Лекции по дискретной математике

М. Вялый В. Подольский А. Рубцов Д. Шварц А. Шень

Оглавление

Предисловие **8**

I Начальные примеры **10**

1 Математическая индукция **11**
1.1 Задача о раскраске плоскости . . . 11
1.2 Общая схема доказательств по индукции . . . 15
1.3 Варианты рассуждений по индукции . . . 16
1.3.1 С чего начинать? . . . 16
1.3.2 Сведение к меньшим . . . 18
1.3.3 Переформулировка: принцип наименьшего числа . . . 19
1.4 Как не надо . . . 20
1.5 Как догадаться, что доказывать? . . . 22
1.6 Доказательства по индукции и без . . . 25
1.7 Индукция и рекурсия . . . 28
1.8 Доказательства неравенств по индукции . . . 31
1.8.1 Неравенство Бернулли . . . 31
1.8.2 Среднее арифметическое и геометрическое . . . 32
1.9 Пример из алгебры: системы однородных уравнений . . . 35
1.10 Коды Грея . . . 38
1.11 Теорема Холла о представителях . . . 40
1.12 Задачи для самостоятельного решения . . . 42

2 Подсчёты **44**
2.1 Правило суммы . . . 44
2.2 Рекуррентное соотношение: пример . . . 48
2.3 Рекуррентное соотношение: число путей . . . 51
2.4 Слова и правило произведения . . . 53
2.5 Выбор с ограничениями . . . 56
2.6 Подсчёты с кратностью . . . 58
2.7 Подмножества и числа сочетаний . . . 61
2.8 Ещё о числах сочетаний . . . 63

2.8.1 Симметрия . . . 63
2.8.2 Сумма чисел в строке . . . 64
2.8.3 Знакочередующаяся сумма . . . 65
2.8.4 Снова о включениях и исключениях . . . 66
2.8.5 Пути, подмножества, слова . . . 67
2.8.6 Соседние числа в строке . . . 69
2.8.7 Монеты и перегородки . . . 70
2.9 Бином Ньютона и производящие функции . . . 72
2.10 Числа Каталана . . . 80
2.10.1 Правильные последовательности скобок . . . 80
2.10.2 Рекуррентная формула . . . 82
2.10.3 Вычисление с помощью отражений . . . 84
2.10.4 Вычисление с производящей функцией . . . 86
2.10.5 Вычисление с теорией вероятностей и поворотами . . . 88
2.10.6 Доказательство по индукции с дробными параметрами . . . 89
2.10.7 Неассоциативные произведения, триангуляции и стековый калькулятор . . . 90
2.11 Что дальше? . . . 94

3 Графы **95**
3.1 Примеры . . . 95
3.1.1 Граф авиарейсов . . . 95
3.1.2 Перестановка коней . . . 96
3.1.3 Эйлер и мосты в Кёнигсберге . . . 98
3.1.4 Рукопожатия . . . 99
3.1.5 Задачи и решения . . . 100
3.1.6 Разбор контрольной* . . . 101
3.1.7 Знакомые и незнакомые . . . 103
3.2 Неориентированные графы . . . 105
3.2.1 Определение . . . 106
3.2.2 Соседи. Степени вершин . . . 107
3.2.3 Связные компоненты . . . 110
3.2.4 Расстояния. Простые пути . . . 117
3.2.5 Деревья . . . 119
3.2.6 Полное бинарное дерево . . . 123
3.3 Ориентированные графы . . . 124
3.3.1 Определение . . . 124
3.3.2 Степени вершин . . . 125
3.3.3 Пути и достижимость . . . 125
3.3.4 Достижимость и разрезы . . . 126
3.3.5 Компоненты сильной связности и ациклические графы . . . 128
3.3.6 Графы преобразований . . . 130
3.4 Эйлеровы циклы . . . 132

3.4.1 Определение . . . 132
3.4.2 Критерий существования . . . 132
3.4.3 Последовательности де Брёйна . . . 134
3.4.4 Гамильтоновы циклы . . . 135
3.5 Двудольные графы . . . 135
3.5.1 Определение . . . 135
3.5.2 Двудольные графы и раскраска в два цвета . . . 136
3.5.3 Степени вершин . . . 137
3.5.4 Паросочетания . . . 138
3.6 Клики и независимые множества . . . 140

4 Арифметика остатков **143**
4.1 Чётные и нечётные числа . . . 143
4.2 Деление на 3 и остатки . . . 144
4.3 Деление с остатком . . . 144
4.4 Сравнения по модулю . . . 147
4.5 Таблицы сложения и умножения по модулю N . . . 149
4.6 Обратимые остатки по модулю N . . . 151
4.7 Обратимые элементы и диофантовы уравнения . . . 153
4.8 Алгоритм Евклида . . . 154
4.9 Алгоритм Евклида и диофантовы уравнения . . . 156
4.10 Однозначность разложения на множители . . . 159
4.11 Китайская теорема об остатках . . . 161
4.12 Малая теорема Ферма . . . 162
4.13 Функция Эйлера и теорема Эйлера . . . 165
4.14 Что дальше? . . . 166

II Основные конструкции **168**

5 Множества и логика **169**
5.1 Основные свойства множеств и операции с множествами . . . 169
5.2 Теоретико-множественные тождества . . . 174
5.3 Логические переменные, логические связки . . . 177
5.4 Наблюдения . . . 180
5.5 Какие связки необходимы? . . . 183
5.5.1 Полнота дизъюнкции, конъюнкции и отрицания . . . 185
5.5.2 Полнота конъюнкции и отрицания . . . 186
5.5.3 Алгебраическое доказательство теоремы 5.1 . . . 186
5.6 Формула включений-исключений . . . 187
5.6.1 Первое доказательство . . . 188
5.6.2 Второе доказательство . . . 189
5.6.3 Формула для симметрической разности . . . 190

6 Функции **191**
6.1 Пример . . . 191
6.2 Функции и связанные с ними понятия . . . 192
6.2.1 Терминология и обозначения . . . 192
6.2.2 Образ множества, полный прообраз . . . 194
6.3 Декартово произведение множеств и графики функций . . . 197
6.4 Инъекции, сюръекции и биекции . . . 201
6.4.1 Определения . . . 201
6.4.2 Биекции и сравнение множеств . . . 204
6.5 Композиции функций . . . 206
6.5.1 Определение . . . 206
6.5.2 Ассоциативность . . . 208
6.5.3 Обратная функция . . . 208
6.5.4 Степени композиций . . . 210
6.6 Подсчёты . . . 212
6.7 Задачи для самостоятельного решения . . . 214

7 Отношения и их графы **216**
7.1 Отношения в естественном языке . . . 216
7.2 Отношения с точки зрения математики . . . 217
7.3 Свойства бинарных отношений . . . 219
7.4 Графы, матрицы и бинарные отношения . . . 221
7.5 Отношения эквивалентности . . . 221
7.6 Композиция отношений . . . 224
7.7 Отношения: что дальше? . . . 227
7.8 Задачи для самостоятельного решения . . . 228

8 Мощность множеств **229**
8.1 Равномощные множества . . . 229
8.1.1 Определение равномощности . . . 229
8.1.2 Свойства равномощности . . . 230
8.1.3 Примеры равномощных множеств . . . 231
8.2 Счётные множества . . . 233
8.2.1 Определение и простейшие примеры . . . 233
8.2.2 Свойства счётных множеств . . . 234
8.3 Несчётные множества . . . 238
8.3.1 Интервал и отрезок равномощны . . . 238
8.3.2 Добавление счётного множества . . . 239
8.3.3 Числа и последовательности . . . 240
8.3.4 Отрезок и квадрат . . . 241
8.4 Диагональный аргумент Кантора и сравнение мощностей . . . 242
8.4.1 Несчётность отрезка . . . 242
8.4.2 Сравнение мощностей . . . 244
8.5 Что дальше? . . . 249

9 Упорядоченные множества **250**
9.1 Отношения порядка . . . 250
9.1.1 Отношения строгого частичного порядка . . . 250
9.1.2 Строгие и нестрогие порядки . . . 251
9.2 Примеры . . . 252
9.3 Операции над частично упорядоченными множествами . . . 255
9.4 Какие порядки считать «одинаковыми»? . . . 256
9.5 Конечные линейные порядки . . . 258
9.6 Порядки и индукция . . . 258
9.7 Антицепи . . . 260

10 Вероятность: первые шаги **263**
10.1 Элементарная теория вероятностей: определения . . . 264
10.2 Вероятность объединения событий . . . 272
10.3 Вероятностный метод . . . 275
10.4 Условные вероятности . . . 276
10.5 Случайная величина, математическое ожидание . . . 284
10.6 Частота орлов при подбрасывании монеты и биномиальные коэффициенты . . . 292
10.7 Большие уклонения: неравенство Чернова . . . 296
10.8 Подробности для любознательных . . . 298
10.8.1 Ещё одна элементарная оценка отношения биномиальных коэффициентов . . . 298
10.8.2 Другое доказательство неравенства Чернова . . . 299

11 Комбинаторные игры **302**
11.1 Позиции . . . 302
11.2 Стратегии . . . 305
11.3 Разбор с конца . . . 308
11.4 Симметричные стратегии . . . 312
11.5 Ним . . . 316
11.6 Сумма игр и функция Шпрага–Гранди . . . 319

III Вычислимость **325**

12 Разрешающие деревья **326**
12.1 Задача об угадывании числа. Деление пополам. Мощностная нижняя оценка . . . 326
12.2 Формализация модели . . . 328
12.3 Угадывание числа, неадаптивный вариант задачи . . . 328
12.4 Ограниченные модели разрешающих деревьев. Сортировка, взвешивания, булевы функции . . . 330
12.5 Рассуждение с противником . . . 333

