	1.7154 7652 B82F E177 7D0F FDA0 D23A 7D11 D6AE F552−
$1/\ln 10 =$	0.6F2D EC54 9B94 38CA 9AAD D557 D699 EE19 1F71 A301+
$\pi =$	3.243F 6A88 85A3 08D3 1319 8A2E 0370 7344 A409 3822+
$1^\circ = \pi/180 =$	0.0477 D1A8 94A7 4E45 7076 2FB3 74A4 2E26 C805 BD78−
$1/\pi =$	0.517C C1B7 2722 0A94 FE13 ABE8 FA9A 6EE0 6DB1 4ACD−
$\pi^2 =$	9.DE9E 64DF 22EF 2D25 6E26 CD98 08C1 AC70 8566 A3FE+
$\sqrt{\pi} = \Gamma(1/2) =$	1.C5BF 891B 4EF6 AA79 C3B0 520D 5DB9 383F E392 1547−
$\Gamma(1/3) =$	2.ADCE EA72 905E 2CEE C8D3 E92C D580 46D8 4B46 A6B3−
$\Gamma(2/3) =$	1.5AA7 7928 C367 8CAB 2F4F EB70 2B26 990A 54F7 EDBC+
$e =$	2.B7E1 5162 8AED 2A6A BF71 5880 9CF4 F3C7 62E7 160F+
$1/e =$	0.5E2D 58D8 B3BC DF1A BADE C782 9054 F90D DA98 05AB−
$e^2 =$	7.6399 2E35 376B 730C E8EE 881A DA2A EEA1 1EB9 EBD9+
$\gamma =$	0.93C4 67E3 7DB0 C7A4 D1BE 3F81 0152 CB56 A1CE CC3B−
$\ln \pi =$	1.250D 048E 7A1B D0BD 5F95 6C6A 843F 4998 5E6D DBF4−
$\phi =$	1.9E37 79B9 7F4A 7C15 F39C C060 5CED C834 1082 276C−
$e^\gamma =$	1.C7F4 5CAB 1356 BF14 A7EF 5AEB 6B9F 6C45 60A9 1932+
$e^{\pi/4} =$	2.317A CD28 E395 4F87 6B04 B8AB AAC8 C708 F1C0 3C4A+
$\sin 1 =$	0.D76A A478 4867 7020 C6E9 E909 C50F 3C32 89E5 1113+
$\cos 1 =$	0.8A51 407D A834 5C91 C246 6D97 6871 BD29 A237 3A89+
$-\zeta'(2) =$	0.F003 2992 B55C 4F28 88E9 BA28 1E4C 405F 8CBE 9FEE+
$\zeta(3) =$	1.33BA 004F 0062 1383 7171 5C59 E690 7F1B 180B 7DB1+
$\ln \phi =$	0.7B30 B2BB 1458 2652 F810 812A 5A31 C083 4C9E B233+
$1/\ln \phi =$	2.13FD 8124 F324 34A2 63C7 5F40 76C7 9883 5224 4685−
$-\ln \ln 2 =$	0.5DD3 CA6F 75AE 7A83 E037 67D6 6E33 2DBC 09DF AA82−

Далее приведено несколько интересных менее распространенных констант, возникающих в связи с анализом, выполняемым в данной книге. Эти константы вычислены с точностью до 40 десятичных цифр в 7.1.4–(90) и 7.2.1.5–(34) и в ответе к упр. 7.1.4–191.

Таблица 3

Значения гармонических чисел, чисел Бернулли и чисел Фибоначчи для малых значений n

n	H_n	B_n	F_n	n
0	0	1	0	0
1	1	−1/2	1	1
2	3/2	1/6	1	2
3	11/6	0	2	3
4	25/12	−1/30	3	4
5	137/60	0	5	5
6	49/20	1/42	8	6
7	363/140	0	13	7
8	761/280	−1/30	21	8
9	7129/2520	0	34	9
10	7381/2520	5/66	55	10
11	83711/27720	0	89	11
12	86021/27720	−691/2730	144	12
13	1145993/360360	0	233	13
14	1171733/360360	7/6	377	14
15	1195757/360360	0	610	15
16	2436559/720720	−3617/510	987	16
17	42142223/12252240	0	1597	17
18	14274301/4084080	43867/798	2584	18
19	275295799/77597520	0	4181	19
20	55835135/15519504	−174611/330	6765	20
21	18858053/5173168	0	10946	21
22	19093197/5173168	854513/138	17711	22
23	444316699/118982864	0	28657	23
24	1347822955/356948592	−236364091/2730	46368	24
25	34052522467/8923714800	0	75025	25
26	34395742267/8923714800	8553103/6	121393	26
27	312536252003/80313433200	0	196418	27
28	315404588903/80313433200	−23749461029/870	317811	28
29	9227046511387/2329089562800	0	514229	29
30	9304682830147/2329089562800	8615841276005/14322	832040	30

Положим $H_x = \sum_{n\geq 1}\left(\frac{1}{n} - \frac{1}{n+x}\right)$ для произвольного x. Тогда

$$\begin{aligned}
H_{1/2} &= 2 - 2\ln 2,\\
H_{1/3} &= 3 - \tfrac{1}{2}\pi/\sqrt{3} - \tfrac{3}{2}\ln 3,\\
H_{2/3} &= \tfrac{3}{2} + \tfrac{1}{2}\pi/\sqrt{3} - \tfrac{3}{2}\ln 3,\\
H_{1/4} &= 4 - \tfrac{1}{2}\pi - 3\ln 2,\\
H_{3/4} &= \tfrac{4}{3} + \tfrac{1}{2}\pi - 3\ln 2,\\
H_{1/5} &= 5 - \tfrac{1}{2}\pi\phi^{3/2}5^{-1/4} - \tfrac{5}{4}\ln 5 - \tfrac{1}{2}\sqrt{5}\ln\phi,\\
H_{2/5} &= \tfrac{5}{2} - \tfrac{1}{2}\pi\phi^{-3/2}5^{-1/4} - \tfrac{5}{4}\ln 5 + \tfrac{1}{2}\sqrt{5}\ln\phi,\\
H_{3/5} &= \tfrac{5}{3} + \tfrac{1}{2}\pi\phi^{-3/2}5^{-1/4} - \tfrac{5}{4}\ln 5 + \tfrac{1}{2}\sqrt{5}\ln\phi,\\
H_{4/5} &= \tfrac{5}{4} + \tfrac{1}{2}\pi\phi^{3/2}5^{-1/4} - \tfrac{5}{4}\ln 5 - \tfrac{1}{2}\sqrt{5}\ln\phi,\\
H_{1/6} &= 6 - \tfrac{1}{2}\pi\sqrt{3} - 2\ln 2 - \tfrac{3}{2}\ln 3,\\
H_{5/6} &= \tfrac{6}{5} + \tfrac{1}{2}\pi\sqrt{3} - 2\ln 2 - \tfrac{3}{2}\ln 3,
\end{aligned}$$

и в общем случае, когда $0 < p < q$ (см. упр. 1.2.9–19),

$$H_{p/q} = \frac{q}{p} - \frac{\pi}{2}\cot\frac{p}{q}\pi - \ln 2q + 2\sum_{1\leq n<q/2}\cos\frac{2pn}{q}\pi\cdot\ln\sin\frac{n}{q}\pi.$$

Читатель, если тебе когда-нибудь придется заниматься вычислениями, предупреждаю — ни в коем случае не бери на эту работу бухгалтера, каким бы честным и быстро работающим с числами он ни был. Ваш компьютер должен работать с определенным количеством цифр, независимо от того, располагаются ли они до десятичной точки или после. Бухгалтер же работает с центами, и будет работать с центами, пока ад не замерзнет. Что бы ни вычислял наш бухгалтер, на всех этапах он сохранял ровно две цифры после запятой... Этого требовала от него его совесть — быть точным до последнего цента; и он был просто не в состоянии понять, что физические величины не измеряются центами, что применяется скользящая шкала значений, в которой центы одной задачи могут быть долларами другой.

— НОРБЕРТ ВИНЕР (NORBERT WIENER),
Я — математик (*I am a Mathematician*) (1956)

ПРИЛОЖЕНИЕ Б

ОСНОВНЫЕ ОБОЗНАЧЕНИЯ

В приведенных далее формулах не поясняемые буквы имеют следующий смысл.

j, k	Целочисленное арифметическое выражение
m, n	Неотрицательное целочисленное арифметическое выражение
p, q	Бинарное арифметическое выражение (0 или 1)
x, y	Действительное арифметическое выражение
z	Комплексное арифметическое выражение
f	Целочисленная, действительная или комплексная функция
G, H	Граф
S, T	Множество или мультимножество
$\mathcal{F}, \mathcal{G}$	Семейство множеств
u, v	Вершина графа
α, β	Строка символов

Место определения обозначения представляет собой либо номер страницы в настоящем томе, либо номер раздела в предыдущем томе (указывается символом §).

Обозначение	**Значение**	**Определено**
$V \leftarrow E$	Присваивает переменной V значение выражения E	§1.1
$U \leftrightarrow V$	Обмен значениями между переменными U и V	§1.1
A_n или $A[n]$	n-й элемент линейного массива A	§1.1
A_{mn} или $A[m, n]$	Элемент в строке m и столбце n прямоугольного массива A	§1.1
$(R?\ a\colon b)$	Условное выражение: обозначает a, если отношение R истинно, b, если R ложно	125
$[R]$	Характеристическая функция отношения R: $(R?\ 1\colon 0)$	§1.2.3
δ_{jk}	Символ Кронекера: $[j = k]$	§1.2.3
$[z^n]\, f(z)$	Коэффициент при z^n в степенном ряду $f(z)$	§1.2.9
$z_1 + z_2 + \cdots + z_n$	Сумма n чисел (даже при n, равном 0 или 1)	§1.2.3
$a_1 a_2 \ldots a_n$	Произведение или строка или вектор из n элементов	
$(x_1, \ldots, x_n)$	Вектор из n элементов	
$\langle x_1 x_2 \ldots x_{2k-1} \rangle$	Медиана (среднее значение после сортировки)	101

Обозначение	Значение	Определено
$\sum_{R(k)} f(k)$	Сумма всех $f(k)$, таких, что отношение $R(k)$ истинно	§1.2.3
$\prod_{R(k)} f(k)$	Произведение всех $f(k)$, таких, что отношение $R(k)$ истинно	§1.2.3
$\min_{R(k)} f(k)$	Минимум среди всех $f(k)$, таких, что отношение $R(k)$ истинно	§1.2.3
$\max_{R(k)} f(k)$	Максимум среди всех $f(k)$, таких, что отношение $R(k)$ истинно	§1.2.3
$\bigcup_{R(k)} S(k)$	Объединение всех $S(k)$, таких, что отношение $R(k)$ истинно	
$\sum_{k=a}^{b} f(k)$	Сокращение для $\sum_{a\le k\le b} f(k)$	§1.2.3
$\{a \mid R(a)\}$	Множество всех a, таких, что отношение $R(a)$ истинно	
$\sum\{f(k) \mid R(k)\}$	Другой способ записи $\sum_{R(k)} f(k)$	
$\{a_1, a_2, \ldots, a_n\}$	Множество или мультимножество $\{a_k \mid 1\le k\le n\}$	
$[x\,..\,y]$	Замкнутый интервал: $\{a \mid x \le a \le y\}$	§1.2.2
$(x\,..\,y)$	Открытый интервал: $\{a \mid x < a < y\}$	§1.2.2
$[x\,..\,y)$	Полуоткрытый интервал: $\{a \mid x \le a < y\}$	§1.2.2
$(x\,..\,y]$	Полузамкнутый интервал: $\{a \mid x < a \le y\}$	§1.2.2
$\lvert S\rvert$	Число элементов множества S	
$\lvert f\rvert$	Число решений (когда f — булева функция): $\sum_x f(x)$	247
$\lvert x\rvert$	Абсолютная величина x: $(x \ge 0?\ x\colon -x)$	
$\lvert z\rvert$	Абсолютная величина z: $\sqrt{z\bar{z}}$	§1.2.2
$\lvert\alpha\rvert$	Длина α: m, если $\alpha = a_1a_2\ldots a_m$	
$\lfloor x\rfloor$	Пол x, наибольшее целое число, не превосходящее x: $\max_{k\le x} k$	§1.2.4
$\lceil x\rceil$	Потолок x, наименьшее целое число, не меньшее x: $\min_{k\ge x} k$	§1.2.4
$x \bmod y$	Функция mod: $\bigl(y = 0?\ x\colon x - y\lfloor x/y\rfloor\bigr)$	§1.2.4
$\{x\}$	Дробная часть (в контексте действительного значения, а не множества): $x \bmod 1$	§1.2.11.2
$x \equiv x'$ (по модулю y)	Отношение конгруэнтности: $x \bmod y = x' \bmod y$	§1.2.4
$j\backslash k$	j делит k: $k \bmod j = 0$ и $j > 0$	§1.2.4
$S \setminus T$	Разность множеств: $\{s \mid s$ в S и s не в $T\}$	
$S \setminus t$	Сокращенная запись $S \setminus \{t\}$	
$G \setminus v$	G с удаленной вершиной v	32
$G \setminus e$	G с удаленным ребром e	32
$G \,/\, e$	G с ребром e, сжатым в точку	521
$S \cup t$	Сокращенная запись $S \cup \{t\}$	
$S \uplus T$	Сумма мультимножеств; т. е. $\{a, b\} \uplus \{a, c\} = \{a, a, b, c\}$	§4.6.3
$\gcd(j, k)$	Наибольший общий делитель: $(j{=}k{=}0?\ 0\colon \max_{d\backslash j, d\backslash k} d)$	§1.1

Обозначение	Значение	Определено
$j \perp k$	j взаимно простое с k: $\gcd(j,k)=1$	§1.2.4
A^T	Транспонированный прямоугольный массив A: $A^T[j,k]=A[k,j]$	
α^R	Левоправое обращение строки α	
α^T	Разбиение, сопряженное к α	449
x^y	x в степени y (при $x>0$): $e^{y\ln x}$	§1.2.2
x^k	x в степени k: $\bigl(k\ge 0?\ \prod_{j=0}^{k-1} x\colon 1/x^{-k}\bigr)$	§1.2.2
x^-	Обращение x: x^{-1}	§1.3.3
$x^{\overline{k}}$	$\Gamma(x+k)/\Gamma(k)=\bigl(k\ge 0?\ \prod_{j=0}^{k-1}(x+j)\colon 1/(x+k)^{\overline{-k}}\bigr)$	§1.2.5
$x^{\underline{k}}$	$x!/(x-k)!=\bigl(k\ge 0?\ \prod_{j=0}^{k-1}(x-j)\colon 1/(x-k)^{\underline{-k}}\bigr)$	§1.2.5
$n!$	Факториал n: $\Gamma(n+1)=n^{\underline{n}}$	§1.2.5
$\binom{x}{k}$	Биномиальный коэффициент: $(k<0?\ 0\colon x^{\underline{k}}/k!)$	§1.2.6
$\binom{n}{n_1,\ldots,n_m}$	Мультиномиальный коэффициент $(n=n_1+\cdots+n_m)$	§1.2.6
$\left[{n \atop m}\right]$	Число Стирлинга первого рода (количество циклов): $\sum_{0<k_1<\cdots<k_{n-m}<n} k_1\ldots k_{n-m}$	§1.2.6
$\left\{{n \atop m}\right\}$	Число Стирлинга второго рода (количество подмножеств): $\sum_{1\le k_1\le\cdots\le k_{n-m}\le m} k_1\ldots k_{n-m}$	§1.2.6
$\left\langle{n \atop m}\right\rangle$	Число Эйлера: $\sum_{k=0}^{m}(-1)^k\binom{n+1}{k}(m+1-k)^n$	§5.1.3
$\left\lvert{n \atop m}\right\rvert$	Количество разбиений n на m частей: $\sum_{1\le k_1\le\cdots\le k_m}[k_1+\cdots+k_m=n]$	454
$(\ldots a_1a_0.a_{-1}\ldots)_b$	Представление числа в позиционной системе счисления с основанием b: $\sum_k a_kb^k$	§4.1
$\Re z$	Действительная часть z	§1.2.2
$\Im z$	Мнимая часть z	§1.2.2
$\overline{z}$	Комплексно сопряженное: $\Re z - i\,\Im z$	§1.2.2
$\sim p$ или $\overline{p}$	Дополнение: $1-p$	74
$\sim x$ или $\overline{x}$	Побитовое дополнение	167
$p\wedge q$	Логическая конъюнкция (И): pq	74
$x\wedge y$	Минимум: $\min\{x,y\}$	88
$x\,\&\,y$	Побитовое И	165
$p\vee q$	Логическая дизъюнкция (ИЛИ): $\overline{\overline{p}\,\overline{q}}$	74
$x\vee y$	Максимум: $\max\{x,y\}$	88
$x\mid y$	Побитовое ИЛИ	165
$p\oplus q$	Логическая исключающая дизъюнкция (ЛИБО): $(p+q)\bmod 2$	74
$x\oplus y$	Побитовое ЛИБО	165
$x\dot{-}y$	Вычитание с насыщением, x монус y: $\max\{0,x-y\}$	§1.3.1′
$x\ll k$	Побитовый сдвиг влево: $\lfloor 2^kx\rfloor$	167
$x\gg k$	Побитовый сдвиг вправо: $x\ll(-k)$	167
$x\ddagger y$	“Функция застежки” для чередования битов, x zip y	180

Обозначение	Значение	Определено
$\log_b x$	Логарифм числа x по основанию b (определен при $x>0$, $b>0$ и $b\neq 1$): y такое, что $x=b^y$	§1.2.2
$\ln x$	Натуральный логарифм: $\log_e x$	§1.2.2
$\lg x$	Логарифм по основанию 2: $\log_2 x$	§1.2.2
λn	Бинарный логарифм (при $n>0$): $\lfloor \lg n \rfloor$	174
$\exp x$	Показательная функция от x: $e^x=\sum_{k=0}^{\infty} x^k/k!$	§1.2.9
ρn	Линеечная функция (при $n>0$): $\max_{2^m\backslash n} m$	172
νn	Контрольная сумма (при $n>0$): $\sum_{k\geq 0}\bigl((n\gg k)\&1\bigr)$	175
$\langle X_n\rangle$	Бесконечная последовательность X_0, X_1, X_2, ... (здесь буква n является частью обозначения)	§1.2.9
$f'(x)$	Производная от f по x	§1.2.9
$f''(x)$	Вторая производная от f по x	§1.2.10
$H_n^{(x)}$	Гармоническое число порядка x: $\sum_{k=1}^{n} 1/k^x$	§1.2.7
H_n	Гармоническое число: $H_n^{(1)}$	§1.2.7
F_n	Число Фибоначчи: $(n\leq 1?\ n\colon F_{n-1}+F_{n-2})$	§1.2.8
B_n	Число Бернулли: $n!\,[z^n]\,z/(e^z-1)$	§1.2.11.2
$\det(A)$	Определитель квадратной матрицы A	§1.2.3
$\operatorname{sign}(x)$	Знак x: $[x>0]-[x<0]$	
$\zeta(x)$	Дзета-функция: $\lim_{n\to\infty} H_n^{(x)}$ (при $x>1$)	§1.2.7
$\Gamma(x)$	Гамма-функция: $(x-1)!=\gamma(x,\infty)$	§1.2.5
$\gamma(x,y)$	Неполная гамма-функция: $\int_0^y e^{-t}t^{x-1}\,dt$	§1.2.11.3
γ	Константа Эйлера: $-\Gamma'(1)=\lim_{n\to\infty}(H_n-\ln n)$	§1.2.7
e	Основание натуральных алгоритмов: $\sum_{n\geq 0} 1/n!$	§1.2.2
π	Отношение длины окружности к ее диаметру: $4\sum_{n\geq 0}(-1)^n/(2n+1)$	§1.2.2
∞	Бесконечность: больше любого числа	
Λ	Нулевая связь (указатель на несуществующий адрес)	§2.1
$\emptyset$	Пустое множество (множество без элементов)	
ϵ	Пустая строка (строка нулевой длины)	
ϵ	Единичное семейство: $\{\emptyset\}$	320
ϕ	Золотое сечение: $\bigl(1+\sqrt{5}\bigr)/2$	§1.2.8
$\varphi(n)$	Функция Эйлера: $\sum_{k=0}^{n-1}[k\perp n]$	§1.2.4
$x\approx y$	x приближенно равно y	§1.2.5
$G\cong H$	G изоморфен H	33
$O\bigl(f(n)\bigr)$	О большое от $f(n)$ при $n\to\infty$	§1.2.11.1
$O\bigl(f(z)\bigr)$	О большое от $f(z)$ при $z\to 0$	§1.2.11.1
$\Omega\bigl(f(n)\bigr)$	Омега большое от $f(n)$ при $n\to\infty$	§1.2.11.1
$\Theta\bigl(f(n)\bigr)$	Тэта большое от $f(n)$ при $n\to\infty$	§1.2.11.1

Обозначение	Значение	Определено
$\overline{G}$	Дополнение графа (или однородного гиперграфа) G	46
$G \mid U$	G, ограниченный вершинами множества U	32
$u \text{—} v$	u, смежная с v	32
$u \not\text{—} v$	u, не смежная с v	32
$u \longrightarrow v$	Имеется дуга от u до v	38
$u \longrightarrow^* v$	Транзитивное замыкание: v достижима из u	193
$d(u, v)$	Расстояние от u до v	35
$G \cup H$	Объединение G и H	47
$G \oplus H$	Прямая сумма (соприкосновение) G и H	47
$G \text{—} H$	Соединение G и H	47
$G \longrightarrow H$	Ориентированное соединение G и H	47
$G \square H$	Декартово произведение G и H	48
$G \otimes H$	Прямое произведение (конъюнкция) G и H	49
$G \boxtimes H$	Сильное произведение G и H	49
$G \triangle H$	Нечетное произведение G и H	49
$G \circ H$	Лексикографическое произведение (композиция) G и H	49
e_j	Элементарное семейство: $\{\{j\}\}$	320
$\mathcal{F} \cup \mathcal{G}$	Объединение семейств: $\{S \mid S \in \mathcal{F}$ или $S \in \mathcal{G}\}$	320
$\mathcal{F} \cap \mathcal{G}$	Пересечение семейств: $\{S \mid S \in \mathcal{F}$ и $S \in \mathcal{G}\}$	320
$\mathcal{F} \setminus \mathcal{G}$	Разность семейств: $\{S \mid S \in \mathcal{F}$ и $S \notin \mathcal{G}\}$	320
$\mathcal{F} \oplus \mathcal{G}$	Симметричная разность семейств: $(\mathcal{F} \setminus \mathcal{G}) \cup (\mathcal{G} \setminus \mathcal{F})$	320
$\mathcal{F} \sqcup \mathcal{G}$	Соединение семейств: $\{S \cup T \mid S \in \mathcal{F}, T \in \mathcal{G}\}$	320
$\mathcal{F} \sqcap \mathcal{G}$	Встреча семейств: $\{S \cap T \mid S \in \mathcal{F}, T \in \mathcal{G}\}$	320
$\mathcal{F} \boxplus \mathcal{G}$	Дельта семейств: $\{S \oplus T \mid S \in \mathcal{F}, T \in \mathcal{G}\}$	320
$\mathcal{F}/\mathcal{G}$	Частное (кофактор) семейств	321
$\mathcal{F} \bmod \mathcal{G}$	Остаток семейств: $\mathcal{F} \setminus (\mathcal{G} \sqcup (\mathcal{F}/\mathcal{G}))$	321
$\mathcal{F} \,\S\, k$	Симметризованное семейство, если $\mathcal{F} = e_{j_1} \cup e_{j_2} \cup \cdots \cup e_{j_n}$	321
$\mathcal{F}^{\uparrow}$	Максимальные элементы $\mathcal{F}$: $\{$из $S \in \mathcal{F} \mid T \in \mathcal{F}$ и $S \subseteq T$ вытекает $S = T\}$	324
$\mathcal{F}^{\uparrow}$	Минимальные элементы $\mathcal{F}$: $\{$из $S \in \mathcal{F} \mid T \in \mathcal{F}$ и $S \supseteq T$ вытекает $S = T\}$	324
$\mathcal{F} \nearrow \mathcal{G}$	Не подмножества: $\{$из $S \in \mathcal{F} \mid T \in \mathcal{G}$ вытекает $S \not\subseteq T\}$	324
$\mathcal{F} \searrow \mathcal{G}$	Не надмножества: $\{$из $S \in \mathcal{F} \mid T \in \mathcal{G}$ вытекает $S \not\supseteq T\}$	324
$\mathcal{F} \swarrow \mathcal{G}$	Подмножества: $\{$из $S \in \mathcal{F} \mid T \in \mathcal{G}$ вытекает $S \subseteq T\} = \mathcal{F} \setminus (\mathcal{F} \nearrow \mathcal{G})$	745
$\mathcal{F} \nwarrow \mathcal{G}$	Надмножества: $\{$из $S \in \mathcal{F} \mid T \in \mathcal{G}$ вытекает $S \supseteq T\} = \mathcal{F} \setminus (\mathcal{F} \searrow \mathcal{G})$	745
▮	Конец алгоритма, программы или доказательства	§1.1

Обозначение	Значение	Определено
$X \cdot Y$	Скалярное произведение векторов: $x_1y_1 + x_2y_2 + \cdots + x_ny_n$, если $X = x_1x_2\ldots x_n$ и $Y = y_1y_2\ldots y_n$	31
$X \subseteq Y$	Содержание векторов: $x_k \le y_k$ для $1 \le k \le n$, если $X = x_1x_2\ldots x_n$ и $Y = y_1y_2\ldots y_n$	167
$\alpha \diamond \beta$	Смесь таблиц истинности	259

Чтобы избежать утомительно частого повторения слов "равно тому-то", я буду ставить (как я часто делаю в своей работе) пару параллельных линий одинаковой длины (т. е. =), поскольку никакие две вещи в мире не могут быть еще более равными...

— РОБЕРТ РЕКОРДЕ (ROBERT RECORDE), *The Whetstone of Witte* (1557)

Профессор Лежандр в труде, который мы будем часто цитировать, использует один и тот же знак и для равенства, и для конгруэнтности. Чтобы избежать неоднозначности, мы вынуждены обозначать их по-разному.

— К. Ф. ГАУСС (C. F. GAUSS), *Disquisitiones Arithmetic?* (1801)

Кое-кто утверждает, что каждое уравнение, включенное мною в книгу, вдвое снижает ее продажи.

— СТИВЕН ХОКИНГ (STEPHEN HAWKING), *Краткая история времени* (*A Brief History of Time*) (1987)

ПРИЛОЖЕНИЕ B

СПИСОК АЛГОРИТМОВ И ТЕОРЕМ

Алгоритм 7B, 43.
Теорема 7B, 37.
Алгоритм 7H, 51.
Следствие 7H, 52.
Лемма 7M, 51.
Алгоритм 7.1.1B, 603.
Алгоритм 7.1.1C, 84.
Следствие 7.1.1C, 94.
Теорема 7.1.1C, 94.
Алгоритм 7.1.1E, 110.
Следствие 7.1.1F, 100.
Теорема 7.1.1F, 99.
Теорема 7.1.1G, 91.
Алгоритм 7.1.1H, 95.
Теорема 7.1.1H, 82.
Подпрограмма 7.1.1I, 95.
Алгоритм 7.1.1K, 619.
Теорема 7.1.1K, 87.
Лемма 7.1.1M, 91.
Алгоритм 7.1.1P, 603.
Теорема 7.1.1P, 89.
Следствие 7.1.1Q, 80.
Теорема 7.1.1Q, 80.
Теорема 7.1.1S, 98.
Теорема 7.1.1T, 103.
Алгоритм 7.1.1X, 108.
Алгоритм 7.1.2L, 128.
Теорема 7.1.2L, 141.
Алгоритм 7.1.2S, 158.
Теорема 7.1.2S, 139.
Алгоритм 7.1.2T, 159.
Алгоритм 7.1.2U, 628.
Лемма 7.1.3A, 190.
Алгоритм 7.1.3B, 188.
Лемма 7.1.3B, 191.
Алгоритм 7.1.3C, 680.
Следствие 7.1.3I, 190.
Алгоритм 7.1.3K, 670.
Следствие 7.1.3L, 191.
Алгоритм 7.1.3M, 673.
Следствие 7.1.3M, 192.
Алгоритм 7.1.3N, 690.
Теорема 7.1.3P′, 193.
Теорема 7.1.3P, 191.
Алгоритм 7.1.3Q, 673.
Теорема 7.1.3R′, 192.
Алгоритм 7.1.3R, 193.
Программа 7.1.3R, 194.
Теорема 7.1.3R, 191.
Алгоритм 7.1.3S, 676.
Алгоритм 7.1.3T, 215.
Алгоритм 7.1.3V, 202.
Следствие 7.1.3W, 190.
Алгоритм 7.1.4A, 697.
Теорема 7.1.4A, 293.
Алгоритм 7.1.4B, 249.
Теорема 7.1.4B, 280.
Алгоритм 7.1.4C′, 697.
Алгоритм 7.1.4C, 247.
Алгоритм 7.1.4H, 713.
Алгоритм 7.1.4I, 694.
Теорема 7.1.4J$^+$, 286.
Теорема 7.1.4J$^-$, 286.
Алгоритм 7.1.4J, 287.
Алгоритм 7.1.4K, 698.
Теорема 7.1.4K, 281.
Теорема 7.1.4M, 255.
Алгоритм 7.1.4N, 709.
Алгоритм 7.1.4R, 257.
Алгоритм 7.1.4S, 264.
Следствие 7.1.4S, 301.
Теорема 7.1.4S, 301.
Алгоритм 7.1.4U, 269.
Теорема 7.1.4U, 277.
Теорема 7.1.4W, 291.
Теорема 7.1.4X, 292.
Следствие 7.1.4Y, 293.
Теорема 7.1.4Y, 293.
Алгоритм 7.2.1.1A, 352.
Алгоритм 7.2.1.1B, 757.
Следствие 7.2.1.1B, 343.
Алгоритм 7.2.1.1C, 756.
Алгоритм 7.2.1.1D, 353.
Теорема 7.2.1.1D, 342.
Алгоритм 7.2.1.1E, 778.
Алгоритм 7.2.1.1F, 356.
Алгоритм 7.2.1.1G, 334.
Алгоритм 7.2.1.1H, 348.
Алгоритм 7.2.1.1J, 757.
Алгоритм 7.2.1.1K, 350.
Алгоритм 7.2.1.1L, 338.
Алгоритм 7.2.1.1M, 330.
Алгоритм 7.2.1.1N, 755.
Алгоритм 7.2.1.1P, 760.
Определение 7.2.1.1P, 355.
Теорема 7.2.1.1P, 355.
Определение 7.2.1.1Q, 356.
Теорема 7.2.1.1Q, 356.
Алгоритм 7.2.1.1R, 353.
Алгоритм 7.2.1.1S, 351.
Алгоритм 7.2.1.1T, 761.
Алгоритм 7.2.1.1U, 773.
Алгоритм 7.2.1.1V, 757.
Алгоритм 7.2.1.1X, 768.
Алгоритм 7.2.1.1Z, 756.
Алгоритм 7.2.1.2C, 388.
Алгоритм 7.2.1.2E′, 401.
Алгоритм 7.2.1.2E, 389.
Алгоритм 7.2.1.2G, 380.
Алгоритм 7.2.1.2H, 386.
Алгоритм 7.2.1.2L′, 781.
Алгоритм 7.2.1.2L, 369.
Алгоритм 7.2.1.2M, 781.
Алгоритм 7.2.1.2P′, 783.

Algorithm 7.2.1.2P, 372.
Алгоритм 7.2.1.2Q, 784.
Алгоритм 7.2.1.2R, 783.
Теорема 7.2.1.2R, 390.
Лемма 7.2.1.2S, 378.
Алгоритм 7.2.1.2T, 374.
Алгоритм 7.2.1.2V, 395.
Алгоритм 7.2.1.2X, 385.
Алгоритм 7.2.1.2Y, 799.
Алгоритм 7.2.1.2Z, 403.
Алгоритм 7.2.1.3A, 805.
Алгоритм 7.2.1.3C, 420.
Следствие 7.2.1.3C, 433.
Алгоритм 7.2.1.3CB, 813.
Алгоритм 7.2.1.3CC, 813.
Алгоритм 7.2.1.3F, 414.
Теорема 7.2.1.3K, 427.
Алгоритм 7.2.1.3L, 411.
Теорема 7.2.1.3L, 413.
Теорема 7.2.1.3M, 427.
Алгоритм 7.2.1.3N, 804.
Теорема 7.2.1.3N, 418.
Теорема 7.2.1.3P, 423.
Алгоритм 7.2.1.3Q, 816.
Алгоритм 7.2.1.3R, 416.
Алгоритм 7.2.1.3S, 805.
Лемма 7.2.1.3S, 431.
Алгоритм 7.2.1.3T, 412.
Алгоритм 7.2.1.3V, 806.
Алгоритм 7.2.1.3W, 815.
Теорема 7.2.1.3W, 431.
Алгоритм 7.2.1.4A, 831.
Алгоритм 7.2.1.4B, 831.
Алгоритм 7.2.1.4C, 828.
Теорема 7.2.1.4C, 458.
Теорема 7.2.1.4D, 453.
Теорема 7.2.1.4E, 455.
Алгоритм 7.2.1.4H, 447.
Теорема 7.2.1.4H, 459.
Алгоритм 7.2.1.4K, 840.
Алгоритм 7.2.1.4M, 838.
Алгоритм 7.2.1.4N, 467.
Алгоритм 7.2.1.4P, 446.
Теорема 7.2.1.4S, 462.
Алгоритм 7.2.1.5E, 846.
Алгоритм 7.2.1.5H, 472.
Алгоритм 7.2.1.5L, 843.
Алгоритм 7.2.1.5M, 485.
Алгоритм 7.2.1.6B, 502.
Алгоритм 7.2.1.6H, 518.
Алгоритм 7.2.1.6L, 506.
Алгоритм 7.2.1.6N, 504.
Алгоритм 7.2.1.6O, 520.
Алгоритм 7.2.1.6P, 499.
Алгоритм 7.2.1.6Q, 530.
Алгоритм 7.2.1.6R, 515.
Алгоритм 7.2.1.6S, 523.
Теорема 7.2.1.6S, 529.
Алгоритм 7.2.1.6U, 510.
Алгоритм 7.2.1.6W, 511.
Алгоритм 7.2.1.6Z, 864.
Алгоритм 7.2.1.7N, 899.

[Обратный список] предоставляет дублированную, избыточную информацию, ускоряющую выборку вторичного ключа.

— ДОНАЛЬД Э. КНУТ (DONALD E. KNUTH),
Сортировка и поиск (*Sorting and Searching*) (1973)

ПРИЛОЖЕНИЕ Г

УКАЗАТЕЛЬ КОМБИНАТОРНЫХ ЗАДАЧ

В данном приложении кратко описаны основные задачи, рассмотренные в данной книге. Одни из них могут быть эффективно решены, в то время как другие представляются очень сложными в общем случае, хотя частные случаи могут оказаться легкими. Никакие указания о сложности задач здесь не приводятся.

Комбинаторные задачи подобны хамелеонам и способны принимать множество обличий. Например, некоторые свойства графов и гиперграфов аналогичны другим свойствам 0–1 матриц; а матрица из нулей и единиц размером $m \times n$ может рассматриваться как булева функция от mn булевых переменных, где 0 представляет TRUE, а 1 представляет FALSE. Каждая задача также имеет много разновидностей: иногда нас интересует только существование решения для определенных ограничений, но обычно нам требуется получить по крайней мере одно явное решение или попытаться подсчитать количество решений, или посетить их все. Часто нам требуется решение, оптимальное в том или ином смысле.

В приведенном списке — который более полезен, чем сколь-нибудь полон — каждая задача представлена в более-менее формализованном виде, как задача "поиска" некоторого целевого объекта. За этой характеристикой следуют неформальный пересказ (в скобках и кавычках) и, возможно, некоторые комментарии.

Любая задача, изложенная в терминах ориентированных графов, автоматически применима к неориентированным графам, если только ориентированный граф не обязан быть ациклическим, поскольку неориентированное ребро u — v эквивалентно двум ориентированным дугам $u \longrightarrow v$ и $v \longrightarrow u$.

• Выполнимость. Для заданной булевой функции f от n булевых переменных найти булевы значения $x_1, \ldots, x_n$, такие, что $f(x_1, \ldots, x_n) = 1$. ("Если возможно, покажите, что f может быть истинна.")

• kSAT. Задача выполнимости, когда f представляет собой конъюнкцию дизъюнктов, где каждый дизъюнкт представляет собой дизъюнкцию не более k литералов x_j или $\bar{x}_j$. ("Могут ли все дизъюнкты быть истинными?") Наиболее важными являются случаи 2SAT и 3SAT. Другой важный частный случай — когда f является конъюнкцией *дизъюнктов Хорна*, каждый из которых имеет не более одного инвертированного литерала $\bar{x}_j$.

• Булева цепочка. Для заданных одной или нескольких булевых функций от n булевых значений $x_1, \ldots, x_n$ найти $x_{n+1}, \ldots, x_N$, такие, что каждое x_k для $n < k \leq N$ является булевой функцией от x_i и x_j для некоторых $i < k$ и $j < k$, и такие, что каждая из данных функций либо представляет собой константу, либо равна x_l для некоторого $l \leq N$. ("Напишите простую линейную программу для вычисления заданного множества функций с совместным использованием промежуточных значений.") ("Разработайте схему для вычисления заданного набора выходов для входов 0, 1, $x_1, \ldots, x_n$, используя логические элементы с двумя входами и неограниченным ветвлением на выходе.") Обычно цель заключается в минимизации N.

• Широкословная цепочка. Аналогична булевой цепочке, но с использованием битовых и/или арифметических операций над целыми числами по модулю 2^d вместо булевых операций над булевыми значениями; заданное значение d может быть произвольно большим. ("Работа над несколькими связанными задачами одновременно.")

• Булево программирование. Для заданной булевой функции f от n булевых переменных вместе с заданными весами $w_1, \ldots, w_n$ найти булевы значения $x_1, \ldots, x_n$, такие, что $f(x_1, \ldots, x_n) = 1$ и $w_1x_1 + \cdots + w_nx_n$ велико, насколько это возможно. ("Как выполнить f с максимальным выигрышем?")

• Паросочетание. Для заданного графа G найти множество несоприкасающихся ребер. ("Разбить вершины на пары так, чтобы каждая вершина имела не более одного партнера?") Обычно цель заключается в поиске наибольшего возможного количества ребер; "идеальное паросочетание" включает все вершины. В двудольном графе с m вершинами в одной части и n вершинами в другой паросочетание эквивалентно выбору множества единиц в матрице размером $m \times n$, состоящей из нулей и единиц, не более чем с одной выбранной в каждой строке и не более чем с одной выбранной в каждом столбце.

• Задача назначений. Обобщение двудольного паросочетания с весами, назначенными каждому ребру; общий вес паросочетания должен быть максимизирован. ("Какое назначение людей на работы наилучшее?") Эквивалентная постановка задачи заключается в таком выборе элементов матрицы размером $m \times n$ не более чем по одному в строке и одному в столбце, чтобы сумма выбранных элементов была наибольшей возможной.

• Покрытие. Для заданной матрицы A_{jk} из нулей и единиц найдите множество строк R, такое, что мы имеем $\sum_{j \in R} A_{jk} > 0$ для всех k. ("Отметьте 1 в каждом столбце и выберите все отмеченные строки?") Цель обычно заключается в том, чтобы минимизировать $|R|$.

• Точное покрытие. Для заданной матрицы A_{jk} из нулей и единиц найдите множество строк R, такое, что $\sum_{j \in R} A_{jk} = 1$ для всех k. ("Покрытие взаимно ортогональными строками?") Задача идеального паросочетания эквивалентна поиску точного покрытия транспонированной матрицы смежности.

• Независимое множество. Для заданного графа или гиперграфа G найдите множество вершин U, такое, что порожденный граф $G|U$ не имеет ребер. ("Выберите несвязанные вершины?") Цель обычно заключается в том, чтобы максимизировать $|U|$. Классические частные случаи включают задачу 8 ферзей, когда G представляет собой граф ходов ферзя на шахматной доске и задачу размещения точек так, чтобы никакие три не были на одной прямой.

• Клика. Для заданного графа G, найдите множество вершин U, такое, что порожденный граф $G \mid U$ является полным. ("Выберите взаимно смежные вершины?") Эквивалентная постановка задачи — найти независимое множество в $\sim G$. Цель обычно заключается в том, чтобы максимизировать $|U|$.

• Вершинное покрытие. Для заданного графа или гиперграфа найдите множество вершин U, таких, что каждое ребро включает как минимум одну вершину из U. ("Пометьте некоторые вершины таким образом, чтобы ни одно ребро не осталось непомеченным?") Эквивалентные формулировки — найти покрытие транспонированной матрицы смежности; найти U, такое, что множество $V \setminus U$ независимое, где V — множество всех вершин. Цель обычно заключается в том, чтобы минимизировать $|U|$.

• Доминирующее множество. Для заданного графа найдите множество вершин U, такое, что каждая вершина, не входящая в U, смежна с некоторой вершиной из U. ("Какие вершины находятся в одном шаге от них всех?") Классическая задача о пяти ферзях является частным случаем, когда G представляет собой граф ходов ферзя на шахматной доске.

• Ядро. Для заданного ориентированного графа найдите независимое множество вершин U, такое, что каждая вершина, не входящая в U, является предшественником некоторой вершины из U. ("В каких независимых позициях игры с двумя игроками ваш противник может заставить вас пропустить ход?") Если граф неориентированный, ядро эквивалентно

максимальному независимому множеству и доминирующему множеству, которое является одновременно минимальным и независимым.

• Раскрашивание. Для заданного графа найдите способ разделить его вершины на k независимых множеств. ("Раскрасьте вершины k красками так, чтобы никакие две смежные вершины не были окрашены одним цветом.") Цель обычно заключается в том, чтобы минимизировать k.

• Кратчайший путь. Для заданных вершин u и v ориентированного графа, в котором каждой дуге назначен вес, найти наименьший общий вес ориентированного пути из u в v. ("Определите наилучший маршрут.")

• Длиннейший путь. Для заданных вершин u и v ориентированного графа, в котором каждой дуге назначен вес, найти наибольший общий вес простого ориентированного пути из u в v. ("Какой маршрут отклоняется сильнее всего?")

• Достижимость. Для заданного множества вершин U в ориентированном графе G найдите все вершины v, такие, что $u \longrightarrow^* v$ для некоторого $u \in U$. ("Какие вершины встречаются на путях, начинающихся в U?")

• Остовное дерево. Для заданного графа G найдите свободное дерево F на тех же вершинах, такое, что каждое ребро F является ребром G. ("Выберите достаточно ребер для связывания всех вершин.") Если каждому ребру назначен вес, *минимальным остовным деревом* является остовное дерево с наименьшим общим весом.

• Гамильтонов путь. Для заданного графа G найдите путь P на том же множестве вершин, такой, что каждое ребро P является ребром G. ("Найдите путь, который проходит через каждую вершину ровно один раз.") Это классическая задача обхода конем, когда G является графом ходов коня на шахматной доске. Когда вершины G представляют собой комбинаторные объекты (например, кортежи, перестановки, сочетания, разбиения или деревья), являющиеся смежными, если они "близки" друг другу, гамильтонов путь часто называют кодом Грея.

• Гамильтонов цикл. Для заданного графа G найдите цикл C на том же множестве вершин, такой, что каждое ребро C является ребром G. ("Найдите путь, который проходит через каждую вершину ровно один раз и возвращается в начальную точку.")

• Задача коммивояжера. Найдите гамильтонов цикл с наименьшим общим весом, когда каждому ребру заданного графа назначен вес. ("Как дешевле всего посетить всех?") Если граф не имеет гамильтонова цикла, мы расширяем его до полного графа, назначая очень большой вес W каждому несуществующему ребру.

• Топологическая сортировка. Для заданного ориентированного графа найдите способ маркировки каждой вершины x различными числами $l(x)$ таким образом, чтобы из $x \longrightarrow y$ следовало $l(x) < l(y)$. ("Разместите все вершины в строку так, чтобы каждая вершина находилась левее своих преемников.") Такая маркировка возможна тогда и только тогда, когда заданный ориентированный граф ацикличен.

• Оптимальное линейное размещение. Для заданного графа найдите способ маркировки каждой вершины x различными целыми числами $l(x)$ таким образом, чтобы $\sum_{u—v} |l(u) - l(v)|$ имела наименьшее возможное значение. ("Разместите вершины в строку, минимизируя суммы длин получающихся ребер.")

• Задача укладывания рюкзака. Для заданной последовательности весов $w_1, \ldots, w_n$, порогового значения W и последовательности значений $v_1, \ldots, v_n$ найти $K \subseteq \{1, \ldots, n\}$, такое, что $\sum_{k \in K} w_k \le W$ и $\sum_{k \in K} v_k$ максимальна. ("Насколько большу́ю ценность можно унести?")

• Ортогональный массив. Для заданных положительных целых m и n найти массив размером $m \times n^2$ с элементами $A_{jk} \in \{0, 1, \ldots, n-1\}$, обладающим тем свойством, что из $j \neq j'$ и $k \neq k'$ следует $(A_{jk}, A_{j'k}) \neq (A_{jk'}, A_{j'k'})$. ("Построить m различных матриц $n \times n$ из n-арных цифр таким образом, что при наложении любых двух матриц встречаются все n^2 возможных пар цифр?") Случай $m = 3$ соответствует латинскому квадрату, а случай $m > 3$ соответствует $m - 2$ взаимно ортогональным латинским квадратам.

• Ближайший общий предок. Для заданных узлов леса u и v найти такой узел w, что каждый включающий предок u и v является также включающим предком w. ("Где кратчайший путь между u и v меняет направление?")

• Задача поиска минимума на отрезке. Для заданной последовательности чисел $a_1, \ldots, a_n$, найти наименьшие элементы каждого подынтервала $a_i, \ldots, a_j$ для $1 \leq i < j \leq n$. ("Выполните все возможные запросы, касающиеся наименьшего значения в каждом заданном диапазоне?") В упр. 150 и 151 из раздела 7.1.3 показана эквивалентность этой задачи задаче поиска ближайшего общего предка.

• Универсальный цикл. Для заданных b, k и N найти циклическую последовательность элементов $x_0, x_1, \ldots, x_{N-1}, x_0, \ldots$ из b-арных цифр $\{0, 1, \ldots, b-1\}$, обладающую тем свойством, что все комбинаторные размещения определенного вида задаются последовательными k-кортежами $x_0x_1 \ldots x_{k-1}, x_1x_2 \ldots x_k, \ldots, x_{N-1}x_0 \ldots x_{k-2}$. ("Вывести все возможности циклическим образом?") Результат именуется циклом де Брейна, если $N = b^k$ и встречаются все возможные k-кортежи; это универсальный цикл сочетаний, если $N = \binom{b}{k}$ и если встречаются все k-сочетания b объектов; и это универсальный цикл перестановок, если $N = b!$, $k = b - 1$ и если в виде k-кортежей встречаются все $(b-1)$-варианты.

В большинстве случаев мы были способны привести множественно-теоретическое определение, полностью описывающее задачу, хотя требование краткости зачастую приводило к некоторому ухудшению интуитивного восприятия задачи.

— М. Р. ГАРИ (M. R. GAREY) и Д. С. ДЖОНСОН (D. S. JOHNSON), *Список NP-полных задач* (*A List of NP-Complete Problems*) (1979)

ПРЕДМЕТНО-ИМЕННОЙ УКАЗАТЕЛЬ

Когда запись в предметно-индексном указателе ссылается на страницу с упражнением, дополнительную информацию можно получить в ответе к этому упражнению. Страницы с упражнениями входят в указатель только в случае ссылки на тему, которая в условии упражнения не упоминается.

2^m-way multiplexer, 309.
C_n, 32
  граф цикла, 541.
  (числа Каталана), 508.
C_{pq} (баллотировочные числа), 509.
e, 535, 579, 854.
k-дольный граф, 36.
k-раскрашивание гиперграфа, 54.
k-я степень графа, 529.
K_n, 32.
M_n (средний биномиальный коэффициент), 517.
n-арная булева функция, 76.
n-кортеж, 445.
  последовательность строк длины n, 330.
n-мерный куб, 378, 528, 544, 891.
n-распределение, 777.
pi, 491.
P_n, 32.
  граф пути, 541.
r-однородность, 54.
r-перестановки, 400.
S_n (число Шрёдера), 539.
t-арное дерево, 538, 538, 874.
t-арные деревья, 879.
∂, 426.
ϱ, 426.
δ-обмен, 177.
δ-сдвиг, 181.
$\exists$, 272.
$\forall$, 272.
γ, 579.
∞, 224.
κ_t, 428.
λ, 186, 234.
λn, 12, 225.
λx, 174, 231, 699.
λ_t, 428.
$[\lg x]$, 174.
μ_t, 428.
ν, 429, 658.
νn, 225, 12.
ϕ, 577, 579.
π, 77, 102, 106, 126, 148, 156, 159, 182, 245, 291, 368, 397, 400, 406, 409, 421, 435, 436, 443, 517, 539, 547, 579, 628, 715, 725, 758, 841, 878, 897, 904.
ρ, 186, 234, 658.
$\rho(k)$, 334.
ρn, 225.
ρx, 172.
σ-цикл, 494.
$\sigma(n)$, 465.
$\tau(x)$, 428.
ϖ_n, 474.
0–1
  матрица, 54, 241, 281, 295, 313, 316, 717.
  принцип, 93, 223.
2-адическая цепочка, 189.
2-адические целые числа, 166.
2-адические целые числа как метрическое пространство, 653.
2-адические числа, 360.
2-адические числа рациональные, 231.
2-номиальный коэффициент, 806.
2SAT, 82, 82.
3CNF, 81.
3SAT, 81, 608.
8-мерный куб, 346.

BDD, 243.
Bitburger Brauerei, 16.

C, 42.
CI-сеть, 98.
CMath, 733.
CNF, 78, 126.
covering, 517.
CRC, 220, 241.

DNF, 77, 126.

endo-order, 422.

GraphBase, 87.

hack, 815.
HAKMEM, 192, 650, 655.
Haralambous, 4.

I Ching (易經), 547, 574.
IBM Type 650, 6.
IEEE Transactions, 11.
ind α, 487.

Ludus Clericalis, 554, 574.

max, 88, 230.
METAFONT, 4, 31, 686.
METAPOST, 4.
min, 88, 230.
MIP-год, 20.
`MMIX`, 184, 231, 404, 439, 652, 658, 665, 667.
`MMIXAL`, 63.
MP3 (MPEG-1 Audio Layer III), 220.
`MXOR`, 405.

NP-полнота, 9, 57, 283, 311, 603, 785.

P-разбиения, 470.
$P = NP(?)$, 80.
PLA, 78.
Prolog, 82.
Proteus, 575.

q-баллотировочные числа, 536.
q-мультиномиальные коэффициенты, 438.
q-номиальная теорема, 831, 835.
q-номиальные коэффициенты, 806, 852, 878.
q-числа
  Каталана, 536.
  Стирлинга, 492.
QDD, 277.

RISC: Reduced Instruction Set Computer, 53.

SGB, 28.
Stanford GraphBase, 10, 28, 29, 31, 41, 44, 52, 297, 322, 368, 488, 527, 636, 888.
  полное руководство, 53.
SWAC, 24.

TEX, 31, 879.
TrueType, 686.

UCS, 240.
Unicode, 240.
UNIVAC 1206 Military Computer, 24.
UTF-8, 240.
UTF-16, 240.

Wikipedia, 242.

XOR, 74, 103, 795.

z-номиальная теорема, 831.

Абелева группа, 470, 597, 890.
Абель, Нильс Хенрик (Abel, Niels Henrik), 833.
Аборги, Сэмюэл Эдмунд Нии Саи (Aborhey, Samuel Edmund Nii Sai), 713.
Абстрактная алгебра, 253.
Аванн, Шервин Паркер (Avann, Sherwin Parker), 116, 798.
Автоматизация вывода, 614.
Автоморфизм, 34, 68, 587, 592, 597, 728.
Аграваль, Дхарма Пракаш (Agrawal, Dharma Prakash (धर्म प्रकाश अग्रवाल)), 650.
Адам, Андрас (Ádám, András), 765.
Адамар, Жак Соломон (Hadamard, Jacques Salomon), 762.
Адачи, Фимие (Adachi, Fumie (安達文江)), 566.
Адена, Майкл Энтони (Adena, Michael Anthony), 750.
Адресные биты, 715.
Айвес, Фредерик Малкольм (Ives, Frederick Malcolm), 401.
Айкен, Говард Хатавей (Aiken, Howard Hathaway), 133.
Айнли, Стивен (Ainley, Stephen), 750.
Айтаи, Миклош (Ajtai, Miklós), 119.
Акерс, Шелдон Букингем-мл. (Akers, Sheldon Buckingham, Jr.), 114, 302, 303, 754.
Аккорд, 417, 439.
Акл, Селим Георг (Akl, Selim George (سليم جورج عقل)), 798.
Активность, 420.
Алгебра
  графов, 65.
  медиан, 90–116.
  семейств, 298, 320, 322, 324, 325, 749.
Алгебраическая связность, 893.
Алгоритм
  без циклов, 338, 348, 358, 367, 398, 435, 436, 529, 757, 784, 798, 809, 814, 815, 829.
  двери-вертушки, 415, 460.
  обработки отношений эквивалентности, 95, 845.
  Орда–Смита, 570.
  разложения, 778.
Алгоритмы на графах, 193, 233.
Алексеев, Валерий Борисович, 626.
Аллуш, Жан-Поль (Allouche, Jean-Paul), 658.
Алмквист, Герт Эйнар Торстен (Almkvist, Gert Einar Torsten), 835.
Алон, Нога (Alon, Noga (נגה אלון)), 622, 647, 648.
Алонсо, Лорен (Alonso, Laurent), 882.
Алфавитный порядок, 60.
аль-Самаваль ибн Яхья ибн Яхуда аль-Магриб (al-Samaw'al (= as-Samaw'al), ibn Yaḥyā ibn Yahūda al-Maghribī (ابن يحيى بن يهودا المغربي السموعل)), 554, 898.
Альберс, Сюзанна (Albers, Susanne), 669.
Альду, Девид Джон (Aldous, David John), 513.
Альпинизм, 476.
Альсведе, Рудольф (Ahlswede, Rudolph), 826.
Альфа-канал, 229.
Альфрик Грамматик, аббат Эйншамский (Ælfric Grammaticus, abbot of Eynsham), 328.
Алюк, Факир Чанд (Auluck, Faqir Chand), 862.
Амано, Казуюки (Amano, Kazuyuki (天野一幸)), 292, 319.
Амарель, Сол (Amarel, Saul), 649.
Амир, Яир (Amir, Yair (יאיר עמיר)), 621.
Анаграмма, 552, 583.
Анализ алгоритмов, 63, 162, 224, 276, 321, 396–401, 404, 411, 434, 436, 437, 458, 508, 536, 541, 544, 666, 869.
Английский язык, 28.
Андерсен, Ларс Довлинг (Andersen, Lars Døvling), 579.
Антисимметричный ориентированный граф, 118.
Антисимметрия, 87.
Антицепочка подмножеств, 517.

Аппель, Кеннет Айра (Appel, Kenneth Ira), 37.
Арабские
математики, 579.
цифры, 574.
Аренс, Вильгельм Эрнст Мартин Георг (Ahrens, Wilhelm Ernst Martin Georg), 579, 749.
Арийоши, Хирому (Ariyoshi, Hiromu (有吉弘)), 674.
Арима, Ёриюки (Arima, Yoriyuki (有馬頼徸)), 566, 756.
Аримура, Хироки (Arimura, Hiroki (有村博紀)), 750, 752.
Арисава, Макото (Arisawa, Makoto (有澤誠)), 787.
Аристоксен (Aristoxenus (Ἀριστόξενος)), 551.
Аристотель из Стагира, сын Никомаха (Aristotle of Stagira, son of Nicomachus (Ἀριστοτέλης Νικομάχου ὁ Σταγιρίτης)), 557.
Арифметика
с многократной точностью, 247.
с плавающей точкой, 247, 896.
Арифметическая прогрессия, 60.
Арифметическое среднее, 470, 495.
Аркю, Дидье (Arquès, Didier), 896.
Арндт, Йорг Уве (Arndt, Jörg Uwe), 655, 665, 760.
Арнольд, Девид Брайан (Arnold, David Bryan), 511.
Архитектура SIMD, 185.
Асимптотические методы, 138, 451–458, 475–482, 494.
Аспвалл, Бенгт Ингемар (Aspvall, Bengt Ingemar), 114, 610.
Ассоциативность, *см.* Закон ассоциативности, 74.
Ассоциаэдр, 535.
Аткин, Артут Оливер Лонсдейл (Atkin, Arthur Oliver Lonsdale), 832.
Аткинсон, Майкл Дэвид (Atkinson, Michael David), 882.
Атомарность, 538.
Атрибуты Бога, 556.
Ауротор, 369.
Аутоморфизм, 378, 398.
Аффинная функция, 123, 629.
Ахмад, Салах (Ahmad, Salah (صلاح أحمد)), 554, 898.
Ациклический ориентированный граф, 200, 243.
Ацикличный граф, 35.
Ашар, Пранав Навинчандра (Ashar, Pranav Navinchandra (प्रणव नवीनचन्द्र आशर)), 704.
Ашбахер, Майкл Джордж (Aschbacher, Michael George), 587.
Ашенхарст, Роберт Ловетт (Ashenhurst, Robert Lovett), 147, 150.