13 Булевы схемы и формулы **337**
13.1 Булевы схемы 338
13.2 Формулы 348

14 Алгоритмическая неразрешимость **353**
14.1 Игра FRACTRAN 353
14.2 Что утверждается? 354
14.3 Отступление о процессорах 355
14.4 Кодирование 356
14.5 Класс вычислимых функций 357
14.6 Определение вычислимости? 358
14.7 Компромисс 358
14.8 Композиция вычислимых функций 359
14.9 Не все функции вычислимы 360
14.10Неразрешимость проблемы остановки 361
14.11Самоприменимость 362
14.12Перечисление останавливающихся программ 363
14.13Как доказать неразрешимость? 364
14.14Язык программирования для доказательства теоремы Конвея 365
14.15Сведение проблемы остановки: от программ к пасьянсам 369

15 Вычислимые функции, разрешимые и перечислимые множества **371**
15.1 Примеры вычислимых функций 373
15.2 Не все функции вычислимы (повторение) 378
15.3 Разрешимые множества 379
15.4 Перечислимые множества 380
15.5 Вычислимость и конечные объекты 387
15.6 Универсальная вычислимая функция 388
15.7 Главная универсальная функция 393
15.8 Теорема Райса – Успенского 397
15.9 Теорема о неподвижной точке 401
15.10Решения задач 404

16 Машины Тьюринга **408**
16.1 Определения 408
16.2 Тезис Чёрча – Тьюринга 412
16.3 Машины Тьюринга и свойства вычислимых функций 413
16.4 Использование машин Тьюринга в доказательствах 415
16.5 Композиция функций, вычислимых по Тьюрингу, и уборка мусора . . 416
16.6 Многоленточные машины Тьюринга 419
16.7 Моделирование многоленточной МТ на одноленточной 422
16.8 Универсальная машина Тьюринга 424
16.9 Универсальная 3-ленточная машина для 1-ленточных машин 426
16.10Соответствие между абстрактной теорией алгоритмов и МТ 429

16.11Машины Тьюринга в доказательствах неразрешимости 433
16.11.1 Задача достижимости на графе подстановок слов 433
16.11.2 Неразрешимость задачи достижимости для графа подстановок слов 434
16.12Решения задач 437

Предисловие

Слова «дискретная математика», входящие в название этой книжки, употребляют в разных значениях. Иногда противопоставляют «дискретную» математику, говорящую о конечных или по крайней мере хорошо различимых объектах, и «непрерывную», где речь идёт о действительных числах, пределах, непрерывности, производных и т.п. Хотя это противопоставление условно и не всегда применимо (скажем, странно было бы разделять «дискретные» алгебраические кривые над конечным полем и «непрерывные» алгебраические кривые над полем комплексных чисел), некоторый смысл оно имеет.

Говоря о «советской школе дискретной математики», имеют в виду немного другое — прежде всего пионерские работы 1950-х и 1960-х годов (О. Б. Лупанов и его школа) по анализу булевых функций, их классов, обобщений на многозначную логику и др.

Наконец, «дискретная математика» как учебный предмет на младших курсах — это сборная солянка из разных понятий и результатов, которые являются частью базовой математической культуры и необходимы будущим математикам и программистам, но не входят в традиционно сложившиеся курсы начального математического цикла (анализ, алгебра, линейная алгебра).

Именно в этом смысле слова «дискретная математика» используются в названии этой книжки, представляющей собой расширенные записки лекций, читавшихся на факультете компьютерных наук Высшей школы экономики. Получилась она разнородной: некоторые темы (скажем, про математическую индукцию или про комбинаторику) — это то, что вполне могло бы изучаться в школе и даже когда-то изучалось.[1] В других случаях целью является освоение некоторого языка (скажем, что такое пересечение множеств или бинарное отношение). Или это может быть прологом к рассказу о некоторой математической теории, попыткой выделить какое-то минимальное содержательное начало, которое имело бы смысл рассказать даже тем, кто в дальнейшем с этим не столкнётся. Или просто какой-то красивый результат, который трудно найти доступно изложенным.

[1] «Гимназист бойко выводил какую-то формулу, со стуком ломая мел о доску, и всё писал, несмотря на то, что профессор уже сказал ему: „Довольно“, — и велел нам взять билеты. „Ну что, ежели достанется теория сочетаний!“ — подумал я, доставая дрожащими пальцами билет из мягкой кипы нарезанных бумажек.» (Лев Толстой, *Юность* из цикла *Детство. Отрочество. Юность*, глава XI, *Экзамен математики*. Странно, но потом герой повести с успехом отвечает про бином Ньютона.)

В каждую «лекцию» (в реальности это могло быть несколько лекций) мы старались включить и достаточно трудный материал, чтобы подготовленным студентам не было скучно. При этом мы не рассчитывали на то, что все это сразу поймут. Такие более трудные места можно и нужно пропускать, если они кажутся непонятными, и двигаться дальше.

Изложение сопровождается задачами; часть из них — это вопросы к слушателям для проверки понимания на лекциях, другие разбирались на семинарах и включались в домашние задания, третьи могут быть предметом самостоятельной работы для заинтересовавшихся студентов и указанием на возможное развитие темы. В общем, как говорил классик, *прими собранье пёстрых глав*...

Черновики книги использовались в преподавании курса «Дискретная математика» на ФКН ВШЭ в 2014–2018 годах. Авторы признательны всем преподавателям и студентам, которые указывали на замеченные неточности в тексте. Практически невозможно вспомнить всех поимённо. Особо отметим В.Ю. Батурина, В.И. Бубнову, С.В. Булгакова, Е.В. Дашкова, Р.Е. Кизилова.

Авторы также благодарны анонимному рецензенту издательства, который сделал много важных замечаний.

Скорее всего, не все неточности и ошибки в книге исправлены. Все они остаются на совести авторов. Мы будем благодарны тем читателям, которые заметят эти неточности и сообщат нам о них.

Часть I

Начальные примеры

Лекция 1

Математическая индукция

Математики часто говорят о «доказательствах по индукции», «принципе математической индукции» и так далее — ничего сложного в этом нет, но нужно к этому привыкнуть. С этим приёмом рассуждений мы будем часто сталкиваться, и начнём с простых примеров.

1.1 Задача о раскраске плоскости

Задача 1.1. На плоскости проведено несколько прямых. Они делят плоскость на области. Докажите, что области можно так раскрасить в два цвета, чтобы соседние области были покрашены в разные цвета.

Соседними считаются области, имеющие общий участок границы (отрезок или луч, точка не считается).

На рисунке 1.1 показана такая раскраска для одной, двух и трёх прямых. Примерно такое обычно рисуют школьники, когда решают эту задачу на математическом кружке — и часто считают, что решили задачу: показали, что раскрасить можно. И действительно, раскраска удовлетворяет поставленным требованиям. Что не так с этим «решением»?

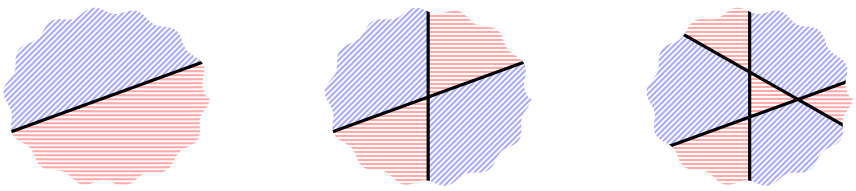


Рис. 1.1: Примеры раскрасок для одной, двух и трёх прямых

Проблема в том, что разобраны не все варианты. Для одной прямой действительно ничего другого быть не может, но уже для двух прямых возможна ситуация, когда

они параллельны. (Понятно, как тогда красить?) Для трёх прямых вариантов ещё больше: некоторые из прямых могут быть параллельными, все три прямые могут проходить через одну точку. Попробуйте перечислить все возможности и убедиться, что в каждом случае требуемая раскраска существует. Нужно быть внимательным, чтобы ничего не пропустить — и чем больше прямых, тем сложнее. Видно, что перебором вариантов задачу не решить, тем более что число прямых может быть сколь угодно большим. Как же быть?

«Ну хорошо, — скажет школьник, недовольный тем, что его решение забраковали. — Давайте я объясню, как раскрашивать в общем случае. Выберем какую-то область и покрасим её в какой-то цвет. Тогда соседние области нужно красить в другой цвет, их соседей снова в первый, и так постепенно всё закрасим. Хотите, нарисуйте прямые, и я так всё закрашу.»

Что на это может возразить преподаватель? Ведь действительно, если нарисовать какие угодно прямые, таким способом можно найти требуемую раскраску. Так что, теперь задача решена?

Увы, нет — проблема в том, не придём ли мы к противоречию. Представьте себе, что мы закрасили начальную область в один цвет, её соседей в другой, соседей соседей снова в первый, и так далее — но дойдя до какой-то области, обнаружили, что она уже граничит с областями разных цветов, и её нельзя закрасить ни так, ни этак. «Так не может быть» — скажет школьник, и он действительно прав в том смысле, что так не бывает. Но это пока не доказано. И, скажем, если бы мы рассматривали не прямые, а лучи, то проблема бы действительно возникла (попробуйте закрасить области на рис. 1.2). А предложенное «решение» с постепенной раскраской никак не использует то, что у нас именно прямые.

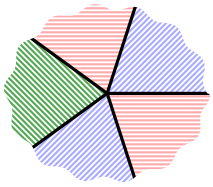


Рис. 1.2: Пять областей нельзя раскрасить в два цвета

Идею с постепенной раскраской можно довести до конца[1], но сейчас мы хотим показать рассуждение по индукции. Рассуждая по индукции, мы идём, так сказать, «от простого к сложному» и постепенно увеличиваем число прямых. Попробуем понять, что происходит при добавлении одной прямой. Пусть есть набор прямых и

[1]Скажем коротко, как это делается. Идя от соседа к соседу, мы меняем цвет. Проблема возникнет, если на одном пути будет чётное число границ, а на другом нечётное. Тогда при обходе туда-обратно мы вернёмся, пересекши в сумме нечётное число границ, а каждую прямую мы должны пересекать чётное число раз, переходя из одной полуплоскости в другую и возвращаясь обратно — противоречие.