Баец, Джон Карлос (Baez, John Carlos), 762.
База БДР, 304, 307, 308, 327.
Базис векторного пространства, 434.
Байт: 8-битовая величина, 240.
Байты, проверка относительного порядка, 186.
Бак, Маршалл Уилберт (Buck, Marshall Wilbert), 439.
Бакли, Майкл Роберт Уоррен (Buckley, Michael Robert Warren), 398.
Бакос, Тибор (Bakos, Tibor), 765.
Балдерик (Baldéric), 555.
Баллотировочные числа, 509, 536.
Баль, Лалит Раи (Bahl, Lalit Rai (ललित राय बहल)), 754.
Банг, Тогер Софус Вильгельм (Bang, Thøger Sophus Vilhelm), 577.
Бандельт, Ганс-Юрген (Bandelt, Hans-Jürgen), 617.
Барбара Милла, Даниэль (Barbará Millá, Daniel), 115.
Барбур Эндрю Дэвид (Barbour, Andrew David), 597.
Барвелл, Брайан Роберт (Barwell, Brian Robert), 399.
Барицентрические координаты, 46, 115.
Барон, Герд (Baron, Gerd), 579.
Бархан, 542.
Батлер, Йон Терри (Butler, Jon Terry), 699.
Батчер, Кеннет Эдвард (Batcher, Kenneth Edward), 226, 662.
Баумгарт, Брюс Гюнтер (Baumgart, Bruce Guenther), 177.
Баухуис, Бернард (Bauhuis, Bernard (= Bauhusius, Bernardus)), 562.
Бах, Йоганн Себастьян (Bach, Johann Sebastian), 17.
БДР, 243.
с подавлением нулей, 293–302.
Безу, Этьен (Bézout, Étienne), 898.
Безье, Пьер Этьен (Bézier, Pierre Etienne), 216, 237, 686.
Бейдлер, Джон Энтони (Beidler, John Anthony), 375.
Бейер, Уэнделл Терри (Beyer, Wendell Terry), 210, 520.
Бейс, Джон Картер (Bays, John Carter), 656.
Бекенбах, Эдвин Форд (Beckenbach, Edwin Ford), 413, 572.
Беккер, Гарольд Вильям (Becker, Harold William), 849, 853, 854, 870.
Беккетт, Сэмюэль Барклай (Beckett, Samuel Barclay), 365.
Беллман, Ричард Эрнест (Bellman, Richard Ernest), 13, 427, 896.
Беллхаус, Девид Ричард (Bellhouse, David Richard), 556.
Белок, 574.
Бенеш, Вацлав Эдвард (Beneš, Václav Edvard), 178.
Беннетт, Грэхем (Bennett, Grahame), 727.
Беннетт, Уильям Ральф (Bennett, William Ralph), 333.

Бентли, Йон Луи (Bentley, Jon Louis), 677.
Берг, Клод (Berge, Claude), 55, 695, 747.
Береле, Аллан (Berele, Allan), 851.
Берлекамп, Элвин Ральф (Berlekamp, Elwyn Ralph), 187, 640, 652, 680.
Берман, Чарльз Леонард (Berman, Charles Leonard), 700.
Берн, Йохен (Bern, Jochen), 713.
Бернайс, Пауль Исаак (Bernays, Paul Issak), 77.
Бернулли, Якоб (Bernoulli, Jacques (Jakob, James)), 424, 547, 561, 564, 565, 783, 899, 900.
Бернштейн, Артур Джай (Bernstein, Arthur Jay), 760.
Бернштейн, Бенджамин Абрам (Bernstein, Benjamin Abram), 599.
Бернштейн, Сергей Натанович, 686.
Берте, Кристиан (Berthet, Christian), 708.
Бесконечное множество, 750.
Бесконечность, 44.
Бесповторная функция, 290, 317.
  обобщенная, 728.
Бесполезная последовательность, 488.
Бессинджер, Джанет Симпсон (Beissinger, Janet Simpson), 728.
Бёрнер, Андерс (Björner, Anders), 820.
Биггс, Норман Линстед (Biggs, Norman Linstead), 35.
Биграф, 37.
Биклаттер, 540.
Биллон, Жан-Поль (Billon, Jean-Paul), 754.
Билогарифм, 833, 835.
Билогарифмическая функция, 465.
Бинарная арифметика, 548.
Бинарная диаграмма решений, 242–257.
Бинарная рекуррентность, 137, 138, 157, 225, 615, 633, 777.
Бинарная решетка мажоризации, 121.
Бинарная строка, 78.
Бинарное векторное пространство, 434.
  дерево, 112, 125, 127, 243, 323, 499, 502–508, 531–539, 571.
  дерево поиска, 234, 263, 537, 546.
  обозначение Дьюи, 879.
  отношение, 472.
  сложение, 136, 256, 308.
  умножение, 310.
  число Грея, 818.
Бинарные деревья поиска, 659.
Бинарные деревья, представление, 515.
Бинарные операции, 71, 304, 319.
  таблица, 73.
Бинарные разбиения, 470.
Бинарный декодер, 634.
Бинарный код Грея, 331–346, 358–362, 366, 372, 415, 544, 633, 652, 664, 671, 774, 809, 847, 883.
Бинарный логарифм, 174, 191, 231.
Бинарный луч, 360.
Бинарный луч Грея, 360.
  оператор, 261.
  поиск, 519.
Бинг, Р. Х. (Bing, R. H.) 577.
Биномиальное дерево, 435, 413, 895.
Биномиальный коэффициент, 440.
  обобщенный, 442.
Биох, Ян Корстиан (Bioch, Jan Corstiaan), 613.
Биразбиения, 485.
Биркхофф, Гарретт (Birkhoff, Garrett), 614, 846.
Бит
  переноса, 136, 158.
  четности, 60, 334.
Битнер, Джеймс Ричард (Bitner, James Richard), 338, 416.
Битовое дополнение, 258.
Битовые операции, 71, 100, 109, 129, 131, 156, 165, 393, 411, 599, 812, 813, 827.
Блейк, Арчи (Blake, Archie), 604.
Блек, Макс (Black, Max), 297.
Ближайший общий предок, 200, 234.
Близкое дерево, 522–527, 888.
Близость к идеалу, 418–424, 437, 504.
Блиссард, Джон (Blissard, John), 848.
Блоки, 471.
Блочный код, 327.
Блюм, Мануэль (Blum, Manuel), 696.
Блюм, Норберт Карл (Blum, Norbert Karl), 153.
Бодо, Жан Морис Эмиль (Baudot, Jean Maurice Émile), 333.
Бозе, Радж Чандра (Bose, Raj Chandra (রাজ চন্দ্র বসু)), 24.
Бойер, Роберт Стефен (Boyer, Robert Stephen), 70.
Бойяи, Янош (Bolyai, János), 203.
Болгарский пасьянс, 471.
Болкер, Этан Девид (Bolker, Ethan David), 854.
Болл, Майкл Оуэн (Ball, Michael Owen), 606, 607.
Боллиг, Беата Барбара (Bollig, Beate Barbara), 281, 287, 315, 718, 721, 732.
Боллобас, Бела (Bollobás, Béla), 626.
Бонди, Джон Адриан (Bondy, John Adrian), 34, 598.
Бонне, Энтони Эдмонд (Bonner, Anthony Edmonde), 898.
Бонферрони, Карло Эмилио (Bonferroni, Carlo Emilio), 835.
Боппана, Рави Бабу (Boppana, Ravi Babu), 647, 648.
Борель, Эмиль Феликс Эдуард Жюстен (Borel, Emile Félix Edouard Justin), 778.
Борковски, Людвик Стефан (Borkowski, Ludwik Stefan), 197.
Борос, Эндре (Boros, Endre), 150, 606.
Боррель, Жан (Borrel, Jean (= Buteonis, Ioannes)), 899.
Босвелл, Джеймс (Boswell, James), 16.

Боссен, Дуглас Крейг (Bossen, Douglas Craig), 582.
Ботерманс, Джекоб (Botermans, Jacobus (= Jack) Petrus Hermana), 772.
Боф, Чарльз Ричмонд (Baugh, Charles Richmond), 619, 621, 625.
Боченьски, Йозеф Мария (Bocheński, Józef (= Innocenty) Maria), 73.
Бошкович, Рудер Иосип (Bošković, Ruđer Josip), 569, 576, 836.
Брайант, Рэндал Эверитт (Bryant, Randal Everitt), 278, 280, 299, 300, 303, 311, 703, 709, 752.
Брайтвелл, Грэхэм Ричард (Brightwell, Graham Richard), 626, 797.
Брандт, Юрген (Brandt, Jørgen), 843.
Браун, Джозеф Алексендер (Brown, Joseph Alexander), 785.
Браун, Джон Уэсли (Brown, John Wesley), 580.
Браун, Дэвид Трент (Brown, David Trent), 220.
Браун, Уильям Гордон (Brown, William Gordon), 587.
Браун, Чарльз Филип (Brown, Charles Philip), 550.
Браунинг, Элизабет Барретт (Browning, Elizabeth Barrett), 493.
Брезенхам, Джек Элтон (Bresenham, Jack Elton), 685.
Брейн, Николас Говерт де (de Bruijn, Nicolaas Govert), 352, 407, 482, 515, 856.
Брейс, Карл Стивен (Brace, Karl Steven), 303.
Брейтбарт, Юрий Яковлевич (Breitbart, Yuri), 711, 753.
Брейтон, Роберт Кинг (Brayton, Robert King), 152, 736.
Брейш, Ричард Льюис (Breisch, Richard Lewis), 786.
Брент, Ричард Пирс (Brent, Richard Peirce), 635, 663.
Бретт, Жан (Brette, Jean), 578.
Бриггс, Генри (Briggs, Henry), 693.
Брилавски, Томас Генри (Brylawski, Thomas Henry), 837.
Бринкманн, Гуннар (Brinkmann, Gunnar), 594.
Бродал, Герт Столтинг (Brodal, Gerth Stølting), 188.
Бродник, Андрей (Brodnik, Andrej), 193.
Бройер, Мелвин Аллен (Breuer, Melvin Allen), 676.
Брон, Конраад (Bron, Coenraad), 674.
Броуновское движение, 512.
Бруальди, Ричард Энтони (Brualdi, Richard Anthony), 580.
Брук, Ричард Хьюберт (Bruck, Richard Hubert), 582.
Брукер, Ральф Энтони (Brooker, Ralph Anthony), 165.
Брукс, Роуланд Леонард (Brooks, Rowland Leonard), 589.
Брюстер, Джордж (Brewster, George), 27.
Буквометик, 375, 384, 400.
  чистый, 376, 398.
Булева матрица, 272.
Булева разность, 273, 710.
Булева решетка, 872.
Булева формула, 443.
Булева функция, 55, 518.
Булева цепочка, 124, 327.
Буль, Джордж (Boole, George), 72, 76, 252, 637.
Бургер, Алевин Петрус (Burger, Alewyn Petrus), 749.
Буркхардт, Иоганн Карл (Burckhardt, Johann Karl (= Jean Charles)), 570.
Бурлей, Вальтер (Burley (= Burleigh), Walter), 75.
Бусина, 243, 256, 260.
Бутеонис, Иоанн (Buteonis, Ioannes (= Borrel, Jean)), 899.
Бутон, Чарльз Леонард (Bouton, Charles Leonard), 650.
Бхаскара Акарья (Bhāskara II, Ācārya, son of Maheśvara (भास्कराचार्य, महेश्वरपुत्र)), 552.
Бхаттотпала (Bhaṭṭotpala (= Utpala, भट्टोत्पल)), 562.
Быстрое преобразование
  Уолша, 362.
  Фурье, 225, 357.
Бью, Ален (Bui, Alain), 20.
Бэббидж, Генри Провост (Babbage, Henry Provost), 636.
Бэббидж, Чарльз (Babbage, Charles), 145, 636, 641, 642.
Бюхи, Юлиус Рихард (Büchi, Julius Richard), 155, 654.
Бюхнер, Морган Меллори-мл. (Buchner, Morgan Mallory, Jr.), 760.

Вагналлс, Адам Уиллис (Wagnalls, Adam Willis), 73.
Вагнер, Эрик Герхардт (Wagner, Eric Gerhardt), 605.
Вада, Эиити (Wada, Eiiti (和田英一)), 655.
Вазони, Эндре (Vázsonyi, Endre), 773.
Вайз, Дэвид Стефен (Wise, David Stephen), 666.
Вайзнер, Луи (Weisner, Louis), 580.
Вайнбергер, Арнольд (Weinberger, Arnold), 635.
Вайндер, Роберт Оуэн (Winder, Robert Owen), 90, 620, 649.
Вайнер, Питер Галлегос (Weiner, Peter Gallegos), 148, 150, 162.
Валентность, *см.* степень, 33.
Ванг, Да-Лун (Wang, Da-Lun (王大倫)), 428, 431, 592.
Ванг, Пинг Янг (Wang, Ping Yang (王平, née 楊平)), 428, 431.
Ванг, Терри Мин Их (Wang, Terri Min Yih), 357.
Ванг, Шинмин Патрик (Wang, Shinmin Patrick (王新民)), 580.

Вандерлих, Чарльз Марвин (Wunderlich, Charles Marvin), 675.
Варди, Александер (Vardy, Alexander (אלכסנדר ורדי)), 754.
Вариация, 397, 400, 786, 796.
Варол, Яков Леон (Varol, Yaakov Leon (ורול יעקב לאון)), 392, 395.
Ватанабе, Масатоси (Watanabe, Masatoshi (渡邊雅俊)), 4.
Ватанабе, Хитоси (Watanabe, Hitoshi (渡部和)), 573.
Ватриквант, Симон (Vatriquant, Simon), 375.
Ватсон, Джордж Невилл (Watson, George Neville), 878.
Ваховский, Евгений Борисович, 891.
Вебер, Карл (Weber, Karl), 606.
Веблен, Освальд (Veblen, Oswald), 212.
Вегенер, Инго Вернер (Wegener, Ingo Werner), 155, 257, 281, 287, 291, 304, 315, 319, 598, 633, 699, 703, 708, 711, 713, 715, 716, 718, 721, 723, 727.
Вегман, Марк Н. (Wegman, Mark N.), 696.
Вегнер, Герд (Wegner, Gerd), 820.
Вейден, Путтило Леонович (Wegner, Peter (= Weiden, Puttilo Leonovich)), 172, 176.
Вейхсель, Пол Моррис (Weichsel, Paul Morris), 591.
Вектор, 78, 136.
  размеров, 442.
  счетов, 469.
Векторное пространство, 434, 540, 690.
Велтер, Корнелиус Петрус (Welter, Cornelis Petrus), 653, 654.
Вергилий (Vergil (= Publius Vergilius Maro)), 563.
Вермут, Удо (Wermuth, Udo), 10.
Вернике, Август Людвиг Пауль (Wernicke, August Ludwig Paul), 23.
Верофф, Роберт Луи (Veroff, Robert Louis), 614.
Верхнер, Ральф (Werchner, Ralph), 291, 727.
Верхняя граница, 139.
Верхняя полуплоскость, 679.
Верхняя тень, 426.
Верхушка, 38.
Вершик, Анатолий Моисеевич (Vershik, Anatoly Moiseevich), 458, 835.
Вершинное покрытие, 233, 305.
Вес Ли, 359.
Вестергомби, Каталин (Vesztergombi, Katalin), 594.
Вестон, Эндрю (Weston, Andrew), 390.
Ветвление, 217, 232.
Вёльфель, Петер Филипп (Wölfel (= Woelfel), Peter Philipp), 292.
Взаимная ортогональность, 59.
Взаимно несравнимые множества, 310.
Взаимоисключение, 115.
Вибольд, епископ Камбрайский (Wibold, bishop of Cambrai (= Wiboldus, Cameracensis episcopus)), 554, 556, 567.
Видеманн, Дуглас Генри (Wiedemann, Douglas Henry), 99, 439, 627, 774.
Видимый узел, 284, 722.
Вик, Кристофер Джон ван (Wyk, Christopher John Van), 682.
Виккерс, Вирджил Юджин (Vickers, Virgil Eugene), 763, 765.
Викли, Виллиам Дуглас (Weakley, William Douglas), 749.
Викулин, Анатолий Петрович, 624.
Вильсон, Вильфрид Джордж (Wilson, Wilfrid George), 373.
Вильсон, Дэвид Уитекер (Wilson, David Whitaker), 675.
Вильсон, Ричард Майкл (Wilson, Richard Michael), 580.
Вильсон, Робин Джеймс (Wilson, Robin James), 35, 65, 595.
Вильф, Герберт Саул (Wilf, Herbert Saul), 390, 415, 467, 541, 573, 800, 806.
Вильямс, Аарон Майкл (Williams, Aaron Michael), 407, 802, 815.
Вильямсон, Стенли Джилл (Williamson, Stanley Gill), 390, 402, 798, 844.
Виман, Макс (Wyman, Max), 481, 853, 855.
Винет, Норберт (Wiener, Norbert), 907.
Винклер, Питер Манн (Winkler, Peter Mann), 346, 347, 365, 764, 882.
Винникомб, Роберт Ян Джеймс (Vinnicombe, Robert Ian James), 398.
Винчи, Леонардо ди сер Пьеро да (Vinci, Leonardo di ser Piero da), 28, 44.
Виртуальный адрес, 309.
Високосный год, 669.
Витале, Фабио (Vitale, Fabio), 202.
Витерби, Эндрю Джеймс (Viterbi, Andrew (= Andrea) James), 754.
Вишкин, Узи Ешуа (Vishkin, Uzi Yehoshua (עוזי יהושע וישקין)), 200.
Вложенность, 326, 499.
Вложенные скобки, 223, 510, 514, 530, 538, 571, 572, 573.
Внешний узел, 499.
Во, Кьем-Фонг (Vo, Kiem-Phong), 885.
Военный рапорт, 329.
Возможность, 233.
Вольф, Маргарет Керолайн (Wolf, Margarete Caroline), 901.
Вольфрам, Стивен (Wolfram, Stephen), 690.
Вонг, Родерик Су-Чун (Wong, Roderick Sue-Chuen (王世全)), 482.
Вонг, Чак-Куен (Wong, Chak-Kuen (黃澤權)), 182, 227, 597.
Временна́я метка, 709.
Время, 230.
Ву, Ван Ха (Vũ, Văn Hà), 622.

Вуд, Фрэнк Вашингтон (Wood, Frank Washington), 141.
Вудкок, Дженнифер Роселинн (Woodcock, Jennifer Roselynn), 739.
Вудрам, Лютер Джей (Woodrum, Luther Jay), 174, 657.
Вудс, Дональд Рой (Woods, Donald Roy), 666.
Вул, Авишаи (Wool, Avishai (אבישי וול)), 621.
Входящая степень, 38, 39, 63, 298.
Выборы, 557.
Выделение
  битов, 172.
  и сжатие, 181.
  старшего бита, 231, 670.
Выполнимая функция, 80.
Выполнимость, 80–112.
Выпуклая оболочка, 94.
Выпуклая оптимизация, 666.
Вырожденность, 535, 537, 894.
Вычисление, 245.
  булевых функций, 124.
Вычисляемые таблицы, *см.* Кэш записей, 268.
Вычитание, 229.
  с насыщением, 12.
Вьено, Жерар Мишель Франсуа Ксавье (Viennot, Gérard Michel François Xavier), 896.
Вэнь-ван (King Wen), 547, 574.
Вюллемин, Жан Этьен (Vuillemin, Jean Etienne), 307, 677, 752.

Габов, Гарольд Нейл (Gabow, Harold Neil), 677, 889.
Гавел, Вацлав (Havel, Václav), 50.
Гаддам, Джерри Уильям (Gaddum, Jerry William), 594.
Гаджиев, Максим Мамаевич, 624.
Гай, Ричард Кеннет (Guy, Richard Kenneth), 640, 652, 680, 714.
Гален, Клавдий (Galen, Claudius (Κλαύδιος Γαληνός)), 73.
Галилей, Галилео (Galilei, Galileo), 567.
Галлаи, Тибор (Gallai (= Grünwald), Tibor), 591.
Галлиан, Джозеф Энтони (Gallian, Joseph Anthony), 794.
Галль, Франц Йозеф (Gall, Franz Joseph), 16.
Галлье, Жан Генри (Gallier, Jean Henri), 85.
Галуа, Эварист (Galois, Evariste), 378.
Гамбург, Майкл Александер (Hamburg, Michael Alexander), 654.
Гамильтон, Уильям Рован (Hamilton, William Rowan), 35.
Гамильтонов
  граф, 35, 584.
  путь, 35, 112, 299, 323, 344, 371, 390, 402, 404, 407, 438, 545, 792, 817, 871.
  цикл, 35, 323, 341, 363, 364, 372, 390, 401, 407, 765, 895.
Гамма-функция, 477, 833.
Ганкель, Герман (Hankel, Hermann), 478, 854.
Гантер, Бернгард (Ganter, Bernhard), 580.
Гарван, Френсис Жерар (Garvan, Francis Gerard), 832.
Гарди, Даниэль (Gardy, Danièle), 898.
Гарднер, Мартин (Gardner, Martin), 27, 30, 208, 389, 397, 559, 612, 641, 680, 750, 772, 780, 843.
Гари, Майкл Рэндольф (Garey, Michael Randolph), 919.
Гарихаран, Рамеш (Hariharan, Ramesh (ரமேஷ் ஹரிஹரன்)), 889.
Гармонические числа, 906–907.
Гарсиа, Адриано Марио (Garsia, Adriano Mario), 852.
Гарсия-Молина, Гектор (García-Molina, Héctor), 115.
Гаусс, Иоганн Фридрих Карл (Gauß (= Gauss), Johann Friderich Carl (= Carl Friedrich)), 23, 36, 203, 913.
Гвиттоне д'Ареццо (Guittone d'Arezzo), 493.
Гвоздяк, Павол (Gvozdjak, Pavol), 364.
Гебхардт, Дитер (Gebhardt, Dieter), 689.
Гейл, Дэвид (Gale, David), 839.
Гейне, Генрих Эдуард (Heine, Heinrich Eduard), 465.
Гекзаметр, 562, 575.
Гексаграмма, 547.
Генерация, 329.
  всех решений, 246, 305.
  перестановок, 562.
    быстрейшая, 391.
    в лексикографическом порядке, 381.
    применение, 396.
  случайных чисел, 124, 368.
  сочетаний, 408–426, 434–440, 443.
Генетический код, 574.
Геометрическая сеть, 59, 60, 62.
Геометрическое среднее, 470, 495.
Геральдика, 566.
Герардини, Лиза (Gherardini, Lisa), 44, 45.
Гершвин, Джордж (Gershwin, George), 560.
Гильберт, Давид (Hilbert, David), 443.
Гильберт, Уильям С. (Gilbert, William S.), 329.
Гильберт, Эдгар Нельсон (Gilbert, Edgar Nelson), 364.
Гильберта основная теорема, 443.
Гинденбург, Карл Фридрих (Hindenburg, Carl Friedrich), 370, 447, 475, 569.
Гипербола, 212, 237, 654, 685.
Гиперболическая плоскость, 202–216, 234.
Гиперболические функции, 495.
Гиперграф, 53, 67, 297, 746.
Гипердуга, 593.
Гиперкуб, 78.
Гиперлес, 67.
Гиперребро, 54.
Гипотеза, 84.
  средних уровней, 816.

Главная профилограмма, 282, 283, 290, 314, 717.
Главный подлес, 349.
Глаголев, Валерий Владимирович, 625.
Гладвин, Хармон Тимоти (Gladwin, Harmon Timothy), 172.
Глобальная переменная, 710.
Глубина булевой
  функции, 128.
  цепочки, 128, 156.
Годдин де ла Вега, Луи Армандо (Goddyn de la Vega, Luis Armando), 364, 766.
Годфри, Майкл Джон (Godfrey, Michael John), 58.
Голдман, Джей Роберт (Goldman, Jay Robert), 852.
Голдштейн, Алан Джей (Goldstein, Alan Jay), 392.
Голль, Филипп (Golle, Philippe), 540, 886.
Головоломки, 26, 578, 585.
Голомб, Соломон Вольф (Golomb, Solomon Wolf), 409, 433, 582, 844.
Голосование, 886.
Гомер (Homer (Ὅμηρος)), 28, 563.
Гомес, Питер Д. (Gomes, Peter J.), 16.
Гомогенность, 417, 418, 424.
Гомоморфизм, 100.
Гонзалес-Моррис, Герман Антонио (González-Morris, Germán Antonio), 786.
Гонне Хаас, Гастон Генри (Gonnet Haas, Gaston Henry), 856.
Гордиев узел, 366.
Гордон, Бэзил (Gordon, Basil), 487.
Гордон, Леонард Джозеф (Gordon, Leonard Joseph), 583.
Горный перевал, 476.
Госпер, Ральф Вильям-мл. (Gosper, Ralph William, Jr.), 223, 226, 241.
ГОСТ, 160.
Гото, Эиичи (Goto, Eiichi (後藤英一)), 619, 620.
Грамматика, 83.
Гранди, Патрик Майкл (Grundy, Patrick Michael), 650.
Грант, Джеффри Ллойдд Джагтон (Grant, Jeffrey Lloydd Jagton), 584.
Граф, 30–57, 61–68, 179, 310, 521–529, 542.
  алгебра, 46.
  Кейли, 390, 407, 597, 792.
  нулевой, 46.
  Петерсена, 33, 35, 46, 61, 65, 67, 68, 589.
  пути P_n, 888.
  Рамануджана, 45.
  ферзя, 67, 674.
  цикла C_n, 888.
  Чватала, 33, 61, 67, 594.
Греческая поэзия, 550.
Григгс, Джерольд Робинсон (Griggs, Jerrold Robinson), 843.
Грини, Кертис (Greene, Curtis), 518, 838, 885.
Грис, Дэвид Джозеф (Gries, David Joseph), 598.
Грос, Люк Агатон Луи (Gros, Luc Agathon Louis), 333.
Грот, Эдвард Джон,-мл. (Groth, Edward John, Jr.), 27.
Группа, 222, 378, 390.
  коммутативная, 470.
Группоид, таблицы умножения, 198.
Грэхем, Рональд Льюис (Graham, Ronald Lewis (葛立恒)), 117, 615, 739, 826, 863.
Грюнбаум, Бранко (Grünbaum, Branko), 594.
Грятцер, Георг (Grätzer, György (= George)), 872.
Гуд, Ирвинг Джон (Good, Irving John), 855.
Гуибас, Леонидас Джон (Guibas, Leonidas John (Γκίμπας, Λεωνίδας Ιωάννου)), 687.
Гун, Сян Цзун (Kung, Hsiang Tsung (孔祥重)), 635.
Гуо, Зиченг Чарльз (Guo, Zicheng Charles (郭自成)), 209, 236.
Гупта, Хансрай (Gupta, Hansraj (हंसराज गुप्ता)), 834.
Гурвиц, Адольф (Hurwitz, Adolf), 590.
Гурвич, Владимир Александрович (Gurvich, Vladimir Alexander), 150, 602.
Гутри, Френсис (Guthrie, Francis), 37.
Гутьяр, Уолтер Джозеф (Gutjahr, Walter Josef), 881.
Гюго, Виктор Мари (Hugo, Victor Marie), 44.
Гюнтер, Вольфганг Альбрехт (Günther, Wolfgang Albrecht), 720.

Дайсон, Фриман Джон (Dyson, Freeman John), 832.
Дакворт, Ричард (Duckworth, Richard), 373.
Дактиль, 550, 563.
Далли, Уильям Джеймс (Dally, William James), 759.
Даллос, Джозеф (Dallos, József), 173, 658.
Далхаймер, Торстен (Dahlheimer, Thorsten), 10, 611, 667, 696, 729, 730.
Данг, Тран-Нгок (Danh, Tran-Ngoc), 826.
Данте Алигьери (Dante Alighieri), 853.
Данхэм, Бредфорд (Dunham, Bradford), 122, 624.
Дарроч, Джон Ньютон (Darroch, John Newton), 860.
Дата, 168, 230.
Даулинг, Уильям Френсис (Dowling, William Francis), 85.
Дахми, Шаддин Фарис (Dughmi, Shaddin Faris (شادن فارس ألدغمي)), 751.
Дважды ограниченные разбиения, 458, 467, 469.
Дважды связанный список, 96, 350, 523, 773.
Двенадцатизадачие, 444, 462, 561.
Дверь-вертушка, 436, 521, 527, 542.
Двоичная система счисления, 71, 330, 548, 551, 560.