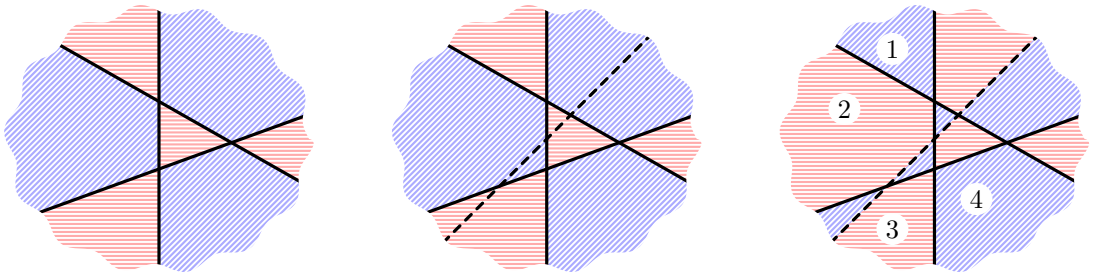


Рис. 1.3: Добавление прямой

области, на которые они разбили плоскость, уже покрашены как требуется (рис. 1.3, левый).

Проведем еще одну прямую (рис. 1.3, средний). Для новой конфигурации старая раскраска не годится, потому что по сторонам этой прямой области одного цвета. Как исправить положение? Если мы поменяем цвет у одного из получившихся кусков и сохраним у другого, то в этом месте проблема будет решена, но потом придётся перекрашивать и другие области, и где гарантия, что мы снова не придём к противоречию?

Будем действовать более глобально. Изменим цвет на противоположный у *всех* областей по одну сторону от новой прямой (рис. 1.3, правый, где цвета поменялись у областей сверху от новой прямой). Мы утверждаем, что теперь получится правильная раскраска (если, конечно, перед добавлением прямой раскраска была правильной). Почему?

Это надо действительно объяснить, иначе к нам можно будет предъявить те же претензии, которые мы раньше предъявляли к школьнику: сформулированное утверждение не доказано. Давайте попробуем. Требование задачи состоит в том, что любой участок границы теперь разделяет области разных цветов. Граница может быть либо на новой прямой, либо на старой. Случай первый: граница на новой прямой. В старой раскраске области с двух сторон были одного цвета, а теперь мы с одной стороны перекрасили, так что получились области разного цвета. Случай второй: граница на старой прямой. Тогда мы либо оставили цвета как было (если перекрашивали с другой стороны от новой прямой, области 3 и 4 на рис. 1.3), либо изменили оба цвета (области 1 и 2 на том же рисунке), но и тогда цвета останутся разными, только поменяются местами.

Что осталось сказать, чтобы закончить решение задачи? Для одной прямой утверждение задачи очевидно. Добавив вторую прямую, и перекрасив области с одной стороны от неё, мы получим решение для двух прямых. При этом не важно, как именно пройдёт вторая прямая (будет она параллельна первой или нет). Теперь можно добавить третью прямую (снова не важно, как именно она проходит) и перекрасить области с одной стороны, и так далее. В конце концов можно получить раскраску для любого числа прямых.

Программисты сказали бы, что мы описали алгоритм построения требуемой раскраски (рис. 1.4). Пусть нам дана произвольная конфигурация из n прямых. Пронумеруем эти прямые от 1 до n и будем добавлять их постепенно («в цикле»). Добавив очередную прямую, мы перекрашиваем все области с одной стороны от неё, восстанавливая правильность раскраски («инвариант цикла», как говорят программисты). Так делаем, пока все n прямых не будут добавлены.

$k = 1$
нарисовать одну прямую и закрасить две полуплоскости в разные цвета
// имеется правильная раскраска для k прямых
while $k \neq n$ {
    добавить $(k+1)$-ю прямую
    изменить цвета всех областей с одной её стороны
    $k = k + 1$
}

Рис. 1.4: Алгоритм построения раскраски для n прямых

Математики, конечно, будут несколько удивлены таким изложением (и вздрогнут, увидев «равенство» $k = k + 1$). Более привычное для них изложение выглядит так.

Докажем индукцией по n, что *требуемая раскраска существует для всех конфигураций из n прямых* — назовём это утверждение A_n, где n — параметр индукции.

Базис индукции. При $n = 1$ утверждение A_1 очевидно: прямая делит плоскость на две полуплоскости, которые можно покрасить в разные цвета (рис. 1.1, левый).

Шаг индукции. Пусть мы уже знаем, что A_n верно. Докажем, что верно A_{n+1}. Рассмотрим произвольную конфигурацию из $(n + 1)$ прямых. Надо доказать, что существует раскраска, удовлетворяющая условию задачи. Выберем какую-то одну прямую (можно взять любую) и временно её удалим. Получится конфигурация из n прямых. Согласно A_n, для неё есть раскраска, удовлетворяющая условию. Теперь вернём удалённую прямую на место и изменим раскраску в одной из полуплоскостей (с одной стороны от удалённой и возвращённой прямой). Получится требуемая раскраска для $(n + 1)$ прямых. (Тут надо повторить рассуждения про два типа границ и почему с обоими типами всё будет в порядке.)

С помощью шага индукции можно перейти от A_1 (базис индукции) к A_2, значит, A_2 тоже верно (для любых двух прямых есть раскраска). Теперь снова применим шаг индукции, но уже при $n = 2$: раз A_2 верно, то верно и A_3. И так далее: раз A_3 верно, то верно и A_4, затем A_5 и т.п.

1.2 Общая схема доказательств по индукции

Давайте повторим схему рассуждения из предыдущего раздела. Доказательства по индукции применяются, когда есть последовательность утверждений

$$A_1, A_2, A_3, \ldots, A_n, \ldots,$$

и мы хотим доказать, что все они верны. Принцип индукции говорит, что для этого достаточно сделать две вещи:

- **Базис индукции**: надо доказать, что A_1 (первое утверждение в цепочке) верно.
- **Шаг индукции**: надо доказать (для произвольного n), что A_{n+1} верно, предполагая известным, что A_n верно.

Шаг индукции ещё называют «индуктивным переходом», и даже понятно, почему — откуда и куда мы переходим: от A_n к A_{n+1}. Ещё говорят, что при этом переходе мы должны доказать «следование»: доказать, что из A_n следует A_{n+1}. Записывают это следование как «$A_n \Rightarrow A_{n+1}$», и шаг индукции состоит в доказательстве (при произвольном n) утверждения $A_n \Rightarrow A_{n+1}$.

Понятно ли, почему такая схема рассуждения законна? Смотрите: мы доказали A_1 (базис). Кроме того, мы доказали, что из A_1 следует A_2 (шаг индукции при $n = 1$). Значит, A_2 тоже верно. Теперь используем шаг индукции при $n = 2$: из A_2 следует A_3. Значит, и A_3 верно. И так далее — мы постепенно дойдём до любого A_n.

Тут люди с философским складом ума спросят: а почему, собственно, мы дойдём до любого натурального числа n? Вот мы прибавляем единицу и прибавляем — а вдруг до какого-то n так дойти нельзя в принципе? Или даже более конкретно: если $n = 10^{1000}$, то ясно, что на практике дойти до такого n нереально (время жизни Вселенной существенно меньше). И что? На этот вопрос трудно ответить убедительно, потому что за ним тянутся другие: а что такое вообще натуральное число? что такое число «семь», можно показать на пальцах, а для 10^{1000} никаких пальцев не хватит — и почему мы уверены, что такое число есть? И где, собственно говоря, оно есть? И какие способы рассуждений о натуральных числах допустимы? И почему мы уверены, что не получим какую-то ерунду? Такие вопросы изучаются в математической логике. Мы не будем даже пытаться пересказать ответы на них. Скажем лишь, что в математической логике принцип математической индукции — одна из аксиом натурального ряда (что бы это ни значило).

Вместо этого мы ещё раз продемонстрируем принцип математической индукции в действии и то, как принято записывать рассуждения с его использованием.

Теорема 1.1. *При любом $n \geqslant 1$ выполнено равенство:*

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}.$$

Доказательство. Обозначим это равенство через A_n и докажем его по индукции.

Базис индукции (A_1):

$$1 = \frac{1 \cdot (1+1)}{2},$$

очевиден.

Шаг индукции. Предположим, что A_n верно, то есть

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}.$$

Прибавим к обеим частям этого верного равенства число $(n+1)$, получим

$$1+2+3+\ldots+n+(n+1) = \frac{n(n+1)}{2}+(n+1) = \frac{n(n+1)+2(n+1)}{2} = \frac{(n+2)(n+1)}{2},$$

то есть A_{n+1} (надо только переставить сомножители). Шаг индукции и тем самым всё доказательство завершены. □

Часто, говоря о рассуждениях по индукции, приводят разные житейские примеры. Почему в автобус можно запихнуть любое количество людей? Потому что один человек туда войдёт, и каким бы полным автобус ни был, всегда можно потесниться и впихнуть ещё одного человека. В порядке занудства скажем, что здесь утверждение A_n состоит в том, что в автобус можно поместить n человек, базис индукции – про одного человека, а индуктивный переход соответствует запихиванию ещё одного. Рассуждение это приводит к неверному результату (тысячу человек в автобус явно не запихнуть), потому что сформулированный в качестве шага индукции принцип «всегда можно потесниться» неверен.

Другой житейский пример: если первым в очереди стоит студент, и за каждым студентом в очереди стоит студент, то все в очереди — студенты. Понятно, в чём состоит A_n, где здесь базис и каков индуктивный переход?

Может быть, вы в детстве выстраивали кости домино в цепочку — если первую кость толкнуть, то она упадёт на вторую, вторая — на третью, третья — на четвертую и они все рухнут. (Таких видео много в интернете.) Можно сказать, что примерно так же происходит и рассуждение по индукции: базис индукции — это когда мы толкаем первую кость, а шаг индукции — когда n-я кость, падая, сбивает $(n+1)$-ю.

Или представьте себе цепочку островов в море. Если (базис индукции) построить мост с материка на первый остров, а затем (шаг индукции) построить мосты с n-го острова на следующий, $(n+1)$-й, то на любой остров можно будет попасть. В этой схеме следование $A_n \Rightarrow A_{n+1}$ соответствует мосту: благодаря нему, если мы можем попасть на n-й остров, то можем попасть и на $(n+1)$-й.

1.3 Варианты рассуждений по индукции

1.3.1 С чего начинать?

В наших примерах мы начинали с единицы, но это совсем не обязательно. Пусть, скажем, мы хотим сравнить, какое число больше: 2^n или n^2. Посмотрим на несколько

первых значений:

n	0	1	2	3	4	5	6	7
n^2	0	1	4	9	16	25	36	49
2^n	1	2	4	8	16	32	64	128

Возникает впечатление, что при малых n бывает больше то одно, то другое, а потом 2^n уверенно обгоняет. То есть первые значения 2^n и n^2 подсказывают, что справедливо такое утверждение.

Теорема 1.2. *$2^n \geqslant n^2$ при $n \geqslant 4$.*

Как это доказать? Вдруг когда-нибудь потом n^2 начнёт быстро расти и наверстает упущенное, хотя это и кажется странным? Тут снова помогает рассуждение по индукции, только начинать надо с $n = 4$.

Доказательство. Рассмотрим утверждения

$$2^n \geqslant n^2 \qquad (A_n)$$

при $n = 4, 5, 6, 7, \ldots$. Рассуждая по индукции, докажем, что первое из них (то есть A_4) верно (базис индукции) и что из A_n следует A_{n+1} при $n \geqslant 4$ (шаг индукции).

Базис очевиден ($16 \geqslant 16$). Проведём шаг индукции. Для этого посмотрим, во сколько раз увеличиваются 2^n и n^2 при переходе от n к $n + 1$. Первое выражение увеличивается в два раза, а второе — в

$$\frac{(n+1)^2}{n^2} = (1 + \frac{1}{n})^2$$

раз. При $n \geqslant 4$ это выражение не больше $1{,}25^2 < 2$, поэтому правая часть увеличивается меньше, чем в два раза и не может превысить левую.

Более формально: пусть $2^n \geqslant n^2$, тогда

$$2^{n+1} = 2^n \cdot 2 \geqslant n^2 \cdot 2 \geqslant n^2 \cdot (1{,}25)^2 \geqslant n^2(1 + \frac{1}{n})^2 = \frac{n^2(n+1)^2}{n^2} = (n+1)^2.$$

Дальше рассуждение по индукции происходит как раньше: мы знаем, что A_4 верно и из A_4 следует A_5, поэтому и A_5 верно; поскольку из A_5 следует A_6, то и A_6 верно, и так далее. □

Другими словами, не обязательно начинать индукцию с единицы. Можно начинать и с любого большего числа, и с нуля (который часто тоже считают натуральным числом). В результате утверждение будет доказано для всех натуральных чисел, не меньших того, с которого мы начали.

Задача 1.2. Докажите, что при больших n (выберите границу сами) выполнено неравенство $2^n > n^3$. Тот же вопрос для $1{,}001^n > n^2$ и $n! > 100^n$.

1.3.2 Сведение к меньшим

Мы записывали шаг индукции как переход от A_n к A_{n+1}, но можно было бы записать его и как переход от A_{n-1} к A_n. Это ничего не меняет по существу, но формулы станут немного другими. Скажем, для суммирования мы должны были бы из

$$1+2+3+\ldots+(n-1)=\frac{(n-1)((n-1)+1)}{2}$$

получить

$$1+2+3+\ldots+n=\frac{n(n+1)}{2}.$$

Понятно, как это сделать, прибавив n?

В таких обозначениях легче объяснить, в чём польза от метода математической индукции. Нас просят доказать A_n для произвольного n. Индукция даёт нам некоторое послабление и разрешает принять на веру предыдущее утверждение A_{n-1} (если оно есть). Ясное дело, что от этого наша задача может стать легче. Опять же вернёмся к примеру с суммированием: найти сумму $1+2+\ldots+(n-1)+n$ становится легче, если нам разрешено использовать формулу для $1+2+\ldots+(n-1)$ (достаточно прибавить n).

Можно пойти ещё немного дальше и разрешить себе использовать не одно, а *все* предыдущие утверждения. Это изменит шаг индукции следующим образом.

- **Шаг индукции:** надо доказать (для произвольного n), что утверждение A_n верно, предполагая что все предыдущие утверждения $(A_1, A_2, \ldots, A_{n-1})$ верны.

Такой вариант принципа индукции не меняет сути дела: раньше мы доказывали, что A_n верно исходя из того, что верно A_{n-1}, а поскольку n было произвольным, то перед тем как доказать, что верно A_{n-1} мы доказали, что верно A_{n-2} и так вплоть до утверждения A_2, которое следует из утверждения A_1, истинность которого мы установили, доказав базис.

Более того, в такой формулировке можно даже не разделять базис индукции и шаг индукции: когда мы доказываем A_1, «принимая на веру все предыдущие», этих самых «всех предыдущих» нет, принимать на веру нечего и мы должны доказать A_1 «с чистого листа», что и даёт базис индукции.

Много лет назад, когда в ходу были трёх- и пятикопеечные монеты (последние называли «пятаки»), этот приём было легко проиллюстрировать такой задачей.

Задача 1.3. Докажите, что любое целое число копеек, начиная с восьми, можно заплатить такими монетами.

Понятно, как это доказать? Начнём пробовать: $8=3+5$, $9=3+3+3$, $10=5+5$, но что дальше? А дальше можно добавлять одну трёхкопеечную монету и из 8 получится 11, из 9 получится 12, из 10 получится 13, потом из 11 получится 14 и так далее.

Более формально это рассуждение можно пересказать так. Мы должны доказать утверждение A_n: *можно заплатить n копеек монетами в 3 и 5 копеек* при всех

$n \geqslant 8$. При этом, рассуждая по индукции, считаем известными все предыдущие утверждения A_m (при $8 \leqslant m < n$). Тут есть четыре случая:

- $n = 8$: платим $5 + 3$;
- $n = 9$: платим $3 + 3 + 3$;
- $n = 10$: платим $5 + 5$;
- $n \geqslant 11$: тогда $m = n - 3$ будет больше или равно 8 и меньше n; рассуждая по индукции, мы считаем, что A_m уже известно, то есть что m копеек уплатить можно, и осталось добавить одну трёхкопеечную монету, чтобы заплатить $m + 3 = n$.

Вот ещё один пример задачи, когда полезно использовать *все* предыдущие утверждения, а не только последнее.

Задача 1.4. Пусть выпуклый 1000-угольник разрезан на треугольники диагоналями, которые не пересекаются во внутренних точках (но могут иметь общие вершины). Докажите, что получилось 998 треугольников.

Будем доказывать это по индукции — точнее, конечно, не это, а общее утверждение о том, что если выпуклый n-угольник (при $n \geqslant 3$) разрезан на треугольники диагоналями, то этих треугольников $n-2$. Итак, докажем это для какого-то n, считая известным для всех меньших n. Посмотрим на какую-то диагональ, по которой разрезали. (Если диагоналей нет, то $n = 3$ и этот треугольник уже разрезан на $n - 2$ треугольников.) Представим себе, что по этой диагонали разрезали первой. Что тогда получится? С одной стороны будет какой-то k-угольник, а с другой стороны какой-то l-угольник. Оба числа k и l будут меньше n. При этом $k + l$ будет равно $n + 2$, поскольку, считая вершины сначала k-, а потом l-угольника, мы посчитаем все вершины многоугольника, причём концы диагонали посчитаем дважды. По предположению индукции при разрезании на треугольники получится $k - 2$ и $l - 2$ треугольников, всего $(k-2)+(l-2) = k+l-4 = (n+2)-4 = (n-2)$ треугольников, что и требовалось доказать.

1.3.3 Переформулировка: принцип наименьшего числа

То же самое решение задачи про монеты можно изложить, и не упоминая индукцию. Будем рассуждать, как говорят математики, «от противного»: предположим, что утверждение неверно, и покажем, что такого быть не может: что-то с чем-то не сойдётся («придём к противоречию»).

Пусть не при всех $n \geqslant 8$ сумму в n копеек можно заплатить трёх- и пятикопеечными монетами. *Возьмём наименьшую сумму, которую заплатить нельзя.* Пусть это будет какое-то n. Может ли n быть равно 8, 9 или 10? Нет, потому что мы знаем, как заплатить столько. Значит, $n \geqslant 11$ и $n - 3 \geqslant 8$. Поскольку n было наименьшим «плохим» числом (которое нельзя заплатить) среди чисел от 8, то $n - 3$ заплатить

можно. В чём противоречие? Понятно: $n-3$ заплатить можно, а n нельзя, хотя видно, что можно (надо добавить трёхкопеечную монету к $n-3$).

По существу это то же самое рассуждение, но только ссылку на принцип индукции мы заменили ссылкой на следующий принцип.

Принцип наименьшего числа. Если есть натуральные числа, обладающие каким-то свойством, то найдётся и наименьшее число с этим свойством.

В приведённом примере мы использовали свойство «неуплачиваемости». Этот принцип равносилен принципу математической индукции.[2]

1.4 Как не надо

В жизни всегда есть место ошибкам, и рассуждения по индукции — не исключение. Мы сейчас приведём несколько (намеренно) неверных рассуждений, надеясь, что Вы проявите бдительность и скажете: «какая глупость, здесь же [...]», указав на ошибку.