Дворжак, Станислав (Dvořák, Stanislav), 805.
Двудольный гиперграф, 594.
Двудольный граф, 37, 44, 45, 54, 57, 62, 64, 66, 67, 150, 180, 297, 586, 593, 679, 739.
Двумерное произведение, 366.
Двумерные данные, 180.
Двустрочная запись перестановки, 377.
Двустрочный массив, 849.
Двухуровневое представление, 300.
Де Брейн, Николас Говерт (de Bruijn, Nicolaas Govert), 352, 407, 482, 515, 856.
Дебай, Питер Джозеф Вильям (Debye, Peter Joseph William (= Debije, Petrus Josephus Wilhelmus)), 476.
Деген, Карл Фердинанд (Degen, Carl Ferdinand), 762.
Дедекинд, Юлиус Вильгельм Рихард (Dedekind, Julius Wilhelm Richard), 453.
Дейкстра, Эдсгер Вибе (Dijkstra, Edsger Wybe), 372, 666.
Действительные числа, 119.
Действительные корни, 496.
Декартово
  дерево, 659, 677.
  произведение, 12, 48, 65, 92, 526, 543.
Декартовы координаты, 212.
Декомпозиция функций, 315.
Декорированные бинарные деревья, 538.
Деление, 168.
  на 10, 190.
Делители, сумма, 465.
Делитель, 560, 567.
Деллак, Ипполит (Dellac, Hippolyte), 318, 730.
Дельта-последовательность, 341, 401, 815, 816.
Денг, Ева Ю-Пинг (Deng, Eva Yu-Ping (邓玉平)), 850.
Дене, Жозе (Dénes, József), 873.
Део, Нарсингх (Deo, Narsingh (नरसिंह देव)), 573.
Дерамида, 659.
Дербе, Йозеф (Derbès, Joseph), 842.
Дерево, 849.
  поиска, 25.
  проигравших, 808.
  рекурсии, 509.
  Фибоначчи, 544, 546.
  Шрёдера, 539.
  Штайнера, 8, 36.
Деревья, 498.
  без корня, *см.* Свободные деревья, 521.
  Грега, 856.
Дершовиц, Наум (Dershowitz, Nachum (נחום דרשוביץ)), 873.
Десятичная запись, 844.
  Дьюи, 510.
Десятичная система счисления, 330, 347, 369.
Детерминант, 553, 891.
Дефект, 538.
Джайн, Джавахар (Jain, Jawahar (जवाहर जैन)), 703.
Джанг, Минг (Jiang, Ming (姜明)), 390.
Джаст, Винфрид (Just, Winfried), 626.
Джевонс, Уильям Стенли (Jevons, William Stanley), 72, 75, 600.
Джей Rosenkrantz, Daniel Jay, 711.
Джейллет, Кристоф Андре Жорж (Jaillet, Christophe André Georges), 20.
Джеймс, Генри (James, Henry), 15.
Джексон, Бредли Уоррен (Jackson, Bradley Warren), 407, 827.
Джелинек, Фредерик (Jelinek, Frederick), 754.
Дженг, Сех-Вунг (Jeong, Seh-Woong (정세웅)), 698.
Дженкинс, Томас Арнольд (Jenkyns, Thomas Arnold), 418.
Дженоччи, Анжело (Genocchi, Angelo), 318, 730, 734.
Джеффри, Девид Джон (Jeffrey, David John), 856.
Джилл, Стенли (Gill, Stanley), 175.
Джиллис, Дональд Брюс (Gillies, Donald Bruce), 176.
Джоичи, Джеймс Томеи (Joichi, James Tomei (城市東明)), 831, 852.
Джонсон Аллан Уильям-мл. (Johnson, Allan William, Jr), 785.
Джонсон, Дэвид Стифлер (Johnson, David Stifler), 674, 919.
Джонсон, Селмер Мартин (Johnson, Selmer Martin), 398.
Джонсон, Сэмюэль (Johnson, Samuel), 6.
Джордан, Мари Эннемонд Камилла (Jordan, Marie Ennemond Camille), 212.
Джха, Пранава Кумар (Jha, Pranava Kumar (प्रणव कुमार झा)), 95.
Дзета-функция, 658, 833, 862.
Диагонализация, 155.
Диагональная матрица, 596.
Диагонально доминирующая матрица, 891.
Диаграмма, 33, 540, 885.
  Феррерса, 448, 454, 457, 461, 482, 837, 839, 843, 851.
  Феррерса, обобщенная, 830.
  Юнга, 536, 873.
Диаконис, Перси Уоррен (Diaconis, Persi Warren), 826, 863.
Диаметр, 64, 65, 68, 590.
  графа, 35, 62.
Дизъюнкт, 107.
  дизъюнкция литералов, 79.
  Крома, 81, 98, 112, 114, 643.
  Хорна, 81, 112, 113, 149, 613, 643.
Дизъюнктивная нормальная форма, 77, 107.
  ортогональная, 119.
  полная, 77.
Дизъюнктивная простая форма, 79, 90, 97, 300.
Дизъюнкция, 197.
Дик, Вальтер Франц Антон фон (Dyck, Walther Franz Anton von), 572.
Дикер Юцель, Мелек (Diker Yücel, Melek), 627.

Дикман, Говард Ллойд (Dyckman, Howard Lloyd), 366, 772.
Диллон, Джон Френсис (Dillon, John Francis), 627.
Динамическое переупорядочение переменных, 287, 295, 738.
Динамическое программирование, 13, 599, 896.
Динамическое распределение памяти, 269.
Диомед (Diomedes (Διομήδης)), 551.
Дирихле, Иоганн Петер Густав Лежён (Dirichlet, Johann Peter Gustav Lejeune), 658.
Дискретное преобразование Фурье, 122, 337, 600, 762.
Дискретный тор, 470.
Дисперсия, 227, 597.
Дистрибутивность, *см.* Закон дистрибутивности, 75.
Дитц, Генри Гордон (Dietz, Henry Gordon), 184, 185, 667.
Длина
  булевой функции, 127.
  внутреннего пути, 537.
  периода, 232.
  серии, 343, 346, 774.
  формулы, 132.
  уголков, 803.
ДНК, 574.
Добиньски, Г. (Dobiński, G.), 475.
Добродетель, 555, 556, 562.
Додекаэдр, 35.
Дойл, Артур Игнатий Конан (Doyle, Arthur Ignatius Conan), 19.
Дойч, Эмерик (Deutsch, Emeric), 883.
Доказательство теоремы, 614.
Доминирующее множество, 325, 680.
  минимальное, 325.
Домино, 296, 321, 322, 444, 497.
Донноло, Шаббетаи бен Авраам (Donnolo, Shabbetai ben Avraham (שבתי בן אברהם דונולו)), 551, 574.
Дополнение, 46, 65, 69, 74, 128, 166, 221, 320, 430, 543, 547, 598, 674, 735, 892.
  простого ориентированного графа, 65.
Дополнительный двоичный код, 166.
Достижимость, 200.
Достоверность, 73.
Дрексель, Иеремия (Drexel (= Drechsel = Drexelius), Jeremias (= Hieremias)), 551, 552.
Дрешлер, Николь (Drechsler, Nicole), 720.
Дрешлер, Рольф (Drechsler, Rolf), 720.
Дроби, египетские, 733.
Дробная точность, 168.
Ду, Розена Ру-Кси (Du, Rosena Ruo-Xia (杜若霞)), 850.
Дуальная генерация перестановок, 386.
Дуальное тождество, 599.
Дуальность, 54, 57, 67, 313, 318, 441, 442, 532, 533, 603, 619, 681, 693, 728, 825, 850.
Дуальные сочетания, 408, 434, 438.
Дуальный
  лес, 506.
  планарный граф, 888.
Дуб, Майкл (Doob, Michael), 894.
Дуброва, Елена Владимировна (Dubrova, Elena Vladimirovna), 694, 720.
Дуга, 38–44.
Дугвайд, Эндрю Мелвилл (Duguid, Andrew Melville), 178.
Дутт, Эдмонд (Doutté, Edmond), 579.
Дуэт: 16-битовая величина, 240.
Дхар, Дипак (Dhar, Deepak (दीपक धर)), 890.
Дьюдени, Генри Эрнест (Dudeney, Henry Ernest), 334, 375, 399, 488, 545, 585, 748, 749, 780, 787, 895, 896.
Дэвидсон, Джордж Генри (Davidson, George Henry (= Dee, G.), 828.
Дэйвис Рой Осборн (Davies, Roy Osborne), 20, 577.
Дэйкин, Дэвид Эдвард (Daykin, David Edward), 819, 826.
Дюваль, Жан-Пьер (Duval, Jean-Pierre), 778.
Дюмон, Доминик (Dumont, Dominique), 318, 730, 734.

Евклид (Euclid (Εὐκλείδης)), 203.
Евклидово расстояние, 29, 31.
Египетские дроби, 733.
Единичная матрица, 692, 891.
Единичный вектор, 430, 540.
Естественное соответствие, 499, 531.

Ёшида, Мицуёши (Yoshida, Mitsuyoshi (吉田光由)), 322.

Жадный алгоритм, 143, 549.
Жанг, Линбо (Zhang, Linbo (张林波)), 4.
Жани, Николя (Janey, Nicolas), 896.
Жао, Ксишун (Zhao, Xishun (赵希顺)), 611.
Жарден, Николя (Jardine, Nicholas), 674.
Жегалкин, Иван Иванович, 76, 600.
Жизнь, 235, 316, 725.
Жоливальд, Филипп (Jolivald, Philippe (= Paul de Hijo)), 826.

Завершение квадрата, 452.
Зависимость от переменной, 244.
Задача
  булева программирования, 249.
  выполнимости для дизъюнктов Хорна, 85.
  двойного покрытия, 249.
  достижимости, 193.
  коммивояжера, 9, 299, 396.
  Лэнгфорда, 27.
  о восьми ферзях, 674, 748.
  о парковке, 890.
  о рюкзаке, 415.
  поиска минимума на отрезке, 234.
  точного покрытия, 20, 25, 27, 59, 295, 321, 565, 578, 579, 740.

Задачи целочисленного программирования, 303.
Заем, 667, 668, 755.
Зайльбергер, Дорон (Zeilberger, Doron (דורון ציילברגר)), 464, 831.
Зайтц, Рихард (Seitz, Richard), 388.
Закон
  ассоциативности, 49, 65, 75, 90, 93, 106, 166, 253, 307, 317, 320, 505, 531, 535, 545, 601, 605, 633, 636, 707.
  больших чисел, 495.
  дистрибутивности, 66, 72, 75, 90, 106, 114, 117, 120, 156, 166, 253, 307, 320, 534, 590, 601, 602, 614, 651, 707.
  дополнения, 75.
  Кирхгофа, 458.
  коммутативности, 75, 90, 117, 166, 317, 320, 605, 633, 650, 707.
  мажоритарности, 90, 117.
  поглощения, 75, 166.
  полудистрибутивности, 535.
  разложения, 252.
  разработки, 77.
  сокращения, 102, 651.
  транзитивности, 405, 581.
Законы де Моргана, 75, 88, 107, 599.
Закс, Шмуль (Zaks, Shmuel (שמואל זקס)), 530, 532, 536, 867, 873.
Зальцер, Герберт Эллис (Salzer, Herbert Ellis), 760.
Замена переменных
  константами, 259.
  функциями, 309.
Замощение, 216.
  плоскости, 68.
Замощение, *см.* задача точного покрытия, 295.
Замыкание, 325.
Зантен, Аренд Ян ван (Zanten, Arend Jan van), 759, 809.
Заполнение
  контура, 212–217.
  рюкзака, 435.
Запоминание, 268.
Зауэрхоф, Мартин Поль (Sauerhoff, Martin Paul), 281, 291, 598, 708, 715, 727, 731.
Звезда, 403, 892.
Звездная транспозиция, 389.
Зейдель, Филипп Людвиг фон (Seidel, Philipp Ludwig von), 730.
Зейлстра, Эрик (Zijlstra, Erik), 198.
Зеркальная пара, 60.
Зехфусс, Иоганн Георг (Zehfuss, Johann Georg), 590.
Зилинг, Детлер Германн (Sieling, Detlef Hermann), 257, 713, 724.
Зито, Дженнифер Снайдер (Zito, Jennifer Snyder), 597.
Знак перестановки, 374.
Знаковая перестановка, 398.
Знаковые биты, представление, 195.
Знаковый сдвиг вправо, 218, 657.
Знакопеременная комбинаторная система счисления, 417.
Знакопеременные сочетания, 436.
Зогби, Антуан Шабан (Zoghbi, Antoine Chaiban (أنطون شيبان الزغبي)), 446.
Золотое сечение, 904.
Зуев, Юрий Анатольевич, 625.
Зыков, Александр Александрович, 47.

И, 74, 735.
Ибараки, Тошихиде (Ibaraki, Toshihide (茨木俊秀)), 150, 606, 612, 613, 711, 728.
Ибн аль-Хадж, Мухаммед ибн Мухаммед (Ibn al-Ḥājj, Muḥammad ibn Muḥammad (محمد بن محمد بن الحاج)), 579.
Ибн Муним, Ахмад аль-Абдари (Ibn Munʿim, Aḥmad al-ʿAbdarī(أحمد العبدري بن منعم)), 561.
Ивани, Антал Миклош (Iványi, Antal Miklós), 780.
Иврит, 58, 551, 559, 899.
Игараши, Ёшихиде (Igarashi, Yoshihide (五十嵐善英)), 624.
Игра, 144, 208, 221.
Игральные
  карты, 22, 444, 497.
  кости, 554, 567, 568, 574, 899.
Игуса, Кийоши (Igusa, Kiyoshi (井草潔)), 843.
Иде, Микио (Ide, Mikio (井手幹生)), 674.
Идеал, 90, 94, 439, 504, 871.
Идеальная хеш-функция, 657.
Идеальное разбиение, 471.
Идеальное тасование, 61, 180, 219, 225, 227, 241, 660, 669.
Идеальные схемы, 423.
Идель, Моше (Idel, Moshe (משה אידל)), 559.
Идес, Питер Деннис (Eades, Peter Dennis), 424, 815.
Изинг, Эрнст (Ising, Ernst), 807.
Изкуэрдо, Себастьян (Izquierdo, Sebastián), 560, 575.
Изображение
  бинарного дерева, 513, 865, 870.
  расширенного бинарного дерева, 545.
Изолированная вершина, 45, 274.
Изометрический подграф, 617.
Изометрия, 653.
  сохранение расстояний, 118.
Изоморфизм, 38.
  БДР, 305.
Изоморфные графы, 33.
Иисус из Назарета, сын Иосифа (Jesus of Nazareth, son of Joseph (ישוע בן יוסף בן נצרת, Ἰησοῦς υἱὸς τοῦ Ἰωσὴφ ὁ ἀπὸ Ναζαρέθ)), 552.
ИЛИ, 74, 735.
Импликанта, 78, 107.
Импликация, 73, 197.
Имрих, Вильфред (Imrich, Wilfried), 50, 95, 615.
Инверсия, 372, 374, 450, 492.
Инволюция, 313, 406, 495, 705, 792, 798, 850.

Индекс, 487, 841.
Индийская математика, 548, 552, 554, 561.
Индийские цифры, 552.
Интегрирование по контуру, 475–482.
Интернет, 10, 14, 29, 434, 573, 584, 844, 889.
Инь и ян, 547.
Ирвин, Джозеф Оскар (Irwin, Joseph Oscar), 848.
Исаак, Гарт Тимоти (Isaak, Garth Timothy), 780, 802.
Исключающая нормальная форма, 76.
Исключающее ИЛИ, 74, 795.
Искусственный интеллект, 758.
Исламские математики, 579.
Исозаки, Хидеки (Isozaki, Hideki (磯崎秀樹)), 745.
Источник, 38, 87.
Истрате, Габриэль (Istrate, Gabriel), 609.
Исходящая степень, 38, 39, 42, 63, 298.
Итерация функций, 361, 760.

Йии, Ае Я (Yee, Ae Ja (이애자)), 831.
Йошигахара Нобуюки (Yoshigahara, Nobuyuki (= Nob) (芦ヶ原伸之)), 399, 786.

Каас, Роберт (Kaas, Robert), 198.
Каасила, Сампо Юхани (Kaasila, Sampo Juhani), 686.
Каббала, 551, 559, 899.
Кавут, Сельчук (Kavut, Selçuk), 627.
Кавьер, Стефан Роберт (Cavior, Stephan Robert), 759.
Каган, Стивен Джей (Kahan, Steven Jay), 780, 785.
Кадр стека, 485.
Кай, Нинг (Cai, Ning (蔡宁)), 826.
Кайл, Генрих (Keil, Heinrich), 551.
Кайма, 774.
Как, Сабхаш Чандра (Kak, Subhash Chandra (सुभाष चन्द्र काक)), 550.
Калаби, Эженио (Calabi, Eugenio), 806.
Калдербанк, Артур Роберт (Calderbank, Arthur Robert), 758.
Камеда, Тико (Kameda, Tiko (= Tsunehiko) (亀田恒彦)), 612, 621.
Камен, Поль Фредерик Роже (Camion, Paul Frédéric Roger), 303.
Камерон, Роберт Дуглас (Cameron, Robert Douglas), 692.
Каммингс, Ларри Джин (Cummings, Larry Jean), 774.
Камминс, Ричард Ли (Cummins, Richard Lee), 887.
Кан, Леонард (Cahn, Leonard), 207.
Канализирующая функция, 104, 157, 307.
Каneда, Такаюки (Kaneda, Takayuki (金田高幸)), 721.
Канонические дельта-последовательности, 765.
Канонический
  базис, 434, 440.
  лес, 519, 540.
Кантор, Георг Фердинанд Людвиг Филипп (Cantor, Georg Ferdinand Ludwig Philipp), 666.
Кантор, Мориц Бенедикт (Cantor, Moritz Benedikt), 570.
Канфилд, Эрл Родни (Canfield, Earl Rodney), 798, 860.
Капланский, Ирвинг (Kaplansky, Irving), 572.
Капур, Санджив (Kapoor, Sanjiv (संजीव कपूर)), 889.
Кардано, Джироламо (Cardano, Girolamo (= Hieronymus Cardanus)), 756.
Карлиц, Леонард (Carlitz, Leonard), 842, 853, 878.
Карманная сортировка, 258, 263.
Карно, Морис (Karnaugh, Maurice), 359.
Карон, Жак (Caron, Jacques), 811.
Кароньски, Михал (Karoński, Michał), 597.
Карп, Ричард Маннинг (Karp, Richard Manning), 151.
Карпиньски, Марек Мечислав (Karpiński (= Karpinski), Marek Mieczysław), 611.
Кастаун, Рудольф Уильям (Castown, Rudolph William), 339.
Кастеризация, 207, 235.
Каталан, Эжен Шарль (Catalan, Eugène Charles), 508, 573, 804.
Катаяйнен, Юрки Юхани (Katajainen, Jyrki Juhani), 202.
Категорийное произведение, 49.
Катлер, Роберт Брайан (Cutler, Robert Brian), 603.
Катона, Гюла (Katona, Gyula), 427, 885.
Каттель, Кевин Майкл (Cattell, Kevin Michael), 778.
Кауффман, Стюарт (Kauffman, Stuart Alan), 104.
Кац, Уильям Холл (Kautz, William Hall), 623.
Квадрадерево, 666.
Квадрат
  графа, 529.
  Дюрфи, 449, 457, 830.
  из слов, 30, 60.
Квадратичная форма, 213.
Квантифицированные формулы, 272.
Квантор
  всеобщности, 114.
  существования, 114.
Квартет, или тетрада: 32-битовая величина, 240.
Кватернионы, 362.
Кедара Бхатта (Kedāra Bhaṭṭa (केदार भट्ट)), 549, 569.
Кейли граф, 401.
Кейли, Артур (Cayley, Arthur), 390, 571, 839.
Кейстер, Уильям (Keister, William), 145, 758.

Келли, Патрик Артур (Kelly, Patrick Arthur), 750.
Келли, Пол Джозеф (Kelly, Paul Joseph), 593.
Кельманс, Александр Кольманович, 893.
Кемп, Рейнер (Kemp, Rainer), 782, 879, 880.
Кемпе, Альфред Брей (Kempe, Alfred Bray), 584.
Кербош, Жозе Аугуст Жерар Мари (Kerbosch, Joseph (= Joep) Auguste Gérard Marie), 674.
Керниган, Брайан Уилсон (Kernighan, Brian Wilson), 125.
Кертис, Герберт Аллен (Curtis, Herbert Allen), 124, 151.
Кёниг, Дене (Kőnig, Dénes), 37.
Кимоно, 566.
Кинг, Эндрю Дуглас (King, Andrew Douglas), 799.
Киркман, Томас Пенингтон (Kirkman, Thomas Penyngton), 35.
Кирхер, Анастасиус (Kircher, Athanasius), 551, 552, 559, 574, 898.
Кирш, Расселл Эндрю (Kirsch, Russell Andrew), 207, 235.
Киршенгофер, Питер (Kirschenhofer, Peter), 881.
Кисс, Стефен Энтони (Kiss, Stephen Anthony), 614.
Китаев, Сергей Владимирович, 844.
Китайская математика, 547.
Китайские кольца, 333, 335, 358, 436, 756, 758.
Кифер, Джек Карл (Kiefer, Jack Carl), 770.
Клавжар, Санди (Klavžar, Sandi), 50, 95.
Клавиатура, 418.
Клайн, Джон Роберт (Kline, John Robert), 614.
Клайн, Питер (Klein, Peter), 158.
Классон, Андерс Карл (Claesson, Anders Karl), 844.
Классы эквивалентности, 472, 543, 626.
Кластеризация, 8.
Клаттер, 310, 442, 517, 704, 736, 746.
Клаузен, Томас (Clausen, Thomas), 23.
Клафам, Кристофер Роберт Джаспер (Clapham, Christopher Robert Jasper), 592.
Клебер, Майкт Стивен (Kleber, Michael Steven), 841.
Клементс, Джордж Френсис (Clements, George Francis), 433, 442, 824, 825.
Клеппис, Грегор (Kleppis, Gregor (= Kleppisius, Gregorius)), 575.
Клеточный автомат, 208, 235.
Кли, Виктор (Klee, Victor), 11.
Клика, 57, 233, 305, 313, 325, 440, 648.
  максимальная, 324.
Кликовое покрытие, 57.
Кликовое число, 57.
Климко, Юджин Мартин (Klimko, Eugene Martin), 828.
Клифт, Нейлл Майкл (Clift, Neill Michael), 692.
Клюгель, Георг Симон (Klügel, Georg Simon), 382, 570.
Кляйне Бюнинг, Ганс Герхард (Kleine Büning, Hans Gerhard), 609–611.
Кляйтман, Даниэль (Kleitman, Daniel J (Isaiah Solomon)), 518, 540, 592, 616, 838, 885.
Кнёдель, Вальтер (Knödel, Walter), 674.
Кноблох, Эбернгард Генрих (Knobloch, Eberhard Heinrich), 567, 568.
Кнопп, Марвин Исадор (Knopp, Marvin Isadore), 833.
Кнотт, Гари Дон (Knott, Gary Don), 573.
Кнут, Дональд Эрвин (Knuth, Donald Ervin (高德纳)), 10, 17, 26, 28, 31, 53, 87, 134, 145, 188, 299, 580, 602, 615, 636, 656, 658, 675, 682, 686, 687, 689, 706, 707, 724, 738, 740, 742, 746, 754, 774, 806, 856, 885, 888, 889, 915.
Ко, Чао (Ko, Chao (= Ke Zhao, 柯召)), 444, 621.
Коалесценция, 489.
Коалиция, 472.
Кобхэм, Алан (Cobham, Alan), 302.
Ковалевски, Арнольд (Kowalewski, Arnold), 589.
Ковер, Томас Меррилл (Cover, Thomas Merrill), 32.
Коган, Александр Юрьевич, 150, 606.
Коген, Филип Майкл (Cohen, Philip Michael), 584.
Код
  ASCII, 229.
  Грея, 344, 490, 521, 573, 578, 780.
  Грея двери-вертушки, 424, 526.
  Грея для n-кортежей, небинарный, 347.
  Грея для вложенных скобок, 537.
  Грея для деревьев Шрёдера, 539.
  Грея для деревьев, 504–508.
  Грея для перестановок, 401, 402.
  Грея для разбиений, 460, 841.
  Грея для разбиений множеств, 473.
  Грея для скобок, 871.
  Грея для слабых порядков, 800.
  Грея для сочетаний, 415–426.
  Морзе, 366, 549, 773.
  Хэмминга, 327.
Кода, Ясунори (Koda, Yasunori (黄田保憲)), 349, 350.
Кодирование
  длин серий, 501, 530, 537, 682, 867.
  тернарных данных, 195.
Кодон, 574.
Коды
  с коррекцией ошибок, 60, 327, 359.
  уровней, 540.
Кок, Джон Краул (Cock, John Crowle), 779.
Кокейн, Эрнст Джеймс (Cockayne, Ernest James), 749.
Колесо, 65, 590.
Колибяр, Милан (Kolibiar, Milan), 614.

Количество
мультиразбиений, таблица, 861.
переходов, 343, 363.
разбиений, 450–457, 465.
таблица, 451, 455, 861.
Коллизия, 268.
Колмен, Уолтер Джон Александер (Colman, Walter John Alexander), 834.
Колоколообразная кривая, 480, 495, 496.
Колтарст, Томас Уоллес (Colthurst, Thomas Wallace), 841.
Комбинаторика, 19–26.
Комбинаторная система счисления, 413, 435, 440, 468, 805, 843.
обобщенная, 442.
Комбинаторное представление, 815.
Комбинаторный взрыв, 6, 266.
Комбинационная сложность, 126.
Комлёш, Янош (Komlós, János), 119.
Коммутативная группа, 470.
Коммутативность, *см.* Закон коммутативности, 75.
Компельмахер, Владимир Леонтьевич, 403.
Компилятор, 83.
Комплексно сопряженное значение, 893.
Комплементарность, 535, 551.
Комплементарный код Грея, 342, 363.
Композиция, 358, 409, 418, 445, 466, 554, 575, 807, 861.
булевых функций, 309.
перестановок, 222.
функций, 275, 636.
Компонент, 62, 63, 65.
Комптон, Роберт Кристофер (Compton, Robert Christopher), 390, 402.
Комте, Луи (Comtet, Louis), 474, 799, 856.
Кон, Мартин (Cohn, Martin), 765, 767, 770.
Конвалина, Джон (Konvalina, John), 626.
Конвей, Джон Хортон (Conway, John Horton), 208, 316, 407, 640, 652, 653, 680, 762.
Конечное поле, 219, 581, 689.
Конечный автомат, 302, 327, 671.
Коническое сечение, 212, 237.
Конкатенация, 355.
Константа Эйлера, 904.
Константы с многократной точностью, 904–906.
Контекстно-свободная грамматика, 112, 733.
Контрольная сумма, 103, 137, 157, 429, 437, 658, 805.
Контрольное сложение, 175, 193, 232, 254, 278, 669, 675.
Контрольное суммирование, 250.
Контур, 511.
Ганкеля, 495.
Конфигурация, 526.
Изинга, 435, 440.
Конфуцианство, 547.
Концевая рекурсия, 708.
Конъюнктивная нормальная форма, 78, 107.
полная, 77.
Конъюнктивная простая форма, 79.
Конъюнкция, 49, 73, 197, 260, 543.
Координаты, 341.
глубины, 869.
уровня, 533, 537.
Копперсмит, Дон (Coppersmith, Don), 597.
Кордемский, Борис Анастасьевич, 863.
Корень, 242, 244, 438, 499, 834.
единицы, 454.
Кори, Роберт (Cori, Robert), 890.
Корлесс, Роберт Малкольм (Corless, Robert Malcolm), 856.
Коробков, Виталий Константинович, 519.
Корр, Жан Пьер ле (Corre, Jean Pierre Le), 178.
Кортежи, 547, 561.
Кортил, Сильвия Мари-Клод (Corteel, Sylvie Marie-Claude), 831.
Корш, Джеймс Ф. (Korsh, James F.), 502, 532, 807, 868, 882.
Коршунов, Алексей Дмитриевич, 625.
Косточка, Александр Васильевич, 894.
Котани, Ёшиюки (Kotani, Yoshiyuki (小谷善行)), 739.
Коук, Джон (Cocke, John), 754.
Кох, Джон Аллен (Koch, John Allen), 37.
Коши, Августин Луи (Cauchy, Augustin Louis), 458, 466.
Краевое представление, 449, 457, 463, 468.
Краецки, Мишель (Krajecki, Michaël), 20.
Крайние
слева биты, 174.
справа биты, 171.
Крама, Ив Жан-Мари Мэтью Франц (Crama, Yves Jean-Marie Mathieu Franz), 606.
Крамер, Карл Гаральд (Cramér, Carl Harald), 858.
Крамп, Кристиан (Kramp, Christian), 475.
Кратохвил, Ян (Kratochvíl, Jan), 69, 597.
Кратчайшая дизъюнктивная нормальная форма, 122.
Кратчайший путь, 9, 30, 35, 53, 91, 720.
Краузе, Карл Христиан Фридрих (Krause, Karl Christian Friedrich), 783.
Креверас, Герман (Kreweras, Germain), 870, 893.
Кремер, Уильям Генри-мл. (Cremer, William Henry, Jr.), 772.
Крестики-нулики, 144.
Кретте де Паллюль, Франсуа (Cretté de Palluel, François), 26.
Крехер, Дональд Лоусон (Kreher, Donald Lawson), 573.
Кривизна, 685.
Криптарифм, 375.
Крист, Вильгельм фон (Christ, Wilhelm von), 551.

Кристи Маллоуэн, Агата Мэри Кларисса Миллер (Christie Mallowan, Agatha Mary Clarissa Miller), 38, 902.
Кричевский, Рафаил Евсеевич, 638.
Кром, Мелвен Роберт (Krom, Melven Robert), 87, 611.
Кронекер, Леопольд (Kronecker, Leopold), 590, 891.
Кронк, Хадсон ван Эттен (Kronk, Hudson Van Etten), 895.
Круйсвейк, Дирк (Kruyswijk, Dirk), 515.
Крускал, Джозеф Бернард-мл. (Kruskal, Joseph Bernard, Jr.), 427, 428.
Ксианг, Лимин (Xiang, Limin (相利民)), 504.
Куайн, Уиллард Ван Орман (Quine, Willard Van Orman), 78, 80, 108, 605.
Куб
  графа, 529.
  из слов, 30, 61.
Кубицка, Ева (Kubicka, Ewa), 886.
Кубический граф, 33.
Кудер, Оливье Рене Раймонд (Coudert, Olivier René Raymond), 304, 697, 708, 738, 745, 746, 751.
Кук, Джордж Эрскин (Cooke, George Erskine), 649.
Кук, Раймонд Марк (Cooke, Raymond Mark), 767, 771, 802.
Кук, Стефен Артур (Cook, Stephen Arthur), 608.
Кумар, Панганамала Виджай (Kumar, Panganamala Vijay (పన్గనమల విజయ్ కుమార్)), 758.
Кумулянт, 495, 858.
Кун, Маркус Гюнтер (Kuhn, Markus Günther), 690.
Кусуба, Таканори (Kusuba, Takanori (楠葉隆徳)), 561.
Кшишанг, Франк Роберт (Kschischang, Frank Robert), 754.
Кэллан, Колумсиль Девид (Callan, Columcille David), 866.
Кэрролл, Льюис (= Доджсон, Чарльз Лютвидж) (Carroll, Lewis (= Dodgson, Charles Lutwidge)), 29–31, 72, 106, 584.
Кэш, 218, 656.
  записей, 268.
Кэш-память, 168, 268.
Кэширование, 312.