Пример 1.5. Докажем, что любое натуральное число n больше 100. В самом деле, принцип индукции позволяет это доказывать, считая известным это утверждение для всех меньших чисел. В частности, мы можем предполагать, что утверждение верно для $n-1$, то есть $n-1 > 100$. Тогда $n > 101$ и тем более $n > 100$, что и требовалось доказать.

Понятно, где тут обман? Вот чуть более сложный пример.

Пример 1.6. Докажем, что произведение любых $n \geqslant 0$ чисел равно нулю, используя индукцию по n. Базис индукции очевиден: при $n = 0$ сомножителей нет, так что перемножать нечего. Шаг индукции. Пусть утверждение верно для некоторого n, то есть произведение любых n чисел $a_1, a_2, \ldots, a_n$ равно нулю. Докажем то же утверждение для любых $n+1$ чисел $a_1, a_2, \ldots, a_{n+1}$. Рассуждая по индукции, мы считаем известным, что

$$a_1 \cdot a_2 \cdot \ldots \cdot a_n = 0.$$

Умножим это равенство на a_{n+1}, получится

$$a_1 \cdot a_2 \cdot \ldots \cdot a_n \cdot a_{n+1} = 0 \cdot a_{n+1} = 0,$$

что и требовалось.

Наверно, и здесь ошибка видна сразу, так что попробуем что-то похитрее.

[2] Эта формулировка о равносильности на самом деле требует уточнений. Что значит, что два принципа равносильны, если каждый из них очевиден? равносильны ли утверждения $2 \times 2 = 4$ и $3 \times 3 = 9$? Уточнение может быть сделано по-разному, один из вариантов мы обсудим в главе об упорядоченных множествах.

Пример 1.7. Докажем по индукции такое утверждение A_n: «в любом наборе из n натуральных чисел все числа равны». (Здесь $n = 1, 2, 3, \ldots$)

С базисом индукции всё в порядке: A_1 означает тривиальное утверждение «каждое число равно самому себе».

Докажем законность индуктивного перехода. Пусть A_n верно, докажем A_{n+1}. Рассмотрим набор из $(n+1)$ числа

$$(a_1,\ a_2,\ \ldots, a_n,\ a_{n+1}).$$

Применим утверждение A_n к наборам

$$(a_1,\ a_2,\ \ldots, a_n) \quad \text{и} \quad (a_2,\ \ldots, a_n,\ a_{n+1});$$

Каждый из этих наборов состоит из n чисел, поэтому к ним можно применить A_n (и ничего страшного, что его надо применить дважды: верное утверждение и несколько раз тоже верно). Таким образом, числа и в том, и в другом наборе равны, то есть

$$a_1 = a_2 = \cdots = a_n \quad \text{и}$$
$$a_2 = a_3 = \cdots = a_{n+1}.$$

Отсюда

$$a_1 = a_2 = \cdots = a_n = a_{n+1},$$

то есть утверждение A_{n+1} верно. Применяя принцип математической индукции, получаем, что A_n верно для всех n.

Если и это рассуждение вы легко разоблачили, вот наш последний пример. Коварство этого примера в том, что мы будем доказывать правильное утверждение, неправильным будет только доказательство.

Пример 1.8. Дадим более простое решение задачи 1.4: если разрезать n-угольник диагоналями[3] на треугольники, то получится $n-2$ треугольника.

Это утверждение имеет смысл начиная с $n = 3$, где оно очевидно (ничего разрезать не надо, и получается как раз $n - 2 = 1$ треугольник). Так что с базисом индукции всё в порядке.

Шаг индукции: предположим, что для n-угольника это уже известно, а теперь нам дан $(n+1)$-угольник. Так как $n \geqslant 3$, то $n + 1 \geqslant 4$, так что можно выбрать две вершины, идущие через одну (разделённые одной вершиной). Соединим их диагональю, отрежется треугольник и останется n-угольник (без одной отрезанной вершины). Мы уже знаем, что он разрезается на $n-2$ треугольника (предположение индукции), и ещё надо учесть первый отрезанный, всего будет $n-1$, то есть как раз $(n+1)-2$, то есть мы доказали требуемое для $(n+1)$-угольника.

[3] Точную формулировку см. на с. 19.

Ну что, понятно, в чём тут ошибка?

На всякий случай скажем, что было неправильно в приведённых рассуждениях.

В первом случае шаг индукции правильный, но с базисом проблема: первое же утверждение неверно.

Во втором случае мы начали со странного утверждения о произведении нуля сомножителей — которое якобы равно нулю, и переход к одному сомножителю был неправильным. (Если уж определять произведения нуля сомножителей, то его надо считать равным единице: тогда при умножении на ещё один сомножитель получится разумный результат.)

В третьем случае в цепочке индуктивных переходов есть разрыв: если мы хотим перейти от $n = 1$ к $n = 2$, то ничего не выйдет (у двух групп равных чисел не будет общего элемента).

Наконец, в четвёртом случае мы доказываем не то, что обещали — мы лишь показываем, что есть способ разрезать на $n - 2$ треугольника, а не что любое разрезание содержит столько треугольников. Для доказательства второго (более сильного) утверждения нужна дополнительная лемма: при любом разрезании на треугольники есть диагональ, соединяющая вершины через одну.[4]

1.5 Как догадаться, что доказывать?

Мы уже доказали формулу

$$1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2},$$

и это было совсем несложно. А если мы теперь хотим найти аналогичную формулу для суммы квадратов? Такая формула действительно есть.

Пример 1.9.

$$1^2 + 2^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6}.$$

Если кто-то сообщит нам эту формулу, мы без труда её докажем по индукции. Базис индукции ($n = 1$):

$$1^2 = \frac{1 \cdot 2 \cdot 3}{6}.$$

Шаг индукции удобно записать как переход от $n - 1$ к n. Для этого запишем нашу формулу для $n - 1$:

$$1^2 + 2^2 + \ldots + (n-1)^2 = \frac{(n-1)n(2n-1)}{6}.$$

[4]Это на самом деле верно, и это можно обосновать с использованием доказываемого утверждения: если n-угольник разрезан на $n-2$ треугольника, то сторон больше, чем треугольников, значит две из них принадлежат одному треугольнику, а это может быть лишь когда они соседние. Но это обоснование нам не подходит — нельзя ссылаться на то, что мы только ещё собираемся доказывать!

Прибавим к обеим частям предположения индукции по n^2:

$$1^2 + 2^2 + \ldots + (n-1)^2 + n^2 = \frac{(n-1)n(2n-1)}{6} + n^2 =$$
$$= \frac{(n-1)n(2n-1) + 6n^2}{6} = \frac{n((n-1)(2n-1) + 6n)}{6} = \frac{n(2n^2 - 3n + 1 + 6n)}{6} =$$
$$= \frac{n(2n^2 + 3n + 1)}{6} = \frac{n(n+1)(2n+1)}{6},$$

что и требовалось. Формула доказана.

Хотя к этому доказательству и не придерёшься — всё корректно и по правилам — но оно оставляет впечатление жульничества: как мы, собственно говоря, узнали, что надо доказывать именно это? Конечно, можно заявить, что это наше неотъемлемое право, что хотим, то и доказываем. Но всё-таки? На такие вопросы лекторы обычно отвечают «хороший вопрос» за неимением действительно убедительного ответа. В этом конкретном случае есть разные способы догадаться до такой формулы. Например, можно из некоторых общих соображений ожидать, что ответ задаётся многочленом третьей степени, и потом подобрать коэффициенты этого многочлена. Ну или просто пробовать разные формулы, в конце концов, при современных компьютерах перебор такого рода вполне реалистичен (хотя, конечно, формула для суммы квадратов была известна задолго до компьютеров). Но в целом, да, увы, проблема есть: многие доказательства по индукции производят именно такое впечатление — доказать сравнительно несложно, если знать, что доказывать, но непонятно, как до этого можно было догадаться. Тем более что авторы математических статей и даже учебников не всегда считают себя обязанными распространяться о том, как они додумались до своих результатов, ограничиваясь доказательством того, что они верны.

Мы приведём ещё несколько примеров, где основная трудность именно понять, что надо доказывать, а само доказательство простое.

Пример 1.10. Докажите, что

$$\frac{1}{1^2} + \frac{1}{2^2} + \ldots + \frac{1}{100^2} < 2.$$

Казалось бы, при чём тут индукция, если вообще нет никакого параметра? Но можно предположить (и это правильный ход), что на самом деле выбор числа 100 роли не играет, и надо доказывать неравенство

$$\frac{1}{1^2} + \frac{1}{2^2} + \ldots + \frac{1}{n^2} < 2, \qquad (A_n)$$

при любом n. В соответствии с принципом математической индукции, мы должны предположить, что A_n верно и доказать A_{n+1}, то есть доказать, что

$$\frac{1}{1^2} + \frac{1}{2^2} + \ldots + \frac{1}{n^2} + \frac{1}{(n+1)^2} < 2, \qquad (A_{n+1})$$

Но как? Мы предположили, что некая сумма меньше 2, и должны доказать, что после прибавления к ней ещё одного (положительного) слагаемого $1/(n+1)^2$ она останется меньше 2. Но с какой стати? Сумма ведь увеличится, почему бы ей не пересечь границу и стать теперь больше двух? Кажется, у нас с этим рассуждением по индукции проблемы...

Оказывается, что индукцию всё же применить можно с помощью такого трюка. Будем доказывать *более сильное утверждение*, а именно, неравенство

$$\frac{1}{1^2}+\frac{1}{2^2}+\ldots+\frac{1}{n^2}<2-\frac{1}{n}. \qquad (A'_n)$$

Рассуждая по индукции, предположим, что A'_n верно. Нам надо вывести отсюда, что верно и A'_{n+1}, то есть что

$$\frac{1}{1^2}+\frac{1}{2^2}+\ldots+\frac{1}{n^2}+\frac{1}{(n+1)^2}<2-\frac{1}{n+1}.$$

Что ж, прибавим к обеим частям A'_n величину $1/(n+1)^2$, получим, что

$$\frac{1}{1^2}+\frac{1}{2^2}+\ldots+\frac{1}{n^2}+\frac{1}{(n+1)^2}<2-\frac{1}{n}+\frac{1}{(n+1)^2}.$$

и это неравенство можно продолжить:

$$2-\frac{1}{n}+\frac{1}{(n+1)^2}<2-\frac{1}{n}+\frac{1}{n(n+1)}=2-\frac{n+1-1}{n(n+1)}=2-\frac{1}{n+1},$$

тем самым получается требуемое A'_{n+1}. Ловко? Ведь действительно возразить нечего — и доказали мы даже больше, чем надо (что не просто меньше 2, а даже ещё на $1/n$ меньше), и индукция теперь сработала...

В следующем примере усиление доказываемого утверждения, пожалуй, выглядит ещё более неожиданным.

Пример 1.11. На встрече компании из 15 человек некоторые из участников пожали друг другу руки. Могло ли получиться так, что каждый из 15 участников сделал три рукопожатия (пожал руки трём другим)? Докажите, что нет.

Тут уж совсем непонятно, как можно рассуждать по индукции: числовых параметров тут два (15 и 3), и кажется, что ни один из них не подходит, чтобы сделать его переменным и вести индукцию по нему. И действительно, мы будем действовать иным образом. Во-первых, мы забудем и про 15, и про 3, а будем доказывать общее утверждение: *число участников, сделавших нечётное число рукопожатий, чётно.* Это действительно более сильное утверждение — потому что если все 15 участников сделали по три рукопожатия, то оно нарушается.

Но остаётся вопрос — индукция будет по чему? по какому параметру? Вот по какому: будем представлять себе, что рукопожатия происходят не одновременно, а по очереди (в произвольном порядке) и будем доказывать, что после любого числа

рукопожатий наше утверждение о чётности выполнено. То есть параметр индукции — это число рукопожатий.

После такой подготовки само индуктивное рассуждение уже совсем простое. Базис индукции: не сделано ни одного рукопожатия, все участники сделали ноль (чётное число) рукопожатий, количество участников, сделавших нечётное число рукопожатий (ноль) чётно.

Шаг индукции. Пусть сделано ещё одно рукопожатие. Посмотрим на то, сколько перед этим рукопожатий было у его участников. Есть три возможности:

- К моменту рукопожатия оба участника сделали нечётное число рукопожатий. Тогда после него число сделанных каждым из двух участников рукопожатий окажется чётным, то есть количество «нечётных» участников уменьшится на 2 и останется чётным (раз было чётным по предположению индукции).
- К моменту рукопожатия оба участника сделали чётное число рукопожатий. Тогда после него число сделанных каждым из двух участников рукопожатий окажется нечётным, то есть количество «нечётных» участников увеличится на 2 и останется чётным (раз было чётным по предположению индукции).
- Один из двух участников к моменту рукопожатия был «чётным», а другой «нечётным». Тогда после увеличения на 1 они поменяются ролями: чётный станет нечётным и наоборот. Общее количество нечётных участников не изменится и останется чётным.

Задача решена — правда, всё просто (хотя и загадочно)?

1.6 Доказательства по индукции и без

Как говорил председатель акустической комиссии у Булгакова, «разоблачение совершенно необходимо. Без этого ваши блестящие номера оставят тягостное впечатление. Зрительская масса требует объяснения». И впрямь, можно ли как-то «разоблачить перед зрителями» эти наши фокусы?

Частично да — хотя в некоторых случаях это будет, пожалуй, уже другое доказательство, а не просто изложение того же самого на более понятном языке. Попробуем это сделать для некоторых из наших примеров.

Сумму $S = 1+2+3+\ldots+n$ можно вычислить так. Запишем её дважды, причём во второй раз в обратном порядке:

$$\begin{aligned} 2S = 1 \quad &+ \quad 2 \quad + \quad 3 \quad + \ldots + n = \\ &+ n + (n-1) + (n-2) + \ldots + 1. \end{aligned}$$

Теперь посчитаем её по столбикам, в каждом сумма $n+1$, а всего их n, так что $2S = n(n+1)$ и $S = n(n+1)/2$.

Про разрезание n-угольника на треугольники можно рассуждать так. Сумма углов в каждом треугольнике равна $180°$. Сложим все углы всех треугольников.

С одной стороны, получится 180°·(число треугольников). С другой стороны, если складывать углы, группируя не по треугольникам, а по вершинам, то получится сумма углов n-угольника, которая — как известно из школьной геометрии — равна $180°(n-2)$. Значит,[5] всего треугольников будет $n-2$.

Оценить сверху сумму обратных квадратов

$$\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{100^2}$$

можно так. Заметим, что при уменьшении знаменателя дроби сама дробь увеличивается, поэтому если мы заменим (начиная с $n=2$) дробь

$$\frac{1}{n^2}$$

на

$$\frac{1}{n(n-1)}=\frac{n-(n-1)}{n(n-1)}=\frac{1}{n-1}-\frac{1}{n},$$

то сумма может только возрасти и достаточно доказать, что новая сумма меньше 2. Но эта новая сумма «сворачивается» (по-английски есть даже выразительное слово telescoping sum — напоминающее о трубе телескопа из нескольких частей, вытягивающихся одна из другой), и получается

$$1+\left(\frac{1}{1}-\frac{1}{2}\right)+\left(\frac{1}{2}-\frac{1}{3}\right)+\ldots+\left(\frac{1}{98}-\frac{1}{99}\right)+\left(\frac{1}{99}-\frac{1}{100}\right)=1+1-\frac{1}{100}<2.$$

В этом доказательстве видно, как появляется усиление оценки на $1/n$ (в данном случае, $1/100$).

Теперь о рукопожатиях. Пусть-таки действительно каждый из 15 человек в компании сделал по 3 рукопожатия. Сколько всего будет рукопожатий? Очевидно, $15\cdot 3=45$? На самом деле нет: в каждом рукопожатии участвуют двое, и оно будет посчитано дважды, как рукопожатие одного и второго участника. Поэтому мы посчитали удвоенное число рукопожатий, а их было $15\cdot 3/2=45/2=22,5$. Получилась ерунда — нецелое число рукопожатий? А почему? потому что мы предположили, что каждый из 15 человек сделал по три рукопожатия — значит, так не бывает, что и требовалось доказать.

Понятно, как таким способом доказать и наше более сильное утверждение: что число «нечётных» участников чётно? (Если сумма целых чисел чётна, то количество нечётных слагаемых в ней чётно.)

[5] В этом рассуждении есть тонкое место. Одно из доказательств формулы для суммы углов n-угольника состоит в том, что многоугольник разрезают на треугольники и замечают, что их $n-2$. Если опираться на такое доказательство (что необязательно, есть и другие), получается порочный круг — вроде как мы используем то, что хотим доказать. На самом деле не совсем (вспомните ошибку в доказательстве по индукции): при вычислении суммы углов важно, что *существует* хотя бы одно разрезание n-угольника на $n-2$ треугольника, а это сразу ясно: можно провести $n-3$ диагонали из одной вершины.

Даже и самую первую задачу — о разрезании плоскости n прямыми и о раскраске частей в два цвета — можно объяснить, не говоря открыто об индукции. Например, можно сказать так. Каждая прямая на (координатной) плоскости задаётся уравнением $ax+by+c=0$, где a, b, c — какие-то числа, причём одно из чисел a и b должно быть ненулевым. Эта прямая делит плоскость на две полуплоскости, в одной из них $ax+by+c$ положительно, в другой — отрицательно.

Ну и что? А вот что. Пусть у нас есть много разных прямых

$$a_1x+b_1y+c_1=0, a_2x+b_2y+c_2=0, \ldots, a_nx+b_ny+c_n=0,$$

Объединение этих прямых задаётся уравнением

$$(a_1x+b_1y+c_1)(a_2x+b_2y+c_2)\cdot\ldots\cdot(a_nx+b_ny+c_n)=0.$$

Вне прямых левая часть не равна нулю, и в каждой из областей сохраняет знак (либо всюду положительна, либо всюду отрицательна).[6] Будем красить положительные области в один цвет, а отрицательные в другой.

Почему эта раскраска удовлетворяет условию? Если перейти с одной стороны границы на другую, то мы пересекаем граничную прямую, и соответствующий сомножитель в произведении меняет знак, а остальные его сохраняют — значит, всё произведение меняет знак. Поэтому цвета областей по разные стороны от общего участка границы разные.

В завершение раздела скажем, что наше разделение доказательств на «по индукции» и «без индукции» глубокого смысла не имеет, это скорее вопрос изложения. Специалисты по математической логике и компьютерным системам проверки доказательств хорошо знают, что безобидные слова «и так далее», «складывая все эти неравенства» и им подобные на самом деле скрывают ссылку на принцип математической индукции, и если действительно честно интересоваться, что можно сформулировать и доказать без аксиомы индукции, то это нужно делать гораздо более аккуратно (и окажется, что почти ничего).

Однако неглубокий смысл в таком различении становится понятным, как только вы попробуете решать задачи, а не только читать их решения. Заранее трудно сказать, какое рассуждение проще придумать — использующее индукцию явно или опирающееся на другие простые соображения.

Вот ещё пример доказательства, в котором мы ничего не говорим про индукцию явно.

Пример 1.12. Рассмотрим *последовательность Фибоначчи*

$$1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots,$$

в которой первые два числа равны единице, а каждое следующее равно сумме двух предыдущих. Мы хотим доказать, что *в последовательности Фибоначчи нет двух подряд идущих чётных чисел.*

[6] Тут тоже есть проблема с обоснованием, по-хорошему надо бы сослаться на непрерывность. Впрочем, если честно, то мы ведь не определяли, что такое «области, на которые прямые делят плоскость», так что тут одной такой ссылкой не обойтись.