Лабель, Жильберт (Labelle, Gilbert), 862.
Лаборд, Жан-Мари (Laborde, Jean-Marie), 605.
Лавджой, Джереми Кеннет (Lovejoy, Jeremy Kenneth), 831.
Лагерр, Эдмон Николя (Laguerre, Edmond Nicolas), 860.
Ладейные полиномы, 491.
Ладнер, Ричард Эмиль (Ladner, Richard Emil), 157, 159.
Ладья, 849.
Лаксер, Гарри (Lakser, Harry), 872.
Ламберт, Иоганн Генрих (Lambert, Johann Heinrich), 569.
Лампорт, Лесли Б. (Lamport, Leslie B.), 115, 184, 185, 230.
Лангдон, Глен Джордж-мл. (Langdon, Glen George, Jr.), 388, 393, 405.
Ландау, Хайман Гаршин (Landau, Hyman Garshin), 469, 586.
Ландер, Леон Джозеф (Lander, Leon Joseph), 656.
Ланнон, Вильям Фредерик (Lunnon, William Frederick), 849.
Лапко, Ольга Георгиевна, 4.
Лаплас (де ла Плас), Пьер Симон, маркиз де (Laplace (= de la Place), Pierre Simon, Marquis de), 477.
Ларвала, Самули Кристиан (Larvala, Samuli Kristian), 663.
Ларриви, Жюль Альфонс (Larrivee, Jules Alphonse), 335.
Латинская поэзия, 562.
Латинский квадрат, 22–26, 59, 60, 581, 594.
Лафферти, Джон Дэвид (Lafferty, John David), 754.
Лахтакия, Ахлеш (Lakhtakia, Akhlesh (अखिलेश लखटकिया)), 654.
Ле Борн, Ив Франсуа Андрэ (Le Borgne, Yvan Franccoise André), 890.
Леви, Поль (Lévy, Paul), 513.
Левиальди Гирон, Стефано (Levialdi Ghiron, Stefano), 210.
Легкая булева функция, 301, 325.
Лейбниц, Гроттфрид Вильгельм, Фрайхерр фон (Leibniz, Gottfried Wilhelm, Freiherr von), 548, 564, 567, 568, 575, 900.
Лейзерсон, Чарльз Эрик (Leiserson, Charles Eric), 224.
Лейтон, Роберт Эрик (Leighton, Robert Eric), 583.
Лек, Уве (Leck, Uwe), 826.
Лексикографическая генерация, 411, 424, 427, 433, 434, 571.
  перестановок, 396.
Лексикографический порядок, 21, 101, 111, 113, 183, 240, 254, 328, 331, 355, 358, 370, 377, 385, 424, 437, 439, 446, 449, 463, 472, 485, 489, 499, 518, 520, 530, 541, 548, 552, 555, 556, 561, 567, 568, 572, 575, 695, 738, 816, 837, 898.
Лексикографическое произведение, 49, 65, 543.
Лемер, Деррик Генри (Lehmer, Derrick Henry), 61, 168, 371, 413, 438, 466, 572, 815, 833.
Лемма
  о сжатии, 441.
  Флойда, 7.
Лемпель, Абрам (Lempel, Abraham (למפל אברהם)), 354.
Ленер, Джозеф (Lehner, Joseph), 455, 466.
Ленстра, Хендрик Виллем-мл. (Lenstra, Hendrik Willem, Jr.), 221, 652, 653.

Ленфант, Жак (Lenfant, Jacques), 660.
Лес, 323, 326, 328, 349, 498.
Леске, Мари (Leske, Marie), 896.
Леттманн, Теодор Август (Lettmann, Theodor August), 609, 610.
Леувен, Марк Аврелий Августин ван (Leeuwen, Marcus Aurelius Augustinus van), 832, 851.
Лёббинг, Мартин (Löbbing, Martin), 281, 287, 315.
Ли, Ганг (Li, Gang (= Kenny) (李钢)), 774, 887.
Ли, Гилберт Цзин Джай (Lee, Gilbert Ching Jye (李景傑)), 640.
Ли, Руби Беи-Лох (Lee, Ruby Bei-Loh (李佩露)), 185, 227, 664.
Ли, Честер Чи Юан (Lee, Chester Chi Yuan (李始元) = Chi Lee (李濟)), 302, 303, 758.
Лианг, Франклин Марк (Liang, Franklin Mark), 134.
Лидгейт, Джон (Lydgate, John), 556.
Лидер, Имре (Leader, Imre), 626.
Лимерик, 493.
Лин, Чен-Шанг (Lin, Chen-Shang (林慶祥)), 711.
Линдон, Роджен Конант (Lyndon, Roger Conant), 355.
Линдстрем, Бернт Леннарт Даниэль (Lindström, Bernt Lennart Daniel), 433, 442, 620, 826.
Линеечная функция, 12, 172, 185, 191, 192, 194, 199, 222, 224, 231, 314, 334, 336, 340, 341, 348, 683, 763.
Линейная функция, 738.
Линейное булево программирование, 246.
Линейное время, 82.
Линейное преобразование, 713.
Линейное программирование, 8, 119.
Линейные неравенства, 618.
Линейный блочный код, 327.
Линейный подграф, 586.
Линн, Ричард Джон (Lynn, Richard John), 548.
Линуссон, Ханс Сванте (Linusson, Hans Svante), 824.
Линч, Уильям Чарльз (Lynch, William Charles), 175.
Линь, Билл Цзи Ва (Lin, Bill Chi Wah (林志華 = 林志华)), 304.
Липски, Витольд-мл. (Lipski, Witold, Jr.), 788.
Липшуц, Сеймур Сол (Lipschutz, Seymour Saul), 807.
Лисковец, Валерий Анисимович, 403.
Лист, 525, 533, 870.
Литерал, 78, 253.
Литтлвуд, Дадли Эрнест (Littlewood, Dudley Ernest), 839.
Литтлвуд, Джон Эденсор (Littlewood, John Edensor), 655, 839, 886.
Лихи, Фрэнсис Теодор-мл. (Leahy, Francis Theodore, Jr. (= Ted)), 577.
Ллойд, Эдвард Кейт (Lloyd, Edward Keith), 35, 862.
Лобачевский, Николай Иванович (Лобачевскій, Николай Ивановичъ), 203.
Ловас, Ласло (Lovász, László), 441, 820.
Ловри, Дункан Хамиш (Lawrie, Duncan Hamish), 661.
Логарифм как многозначная функция, 478, 856.
Логика, 154.
  второго порядка, 154.
Ложное заключение, 73.
Лоизоу, Георгиос (Loizou, Georghios (Λοΐζου, Γεώργιος)), 829.
Лойд, Сэм (Loyd, Sam), 16.
Лойд, Сэмюэл (Loyd, Samuel), 656, 748.
Лойд, Уолтер (Loyd, Walter (= “Sam Loyd, Jr.”)), 19.
Локальные изменения, 151.
Лондон, Джон (London, John), 557.
Лоренц, Макс Отто (Lorenz, Max Otto), 839.
Лоуренс, Джордж Мелвин (Lawrence, George Melvin), 343, 766.
Лукакис, Эммануэль (Loukakis, Emmanuel (Λουκάκης, Μανώλης)), 674.
Лукашевич, Ян (Łukasiewicz, Jan), 197, 233.
Луллий, Раймунд (Llull, Ramon (= Lullus, Raimundus)), 556, 574.
Лупанов, Олег Борисович, 139, 141, 160, 632, 638, 640.
Луц, Рюдигер Карл (Lutz, Rüdiger Karl), 684.
Луч, 417.
Лучак, Мальвина Джоанна (Luczak (= Łuczak), Malwina Joanna), 882.
Лушар, Гай (Louchard, Guy), 513.
Льювен, Марк ван (Leeuwen, Marc van), 10.
Льюис, Гарри Рой (Lewis, Harry Roy), 610.
Лэнгфорд, Чарльз Дадли (Langford, Charles Dudley), 26, 27, 578.
Лю, Чао-Нинг (Liu, Chao-Ning (劉兆寧)), 416.
Лю, Юан (Lu, Yuan (吕原)), 703.
Люка, Джоан Мари (Lucas, Joan Marie), 506.
Люка, Франсуа Эдуард Анатоль (Lucas, François Edouard Anatole), 826.
Люнебург, Хайнц (Lüneburg, Heinz), 808.
Ляо, Хех-Тян (Liaw, Heh-Tyan (廖賀田)), 711.
Ляхдесмяки, Гарри (Lähdesmäki, Harri), 626.

Магическая маска, 173, 180, 224, 655, 658, 660, 662, 664, 665, 678, 687.
Магический квадрат, 58, 640.
Мадр, Жан Кристоф (Madre, Jean Christophe), 697, 708, 745, 751.
Маеда, Ватару (Mayeda, Wataru (前田渡)), 573.
Мажоризация, 51, 120, 133, 193, 468, 839.
Мажоритарный элемент, 70.
Майерс, Юджин Уимберли-мл. (Myers, Eugene Wimberly, Jr.), 889.
Майкрофт, Алан (Mycroft, Alan), 185.
Майнель, Кристоф (Meinel, Christoph), 713.
Майорана, Джеймс Энтони (Maiorana, James Anthony), 356, 357.

Майр, Эрнст Вильгельм (Mayr, Ernst Wilhelm), 617.
Мак-Дональд, Питер С. (MacDonald, Peter S.), 398.
Мак-Каллох, Уоррен Стергис (McCulloch, Warren Sturgis), 101.
Мак-Карти, Дэвид (McCarthy, David), 418.
Мак-Кей, Брендон Дамьен (McKay, Brendan Damien), 580, 793, 826.
Мак-Кей, Джон Кейт Стюарт (McKay, John Keith Stuart), 446.
Мак-Келлар, Арчи Чарльз (McKellar, Archie Charles), 148, 150, 162.
Мак-Класки, Эдвард Джозеф-мл. (McCluskey, Edward Joseph, Jr.), 79.
Мак-Клинток, Уильям Эдвард (McClintock, William Edward), 343.
Мак-Крейви, Эдвин Паркер-мл. (McCravy, Edwin Parker, Jr.), 398.
Мак-Крейни, Джадсон Шаста (McCranie, Judson Shasta), 675.
Мак-Кьюн, Уильям Уолкер-мл. (McCune, William Walker, Jr.), 614.
Мак-Манус, Кристофер Декормис (McManus, Christopher DeCormis), 60.
Мак-Миллан, Кеннет Лахлин (McMillan, Kenneth Lauchlin), 700.
Мак-Мэган, Перси Александер (MacMahon, Percy Alexander), 470, 471, 485, 876, 882.
Мак-Мюллен, Кертис Траси (McMullen, Curtis Tracy), 736.
Мак-Нейш, Гаррис Франклин (MacNeish, Harris Franklin), 23.
Макино, Казухиса (Makino, Kazuhisa (牧野和久)), 621, 728.
Маклин, Ян Синклер (McLean, Iain Sinclair), 557.
Маколей, Френсис Соверби (Macaulay, Francis Sowerby), 427, 443, 819.
Максимальная клика, 324.
Максимальная цепочка, 534.
Максимальное независимое множество, 56, 67, 674.
Максимальное правдоподобие, 328.
Максимальные перекрывающиеся семейства, 115.
Максимальные элементы, 324.
Максимальный
  планарный граф, 61.
  подкуб, 79.
  элемент, 797.
Малгойрес, Реми (Malgouyres, Rémy), 682.
Малдер, Генри Мартин (Mulder, Henry Martyn), 614.
Малфатти, Джованни Франческо Джузеппе (Malfatti, Giovanni Francesco Giuseppe), 834.
Манди, Питер (Mundy, Peter), 373.
Манн, Вильям Фредерик (Mann, William Fredrick), 680.
Манн, Генри Бертольд (Mann, Henry Berthold), 579.
Мантель, Виллем (Mantel, Willem), 352.
Маргенштерн, Морис (Margenstern, Maurice), 204.
Мария, Святая (Mary, Saint (Ἅγια Μαρία Θεοτόκος, Παναγία, Παρθένος)), 562.
Маркерт, Жан-Франсуа (Marckert, Jean-François), 513.
Маркизова, Тамара (Marcisová, Tamara), 614.
Марковски, Джордж (Markowsky, George), 605, 624, 872.
Марковский процесс, 548.
Маркс, Адольф (Marx, Adolph (= Arthur = Harpo)), 903.
Марсо, Марсель (Marceau (= Mangel), Marcel), 903.
Мартин, Монро Харниш (Martin, Monroe Harnish), 174, 357.
Мартинелли, Андре (Martinelli, Andrés), 720.
Маруока, Акира (Maruoka, Akira (丸岡章)), 292, 319.
Маршалл, Альберт Уолдрон (Marshall, Albert Waldron), 840.
Маска, 181, 183, 186, 218, 219, 240.
Маска: битовый шаблон с 1 в ключевых позициях, 173.
Матон, Рудольф Антон (Mathon, Rudolf Anton), 580.
Матрица
  Адамара, 696.
  инциденций, 54, 57, 67, 891.
  перестановки, 40, 219, 281.
  смежности, 40, 46, 48, 63, 66, 153, 164, 194, 233, 313, 593, 598, 705, 891.
Мафут, Халед (Maghout, Khāled (خالد ماغوط)), 303.
Мацумото, Макото (Matsumoto, Makoto (松本眞)), 822.
Мацунага, Ёшисуке (Matsunaga, Yoshisuke (松永良弼)), 475, 566.
Мачаруло, Лука (Macchiarulo, Luca), 694.
Машина Тьюринга, 302, 680.
Медиана, 12, 88, 130, 156, 186, 242, 253, 309, 361, 668, 668.
  пяти элементов, 93, 97, 114, 118.
  семи значений, 165.
Медианная цепочка, 164.
Медианные метки, 92–101, 615.
Медианный граф, 92–101, 616.
Медицина, 554, 557.
Мезеи, Йорг Эстебан (Mezei, Jorge Esteban (= György István)), 119.
Мейер, Альберт Рональд да Сильва (Meyer, Albert Ronald da Silva), 154, 155, 647.
Мейеровиц, Аарон Давид (Meyerowitz, Aaron David), 116, 616.
Мейнерт, Элисон (Meynert, Alison), 580.
Мейснер, Отто (Meißner, Otto), 842.

Менон, Ваирелил Вишванат (Menon, Vairelil Vishwanath), 589.
Мерингер, Маркус Рейнхард (Meringer, Markus Reinhard), 594.
Меро, Ласло (Mérő, László), 689.
Мерсенн, Марен (Mersenne, Marin), 551, 552, 567, 574.
Метка, 445.
Метод
  Монте-Карло, 791.
  Ньютона, 479, 858.
  с возвратом, 565.
  седловой точки, 453, 475–482, 494, 862.
Метрическая стопа, 550, 563.
Метрополис, Николас Константин (Metropolis, Nicholas Constantine (Μητρόπολης, Νικόλαος Κωνσταντίνου)), 624.
Миерс, Чарльз Роберт (Miers, Charles Robert), 778.
Мииллер, Генри Седвик (Miiller, Henry Sedwick), 90.
Миками, Ёсио (Mikami, Yoshio (三上義夫)), 553.
Милето, Франко (Mileto, Franco), 606.
Миллер, Джеффри Чарльз Перси (Miller, Jeffrey Charles Percy), 176.
Миллер, Джоан Элизабет (Miller, Joan Elizabeth), 416, 436.
Миллер, Дональд Джон (Miller, Donald John), 591.
Миллс, Бартон Э. (Mills, Burton E.), 601, 605.
Милн, Стефен Карл (Milne, Stephen Carl), 489.
Милнор, Джон Уиллард (Milnor, John Willard), 115, 612.
Мильтерсен, Питер Бро (Miltersen, Peter Bro), 193.
Минато, Шин-ичи (Minato, Shin-ichi (湊真一)), 293, 303, 304, 326, 737, 745, 750, 752, 754.
Минимальная память, 130–135, 156, 157, 633.
Минимальное доминирующее множество, 325.
Минимальное покрытие, 56.
Минимальное решение, 300.
Минимальный элемент, 324, 797.
  подмассива, 234.
Минимум, 56, 468.
Минник, Роберт Чарльз (Minnick, Robert Charles), 104, 619.
Мински, Марвин Ли (Minsky, Marvin Lee), 236.
Минус с точкой, 12.
Минхардт, Кристина (Кика) Магдалена (Mynhardt, Christina (Kieka) Magdalena), 749.
Мирвольд, Венди Джоанна (Myrvold, Wendy Joanne), 580, 796.
Мисра, Джаядев (Misra, Jayadev (ଜୟଦେବ ମିଶ୍ର)), 598, 757.
Мисюревич, Михал (Misiurewicz, Michał), 842.
Миттаг-Леффлер, Магнус Геста (Mittag-Leffler, Magnus Gösta (= Gustaf)), 570.
Митчелл, Кристофер Джон (Mitchell, Christopher John), 354.
Мичели, Джованни де (Micheli, Giovanni De), 152.
Многобайтная обработка, 184.
Многобайтное вычитание, 229.
Многобайтное сложение, 184.
Многобайтные min и max, 230.
Многогбайтное кодирование, 240.
Многогранник, 426, 442, 535.
Многоканальный вывод, 135.
Многосвязная структура данных, 85.
Многоуровневый логический синтез, 152.
Многофазная сортировка, 549.
Множество комбинаций, *см.* Семейство множеств, 294.
Модальная логика, 197.
Модули в сети, 308.
Модуль
  инвертора, 98.
  компаратора, 98.
Модульная арифметика, 578.
Модульный m-арный код Грея, 353, 774.
Модульный десятичный код Грея, 347.
Модульный код Грея, 771, 794.
Модульный универсальный цикл, 802.
Модулярная арифметика, 247.
Мозаика, 203, 234.
Мозер, Лео (Moser, Leo), 481, 595, 853, 855.
Момент, 490.
Моменты распределения, 861.
Монадическая логика: логика только с унарными операторами, 154.
Монеты, 124, 463.
Монмор, Пьер Ремон де (Montmort, Pierre Rémond de), 568, 576.
Мономиальная симметричная функция, 568, 839.
Мономино, 296, 322.
Монотонная булева функция, 114, 300, 310, 325, 540.
Монотонная булева цепочка, 163.
Монотонная самодуальная булева функция, 96, 116.
Монотонная самодуальная функция, 104, 115.
Монотонная функция, 112, 155, 163, 303.
Монотонно убывающая функция, 733.
Монотонность, 79, 88, 104, 108, 111, 123, 273, 301, 312, 326, 344, 443, 518, 601, 616.
Монус, 12.
Мор, Моше (Mor, Moshe (משה מור)), 800, 808.
Моралес, Линда (Morales, Linda), 801.
Морган, Огастес де (Morgan, Augustus de), 75, 107, 408, 466.
Моргенштерн, Оскар (Morgenstern, Oskar), 616, 695.
Морзе, Гарольд Келвин Марстон (Morse, Harold Calvin Marston), 250, 305, 696.
Морреаль, Эженио (Morreale, Eugenio), 604.

Моррис, Скот Андерсон (Morris, Scot Anderson), 61, 786.
Моррис, Эрнест (Morris, Ernest), 373.
Мортон, Гай Макдональд (Morton, Guy Macdonald), 666.
Мост, 523, 541, 544, 894.
Моханрам, Картик (Mohanram, Kartik (ಕಾರ್ತಿಕ ಮೊಹನ್ನಾಮ)), 703.
Мохар, Боян (Mohar, Bojan), 893.
Моцкин, Теодор Сэмюэль (Motzkin, Theodor (= Theodore) Samuel (תיאודור שמואל מוצקין)), 487, 848, 861.
Муавр, Абрахам де (Moivre, Abraham de), 509, 568, 569, 570, 576.
Муди, Джон Кеннет Монтегю (Moody, John Kenneth Montague (= Ken)), 233.
Музыка, 549, 551, 559, 567.
Музыкальный граф, 68.
Мультигипергграф, 593.
Мультиграф, 32, 39, 42, 62, 64, 67, 597.
Мультилинейное представление, 76, 106, 161, 273, 601, 691, 752.
Мультимножество, 38, 369, 371, 397, 409, 552, 802, 815.
Мультиномиальная теорема, 568.
Мультиномиальный коэффициент, 358, 397.
Мультиплексор, 138, 254, 278.
Мультиразбиения, 485, 496.
Мультисемейство, 326.
Мультисочетание, 409, 426, 433, 442, 445, 448, 554, 561, 804.
Мун, Джон Уэсли (Moon, John Wesley), 595.
Мунданос, Константинос (Moundanos, Konstantinos (= Dinos; Μουνδάνος, Κωνσταντίνος)), 703.
Мунро, Джеймс Ян (Munro, James Ian), 193.
Мур, Д. Стротер (Moore, J Strother), 70.
Мур, Рональд Вильямс (Moore, Ronald Williams), 602.
Мур, Эдвард Форрест (Moore, Edward Forrest), 753.
Мур, Элиаким Гастингс (Moore, Eliakim Hastings), 581.
Мурасаки Сикибу (Murasaki Shikibu (= Lady Murasaki, 紫式部)), 565.
Мурога, Сабуро (Muroga, Saburo (室賀三郎)), 103, 104, 603, 618, 619, 621, 625.
Мусор, 193.
Мусульманская математика, 554.
Мюллер, Дэвид Юджин (Muller, David Eugene), 176, 628, 634.
Мюрхед, Роберт Франклин (Muirhead, Robert Franklin), 839.

Надежность, 107.
Наибольшая нижняя граница, 120, 533.
Наибольшее независимое множество, 57.
Наименьшая верхняя граница, 120, 533.
Наименьшее вершинное покрытие, 68, 602.
Наименьшее остовное дерево, 306.
Наименьшие части, 467.
Найлан, Майкл (Nylan, Michael), 548.
Найсли, Томас Рэй (Nicely, Thomas Ray), 656.
Накагава, Нориюки (Nakagawa, Noriyuki), 573.
Напарник, 539.
Нараяна Пандита, сын Нрсимха (Nārāyaṇa Paṇḍita, son of Nṛsiṁha (नारायण पण्डित, नृसिंहस्य पुत्रः)), 370, 549, 553, 561, 569, 575, 781, 899.
НДР, 302.
  в сравнении с БДР, 320, 739, 740.
Небески, Ладислав (Nebeský, Ladislav), 614.
Негадесятичная система счисления, 205.
Неевклидова геометрия, 202.
Независимое множество, 55, 57, 233, 251, 273, 294, 733.
Независимость, 55.
Нейман, Джон фон (Neumann, John von (= Margittai Neumann János)), 616, 695.
Нейман, Эржи (Neyman, Jerzy), 770.
Нейронная сеть, 101.
Немет, Эвелин Холлистер Пратт (Nemeth, Evelyn (= Evi) Hollister Pratt), 766.
Немхаузер, Джордж Ланн (Nemhauser, George Lann), 607.
Необходимость, 233.
Неориентированные циклы, 398.
Непересекаемость, 228, 533.
Непересекающееся разложение, 147.
Непересекающиеся графы, 47.
Непересекающиеся хорды, 502, 530.
Неполная гамма-функция, 477.
Неполный куб, 118.
Непомеченные свободные деревья, 521, 541.
Неравенство
  Адамара, 618.
  треугольника, 35, 39.
Неразделительная дизъюнкция, 73.
Несмещенное округление, 229, 667.
Нетто, Отто Эрвин Иоганн Ойген (Netto, Otto Erwin Johannes Eugen), 570, 571, 572.
Нечетная перестановка, 374, 791.
Нечетное произведение, 49, 65.
Нечипорук, Эдуард Иванович, 640.
Нешетржил, Ярослав (Nešetřil, Jaroslav), 590.
Неявные структуры данных, 198–206.
Неявный граф, 275.
Ниббл, или полубайт: 4-битовая величина, 240.
Нивергельт, Юрг (Nievergelt, Jürg), 573.
Нигматуллин, Рошаль Габдулхаевич, 625.
Нидхам, Ноэль Джозеф Теренс Монтгомери (Needham, Noel Joseph Terence Montgomery (李約瑟)), 548.
Нижние границы комбинационной сложности, 152.
Нижняя граница, 138.
Нийон, Герман (Nijon, Herman), 398.
Никольская, Людмила Николаевна (Nikolskaia, Ludmila Nikolaievna), 716.

Никольская, Мария Николаевна (Nikolskaia, Maria (= Macha) Nikolaievna), 716.
Ним, 166, 221.
Нип: 2-битовая величина, 240.
Новра, Генри (Novra, Henry), 772.
Ноде, Филипп-мл. (Naudé, Philippe, junior), 450.
Нойман, Франтишек (Neuman, František), 545.
Ноотен, Баренд Адриан Анске Йоханнес ван (van Nooten, Barend Adrian Anske Johannes), 549.
Нордстром, Алан Уэйн (Nordstrom, Alan Wayne), 359.
Нордхаус, Эдвард Альфред (Nordhaus, Edward Alfred), 594.
Нормализация, 628.
  булевой функции, 128.
Нормальная булева функция, 128, 131, 632.
Нормальная форма Смита, 596.
Нормальная цепочка, 143.
Нормальные числа, 778.
Нуклеотид, 574.
Нулевой граф, 46, 64.
Нулевой случай, 561.
Нуль-один принцип, 93, 223.
Ньенхьюз, Альберт (Nijenhuis, Albert), 390, 415, 467, 541, 573.

Обер, Жак (Aubert, Jacques), 771.
Обертывающий ряд, 457, 466, 496, 857.
Облексная схема, 813.
Облексный метод, 437, 818.
Облексный порядок, 417–421, 424, 436, 899.
Облексный путь Грея, 439.
Обмен, 649.
  битов, 177.
  конечных элементов, 438.
  первого элемента, 388, 424, 439.
  соседних переменных, 284.
  соседних уровней, 284.
Обменная поразрядная сортировка, 673.
Обобщенное число
  Белла, 491, 495.
  Стирлинга, 847.
Обобщенные числа
  Каталана, 536.
  Стирлинга, 492.
Обобщенный консенсус, 110.
Обобщенный тор, 68.
Обозначения, 11, 908.
Обратная импликация, 73.
Обратная перестановка, 394, 398.
Обратная функция, 332, 360.
Обратнопрямой порядок обхода, 528, 544.
Обратный порядок обхода, 435, 498, 501, 505, 506, 531, 676, 865, 870.
Обратный солексный порядок, 377, 381, 384, 396, 781.
Обращение
  битов, 357, 361, 662.
  перестановки, 506.
  порядка битов, 176.
  ряда, 808.
  степенного ряда, 808.
Обращения к памяти, 20, 459, 528.
Обрезка и прививка, 508, 532.
Обхват, 63, 68.
  графа, 35, 62, 584.
Обход, 329, 528.
  дерева, 463, 528.
Объединение, 47, 320.
  мультимножеств, 326.
Объединенное разбиение, 464.
Ограничение, 707.
  булевой функции, 259, 308, 702.
Ограниченно растущие строки, 327, 472, 488, 849, 854, 861, 869, 901.
Ограниченные композиции, 426, 439, 902.
Ограничивающая кривая, 212.
Одиночный узел, 284, 722.
Одлызко, Эндрю Майкл (Odlyzko, Andrew Michael), 454.
Однородная последовательность, 810.
Однородность, 437, 814, 817.
Однородный гиперграф, 426.
Однородный граф, 543.
Однородный полином, 443.
Однородный порядок, 437.
Одночлен, 443.
Озанам, Жак (Ozanam, Jacques), 22, 26, 27.
Оккам, Уильям (Уильям из Оккама) (Ockham, William of (= Guilielmus ab Occam)), 75.
Окно, 290.
Округление, 165, 199.
  к четному числу, 667.
Окружность, 212.
Октакод, 359.
Октаэдрическая группа, 785.
Октет: 64-битовая величина, 240.
Окуно, Хироши (Okuno, Hiroshi “Gitchang” (奥乃博)), 745.
Олив, Глория (Olive, Gloria), 814.
Олкин, Ингрем (Olkin, Ingram), 840.
Ольвер, Фрэнк Вильям Джон (Olver, Frank William John), 482.
Онегин, Евгений, 493.
Оптимальное булево вычисление, 131.
Оптимальность, 519.
Оптимизация порядка переменных, 282.
Оптическое распознавание символов, 208.
Орд-Смит, Ричард Альберт Джеймс (Ord-Smith, Richard Albert James (= Jimmy)), 382, 387, 400, 782.
Оре, Ойстейн (Ore, Øystein), 10, 65.
Ориентированное бинарное дерево, 637.
Ориентированное дерево, 211, 488, 519, 857.
Ориентированное расстояние, 39.
Ориентированное соединение, 47.