Почему? Закон образования этой последовательности можно переформулировать так: если a, b, c — подряд идущие члены, то $a = c - b$. Значит, если b и c два подряд идущих чётных члена, то и a тоже чётное (разность двух чётных чисел чётна), так что пара чётных соседей встречается и раньше (a и b). А тогда и предыдущее число чётно, и так далее до начала последовательности (а первый член нечётный — противоречие).

Как это сказать более формально? Можно сослаться на принцип наименьшего числа, и сказать, что из всех пар соседних чётных членов мы берём первую, а потом замечаем, что она не первая. А можно использовать принцип индукции — понятно, как сформулировать утверждение A_n?

Один из вариантов такой: *из двух последовательных членов F_{n-1} и F_n последовательности Фибоначчи хотя бы один нечётный.* Докажем, что из A_n следует A_{n+1}, то есть что из чисел F_n и F_{n+1} хотя бы одно нечётное. Два варианта: если F_n нечётно, то доказывать нечего, а если F_n чётно, то из предположения индукции следует, что F_{n-1} нечётно. Тогда $F_{n+1} = F_{n-1} + F_n$ есть сумма нечётного и чётного числа, то есть нечётное число.

Понятно, почему все три рассуждения — это разные варианты по существу одного и того же доказательства?

1.7 Индукция и рекурсия

Приём сведения к меньшим значениям параметра знаком и математикам, и программистам. Математики называют его индукцией, а программисты — рекурсией. Мы сравним их подходы на примере старинной головоломки о «ханойских башнях».

Есть три штырька, на которые можно надевать диски. Есть n дисков разного размера. Сначала все диски на одном штырьке от большего в меньшему (рис. 1.5.)

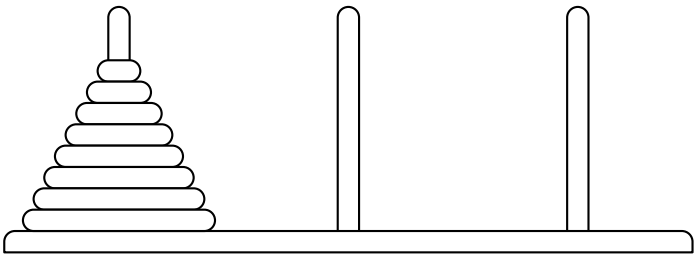


Рис. 1.5: Начальное положение

Задание состоит в том, чтобы *переместить эту пирамидку на другой штырёк, соблюдая правила игры.* Правила эти такие (рис. 1.6). За один ход разрешается переносить верхний в стопке диск на любой другой штырек, но нельзя класть больший диск на меньший.

Почему поставленная задача разрешима при любом n? Математик будет доказывать это индукцией по n. Если $n = 1$, то есть диск всего один, никаких препятствий

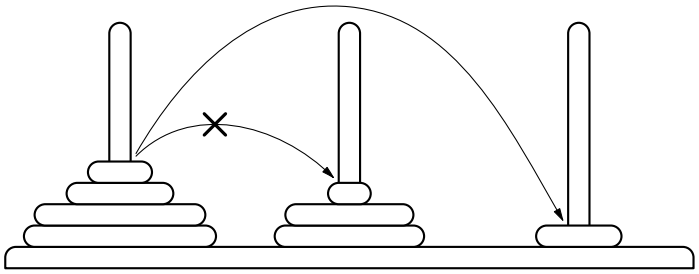


Рис. 1.6: Верхний диск с левого штырька можно переложить на правый, но не на средний, потому что там он ляжет на меньший диск.

нет. Задача решается за один ход.

(Прежде чем читать рассуждение дальше, попробуйте решить задачу для случая двух и трёх дисков — для проверки скажем, что понадобится минимум 3 и 7 ходов соответственно.)

Продолжаем рассуждение: шаг индукции. Пусть известно, что для n дисков задача разрешима. Надо перенести $n+1$. Это можно сделать в три стадии (рис. 1.7).

(а) Перенесём верхние n дисков с первого штырька на второй. Предположение индукции говорит, что это возможно — правда, в отсутствие самого большого диска. Но он тихо и спокойно лежит себе на первом штырьке и ничему не мешает (на него можно класть что угодно).

(б) Перенесём самый большой диск с первого штырька на третий.

(в) Перенесём n дисков со второго штырька на третий. На третьем штырьке уже лежит самый большой диск, он ничему не мешает, а после завершения этой стадии все диски будут на третьем штырьке в нужном порядке.

Программист — не математик, и его скорее интересует не принципиальная разрешимость задачи, а алгоритм её решения. Он скажет: давайте напишем рекурсивную процедуру переноса n дисков со штырька с номером i на штырёк с номером j (считаем, что штырьки пронумерованы от 1 до 3).

```
MOVE (i, j, n) : // перенести n ⩾ 1 верхних дисков с i на j,
                 // предполагая, что остальные диски больше
    if n = 1:
        перенести диск с i на j
    else
        k = 6 − i − j // хак: k — третий штырёк, ибо 1 + 2 + 3 = 6
        MOVE (i, k, n − 1)
        перенести диск с i на j
        MOVE (k, j, n − 1)
```

Три стандартных вопроса, которые должны возникать по поводу любого алгоритма:

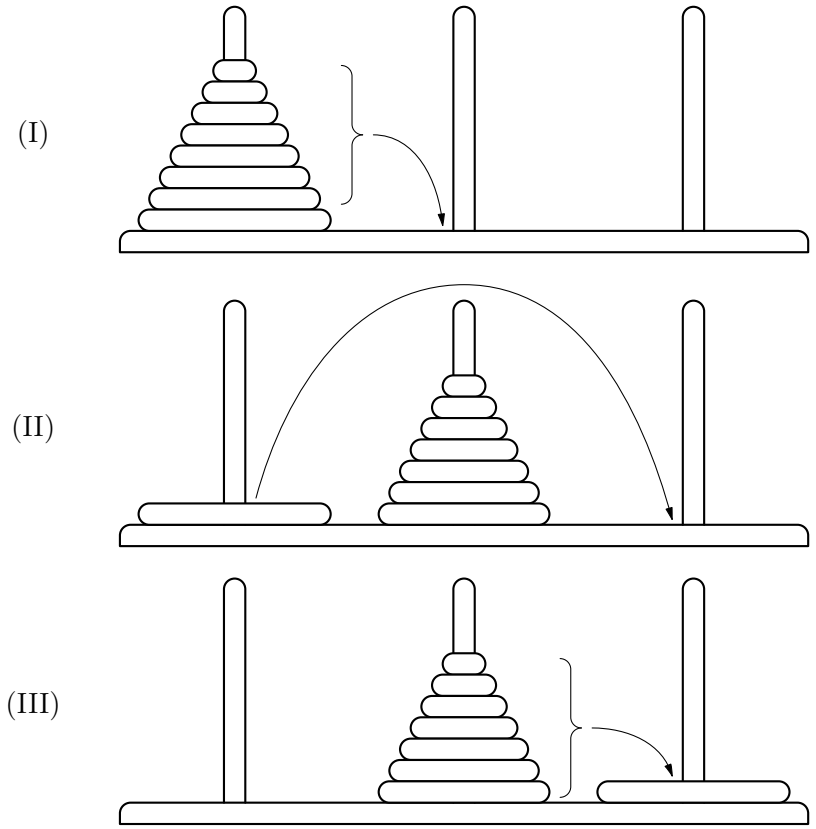


Рис. 1.7: Шаг индукции для ханойских башен.

- почему он правильный?
- сколько шагов он потребует?
- а нет ли лучшего алгоритма (с меньшим числом шагов)?

Математик и программист, наверно, легко согласятся, что алгоритм правильный (единственная тонкость, про которую важно не забыть — объяснить, почему лежащие на штырьках больши́е диски не мешают).

Число шагов они тоже совместными усилиями легко вычислят: если считать только переносы дисков и обозначить через $T(n)$ число переносов дисков для задачи с n дисками, то из текста процедуры видно, что

- $T(1) = 1$;
- $T(n) = 2T(n-1) + 1$ при $n > 1$.

Отсюда $T(2) = 3, T(3) = 7, T(4) = 15, T(5) = 31$ и так далее. Программист (который, разумеется, наизусть помнит степени двойки) сразу же предположит, что

$$T(n) = 2^n - 1,$$

а математик обрадуется и сразу скажет, что это легко доказать по индукции: если $T(n) = 2^n - 1$, то $T(n+1) = 2T(n)+1 = 2\cdot(2^n-1)+1 = 2^{n+1}-1$. Довольные, они разойдутся по домам, и по пути домой математик сообразит, как доказать, что меньшим числом переносов не обойтись. Это тоже делается по индукции: пусть $T'(n)$ — число переносов при самом экономном решении задачи для n дисков. При решении задачи для $(n+1)$ дисков нам не избежать переноса самого большого диска. (Может быть, его будут двигать несколько раз, но рассмотрим какой-то из них, скажем, первый.) В момент этого переноса два штырька (откуда и куда переносят наибольший диск) должны быть свободны. Значит, все диски (и в единственно возможном порядке — пирамидка) должны быть на третьем диске. То есть до выбранного момента (перенос самого большого диска) и после мы волей-неволей будем решать задачу переноса n дисков. Отсюда $T'(n+1) \geqslant 2T'(n)+1$, откуда индукцией получаем $T'(n) \geqslant 2^n - 1$.

Для порядка скажем ещё (следуя французской википедии), что эту задачу придумал французский математик XIX века Эдуард Люка, равно как и название «ханойские башни» и байку о том, что буддийские монахи перекладывают пирамидку из 64 дисков в ожидании конца света (который настанет, как мы теперь знаем, через $2^{64} - 1$ шагов — не так и долго при современных мощностях процессоров).

1.8 Доказательства неравенств по индукции

До сих пор мы приводили разные примеры рассуждений по индукции не потому, что именно эти результаты зачем-то важны, а потому, что они показывали, как индукция работает. Теперь мы потренировались и можем перейти к более важным результатам. В этом разделе мы используем индукцию для доказательства двух неравенств.