Ориентированные деревья, количество, 519.
Ориентированные остовные деревья, 542, 893.
Ориентированный ациклический граф, 53.
Ориентированный гиперграф, 67.
Ориентированный граф, 31, 38–43, 65, 68, 543, 544.
  Кейли, 68.
  случайный, 45.
Ориентированный лес, 200, 519, 540.
Ориентированный остовный путь, 62.
Ориентированный путь, 39, 64, 193, 298.
Ориентированный тор, 404, 893.
Ориентированный цикл, 39, 63, 64, 305, 324.
Ортогональность, 320, 362.
Ортогональные веторы, 59.
Ортогональные латинские квадраты, 22–26.
Ортогональные строки, 59.
Ортогональный массив, 59, 60, 582.
Основная теорема Гильберта, 443.
Остаток, 321.
  по модулю $2^n - 1$, 176.
Остергард, Патрик Ральф Йохан (Östergård, Patric Ralf Johan), 749.
Остин, Ричард Брюс (Austin, Richard Bruce), 714.
Остовные деревья, 301, 306, 521, 541, 543, 573.
  перечисление, 543.
Остовный подграф, 251.
Отели Лас-Вегаса, 85.
Относительное дополнение, 518.
Отношение
  промежуточности, 117.
  эквивалентности, 472, 488, 581.
Отображение, 182.
  трех элементов в двухбитовые коды, 195.
Отражение, 531, 679, 870.
Отрезок, 91, 92.
Отрицание, 221.
  дизъюнкции, 73.
  импликации, 73.
  конъюнкции, 73.
Оттингер, Людвиг (Oettinger, Ludwig), 842.
Офман, Юрий Петрович, 637, 665.
Оффорд, Альберт Сирил (Offord, Albert Cyril), 886.
Оцифровка, 212.
Очередь, 797.
  с приоритетами, 202.
Ошибки, 573.

Пайк, Роберт К. (Pike, Robert C.), 125.
Пак, Игорь Маркович (Pak, Igor Markovich), 791, 792, 831.
Палиндром, 60, 609.
Панхольцер, Алоиз (Panholzer, Alois), 881, 882.
Пападимитриу, Христос Харилаос (Papadimitriou, Christos Harilaos (Παπαδημητρίου, Χρίστος Χαριλάου)), 11, 674.
Парабола, 212, 237, 685.
Параллельная обработка подслов, 184.
Параллельные вычисления, 118, 137, 405, 637, 784.
Параллельные прямые, 59.
Паркер, Эрнест Тильден (Parker, Ernest Tilden), 24, 26, 580.
Паркин, Томас Рэнделл (Parkin, Thomas Randall), 656.
Парковка, 873.
Паросочетание, 57, 502, 530.
Пары Лэнгфорда, 19, 58, 322.
Паскаль, Эрнесто (Pascal, Ernesto), 413.
Патент, 333, 663, 692.
Патерсон, Кеннет Грехэм (Paterson, Kenneth Graham), 354.
Патерсон, Майкл Стюарт (Paterson, Michael Stewart), 156, 158, 188, 671, 672.
Паттенхам Георг и/или Ричард (Puttenham, George and/or Richard), 566, 575.
Паттерсон, Николас Джеймс (Patterson, Nicholas James), 627.
Пейдж, Лоувелл Д. (Paige, Lowell J.), 24, 26.
Пейли, Раймонд Эдвард Алан Кристофер (Paley, Raymond Edward Alan Christopher), 223, 361, 655, 761.
Пейн, Вильям Гаррис (Payne, William Harris), 416, 436.
Пейперт, Сеймур Обри (Papert, Seymour Aubrey), 236.
Пелед, Ури Натан (Peled, Uri Natan (אורי נתן פלד)), 728.
Пентагрид, 204, 235.
Пеппердайн, Эндрю Говард (Pepperdine, Andrew Howard), 800.
Перевернутая пирамида, 808.
Переворот, 381.
Перекрестно пересекающиеся множества, 440.
Перекрестный порядок, 429–433, 441, 826.
Перекрывающиеся поддеревья, 125, 303.
Перекрывающиеся семейства множеств, 115.
Перельман, Григорий Яковлевич, 577.
Переменная
  вершины, 43.
  дуги, 43.
Перенос, 158, 184, 185, 191, 257, 330, 636, 667, 668, 731, 779.
Переносимость, 42, 171.
Переполнение, 673.
Пересечение, 320, 576.
  границы, 212.
Перестановка, 222, 279, 318, 445, 488, 539, 551, 570, 571.
  $\phi(k)$, 381.
  байтов, 219.
  битов, 178, 191, 219.
  Грея, 361.
  индексных цифр, 225.
  мультимножества, 422, 569.
  “на месте”, 357, 361.
  соседних элементов, 438.

Перестановки мультимножества, 394, 411, 438, 450, 807, 842.
Перестановочная сеть, 178, 226.
Перестановочный ориентированный граф, 63.
Переупорядочение, 699.
Перец, Арам (Perez, Aram), 220.
Перечисление, 329.
  булевых функций, 104.
Перманент, 149.
Пермутаэдр, 535.
Перпендикулярные прямые, 203.
Перрин, Франсуа Оливье Рауль (Perrin, François Olivier Raoul), 695, 714.
Петерсен, Юлиус Питер Кристиан (Petersen, Julius Peter Christian), 23, 33, 589.
Петерсон, Уильям Уэсли (Peterson, William Wesley), 220.
Петля, 32, 38, 39, 64, 521, 523, 593.
Петрарка, Франческо (Petrarca, Francesco (= Petrarch)), 493.
Пехушек, Джозеф Дениэль (Pehoushek, Joseph Daniel), 114.
Печарски, Марцин Пётр (Peczarski, Marcin Piotr), 797.
Пианино, 418, 439.
Пиковер, Клиффорд Алан (Pickover, Clifford Alan), 654.
Пиксель, 44, 52, 207, 238.
Пиксельная алгебра, 208.
Пингала, Акарья (Piṅgala, Ācārya (आचार्य पिङ्गल)), 548–550, 574.
Пиппенджер, Николас Джон (Pippenger, Nicholas John), 625, 640.
Пирамида, 116, 199.
Пиррихий, 550.
Пирс, Чарльз Сантьяго Сандерс (Peirce, Charles Santiago Sanders), 73, 74, 77, 474, 607.
Пистолет Дюмона, 318, 734.
Питман, Джеймс Вильям (Pitman, James William), 496, 848, 860.
Питтвей, Майкл Ллойд Виктор (Pitteway, Michael Lloyd Victor), 213.
Питтель, Борис Гершонович, 484.
Питтс, Уолер Гарри (Pitts, Walter Harry), 101.
Планарность, 58.
Планарный граф, 33, 34, 37, 44, 68, 275, 584, 591.
Пласс, Майкл Фредерик (Plass, Michael Frederick), 114.
Платон = Аристокл, сын Аристона (Plato = Aristocles, son of Ariston (Πλάτων = Ἀριστοκλῆς Ἀρίστωνος)), 498.
Плезантс, Питер Артур Барри (Pleasants, Peter Arthur Barry), 849.
Плещински, Стефан (Pleszczyński, Stefan), 796.
Пловер, Кори Майкл (Plover, Corey Michael), 144, 639.
Плотная упаковка, 8.
Побитовое
  И, 42.
  ИЛИ, 74.
Поверхность Римана, 856.
Поворот, 505.
Поглощение, 301.
Подбрасывание монет, 496.
Подграф, 32, 37.
  гиперкуба, 117.
Подкуб, 79, 109, 160, 234, 302, 439, 443, 624, 638, 736.
Подлес, 349, 366.
Подмножество, 183, 330.
Подстановка
  констант вместо переменных, 259.
  функций вместо переменных, 309.
Подсчет
  единиц, 175.
  решений, 246, 247.
Подтаблица, 260, 712.
Подфункция, 256.
Познер, Эдвард Чарльз (Posner, Edward Charles), 582.
Поиск
  в глубину, 43, 63, 258, 310, 643.
  в таблице, 666.
    путем сдвига, 189, 240, 657, 669, 692.
  в ширину, 95, 310, 673, 679, 887.
  вглубь, 887.
Пойа, Гёрги (Pólya, György (= George)), 15, 37, 655, 829, 839.
Показательное множество, 746.
Покрытие, 29, 55, 57, 120, 468, 489, 517.
  в решетке, 533.
  ребер, 366.
Поле, 74, 221, 361.
  Грея, 361.
Полимино, 296, 322.
Полином, 107, 227, 252.
  доступности булевой функции, 252.
  надежности, 246, 252, 306, 307, 314.
  Пирса, 852.
Полиномиальный идеал, 443.
Полиномиальный коэффициент, 567.
Полиномы
  Фибоначчи, 239.
  Чебышева, 688, 892.
Полиньяк, Камилл Арман Жюль Мари де (Polignac, Camille Armand Jules Marie de), 35.
Полиэдрическая комбинаторика, 8.
Поллак, Гёрги (Pollák, György), 765.
Полная дизъюнктивная
  нормальная форма, 77, 107.
  форма, 624.
Полная конъюнктивная нормальная форма, 77.
Полное бинарное дерево, 108, 199, 653, 808.
Полное иерархическое тернарное дерево, 548.
Полное упорядочение, 394.

Полностью детализированная таблица истинности, 253, 306, 307, 321.
Полный k-дольный граф, 37, 62, 67.
Полный r-однородный гиперграф, 54.
Полный биграф, 164, 586, 648, 891.
Полный бинарный луч, 61.
Полный граф, 32, 47, 64, 542, 556, 589, 765, 826.
Полный двудольный r-однородный гиперграф, 67.
Полный двудольный граф, 37, 559.
Полный ориентированный граф, 38.
Полный сумматор, 136, 157, 234, 632.
Полный тернарный луч, 584.
Положительная пороговая функция, 101.
Положительная функция, 601.
Положительно полуопределенная матрица, 890.
Полумодулярность, 533.
Полуопределенное программирование, 8.
Полупомеченное дерево, 488.
Полусумматор, 136, 233.
Получение по рангу
  бинарного дерева, 573.
  кортежа, 357.
  объекта, 332, 348, 405, 548, 574.
  перестановки, 552, 562.
  разбиения, 468.
  разбиения множества, 488.
  сочетания, 435, 436, 437.
  строки вложенных скобок, 510, 537.
Поль, Вольфганг Якоб (Paul, Wolfgang Jakob), 163.
Поль, Ира Шелдон (Pohl, Ira Sheldon), 21.
Польская запись, 88.
Померанц, Карл (Pomerance, Carl), 860.
Помеченные объекты, 856.
Поразрядная сортировка, 486.
Пороговая функция, 101, 104, 123, 254, 307, 308, 315, 519, 634, 728, 885.
  Фибоначчи, 119, 120, 156, 307.
Пороговые функции пороговых функций, 103, 119.
Порожденный подграф, 65.
Порядковый идеал, 442, 620, 871.
Порядок, 32, 38, 68, 789, 846.
  двери-вертушки, 437.
  доминирования, *см.* Мажоризация, 120, 468.
  кода Грея, 827.
  латинского квадрата, 59.
  органных труб, 282, 313, 422, 578, 699, 721, 792, 800, 888.
Последовательно-параллельные графы, 524, 541, 883.
  представление в виде дерева, 525.
Последовательное представление БДР, 246, 305, 308.
Последовательности со сдвигом регистра, 351–357.
Последовательность
  степеней, 50, 66, 69.
    в прямом порядке обхода, 532.
  Фибоначчи, 549.
    обобщенная, 549.
  Чейза, 121.
Последовательные алгоритмы, 9.
Последовательные целые числа, 464.
Пост, Эмиль Леон (Post, Emil Leon), 89, 94, 611.
Пост, Ян Томас (Post, Ian Thomas), 751.
Построчно-эшелонный вид, 806.
Поток Грея, 364.
Поэзия, 493, 548, 556, 584.
Прёмель, Ганс Юрген (Prömel, Hans Jürgen), 797.
Правило резолюций, 605.
Правильный многогранник, 442.
Пратт, Воган Рональд (Pratt, Vaughan Ronald), 155, 224, 228, 661, 665, 670, 686.
Предок, 864.
Предпростая строка, 356, 368.
Представление
  в виде количества частей, 448, 463, 488.
  графа, 233.
  дерева в виде бинарного, 435.
  множества, 233.
  произвольной перестановки, 227.
  трех состояний двумя битами, 195.
Преобразование
  Адамара, 337, 600, 761, 762.
  Меллина, 452, 465.
  Уолша, 337, 362.
  Фурье, дискретное, 357.
Препарата, Франко Паоло (Preparata, Franco Paolo), 628, 634.
Престе, Жан (Prestet, Jean), 564, 575.
Префикс, 157, 355.
  строки, 163.
Приближение Стирлинга, 477, 479, 481, 859.
Приведение к БДР, 258.
Приведенная бинарная диаграмма решений, 243.
Приведенное медианное множество, 98.
Примитивный полином, 352, 761.
  по модулю 2, 699.
Принс, Джирт Калеб Эрнст (Prins, Geert Caleb Ernst), 571.
Принцип включения и исключения, 335, 456, 457, 466, 605, 741, 853, 859, 897.
Приоритет, 76.
Приоритетная очередь, 202.
Притчард, Пол Эндрю (Pritchard, Paul Andrew), 656.
Прован, Джон Скотт (Provan, John Scott), 606.
Проверка
  двудольности, 43.
  достижимости, 887.
  принадлежности диапазону, 230.
  связности графа, 887.

эквивалентности булевых функций, 267, 305.
Программа автоматического доказательства теорем Otter, 614.
Программируемая логическая матрица, 78.
Продингер, Гельмут (Prodinger, Helmut), 657, 853, 880, 881, 882.
Проективная плоскость, 581, 594.
Проецирующая функция, 74, 320, 321, 616.
Произведение, 270.
Кронекера, 590, 891.
перестановок, 377.
Произведения графов, 65.
Производная, 252, 306, 441, 545.
Производящая функция, 225, 227, 246, 251, 300, 306, 307, 397, 437, 450, 455, 464, 466, 471, 475, 492, 508, 537, 539, 714, 715, 725, 744, 747, 748, 755, 799, 816, 862, 883, 901.
Дирихле, 658.
на основе НДР, 738.
Прокоп, Гаральд (Prokop, Harald), 224.
Пропорциональный граф, 69.
Пропуск блоков перестановок, 382.
Просвет, 464.
Просеивание, 287, 295, 715.
частичное, 289.
Проскуровски, Анджей (Proskurowski, Andrzej), 504, 537.
Просодия, 548.
Простая импликанта, 78, 89, 97, 109, 116, 119, 122, 123, 160, 234, 300, 303, 645, 649, 704, 728.
Просто связный компонент, 211.
Простое число, 140, 168, 223, 465.
Простой дизъюнкт, 123, 160, 325.
Простой ориентированный граф, 38, 66, 589, 590.
Простой путь, 298.
Пространственно-временной компромисс, 261.
Пространство бинарных векторов, 440.
Простые изменения, 372, 387, 392, 395, 397, 403, 417.
Простые импликанты функции мажоризации, 619.
Простые строки, 355, 368.
Противоречие, 73.
Профиль, 276, 308, 310, 318, 704.
Пруссе, Гара (Pruesse, Gara), 798.
Прямая сумма
графов, 48, 65.
двух матриц, 47.
Прямое произведение, 543.
графов, 49, 65.
матриц, 66, 891.
Прямой порядок обхода, 200, 381, 383, 414, 435, 498, 501, 502, 507, 508, 531, 573, 676, 811, 870, 888.
Прямообратный порядок, 544.
обхода, 528.
Прямоугольник Дюрфи, 830.
Псевдодополнение, 873.
Псевдослучайные биты, 368.
Птолемей, Клавдий, из Александрии (Ptolemy, Claudius, of Alexandria (Πτολεμαῖος Κλαύδιος ὁ Ἀλεξανδρινός)), 563.
Пуанкаре, Жюль Энри (Poincaré, Jules Henri), 577.
Пуансо, Луи (Poinsot, Louis), 443, 574, 827.
Пуаро, Эркюль (Poirot, Hercule), 38, 902.
Пуассон, Симон Дени (Poisson, Siméon Denis), 833.
Пудлак, Павел (Pudlák, Pavel), 846.
Пузырьковая сортировка, 371.
Пуркисс, Генри Джон (Purkiss, Henry John), 358.
Пурнадер, Рузбен (Pournader, Roozbeh (روزبه ورنادر)), 690.
Пустая строка, 883.
Пустое множество, 561.
Пустой граф, 891.
Путеанус, Эрикус (Puteanus, Erycius (= de Putte, Eerrijk)), 562, 564, 575, 900.
Путцолу, Жанфранко (Putzolu, Gianfranco), 606.
Путь, 32, 64.
в решетке, 409.
в сети, 450.
в сетке, 409, 433.
Грея, 344, 470, 840.
червяка, 511, 530, 881, 884.
Пушкин, Александр Сергеевич, 493.
Пфафф, Иоганн Фридрих (Pfaff, Johann Friedrich), 798.
Пятибуквенные слова, 28, 65, 297, 322, 324, 325, 339, 362, 368, 399, 488.
Пятиугольник, 535.
Пятиугольные числа, 451.

Рабин, Майкл Озер (Rabin, Michael Oser (מיכאל עוזר רבין)), 665.
Равенство байтов, 186.
Равив, Джозеф (Raviv, Josef (יוסף רביב)), 754.
Радемахер, Ганс (Rademacher, Hans), 336, 453, 454, 466, 467.
Радо, Ричард (Radó, Richard), 444, 621.
Радойчич, Радош (Radoičić, Radoš), 792.
Разбиения, 46, 51, 399, 444–497, 571, 576, 593, 803, 806, 852.
векторов, 485, 496.
без единиц, 454.
без одноэлементных частей, 493, 854.
дважды истинные, 399.
множества, 327, 446, 471–497, 533, 565, 573, 575, 701.
мультимножеств, 484, 496, 567, 576, 863.
целых чисел, 446–471, 484, 491, 567.
Разборов, Александр Александрович, 155.
Разделительная дизъюнкция, 73.
Разделяй и властвуй, 138, 181, 634.
Различные части, 464, 465, 466, 468.

Разложение, 162, 567, 576.
строк, 368.
функций, 147.
частично определенных функций, 150.
Эджворта, 858.
Размах, 430, 441.
Размен, 463.
Размер, 32, 38, 68.
Размерность, 435.
Разность, 320.
Разреженная функция, 298.
Разреженный граф, 41, 44.
Райзер, Герберт Джон (Ryser, Herbert John), 59, 580, 839.
Райт, Эдвард Метланд (Wright, Edward Maitland), 861, 862.
Райтвизнер, Георг Вальтер (Reitwiesner, George Walter), 224.
Раман, Раджив (Raman, Rajeev), 666.
Рамануджан Иенгар, Сриниваза (Ramanujan Iyengar, Srinivasa (ஸ்ரீனிவாஸராமானுஜன் ஐயங்கார்)), 453, 454, 465, 466, 467, 831, 832.
Рамрас, Марк Бернард (Ramras, Mark Bernard), 766.
Ранг, 239, 327, 348, 397, 405, 537, 539, 548, 552, 562, 573, 574, 796, 808, 812.
Рандомизация, 148.
Рандомизированные структуры данных, 659.
Рандрианаривоны, Артур (Randrianarivony, Arthur), 730.
Рани, Джордж Нил (Raney, George Neal), 538.
Ранкин, Роберт Александер (Rankin, Robert Alexander), 390, 404, 793.
Рао, Калямпуди Радхакришна (Rao, Calyampudi Radhakrishna (కళంపూడి రాధాకృష్ణ రావు)), 581.
Рапапорт, Эльвира Штрассер (Rapaport, Elvira Strasser), 792.
Раски, Фрэнк (Ruskey, Frank), 160, 349, 350, 357, 360, 363, 390, 405, 438, 439, 473, 490, 504, 506, 510, 513, 537, 573, 739, 774, 778, 792, 793, 796, 798, 802, 875, 887.
Раскрашивание, 70, 290.
графа, 36, 151, 276, 303.
Раскрутка, 632, 637.
Раскрытие медианы, 114.
Распаковка, 168.
Распознавание шаблонов, 208.
Распределение
памяти, 188, 224.
Пуассона, 483, 490.
Распределенные системы, 115.
Распространение битов, 172.
Рассеянная разность, 228.
Рассеянная сумма, 184.
Рассеянный аккумулятор, 228.
Рассеянный сдвиг, 228.
Расстояние, 35, 66.
кода, 60.
обобщенное, 36.
Хэмминга, 31, 49, 60, 118, 323, 615, 696.
Растр, 235.
Растровая графика, 207–217.
Расходящиеся прямые, 203.
Расширение, 356.
Расширенная таблица истинности, 285.
Расширенное бинарное дерево, 514, 532, 545.
Расширенное тернарное дерево, 532.
Расширенные действительные числа: действительные числа с $-\infty$ и $+\infty$, 88.
Рашед, Рощди (Rashed, Roshdi (= Rashid, Rushdi) (رشدي راشد)), 554, 898.
Реберный граф, 47, 57, 65, 586, 590, 593.
Ребра как пары дуг, 39, 42.
Регистр, 130.
Регулярная функция, 121, 310.
Регулярный граф, 33, 45, 65, 68.
Регулярный язык, 231, 327.
Редей, Ласло (Rédei, László), 586.
Редькин, Николай Петрович, 137, 162, 644.
Резольвента, 605.
Результант, 892.
Резулярная функция, 123.
Рейнгольд, Эдвард Мартин (Reingold, Edward Martin (ריינגולד, יצחק משה בן חיים)), 338, 416, 573.
Рейсс, Мишель (Reiss, Michel), 826.
Рейтинги упражнений, 15.
Рекорде, Роберт (Recorde, Robert), 913.
Рекуррентное соотношение Фибоначчи, 451.
Рекуррентность, 239, 313, 352, 434, 451, 459, 465, 602, 633, 634, 695, 714, 717, 725, 728, 746, 750, 761, 770, 782, 809, 810, 811, 861, 873, 880, 894.
Рекурсивная подпрограмма, 95.
Рекурсивная процедура, 628, 847, 887.
Рекурсивная структура, 864, 874.
Рекурсивная формулировка, 266.
Рекурсивные подпрограммы, 424.
Рекурсивный алгоритм, 310, 312, 573.
Рекурсия, 183, 221, 313, 417, 523, 633, 634, 727, 874.
и итерация, 419, 438.
Рекхау, Роберт Аллен (Reckhow, Robert Allen), 608.
Реми, Жан-Люк (Rémy, Jean-Luc), 514, 538, 882.
Реммель, Джеффри Брайан (Remmel, Jeffrey Brian), 852.
Ретракция, 100, 119.
Рефлексивный
десятичный код Грея, 347.
код Грея, 436, 505, 772, 789, 798, 841.
для смешанных оснований, 372.

Решетка, 120, 251, 298, 306, 468, 489, 544, 893.
деревьев, 533.
Кревераса, 533.
мажоризации, 846.
нижняя полумодулярная, 846.
разбиений, 489.
Стенли, 534.
Тамари, 533, 535.
Решето, 842.
Эратосфена, 169, 224.
Решетчатый граф, 297, 528.
Ривест, Рональд Линн (Rivest, Ronald Linn), 81, 307.
Рид, Рональд Седрик (Read, Ronald Cedric), 424, 815.
Римские цифры, 900.
Рингель, Герхард (Ringel, Gerhard), 366, 589.
Риордан, Джон (Riordan, John), 637, 878.
Ритм, 549.
Ричардс, Дана Скотт (Richards, Dana Scott), 366, 532, 536.
Риччи, Алистер Инглиш (Ritchie, Alistair English), 758.
Роббинс, Девид Питер (Robbins, David Peter), 854.
Робертсон, Джордж Нейл (Robertson, George Neil), 37.
Робинсон, Джон Алан (Robinson, John Alan), 605.
Робинсон, Джон Пауль (Robinson, John Paul), 359, 765, 767.
Робинсон, Жильберт де Борегард (Robinson, Gilbert de Beauregard), 540.
Робинсон, Роберт Вильям (Robinson, Robert William), 827, 857.
Роджет, Джон Льюис (Roget, John Lewis), 44.
Роджет, Питер Марк (Roget, Peter Mark), 28, 44.
Родригес, Бенджамин Олинде (Rodrigues, Benjamin Olinde), 514.
Родственные леса, 531.
Роза, Александер (Rosa, Alexander), 580.
Розенбаум, Джозеф (Rosenbaum, Joseph), 770.
Розенкранц, Дэниэл (Джей Rosenkrantz, Daniel Jay), 711.
Розенфельд, Азриэль (Rosenfeld, Azriel (עזריאל רוזנפלד)), 210.
Рой, Мохит Кумар (Roy, Mohit Kumar (মোহিত কুমার রায়)), 784.
Ройш, Бернд (Reusch, Bernd), 751.
Рокицки, Томас Герхард (Rokicki, Tomas Gerhard), 225, 659.
Ролантс ван Баронайген, Доминик (Rœlants van Baronaigien, Dominique), 504, 506, 875.
Россин, Доминик Жиль (Rossin, Dominique Gilles), 890.
Рот, Джон Пол (Roth, John Paul), 605.
Рота, Жан-Карло (Rota, Gian-Carlo), 444, 624.
Роте, Генрих Август (Rothe, Heinrich August), 570, 781.
Роте, Гюнтер (Rote, Günter (= Rothe, Günther Alfred Heinrich)), 237.
Ротем, Дорон (Rotem, Doron (דורון רותם)), 392, 395.
Ротхаус, Оскар Сеймур (Rothaus, Oscar Seymour), 627.
Рудеану, Серж (Rudeanu, Sergiu), 303.
Руделл, Ричард Лайл (Rudell, Richard Lyle), 272, 285, 287, 288, 303, 315, 708, 724.
Руза, Имре Золтан (Ruzsa, Imre Zoltán), 822.
Рукопожатие, 502, 530.
Рутовиц, Дэнис (Rutovitz, Denis), 208.
Руциньски, Анджей (Ruciński, Andrzej), 597.
Рфлексивный код Грея, 350.
Рэй, Луи Чарльз (Ray, Louis Charles), 207.
Рэйнод-Ричард, Пьер (Raynaud-Richard, Pierre), 690.
Рэмшоу, Лайл Гарольд (Ramshaw, Lyle Harold), 187.
Рэндж, Нико (Range, Niko), 721.
Рэндолл, Кейт Гарольд (Randall, Keith Harold), 224.
Рюдигер, Христиан Фридрих (Rüdiger, Christian Friedrich), 370.
Рюкзак, 101, 414.
Ряд
Тейлора, 32, 481, 856.
Фурье, 335, 452.

Сабо, Йозеф (Szabó, József), 785.
Савада, Джозеф Джеймс (Sawada, Joseph James), 778.
Савицки, Петр (Savický, Petr), 723.
Саган, Брюс Эли (Sagan, Bruce Eli), 803.
Сазерленд, Стюарт (Sutherland, Stuart), 11.
Саймон, Имре (Simon, Imre), 697.
Сак, Йорг-Рюдигер Вольфганг (Sack, Jörg-Rüdiger Wolfgang), 882.
Сака, Масанобу (Saka, Masanobu (坂正永)), 566.
Саккери, Джованни Джироламо (Saccheri, Giovanni Girolamo), 203.
Сакс, Майкл Эзра (Saks, Michael Ezra), 615.
Самет, Ханан (Samet, Hanan (חנן סמט)), 666.
Самодополнительный граф, 64, 65.
Самодуализация, 119.
Самодуальная пороговая функция, 104, 119.
Самодуальная функция, 107.
Самодуальность, 88, 104, 123, 315, 536.
Самоорганизующиеся структуры данных, 615.
Самосопряженность, 464, 490, 866.
Самосопряженные разбиения, 840.
Самсон, Эдвард Уолтер (Samson, Edward Walter), 601, 605.
Сандерс, Даниэль Престон (Sanders, Daniel Preston), 37.
Санджованни-Винсентелли, Альберто Луиджи (Sangiovanni-Vincentelli, Alberto Luigi), 152.
Санскрит, 548, 552.

Сарнгадева, сын Содхаладева (Śārṅgadeva, son of Soḍhaladeva (शार्ङ्गदेव, सोढलदेव पुत्रः)), 553, 562, 781, 899.
Сартена, Кристиан (Sartena, Christian), 598.
Сасаки, Фукаши (Sasaki, Fukashi (佐々木不可止)), 104.
Сасао, Цутому (Sasao, Tsutomu (笹尾勤)), 699.
Сатклифф, Алан (Sutcliffe, Alan), 845.
Сатнер, Клаус (Sutner, Klaus), 689.
Сахс, Хорст (Sachs, Horst), 589, 891, 894.
Сачков, Владимир Николаевич, 484.
Сбалансированность, 463, 538, 740.
Сбалансированные тернарные числа, 233.
Сборка мусора, 269, 310.
Сверхребро, 524.
Свиннертон-Дайр, Генри Питер Френсис (Swinnerton-Dyer, Henry Peter Francis), 832.
Свифт, Джонатан (Swift, Jonathan), 171, 692.
Свободная алгебра медиан, 96, 118.
Свободное дерево, 894.
Свободные группы, 572.
Свободные деревья, 36, 37, 68, 92, 521, 541, 545, 571, 645, 873.
Свободные левые скобки, 885.
Свойство двери-вертушки, 437.
Связанные бинарные деревья, 502–508, 514, 532, 538.
Связанные компоненты, 764.
Связанный список, 385, 435, 722.
Связанный узел, 284, 722.
Связи, 488.
Связность, 9, 54, 251, 595.
  вершин, 592.
  ориентированного графа, 39.
Связный гиперграф, 54.
Связный граф, 35, 68, 523.
Связь, 463.
Сдвиг, 185, 232.
Сегё, Габор (Szegő, Gábor), 15.
Седжвик, Роберт (Sedgewick, Robert), 391, 573.
Сеймур, Пол Дуглас (Seymour, Paul Douglas), 37.
Секанина, Милан (Sekanina, Milan), 529.
Секели, Ласло Аладар (Székely, László Aladár), 845.
Секи, Такаказу (Seki, Takakazu (關孝和)), 553, 566, 574.
Секущая, 25.
Селаски, Ганс Петтер Вильям Сиреваг (Selasky, Hans Petter William Sirevaag), 241.
Селе, Тибор (Szele, Tibor), 586.
Семба, Ичиро (Semba, Ichiro (仙波一郎)), 499, 700, 753.
Семейство
  множеств, 54, 294, 310, 734.
  подмножеств, 115.
Семереди, Эндре (Szemerédi, Endre), 119.
Семисегментный дисплей, 141, 143.
Семь смертных грехов, 556.
Сервисное поле, 588.
Серени, Балас (Szörényi, Balázs), 605.
Серии битов, 172.
Серра, Микаэла (Serra, Micaela), 778.
Сет, Викрам (Seth, Vikram (विक्रम सेठ)), 494.
Сетевая модель вычислений, 254, 308.
Сеть, 52, 680.
Сжатие, 825.
  данных, 244.
  множества, 431.
  ребра, 521.
Сжатый граф, 521.
Сибсон, Робин (Sibson, Robin), 674.
Сигель, Карл Людвиг (Siegel, Carl Ludwig), 715.
Сигнатура буквометика, 375.
Сили, Коломан фон (Szily, Koloman von), 749.
Сильвер, Альфред Линдсей Лей (Silver, Alfred Lindsey Leigh), 790.
Сильверман, Джерри (Silverman, Jerry), 763, 765.
Сильвестер, Джеймс Джозеф (Sylvester, James Joseph), 362, 464, 733, 762, 832.
Сильно связанный компонент, 62, 586.
Сильное произведение, 49, 65, 543, 596.
Символы Якоби, 833.
Симметричная булева функция, 319.
Симметричная группа, 653.
Симметричная пороговая функция, 717.
Симметричная разность, 320.
Симметричная функция, 163, 253, 260, 274, 715.
Симметричное среднее, 470.
Симметричные булевы функции, 103, 302, 307, 308, 313.
Симметричные полиномы, 901.
Симметричные функции, 104, 122, 126–133, 137, 146, 156, 157, 321, 325, 448, 631, 632, 699, 716, 839.
Симметричный порядок обхода, 200, 499, 505, 506, 513, 537, 871.
Симметрия, 34, 61, 133, 378, 398, 716, 750.
Симоэс Перейра, Хосе Мануэль дос Сантос (Simões Pereira, José Manuel dos Santos), 807.
Симплекс, 426.
Симплициальный комплекс, 442, 825.
Симплициальный мультикомплекс, 442.
Симпсон, Джеймс Эдвард (Simpson, James Edward), 577.
Симс, Чарльз Коффин (Sims, Charles Coffin), 378.
Синглтон, Роберт Ричмонд (Singleton, Robert Richmond), 587.
Сингмастер, Дэвид Брейер (Singmaster, David Breyer), 578, 689.
Сингх, Пармананд (Singh, Parmanand (परमानन्द सिंह)), 549, 561.
Синтаксис, 83, 545.

Синтез в глубину, 266.
Система
множеств, *см.* Гиперграф, 54.
счисления Фибоначчи, 204, 235.
троек, 54, 67.
Скалигер, Юлиус (Scaligero, Giulio (= Scaliger, Julius Caesar)), 563.
Скалярное произведение, 31, 55, 59.
Скарбек, Владислав Казимеж (Skarbek, Władysław Kazimierz), 502, 877.
Сквайр, Мэттью Блейз (Squire, Matthew Blaze), 774.
Склянски, Джек (Sklansky, Jack), 635.
Скобки, 326, 498–502, 517, 571, 572, 864.
Скоинс, Губерт Ян (Scoins, Hubert Ian), 520, 573.
Сколем, Альберт Торальф (Skolem, Albert Thoralf), 10, 27, 58, 577, 578.
Скошенная пирамида, 234.
Скрученное биномиальное дерево, 544.
Скрыто взвешенная битовая функция, 278, 308, 313, 316, 753.
Скрытый узел, 284, 722.
Скютелла, Мария Грация (Scutellà, Maria Grazia), 85.
Слабая логика второго порядка, 154.
Слабое упорядочение, 571.
Слабый порядок, 406.
Сланина, Маттео (Slanina, Matteo), 653.
След, 449, 457, 464, 535, 830, 872.
Слепиан, Дэвид (Slepian, David), 178.
Слип, Майкл Ронан (Sleep, Michael Ronan), 511.
Слитор, Дэниэль Доминик Каплан (Sleator, Daniel Dominic Kaplan), 167, 680.
Слияние, 259, 286.
Слоан, Нейл Джеймс Александер (Sloane, Neil James Alexander), 581, 758.
Слоан, Роберт Хал (Sloan, Robert Hal), 605.
Слободова, Анна Миклашова (Slobodová, Anna Miklášová), 713.
Слова
Дика, 572.
Линдона, 355.
Словарь, 28, 71, 268.
Сложение по модулю 3 и 5, 160.
Сложность булевой функции, *см.* длина булевой функции, 127.
Слокам, Джеральд Кеннет (Slocum, Gerald Kenneth (= Jerry)), 772.
Слуцки, Жора (Slutzki, Giora (סלוצקי גיורא)), 95.
Случайная булева функция, 81.
Случайное блуждание, 68.
Случайное дерево Каталана, 546.
Случайное дерево Шрёдера, 883.
Случайное ориентированное дерево, 541.
Случайное решение, 246.
уравнения $f(x) = 1$, 248.
Случайные бинарные деревья, 514, 538.
Случайные разбиения множеств, 482.
генерация, 484.
Случайные разбиения, генерация, 467.
Случайный лес, 538.
Смежность, 32.
Смежные обмены, 285, 371–376, 406, 800.
Смежные подмножества вершин, 274.
Смесь НДР и БДР, 302.
Смешанная позиционная система счисления, 330, 348.
Смит, Генри Джон Стефен (Smith, Henry John Stephen), 596.
Смит, Дерек Алан (Smith, Derek Alan), 762.
Смит, Джон Линн (Smith, John Lynn), 635.
Смит, Малкольм Джеймс (Smith, Malcolm James), 521, 523, 526, 527.
Смит, Марк Эндрю (Smith, Mark Andrew), 124.
Снир, Марк (Snir, Marc (מרק שניר)), 158, 163.
Собственное значение, 63, 543, 803.
Собственный вектор, 63, 892.
Сове, Жозе (Sauveur, Joseph), 579.
Совершенное паросочетание, 149, 297.
Сограф, 65.
Соединение, 47, 320, 595.
Сойерс, Дороти Лей (Sayers, Dorothy Leigh), 371.
Сократ, сын Софрониска (Socrates, son of Sophroniscus of Alopece (Σωκράτης Σωφρωνίσκου Ἀλωπεκῆθεν)), 498.
Соле, Патрик (Solé, Patrick), 758.
Солексный порядок, 281, 377, 413, 447, 463, 532, 549, 562, 574, 659, 798, 838, 899, 901.
Соменци, Фабио (Somenzi, Fabio), 304, 698, 706, 707, 713.
Сонет, 493.
Соприкосновение, 47.
Сопрограмма, 403.
Сопряжение, 66, 382, 449, 463, 468, 470, 490, 506, 531, 536, 544, 837.
объединенных разбиений, 831.
Сопряженное разбиение, 50, 592.
Сортировка, 231, 285.
выбором с замещением, 808.
слиянием, 218.
Сортирующая сеть, 119, 158, 633.
Соседи, 32.
Соул, Стефен Парк (Soule, Stephen Parke), 668.
Сочетание, 445, 501, 504, 554–562, 571, 655, 867, 875, 898, 901.
мультимножеств, 433, 574.
с повторениями, 409, 418, 426, 554, 561, 575.
Спектр, 577.
графа, 891, 894.
Спенсер, Эдмунд (Spenser, Edmund), 493.
Спернер, Эмануэль (Sperner, Emanuel), 517, 826.
Спитковски, Валентин Ильич (Spitkovsky, Valentin Ilyich), 727.
Сплайны Безье, 216, 237.
Спондей, 550, 563.

Спруньоли, Ренцо (Sprugnoli, Renzo), 537.
Спрэг, Роланд Персиваль (Sprague, Roland Percival), 650.
Спуск, 486, 539.
Сравнение
  байтов, 186, 230.
  бинарных чисел, 120.
Среднее, 184, 470.
Средний бит, 291.
Стам, Аарт Йоханнес (Stam, Aart Johannes), 484, 496.
Стандартная последовательность, 355.
Стандартное множество, 430, 441.
Станке, Уэйн Ли (Stahnke, Wayne Lee), 352.
Стантон, Дэннис Уоррен (Stanton, Dennis Warren), 831.
Стапперс, Филип Ян Джос (Stappers, Filip Jan Jos), 723.
Старший бит, 291.
Статистика, 129.
Стафкен, Джон (Stufken, John), 581.
Стаховяк, Гжегош (Stachowiak, Grzegorz), 815.
Стедалл, Жаклин Анна (Stedall, Jacqueline Anne), 567.
Стедман, Фабиан (Stedman, Fabian), 373.
Стейглиц, Кеннет (Steiglitz, Kenneth), 760.
Стек, 43, 84, 603.
Стенли, Ричард Питер (Stanley, Richard Peter), 32, 422, 444, 448, 536, 561, 620, 820, 842, 843, 850, 871, 883.
Степень, 33, 39, 61, 66, 67, 532, 593.
  вершины, 544.
    графа, 523.
  графа, 529, 544.
Стефенс, Нельсон Малкольм (Stephens, Nelson Malcolm), 849.
Стибитц, Джордж Роберт (Stibitz, George Robert), 333, 335.
Стивенс, Бретт (Stevens, Brett), 365.
Стивенсон, Дэвид Ян (Stevenson, David Ian), 143, 631, 640, 641.
Стигер, Анжелика (Steger, Angelika), 797.
Стил, Гай Льюис-мл. (Steele, Guy Lewis, Jr.), 181, 227, 660, 663.
Стинсон, Дуглас Роберт (Stinson, Douglas Robert), 573.
Стирлинг, Джеймс (Stirling, James), 566.
Стихи, 567.
Стихи-хамелеоны, 563.
Стойменович, Иван Данча (Stojmenović, Ivan Danča), 446.
Сток, 38, 87, 285.
Стокмейер, Ларри Джозеф (Stockmeyer, Larry Joseph), 9, 154, 155, 163, 632, 647, 661, 665.
Стоктон, Фред Грант (Stockton, Fred Grant), 685.
Столфи, Йорг (Stolfi, Jorge), 687.
Страхлер, Артур Невелл (Strahler, Arthur Newell), 545, 584, 896.
Страчи, Кристофер (Strachey, Christopher), 176.
Строка, 185.
  вложенных скобок, 871.
  Стирлинга, 846.
  Фибоначчи, 58.
Структура связности, 209.
Структуры данных для графов, 523.
Стюарт, Ян Николас (Stewart, Ian Nicholas), 369.
Суби, Карлос Сэмюэль (Subi, Carlos Samuel), 363.
Субрамани, Кришнамурти (Subramani, Krishnamurthy (கிருஷ்ணமூர்த்தி சுப்ரமணி)), 611.
Субраманиан, Ашок (Subramanian, Ashok), 617.
Судборух, Айвен Хал (Sudborough, Ivan Hal), 801.
Сумасшедшая петля, 366.
Сумма
  битов, *см.* Контрольная сумма, 175.
  Дедекинда, 453.
  квадратов, 362.
  по всем разбиениям, 448, 475, 855, 858.
Суммы
  столбцов, 469.
  строк, 469.
Супарта Ай Ненга (Suparta, I Nengah), 759.
Суперкорень, 544.
Суповит, Кеннет Джей (Supowit, Kenneth Jay), 720.
Сусрута (Suśruta (सुश्रुत)), 554.
Суффикс, 355.
  строки, 163.
Схема, близкая к идеальной, 424.
Схемы рифм, 472, 493, 567, 575.
Счетчик
  команд, 192.
  ссылок, 270, 310, 315, 710.
Съёстранд, Йонас Эрик (Sjöstrand, Jonas Erik), 690.
Сылов, Петер Людвиг Мейдель (Sylow, Peter Ludvig Mejdell), 653, 661.
Сэведж, Джон Эдмунд (Savage, John Edmund), 637.
Сэведж, Карла Диана (Savage, Carla Diane), 346, 347, 357, 363, 365, 461, 573, 764, 792, 800, 816.
Сэмпсон, Джон Лоуренс (Sampson, John Laurence), 763.
Сяо, Бен Му-Юе (Hsiao, Ben Mu-Yue (蕭慕岳 = 萧慕岳)), 582.

Таблица, 50.
  инверсий, 372, 405, 501, 781, 797, 798, 864.

истинности, 72, 73, 75–77, 97, 122, 124, 126, 129, 130, 140, 145, 164, 173, 241, 243, 246, 253, 256, 260, 261, 265, 276–278, 294, 302, 305, 308, 310, 326, 598, 636, 720, 726, 753.
количеств булевых функций, 104.
Симса, 378–386, 399.
уникальности, 268.
Юнга, 394.
Таблицы чисел, 904–907.
Табличное представление, 450.
Тавтология, 73, 607, 735.
Тайлер, Дуглас Блейн (Tyler, Douglas Blaine), 536.
Такаги, Теиджи (Takagi, Teiji (高木貞治)), 428, 821.
Такасу, Сатору (Takasu, Satoru (高須達)), 618.
Такахаши, Хидетоши (Takahasi, Hidetosi (高橋秀俊)), 619.
Такенага, Ясухико (Takenaga, Yasuhiko (武永康彦)), 721.
Таки, Джон Уильдер (Tukey, John Wilder), 71.
Тако, Андрэ (Tacquet, André), 560.
Танг, Дональд Тао-Нан (Tang, Donald Tao-Nan (唐道南)), 416, 504.
Танненбаум, Мейер (Tannenbaum, Meyer), 623.
Танцующие связи, 20, 25, 30, 523, 579, 740.
Таржан, Роберт Эндре (Tarjan, Robert Endre), 19, 87, 114, 200, 643, 677.
Тарри, Гастон (Tarry, Gaston), 23, 826.
Тарталья, Никколо Фонтана (Tartaglia, Niccolò Fontana), 561.
Тейлор, Брук (Taylor, Brook), 481, 856.
Тейлор, Ллойд Вильям (Taylor, Lloyd William), 333.
Телевидение, 333.
Телефон, 333.
Темперли, Гарольд Невилл Вазелл (Temperley, Harold Neville Vazeille), 457.
Теневое исчисление, 848.
Тензорное произведение, 49.
Тень, 426–433, 440, 489.
бинарной строки, 443.
подкуба, 443.
Теорема
Джордан о кривых, 212.
Крускала-Катоны, 427.
о вычетах, 475, 478.
о матрице дерева, 802.
Спернера, 539, 885.
Ферма, 368.
четырех красок, 37.
Теория
автоматов, 327.
графов, 303.
введение, 32–40.
Терквем, Олри (Terquem, Olry), 444.
Терминология, 601.
Тернарная операция, 88, 271, 309, 601, 649.
Тернарное дерево, 536.
Тернарность, 436.
Тернарный вектор, 198.
Тернарный код Грея, 348.
Тесленко, Максим Васильевич (Teslenko, Maxim Vasilyevich), 720.
Тетраэдр, 46.
Тимонье, Лойз (Thimonier, Loÿs), 898.
Типпетт, Леонард Генри Кейлеб (Tippett, Leonard Henry Caleb), 834.
Тисон, Пьер Луи Джозеф (Tison, Pierre Louis Joseph), 605.
Тобальд, Торстен (Theobald, Thorsten), 713.
Тода, Ивао (Toda, Iwao (戸田巌)), 618.
Тодоров, Добромир Тодоров (Todorov, Dobromir Todorov), 580.
Тождества биномиальных коэффициентов, 537.
Тождественная перестановка, 378.
Тождество Ньютона, 891.
Токушиге Норихиде (Tokushige, Norihide (徳重典英)), 822.
Толстой, Лев Николаевич, 28.
Томас, Герберт Кристофер (Thomas, Herbert Christopher (= Ivo)), 73.
Томас, Робин (Thomas, Robin), 37.
Томпкинс, Чарльз Браун (Tompkins, Charles Brown), 24, 26, 388, 572.
Томпсон, Кеннет Лейн (Thompson, Kenneth Lane), 240.
Топологическая сортировка, 85, 100, 112, 126, 298, 394, 405, 470, 815, 844.
Тор, 12, 49, 235, 359, 404, 428–433, 441, 527, 528, 544, 669, 758, 890, 893.
де Брейна, 369.
Торо, Дэвид Генри (Thoreau, David Henry (= Henry David)), 124.
Торуп, Миккель (Thorup, Mikkel), 669, 687.
Тот, Золтан (Tóth, Zoltán), 780.
Точер, Кейт Дуглас (Tocher, Keith Douglas), 165, 229, 666.
Точка сочленения, 300.
Точное покрытие, 57.
Траляля (Tweedledee), 72, 106.
Транзитивное замыкание, 193, 200.
Транзитивный турнир, 48, 62, 64.
Транспонирование, 225, 241, 531, 536.
матрицы, 180.
Треугольная решетка, 46, 589.
Треугольная сетка, 871.
Треугольник, 428.
Каталана, 509, 517, 536.
t-арный, 874.
Пирса, 474, 490, 491, 492, 494, 854, 862.
Шрёдера, 883.
Трехвалентный граф, 33.
Трехзначная логика, 197, 233, 558.
Трехрегистровый алгоритм, 213.
Триангуляция, 806.
Делоне, 44.
Тривиальная функция, 74, 88, 89.
Тривиальные деревья, 866.
Трижды связанный лес, 530, 540, 869.

Тримино, 296, 322.
Триразбиения, 485.
Троичная система счисления, 548.
Тройки Лэнгфорда, 58.
Тройное произведение, 451, 465.
Трост, Эрнст (Trost, Ernst), 842.
Троттер, Уильям Томас (Trotter, William Thomas), 794.
Троттер, Хейл Фриман (Trotter, Hale Freeman), 373.
Троубридж, Терри Джей (Trowbridge, Terry Jay), 785.
Трюки и технологии, 165.
Тулиани, Джонатан Р. (Tuliani, Jonathan R.), 777.
Тума, Иржи (Tůma, Jiří), 846.
Туран, Герги (Turán, György), 605.
Турнир, 62, 469, 893.
Тутилл, Джеффри Колин (Tootill, Geoffrey Colin), 342, 756.
Тушар, Жак (Touchard, Jacques), 848.
Тью, Аксель (Thue, Axel), 10, 250, 305, 696.
Тьюринг, Алан Мэтисон (Turing, Alan Mathison), 165.

Уайт, Артур Томас (White, Arthur Thomas), 373.
Уайт, Деннис Эдуард (White, Dennis Edward), 852.
Удаление битов, 172.
Узел графа, 42.
Уилкис, Морис Винсент (Wilkes, Maurice Vincent), 175.
Уиллард, Дан Эдвард (Willard, Dan Edward), 187, 231.
Уиллер, Дэвид Джон (Wheeler, David John), 175.
Уиппл, Френсис Джон Велш (Whipple, Francis John Welsh), 431.
Уитворт, Вильям Аллен (Whitworth, William Allen), 475, 564, 575.
Указатели на родительские узлы, 540.
Указатель на родителя, 519, 529.
Улам, Станислав Марцин (Ulam, Stanisław Marcin), 675.
Улиг, Дитмар (Uhlig, Dietmar), 162.
Умножение, 232, 291, 318.
  бинарное, 270.
  знаковых битов, 195.
  матриц, 40, 219, 226, 649, 811.
  полиномов, 241.
  слева, 381.
Унгер, Стивен Герберт (Unger, Stephen Herbert), 185.
Универсальная алгебра, 599.
Универсальное семейство, 322.
Универсальное хеширование, 318.
Универсальные последовательности разбиений, 497.
Универсальный набор символов, 240.
Универсальный цикл, 443, 559, 574.
  перестановок, 407.
Унимодальные последовательности, 496.
Уножение слева, 383, 799.
Уокерли, Джон Френсис (Wakerly, John Francis), 642.
Уоллис, Джон (Wallis, John), 334, 553, 561, 564, 567, 575, 576, 756.
Уолш, Джозеф Леонард (Walsh, Joseph Leonard), 335, 336, 761.
Уолш, Тимоти Роберт Стефен (Walsh, Timothy Robert Stephen), 417, 798, 799, 817.
Уоррен, Генри Стенли-мл. (Warren, Henry Stanley, Jr.), 172, 175, 177, 190, 220, 221, 649, 658, 663, 667, 691.
Уоррен, Йон (Warren, Jon), 513.
Уоткинс, Джон Джегер (Watkins, John Jaeger), 595.
Уошбурн, Сет Харвуд (Washburn, Seth Harwood), 758.
Упакованные данные, операции над ними, 184.
Упаковка, 168, 181, 198, 224, 240.
  данных, 229.
Упорядочение сочетаний, 413, 417, 437.
Упорядоченная бинарная диаграмма решений, 243.
Упорядоченное разбиение, 575.
Упорядоченные деревья, 573.
Упорядоченные разбиения, 842.
Упорядоченный лес, 571.
Управляющая сетка, 133.
Упражнения, примечания, 13–15.
Уравновешенная троичная система счисления, 809.
Урбан, Жанвье Хокинс (Urban, Genevie Hawkins), 207.
Ури, Дарио (Uri, Dario), 689, 786.
Урны, 444.
Уровень, 502.
Усреднение, 229.
Устойчивая сортировка, 486.
Устойчивое состояние, 542.
Устойчивость, 55.
Утверждение, 73.
Утончение, 489.
Ушийима, Казуо (Ushijima, Kazuo (牛島和夫)), 504.
Уэйн, Алан (Wayne, Alan), 399.
Уэллс, Марк Бримхолл (Wells, Mark Brimhall), 572, 573, 788.
Уэсте, Нейл Гарри Эрл (Weste, Neil Harry Earle), 212.

Факториал, 574.
Факториальная
  линеечная функция, 401.
  система счисления, 781.
Факториальное представление, 562.
Факторная таблица, 426, 439, 469.
Фалкерсон, Делберт Рэй (Fulkerson, Delbert Ray), 592.

Фалутсос, Христос (Faloutsos, Christos (Φαλούτσος, Χρήστος)), 759.
Фанк, Исаак Кауффман (Funk, Isaac Kauffman), 73.
Фачс, Дэвид Рэймонд (Fuchs, David Raymond), 682.
Федер, Томас (Feder, Tomás), 99, 363, 618.
Фейнман, Ричард Филлипс (Feynman, Richard Phillips), 693.
Феллер, Виллибальд (Feller, Willibald (= Vilim = Willy = William)), 882.
Фельзенштейн, Джозеф (Felsenstein, Joseph), 857.
Фенвик, Питер Маколей (Fenwick, Peter McAulay), 636.
Феничел, Роберт Росс (Fenichel, Robert Ross), 434.
Феннер, Тревор Ян (Fenner, Trevor Ian), 829.
Фидлер, Мирослав (Fiedler, Miroslav), 893.
Фидуччия, Чарльз Майкл (Fiduccia, Charles Michael), 597.
Филлипс, Джон Патрик Норман (Phillips, John Patrick Norman), 781.
Филогенетические деревья, 856.
Филон из Мегары (= Филон Диалектик) (Philo of Megara (= Philo the Dialectician, Φίλων ὁ Μεγαρίτης)), 73.
Фиников, Борис Иванович, 156.
Финк, Ганс-Иоахим (Finck, Hans-Joachim), 594.
Финк, Иржи (Fink, Jiří), 765.
Финоженок, Дмитрий Николаевич (Finozhenok, Dmitriy Nikolaevich), 749.
Фишер, Иоганн Кристиан (Fischer, Johannes Christian), 677.
Фишер, Людвиг Йозеф (Fischer, Ludwig Joseph), 783.
Фишер, Майкл Джон (Fischer, Michael John), 157, 159.
Фишер, Петр (Fišer, Petr), 79.
Фишер, Рональд Эйлмер (Fisher, Ronald Aylmer), 834.
Фишер, Рэндолл Джеймс (Fisher, Randall James), 185, 667.
Фишлер, Мартин Алвин (Fischler, Martin Alvin), 623.
Флажоле, Филипп Патрик Мишель (Flajolet, Philippe Patrick Michel), 898.
Флайт, Колин (Flight, Colin), 857.
Флаудс, Лесли Ричард (Foulds, Leslie Richard), 857.
Фли Сан-Мари, Камилла (Flye Sainte-Marie, Camille), 827.
Флойд, Роберт В. (Floyd, Robert W.), 228.
Флорес, Айвен (Flores, Ivan), 770.
Фляйшнер, Герберт (Fleischner, Herbert), 895.
Фойсснер, Фридрих Вильгельм (Feussner, Friedrich Wilhelm), 521.
Фокс, Ральф Хартцлер (Fox, Ralph Hartzler), 355.
Фокус, 497.
Фокусные указатели, 338, 349, 350, 526, 542, 773.
Фолланд, Джеральд Б. (Folland, Gerald B.), 6.
Фомин, Сергей Владимирович, 851.
Фон, 211.
Форкейд, Родни Уорринг (Forcade, Rodney Warring), 597.
Форма
  разбиения множества, 482.
  случайного бинарного дерева, 513.
  случайного леса, 511.
  случайного разбиения, 457, 467.
Формула
  Арбогаста, 475.
  обращения Лагранжа, 856.
  суммирования
    Пуассона, 452, 465.
    Эйлера, 452, 465, 787.
Форни, Джордж Дэвид-мл. (Forney, George David, Jr.), 754.
Форте, Робер Мари (Fortet, Robert Marie), 303.
Форчун, Стивен Джонатон (Fortune, Steven Jonathon), 302.
Фрагментированные поля, 183.
Фраер, Ранан (Fraer, Ranan (רענן פרייר)), 114.
Фрактал, 239, 539, 658.
Франкл, Петер (Frankl, Péter), 820, 822.
Франклин, Фабиан (Franklin, Fabian), 464, 467.
Фредман, Майкл Лоуренс (Fredman, Michael Lawrence), 187, 231, 362, 603, 750, 763.
Фредриксен, Гарольд Марвин (Fredricksen, Harold Marvin), 356, 357.
Фрей, Питер Уильям (Frey, Peter William), 676.
Фрейсс, Генри (Fraisse, Henri), 745.
Френкель, Авизри С. (Fraenkel, Aviezri S. (אביעזרי פרנקל)), 800, 808.
Фрид, Эдвин Карл (Freed, Edwin Earl), 225.
Фридман, Стивен Джеффри (Friedman, Steven Jeffrey), 720.
Фридшал, Ричард (Fridshal, Richard), 122, 624.
Фриз, Ральф Стенли (Freese, Ralph Stanley), 873.
Фристедт, Берт (Fristedt, Bert), 837.
Фрэнк, Грей (Gray, Frank), 333.
Функции
  Крускала–Маколея, 428.
  от пяти переменных, 134, 156.
  от четырех переменных, 128, 157.
  Радемахера, 336, 762.
  Уолша, 335, 361.
Функциональная композиция, 275, 312.
Функция
  ν, 314.
  Бесселя, 453.
  ветвления, 222, 226.
  дерева, 480, 856.
  застежки, 180, 227, 286.
  Крома, 85, 104, 108, 123, 312.
  Крускала, 537.
    κ_t, 440.

λ_t, 440.
Маколея
μ, 820.
μ_t, 440.
монотонной функции, 265, 310.
обращения к памяти, 254.
ограниченно растущей строки, 488.
остовного дерева, 325.
от многих переменных, 151.
Пейли, 361.
перестановки, 281.
Радемахера, 441.
распределения вероятностей, 455.
связности, 325.
сравнения, 150, 644.
Такаги, 441.
Хорна, 82, 104, 112, 123, 312, 318, 736, 750.
определенная, 123.
переименованная, 114.
четности, 104, 122, 133, 193, 232, 631.
не дифференцируемая ни в одной точке, 441.

Хагауэр, Иоганн (Hagauer, Johann (= Hans)), 95.
Хадсон, Ричард Говард (Hudson, Richard Howard), 656.
Хайнен, Франц (Heinen, Franz), 36.
Хайт, Стюарт Ли (Hight, Stuart Lee), 150.
Хак Госпера, 167, 223.
Хакен, Армин (Haken, Armin), 608.
Хакен, Вольфганг (Haken, Wolfgang), 37.
Хакими, Сефолла Луи (Hakimi, Seifollah Louis), 573.
Хамли, Уильям, и сыновья (Hamley, William, and sons), 772.
Хаммер, Петер Ласло (Hammer, Péter László (= Peter Leslie = Ivănescu, Petru Ladislav)), 150, 303, 606.
Хаммонд, Элеонор Прескотт (Hammond, Eleanor Prescott), 556.
Хаммонс, Артур Роджер-мл. (Hammons, Arthur Roger, Jr.), 758.
Хансель, Джордж (Hansel, Georges), 518, 540.
Хант, Гарри Бовен (Hunt, Harry Bowen, III), 711.
Хант, Нейл (Hunt, Neil), 687.
Хантер, Джеймс Алстон Хоуп (Hunter, James Alston Hope), 375.
Хантингтон, Эдвард Вермайл (Huntington, Edward Vermilye), 614.
Характеристический полином, 543, 692, 761, 891.
булевой функции, 252.
Харари, Фрэнк (Harary, Frank), 16, 37, 571, 586, 887, 895.
Харди, Годфри Гарольд (Hardy, Godfrey Harold), 453, 454, 466, 467, 655, 831, 839.
Харе, Девид Эрвин Джордж (Hare, David Edwin George), 856.
Харел, Дов (Harel, Dov (דב הראל)), 200.
Харриот, Томас (Harriot, Thomas), 567.
Хартли, Вильям Эрнест (Hartley, William Ernest), 564, 575.
Хартмут, Хенниг Фридольф (Harmuth, Henning Friedolf), 335.
Хастад, Йохан Торкель (Håstad, Johan Torkel), 121, 672.
Хатчинсон, Джордж Аллен (Hutchinson, George Allen), 472, 488.
Хаффман, Дэвид Альберт (Huffman, David Albert), 122, 753.
Хачиян, Леонид Генрихович (Khachiyan, Leonid Genrikhovich), 602, 603, 750.
Хачтел, Гери Дин (Hachtel, Gary Deane), 152.
Хаяси, Цуруичи (Hayashi, Tsuruichi (林鶴一)), 901.
Хвостовой коэффициент, 844.
Хедаят, Самад (Hedayat, Samad (= Abdossamad, عبدالصمد هدایت)), 580, 581.
Хедетниеми, Сара Ли Митчелл (Hedetniemi, Sarah Lee Mitchell), 520.
Хейгерап, Торбен (Hagerup, Torben), 669.
Хеккель, Пол Чарльз (Heckel, Paul Charles), 662.
Хелл, Павол (Hell, Pavol), 100.
Хеллерман, Лео (Hellerman, Leo), 133.
Хенричи, Питер Карл Юджин (Henrici, Peter Karl Eugen), 452.
Хенсель, Курт Вильгельм Себастьян (Hensel, Kurt Wilhelm Sebastian), 590.
Херрманн, Франсин (Herrmann, Francine), 204.
Хеш-таблица, 263.
Хеширование, 268.
Хёльберт, Гленн Холанд (Hurlbert, Glenn Howland), 780, 802, 827.
Хиккерсон, Дин Роберт (Hickerson, Dean Robert), 536, 739.
Хикки, Томас Батлер (Hickey, Thomas Butler), 424, 815.
Хилевиц, Едидя (Hilewitz, Yedidya), 227.
Хилтон, Энтони Джон Вильям (Hilton, Anthony John William), 440, 820.
Хиндман, Нейл Брюс (Hindman, Neil Bruce), 616.
Хип, Брайан Ричард (Heap, Brian Richard), 382, 384, 391, 400, 404, 784.
Хип, Марк Эндрю (Heap, Mark Andrew), 699.
Хлавичка, Ян (Hlavička, Jan), 79.
Хо, Чин-Чанг Дэниел (Ho, Chih-Chang Daniel (何志昌)), 843.
Хоар, Артур Говард Мэлорти (Hoare, Arthur Howard Malortie), 843.
Хобби, Джон Дуглас (Hobby, John Douglas), 217, 686.
Ходжес, Джозеф Лоусон-мл. (Hodges, Joseph Lawson, Jr.), 882.
Ходллис, Джеффри Джон (Hollis, Jeffrey John), 233.
Хойн, Фолкер (Heun, Volker), 677.

Хокинг, Стивен Уильям (Hawking, Stephen William), 913.
Холл, Маршалл-мл. (Hall, Marshall Jr.), 10, 577, 581, 834, 849.
Холл, Ричард Уэсли-мл. (Hall, Richard Wesley, Jr.), 209, 236.
Холмс, Томас Шерлок Скотт (Holmes, Thomas Sherlock Scott), 19.
Хольтон, Дерек Аллан (Holton, Derek Allan), 596, 750.
Хольцман Пуассон, Карлос Альфонсо (Holzmann Poisson, Carlos Alfonso), 887.
Хонда, Тошиаки (Honda, Toshiaki (本田利明)), 566, 575.
Хопкрофт, Джон Эдвард (Hopcroft, John Edward), 302, 760.
Хорей, 550, 563.
Хорияма, Такаши (Horiyama, Takashi (堀山貴史)), 711.
Хорн, Альфред (Horn, Alfred), 82, 114.
Хорн, Гевин Бернард (Horn, Gavin Bernard), 754.
Хортон, Роберт Элмер (Horton, Robert Elmer), 545, 584, 896.
Хосака, Казухиса (Hosaka, Kazuhisa (保坂和寿)), 721.
Хоффдинг, Вассили (Hoeffding, Wassily), 860.
Хоффман, Алан Джером (Hoffman, Alan Jerome), 587.
Храпченко, Валерий Михайлович, 635.
Хроматическое число, 57, 68, 70.
Хэй, Джон (Haigh, John), 484.
Хэмминг, Ричард Уэсли (Hamming, Richard Wesley), 754.

Цветкович, Драгош Младен (Cvetković, Dragoš Mladen), 528, 894.
Целочисленное мультилинейное представление, 107, 121, 621.
Целочисленное программирование, 8.
Центр тяжести, 346.
Центроид, 541.
Цепочка, 405, 517.
Цепочки подмультимножеств, 885.
Цепь Эйлера, 559, 802, 826, 827, 863.
Цзянь, Роберт Дянь Вень (Chien, Robert Tien Wen (錢天聞)), 582.
Циглер, Гюнтер Маттиас (Ziegler, Günter Matthias), 871.
Цикл, 12, 32, 54, 64, 65, 67, 299, 323, 815, 845.
  Грея, 341.
  де Брейна, 174, 351–357, 367, 550, 699, 733, 780.
  колеблющейся диаграммы, 491.
  универсальный, 443, 559, 574.
Циклическая запись перестановки, 377.
Циклическая перестановка, 406, 494, 535.
Циклические числа Стирлинга, 860.
Циклический сдвиг, 355, 388, 390, 393, 538, 662, 667.
  вправо, 225.
Циклический список, 233, 683.
Циклы в графе, 180.
  генерация всех, 299.
Цилиндр, 49, 206, 528, 544, 679, 893.
Циммерманн, Поль Винсент Мари (Zimmermann, Paul Vincent Marie), 663.
Цифры, санскрит, 552.
Цубои, Тейичи (Tsuboi, Teiichi (坪井定一)), 619, 621, 625.
Цузе, Конрад (Zuse, Konrad), 637.
Цукияма, Шуджи (Tsukiyama, Shuji (築山修治)), 674.

Чайлдс, Рой Сидней (Childs, Roy Sydney), 785.
Чамберс, Эфраим (Chambers, Ephraim), 6.
Чанг Грэхем, Фан Ронг Кинг (Chung Graham, Fan Rong King (鍾金芳蓉)), 615, 826, 863.
Чанг, Ангел Ксуан (Chang, Angel Xuan (章玄)), 727.
Чанг, Кай Лаи (Chung, Kai Lai (鍾開萊)), 882.
Чандра, Ашок Кумар (Chandra, Ashok Kumar (अशोक कुमार चन्द्रा)), 605, 624, 696.
Частично определенная функция, 143, 162, 163.
Частично симметричные булевы функции, 316.
Частичное упорядочение, 394, 405.
Частное, 321.
Чватал, Вацлав (Chvátal, Václav (= Vašek)), 33.
Чебышев, Пафнутий Львович, 688, 858.
Чезаре, Джулио (=Цезарь, Юлий) (псевдоним Дани Феррари, Луиджи Рафаиани, Луиджи Морелли и Дарио Ури) (Cesare, Giulio (pen name of Dani Ferrari, Luigi Rafaiani, Luigi Morelli, and Dario Uri)), 786.
Чейз, Филлип Джон (Chase, Phillip John), 419, 424, 436, 437, 814.
Чен, Вильям Юнг-Чуан (Chen, William Yong-Chuan (陈永川)), 850.
Чен, Йирнг-Ан (Chen, Yirng-An (陳盈安)), 752.
Чен, Ку-Цай (Chen, Kuo-Tsai (陳國才)), 355.
Ченг, Мэтью Чао (Cheong, Matthew Chao (張宙)), 704.
Ченг, Чинг-Сю (Cheng, Ching-Shui (鄭清水)), 770.
Червяк, 498, 510, 530, 531, 864, 871.
Черны, Карл (Czerny, Carl), 816.
Черчение на растре, 217.
Четная перестановка, 374.
Четность, 127, 162, 163, 250, 358, 652, 659.
Чеширский кот, 210, 236, 682.
Чжоу, Вэнь-ван, *см.* Вэнь-ван (Zhou Wenwang, *см.* King Wen), 547.
Чжоу, Чжаю Конг (Chow, Chao Kong (周紹康)), 103.
Чжун, Цзинь-Мань (Chung, Kin-Man (鍾建民)), 182, 227.
Чима, Махиндр Сингх (Cheema, Mohindar Singh), 487, 861.
Чимани, Маркус (Chimani, Markus), 595.

Числа
  Белла, 474, 490–495, 567, 842, 845, 901.
    асимптотическое значение, 478.
  Бернулли, 474, 730, 833, 906.
  Дженоччи, 730, 734.
  Евклида, 733.
  Каталана, 326, 508, 514, 536, 571, 755, 819.
    таблицы, 508.
  Люка, 695.
  Перрина, 602, 695, 714.
  со смешанным основанием, 349, 781.
  Стирлинга, 567.
    асимптотика, 480, 566.
  Улама, 234.
  Фибоначчи, 204, 291, 366, 407, 633, 635, 695, 752, 797, 897, 906.
    с отрицательными индексами, 204.
  Шрёдера, 539, 895.
  Эйлера, 495, 539, 833.
Число независимости, 57.
Чистый буквометик, 376, 488.
Чосер, Джеффри (Chaucer, Geoffrey), 583.
Чун-порядок, 437, 504.
Чуэнте, Морис (Tchuente, Maurice), 792.

Шаблон рождественской елки, 515, 539, 624, 883.
Шаллит, Джеффри Аутлав (Shallit, Jeffrey Outlaw), 658, 854.
Шао Юн (Shao Yung (邵雍)), 548.
Шапиро, Гарольд Сеймур (Shapiro, Harold Seymour), 363.
Шары, 444.
Шаффлер, Теодор Генрих Отто (Schäffler, Theodor Heinrich Otto), 333.
Шахматы, 35, 46, 198, 234, 323, 325.
Швенк, Аллен Джон (Schwenk, Allen John), 895.
Швец, Франк Йозеф (Swetz, Frank Joseph), 548.
Шекспир, Уильям (Shakespeare (= Shakspere), William), 19, 493.
Шен, Винсент Юн-Шен (Shen, Vincent Yun-Shen (沈運申)), 148, 150, 162.
Шеннон, Клод Элвуд-мл. (Shannon, Claude Elwood, Jr.), 71, 139, 302, 626, 637.
Шенстед, Крейг Юджин (Schensted, Craige Eugene), 114, 115, 116, 540, 612.
Шестнадцатеричные константы, 12, 905.
Шестнадцатеричные цифры, 240, 378.
Шефер, Томас Джером (Schaefer, Thomas Jerome), 97.
Шеффер, Генри Морис (Sheffer, Henry Maurice), 74, 106.
Ши, Жи-Жи Джерри (Shi, Zhi-Jie Jerry (史志杰)), 664.
Шибер, Барух Менахем (Schieber, Baruch Menachem (ברוך מנחם שיבר)), 200.
Шилдс, Ян Бомон (Shields, Ian Beaumont), 816.
Шиллингер, Иосиф Моисеевич (Schillinger, Joseph Moiseyevich), 559, 574.
Ширакава, Исао (Shirakawa, Isao (白川功)), 674.
Широкословная
  инструкция, 304.
  цепочка, 189, 232, 304, 678.
    сильная, 231.
Шихан, Джон (Sheehan, John), 596.
Шлово, 187.
Шляфли, Людвиг (Schläfli, Ludwig), 679.
Шмидт, Эрик Майнехе (Schmidt, Erik Meineche), 302.
Шмитт, Питер Ганс (Schmitt, Peter Hans), 611.
Шмулевич, Илья Владимирович (Shmulevich, Ilya Vladimir), 626.
Шнайдер, Бернадетт (Schneider, Bernadette), 771.
Шнорр, Клаус-Петер (Schnorr, Claus-Peter), 163.
Шоландер, Марлоу Кэнон (Sholander, Marlow Canon), 117, 614.
Шоломов, Лев Абрамович, 160.
Шотт, Рене Пьер (Schott, René Pierre), 882.
Шрёдер, Фридрих Вильгельм Карл Эрнст (Schröder, Friedrich Wilhelm Karl Ernst), 106, 600.
Шрёппель, Ричард Крабтри (Schroeppel, Richard Crabtree), 157, 192, 221, 662.
Шрикханде, Шарадчандра Шанкар (Shrikhande, Sharadchandra Shankar (शरदचन्द्र शंकर श्रीखंडे)), 24.
Штайн, Шерман Копальд (Stein, Sherman Kopald), 640.
Штейнер, Якоб (Steiner, Jacob), 36.
Штейнеровская система троек, 27.
Штрассен, Фолькер (Strassen, Volker), 649.
Шумахер, Генрих Кристиан (Schumacher, Heinrich Christian), 23, 36.
Шур, Иссай (Schur, Issai), 839.
Шутен, Франс ван (Schooten, Frans van), 560, 574.
Шютценбергер, Марсель Пауль (Schützenberger, Marcel Paul), 427, 777, 820, 825.

Эббенхорст Тенгберген, Корнелия ван (van Ebbenhorst Tengbergen, Cornelia), 515.
Эва Тёрёк (Török, Éva), 814.
Эвинг, Энн Кэтрин (Ewing, Ann Catherine), 605.
ЭВМ,
  EDSAC, 165, 176.
  ILLIAC I, 176.
  Manchester Mark I, 165.
  Mark II (Manchester/Ferranti), 165.
  БЭСМ-6, 663.
Эволюционные деревья, 856.
Эгиазарян, Карен (Egiazarian, Karen), 626.
Эдельман, Поль Генри (Edelman, Paul Henry), 873.

Эджворт, Френсис Исидро (Edgeworth, Francis Ysidro), 858.
Эйлер, Леонард (Ейлеръ, Леонардъ; Euler, Leonhard), 22, 25, 26, 58, 450, 459, 464, 465, 569, 580, 730, 841.
Эйроллс, Жорж (Eyrolles, Georges), 896.
Эйтер, Томас Роберт (Eiter, Thomas Robert), 728.
Эквивалентность, 73, 233, 599.
по отношению к перестановкам, 104.
по отношению к перестановкам и дополнениям, 104.
Экин, Оя (Ekin, Oya), 606.
Экланд, Брайан Дэвид (Ackland, Bryan David), 212.
Экспоненциальная производящая функция, 854.
Экспоненциальный рост, 267, 279, 282, 451.
Экспоненциальный ряд, частичные суммы, 782.
Экхофф, Юрген (Eckhoff, Jürgen), 820.
Элгот, Келвин Крестон (Elgot, Calvin Creston), 155, 618.
Электровакуумная лампа, 133, 157.
Элемент, 547.
Элементы (земля, воздух, огонь, вода), 558.
Элемментарные симметричные функции, 839.
Эллипс, 212, 514, 685.
Эллиптические функции, 453.
Эмде Боас, Петер ван (Emde Boas, Peter van), 198.
Энггрен, Вилли (Enggren, Willy), 398, 399.
Энгель, Конрад Вольфганг (Engel, Konrad Wolfgang), 826.
Эндрюс, Джордж В. Эйр (Andrews, George W. Eyre), 446, 835.
Эннс, Теодор Кристиан (Enns, Theodore Christian), 815.
Эппштейн, Дэвид Артур (Eppstein, David Arthur), 692, 785.
Эр, Менг Чу (Er, Meng Chiau (余明昭)), 385, 867.
Эратосфен из Кирена (Eratosthenes of Cyrene (Ἐρατοσθένης ὁ Κυρηναῖος)), 224.
Эрдеш, Пал (Erdős, Pál (= Paul)), 427, 444, 455, 466, 572, 591, 593, 595, 616, 621, 794.
Эрдеш, Петер Л. (Erdős, Péter L.), 845.
Эрикссон, Киммо (Eriksson, Kimmo), 690.
Эрикссон, Хенрик (Eriksson, Henrik), 690.
Эрлих, Гидеон (Ehrlich, Gideon (גדעון ארליך)), 338, 388, 402, 416, 463, 473, 474, 784, 810, 853.
Эррера, Альфред (Errera, Alfred), 864.
Эттингсхаузен, Андреас фон (Ettingshausen, Andreas von), 570.
Этцион, Туви (Etzion, Tuvi (טובי עציון, born טובי הולצר)), 354.
Этьен, Гвен (Etienne, Gwihen), 843.
Эшер, Джорджио Арнальдо (Escher, Giorgio Arnaldo (= George Arnold)), 204.
Эшер, Маурис Корнелис (Escher, Maurits Cornelis), 204.

Юен, Чанг Квонг (Yuen, Chung Kwong (阮宗光)), 759.

Яблонский, Сергей Всеволодович, 624.
Яджима, Шузо (Yajima, Shuzo (矢島脩三)), 700, 721, 753.
Ядро, 83, 113, 273, 294, 305, 430, 441, 608, 611.
Язык программирования, 83.
Язык, регулярный, 231.
Якоби, Карл Густав Якоб (Jacobi, Carl Gustav Jacob), 451, 465.
Якубович, Юрий Владимирович, 458, 484.
Ян Сюн (Yang Hsiung (揚雄 or 楊雄)), 548.
Ян, Катарина Хуафей (Yan, Catherine Huafei (颜华菲)), 850.
Яниш, Карл Фридрих Андреевич, де (Jaenisch, Carl Friedrich Andreevitch de (Янишъ Карлъ Андреевичъ)), 748.
Яннакакис, Михалис (Yannakakis, Mihalis (Γιαννακάκης, Μιχάλης)), 674.
Яно, Тамаки (Yano, Tamaki (矢野環)), 565, 566.
Янсон, Карл Сванте (Janson, Carl Svante), 69, 597.
Японская математика, 553, 565, 756.
Ятс, Фрэнк (Yates, Frank), 337.

ИСКУССТВО ПРОГРАММИРОВАНИЯ

ТОМ 1. ОСНОВНЫЕ АЛГОРИТМЫ,
3-Е ИЗДАНИЕ

ТОМ 2. ПОЛУЧИСЛЕННЫЕ МЕТОДЫ,
3-Е ИЗДАНИЕ

ТОМ 3. СОРТИРОВКА И ПОИСК,
2-Е ИЗДАНИЕ

Дональд Э. Кнут

в продаже

www.williamspublishing.com

На мировом рынке компьютерной литературы существует множество книг, предназначенных для обучения основным алгоритмам и используемых при программировании. Их довольно много, и они в значительной степени конкурируют между собой. Однако среди них есть особая книга.
Это трехтомник *“Искусство программирования”* Д. Э. Кнута, который стоит вне всякой конкуренции, входит в золотой фонд мировой литературы по информатике и является настольной книгой практически для всех, кто связан с программированием.
Ценность книги в том, что она предназначена не столько для обучения технике программирования, сколько для обучения, если это возможно, “искусству” программирования, предлагает массу рецептов усовершенствования программ и, что самое главное, учит самостоятельно находить эти рецепты.

Алгоритмы: построение и анализ *3-е издание*

Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн

www.williamspublishing.com

Фундаментальный труд известных специалистов в области информатики достоин занять место на полке любого человека, чья деятельность так или иначе связана с вычислительной техникой и алгоритмами. Для профессионала эта книга может служить настольным справочником, для преподавателя — пособием для подготовки к лекциям и источником интересных нетривиальных задач, для студентов и аспирантов — отличным учебником.

Строгий математический анализ и обилие теорем сопровождаются большим количеством иллюстраций, элементарными рассуждениями и простыми приближенными оценками. Широта охвата материала и степень строгости его изложения дают основания считать эту книгу одной из лучших книг, посвященных разработке и анализу алгоритмов.

Третье издание этого классического труда в большой степени доработано. В нем появились новые главы, в том числе посвященные такой важной в последнее время теме, как многопоточные алгоритмы, а старые подверглись переработке, местами весьма существенной, когда уже имевшийся во втором издании материал излагается с иных позиций, чем ранее.

Данная книга будет не лишней как на столе студента и аспиранта, так и на рабочей полке практикующего программиста.

ISBN 978-5-8459-1794-2 **в продаже**

АЛГОРИТМЫ НА C++
АНАЛИЗ, СТРУКТУРЫ ДАННЫХ, СОРТИРОВКА, ПОИСК, АЛГОРИТМЫ НА ГРАФАХ

Роберт Седжвик

www.williamspublishing.com

Эта классическая книга удачно сочетает в себе теорию и практику, что делает ее популярной у программистов на протяжении многих лет. Кристофер Ван Вик и Роберт Седжвик разработали новые лаконичные реализации на C++, которые естественным и наглядным образом описывают методы и могут применяться в реальных приложениях. Каждая часть содержит новые алгоритмы и реализации, усовершенствованные описания и диаграммы, а также множество новых упражнений для лучшего усвоения материала. Акцент на АТД расширяет диапазон применения программ и лучше соотносится с современными средами объектно-ориентированного программирования. Книга предназначена для широкого круга разработчиков и студентов.

ISBN 978-5-8459-1650-1

в продаже

АЛГОРИТМЫ НА JAVA
4-Е ИЗДАНИЕ

Роберт Седжвик,
Кевин Уэйн

www.williamspublishing.com

Последнее издание из серии бестселлеров Седжвика, содержащее самый важный объем знаний, наработанных за последние несколько десятилетий. Содержит полное описание структур данных и алгоритмов для сортировки, поиска, обработки графов и строк, включая пятьдесят алгоритмов, которые должен знать каждый программист. В книге представлены новые реализации, написанные на языке Java в доступном стиле модульного программирования — весь код доступен пользователю и готов к применению. Алгоритмы изучаются в контексте важных научных, технических и коммерческих приложений. Клиентские программы и алгоритмы записаны в реальном коде, а не на псевдокоде, как во многих других книгах. Дается вывод точных оценок производительности на основе соответствующих математических моделей и эмпирических тестов, подтверждающих эти модели.

ISBN 978-5-8459-1781-2

в продаже

КОНКРЕТНАЯ МАТЕМАТИКА
2-е издание

Рональд Л. Грэхем,
Дональд Э. Кнут,
Орен Паташник

www.williamspublishing.com

В основу данной книги положен одноименный курс лекций Станфордского университета. Название “конкретная математика” происходит от слов “КОНтинуальная” и “дисКРЕТНАЯ” математика. Назначение данной книги — обеспечить читателя техникой оперирования с дискретными объектами, что совершенно необходимо для математиков, работающих в области информатики. Книга ориентирована в первую очередь на практиков (хотя и теоретики найдут в ней много полезного), и изобилует массой конкретных примеров и упражнений. Конкретность изложения абстрактного материала — еще одно пояснение названия книги. Широта охвата столь различных тем в одной книге могла бы вызвать подозрения в некоторой легковесности, если бы не имена ее авторов — известных американских математиков. Тем не менее слово “легкий” к книге вполне применимо, так как стиль изложения достаточно далек от сухого академизма. Как признаются сами авторы, они считают математику развлечением, и они сделали все, чтобы читатели книги получили от ее прочтения не только знания, но и удовольствие.

Книгу можно рекомендовать всем математикам, но в первую очередь она предназначена для студентов, обучающихся математике и информатике.

ISBN 978-5-8459-1588-7

в продаже

Искусство программирования

ДОНАЛЬД Э. КНУТ

Дональд Э. Кнут — автор всемирно известной серии книг, посвященной основным алгоритмам и методам вычислительной математики, а также создатель настольных издательских систем TEX и METAFONT, предназначенных для верстки физико-математической литературы. Его перу принадлежат 26 книг и более 160 статей. Дональд Кнут является почетным профессором Станфордского университета в области программирования и вычислительной математики. В настоящее время он полностью занят написанием новых книг серии *Искусство программирования*. Работу над первым томом он начал еще в 1962 году, сразу после окончания Калифорнийского технологического института (California Institute of Technology).
Профессор Кнут удостоен многочисленных премий и наград, среди которых можно отметить ACM Turing Award, Medal of Science президента Картера и ASM Steele Prize за серию научно-популярных статей. В ноябре 1996 года Дональд Кнут был удостоен престижной награды Kyoto Prize в области передовых технологий.
Он проживает в городке Станфордского университета со своей женой Джилл. Дополнительная информация об этой книге и будущих томах серии имеется на персональной странице Кнута по адресу `www-cs-faculty.stanford.edu/~knuth`.

Этот многотомный труд широко известен как полное изложение информатики. В течение десятилетий первые три тома служили бесценным источником информации по теории и практике программирования для студентов, теоретиков и практиков. Ученые восхищались красотой и изяществом анализа Кнута, в то время как практикующие программисты успешно применяли его "поваренную книгу" для решения ежедневных задач.

> *Уровень первых трех томов столь высок, и в них проявлено столь широкое и глубокое знакомство с искусством программирования, что вполне достаточным обзором будущих томов будет краткое "Вышел том n Искусства программирования Кнута".*
>
> *— Data Processing Digest*

Вышел том n *Искусства программирования* Кнута, где n = 4, A.

> *В этом долгожданном новом томе старый мастер уделяет внимание как ряду своих издавна любимых тем — широкословным вычислениям и комбинаторной генерации (исчерпывающему перечислению фундаментальных комбинаторных объектов, таких как перестановок, разбиений или деревьев), так и более поздним увлечениям, таким как бинарные диаграммы решений.*
>
> *Признаки качества, отличающие его прежние тома, проявились и в новом томе: детальное описание основ, иллюстрация хорошо подобранными примерами, иногда экскурсы в более эзотеричные темы и задачи на острие ведущихся исследований; безупречный стиль изложения, приправленный долей юмора; обширные наборы упражнений — все с решениями или полезными указаниями; должное внимание историческим вопросам; реализация множества алгоритмов в его классическом пошаговом стиле.*
>
> *На каждой странице книги имеется удивительное количество информации. Очевидно, Кнут долго и тщательно размышлял о том, какие результаты являются наиболее центральными и важными, и о том, как наиболее интуитивно понятно и кратко изложить этот материал. Поскольку области, охваченные этим томом, увеличились с момента первых черновых заметок о них просто взрывным образом, это просто удивительно — как он сумел втиснуть столь тщательное рассмотрение в такой небольшой объем.*
>
> *— Фрэнк Раски (Frank Ruskey), факультет информатики университета Виктории (Department of Computer Science, University of Victoria)*

Эта книга представляет собой том 4, A, поскольку сам том 4 является многотомником. Комбинаторный поиск — богатая и важная тема, и Кнут приводит слишком много нового, интересного и полезного материала, чтобы его можно было разместить в одном или двух (а может быть, даже в трех) томах. Одна эта книга включает около 1500 упражнений с ответами для самостоятельной работы, а также сотни полезных фактов, которые вы не найдете ни в каких других публикациях. Том 4, A определенно должен занять свое место на полке рядом с первыми тремя томами этой классической работы в библиотеке каждого серьезного программиста.

ТОМ 1. ОСНОВНЫЕ АЛГОРИТМЫ.

Третье издание, ISBN 978-5-8459-0080-7

Первый том серии книг *Искусство программирования* начинается с описания основных понятий и методов программирования. Затем автор переходит к рассмотрению информационных структур — представлению информации внутри компьютера, структурных связей между элементами данных и способам эффективной работы с ними. Для методов имитации, символьных вычислений, числовых методов, методов разработки программного обеспечения даны примеры элементарных приложений.

ТОМ 2. ПОЛУЧИСЛЕННЫЕ АЛГОРИТМЫ

Третье издание, ISBN 978-5-8459-0081-4

Во втором томе представлено полное введение в теорию получисленных алгоритмов, причем случайным числам и арифметике посвящены отдельные главы. В книге даны основы теории получисленных алгоритмов, а также их основные примеры. Тем самым установлено прочное связующее звено между компьютерным программированием и численным анализом.

ТОМ 3. СОРТИРОВКА И ПОИСК

Второе издание, ISBN 978-5-8459-0082-1

Во втором издании третьего тома содержится исчерпывающий обзор классических алгоритмов сортировки и поиска. Представленная в нем информация дополняет приведенное в первом томе обсуждение структур данных. Автор рассматривает принципы построения больших и малых баз данных, а также внутренней и внешней памяти.

ТОМ 4, A. КОМБИНАТОРНЫЕ АЛГОРИТМЫ, часть 1

ISBN 978-5-8459-1980-9

В этом томе рассматриваются методы, позволяющие компьютерам эффективно работать с задачами гигантского размера. Рассматриваемый материал начинается с булевых функций и технологий и трюков работы с битами, затем всесторонне рассматривается генерация всех кортежей и перестановок, всех сочетаний и разбиений, и всех деревьев.

Addison-Wesley
Addison Wesley Corporate & Professional
is an imprint of Addison Wesley Longman, Inc.
www.aw.com/cseng/authors/knuth

ВИЛЬЯМС
Издательский дом "Вильямс"
www.williamspublishing.com

ISBN 978-5-8459-1980-9
15059
9 785845 919809