1.8.1 Неравенство Бернулли

Первое из них совсем простое и называется «неравенство Бернулли» в честь математика XVII века Якоба Бернулли (хотя, вероятно, и до него это мало бы кого удивило). Оно говорит, что если каждый год деньги в банке растут на 1%, то за сто лет они более чем удвоятся. Более чем — потому, что платят 1% не от исходной суммы, а от текущей, которая больше (это иногда называют «сложными процентами»). Велика ли разница? Можно подсчитать:

$$(1 + 0{,}01)^{100} \approx 2{,}7048\ldots > 2$$

В курсе математического анализа объяснятся, что так получаются приближения к основанию натуральных логарифмов e, но нас сейчас интересуют более простые вещи.

Теорема 1.3 (неравенство Бернулли)**.**

$$(1+h)^n \geqslant 1+nh$$

при любом действительном $h \geqslant -1$ *и любом натуральном* n*.*

При $h \geqslant 0$ доказательство по существу уже дано: при вычислении очередной степени мы умножаем число (уже большее 1) на $(1+h)$ и потому увеличиваем его по крайней мере на h. При $h < 0$ это рассуждение применять нельзя, потому что $(1+h)^n$ меньше 1. Но неравенство тоже легко доказать наглядно. Изменим знак у h и будем доказывать, что

$$(1-h)^n \geqslant 1-nh$$

при $0 < h < 1$ и любом натуральном n. Что это значит в банковских терминах? что если в год берут, скажем, 1% процент за хранение (отрицательные проценты реально бывают, кстати), то за десять лет возьмут не больше 10% исходной суммы. И понятно, почему: в следующие годы возьмут процент с текущего капитала, и это будет меньше, чем от начального.

Интересно, что при доказательстве по индукции различать эти два случая нет необходимости.

Базис индукции. При $n = 0$ неравенство Бернулли очевидно: $1 \geqslant 1$.

Шаг индукции. Пусть $(1+h)^n \geqslant 1+nh$ при некотором натуральном n (индуктивное предположение). Поскольку $h \geqslant -1$, то $1+h \geqslant 0$ и умножение на $1+h$ обеих частей индуктивного предположения сохраняет неравенство. Получается

$$(1+h)^{n+1} = (1+h)^n(1+h) \geqslant (1+nh)(1+h) = 1+(n+1)h+nh^2 \geqslant 1+(n+1)h$$

(в последнем переходе мы использовали, что квадрат всегда неотрицателен, независимо от знака h).

Задача 1.13. Докажите, что $0.99^{100} \leqslant 1/2$.

1.8.2 Среднее арифметическое и геометрическое

Среднее арифметическое n чисел $a_1, \ldots, a_n$ определяется как

$$\frac{a_1 + \ldots + a_n}{n}.$$

Слово «среднее» тут уместно в том смысле, что среднее арифметическое находится между наименьшим и наибольшим из чисел (понятно, почему?). Но, конечно, оно не будет «средним» в обычном смысле: из пяти чисел $1, 2, 3, 4, 10$ среднее арифметическое равно 4, а не 3. Статистики скажут, что среднее равно 4, а *медиана* равна 3 — наверно, вы слышали сетования, что из-за сильного неравенства медианный доход сильно меньше среднего.

Так или иначе, нас сейчас интересует сравнение среднего арифметического со *средним геометрическим*. Это второе определяется для положительных чисел как

$$\sqrt[n]{a_1 a_2 \cdot \ldots a_n}$$

Теорема 1.4 (неравенство о среднем арифметическом и геометрическом)**.** *Среднее геометрическое положительных чисел не больше их среднего арифметического.*

Скажем, среднее геометрическое чисел 1 и 9 равно 3 и меньше их среднего арифметического (которое равно 5).

Для двух чисел получается неравенство

$$\sqrt{ab} \leqslant \frac{a+b}{2}.$$

Если умножить его на 2 и возвести в квадрат, то получится равносильное неравенство

$$4ab \leqslant (a+b)^2,$$

или (перенося всё в правую часть)

$$a^2 - 2ab + b^2 \geqslant 0.$$

В левой части легко узнать квадрат разности, так что для двух чисел неравенство доказано (и заодно видно, что оно обращается в равенство лишь при $a = b$ — понятно, почему?).

Но как доказать его для большего количества чисел? Есть много разных способов. Например, любители математического анализа заметили бы, что среднее геометрическое получается, если перейти к логарифмам, взять среднее арифметическое, а потом вернуться обратно. Свойство функции «логарифм». называемое выпуклостью вверх и проверяемое вычислением знака второй производной, говорит, что после этого перехода туда-обратно получится меньше, чем просто среднее арифметическое.

Но мы обойдёмся без таких экскурсов, рассуждая по индукции. Для начала сформулируем такое следствие неравенства:

> *Если произведение n положительных чисел равно 1, то их сумма не меньше n.*

Это частный случай неравенства, когда среднее геометрическое равно единице (понятно, почему?).

Теперь заметим, что достаточно доказать этот частный случай, от него можно перейти к общему. В самом деле, если умножить все n чисел на какую-то константу $c > 0$, то и среднее арифметическое, и среднее геометрическое умножатся на c, поэтому неравенство между ними останется верным, если оно было верным, и останется неверным, если было неверным. Поэтому, если мы хотим доказать неравенство о средних для произвольных чисел, можно поделить все их на такую константу, чтобы среднее геометрическое стало равным 1 (то есть на это самое среднее геометрическое), потом воспользоваться следствием и потом вернуться обратно.

Итак, осталось доказать следствие. Мы сделаем это по индукции.

Базис индукции: при $n = 1$ есть единственное число, равное 1, и сумма с единственным таким слагаемым не меньше 1. (Тут всё чисто, но если у вас есть сомнение, не будет ли какой-то беды от рассмотрения суммы с единственным слагаемым,

заметьте, что случай $n = 2$ мы уже тоже разобрали и в крайней случае можно сослаться на него.)

Шаг индукции. Пусть для n чисел это уже известно, и мы рассматриваем $n+1$ чисел, для которых

$$a_1 a_2 \cdot a_n \cdot a_{n+1} = 1.$$

Надо показать, что $a_1 + a_2 + \ldots + a_n + a_{n+1} \geqslant n+1$. Как воспользоваться предположением индукции? Откуда взять n чисел, произведение которых равно 1? Ничего лучшего, чем соединить два числа в одно, тут в голову не приходит, так что попробуем так: пусть $a = a_1 a_2$, тогда

$$a \cdot a_3 \cdot \ldots a_n a_{n+1} = 1,$$

и потому

$$a + a_3 + \ldots + a_n + a_{n+1} \geqslant n,$$

другими словами,

$$a_1 a_2 + a_3 + \ldots + a_n + a_{n+1} \geqslant n.$$

Чего мы добились таким способом? В левой части у нас $a_1 a_2$, а нужно $a_1 + a_2$, а в правой части n, а нужно $n+1$. Чтобы перейти от того. что есть, к тому, что нужно, нам хорошо было бы знать. что

$$a_1 + a_2 \geqslant a_1 a_2 + 1, \qquad (*)$$

но надежд на это мало: числа a_1 и a_2 могут быть произвольными, и видно, скажем, что при $a_1 = a_2 = 2$ левая часть 4, а правая 5, и неравенство $(*)$ не выполняется.

Можно даже понять, когда неравенство $(*)$ выполняется и когда нет. Перенесём всё в правую часть и перепишем его в равносильном виде

$$a_1 a_2 + 1 - a_1 - a_2 \leqslant 0.$$

Левую часть теперь можно разложить на множители:

$$(a_1 - 1)(a_2 - 1) \leqslant 0,$$

и видно, что $(*)$ выполняется, когда одно из чисел a_1 меньше единицы, а второе больше (и равенство тоже допустимо). Но этого нам никто не гарантирует, что же делать?

В этом месте пора кричать «Эврика!», как Архимед в ванне. Да, нам никто не гарантирует, что $a_1 \leqslant 1$ и $a_2 \geqslant 1$ (или наоборот). Но ведь числа $a_1, a_2, \ldots, a_n, a_{n+1}$ все равноправны, мы можем их переставлять в любом порядке, от этого ни сумма, ни произведение не изменятся. Раз их произведение равно единице, то среди них обязательно должно быть число, не большее единицы (если они все были бы больше, то и произведение было бы больше). Произведение остальных чисел не меньше единицы, значит. среди них есть число, не меньшее единицы. Так и возьмём их в качестве a_1 и a_2, и наше рассуждение пройдёт!

Другими словами, нужно рассуждать так: заметим, что в произведении $n+1$ чисел можно взять сомножитель, не больший 1, а из оставшихся выбрать сомножитель, не меньший 1, и *именно эти два сомножителя* объединить, применив предположение индукции к их произведению и к оставшимся $(n-1)$ числам. При таком подходе рассуждение благополучно завершается, и наше следствие, а с ним и неравенство о среднем арифметическом и геометрическом, доказаны.

1.9 Пример из алгебры: системы однородных уравнений

Есть такая народная мудрость: если в задаче n неизвестных, то чтобы все их определить, нужно составить n уравнений, меньшего числа не хватит, останется неоднозначность. Иногда ещё говорят, что есть n «степеней свободы», каждое уравнение отбирает одну из них, и если уравнений меньше n, то какая-то свобода останется.

Как всегда, при буквальном понимании это неверно: скажем, уравнение $x^2 + y^2 = 0$ определяет сразу две переменные: $x = 0$ и $y = 0$ (иначе сумма квадратов положительна). И таких примеров много, скажем, уравнение $x^2-2xy+2y^2-2y+1 = 0$ тоже однозначно определяет обе переменные (и люди, недавно готовившиеся к «вступительным экзаменам», это сразу же увидят). Но принцип этот тем не менее имеет смысл и в каких-то ситуациях работает, и в этом разделе мы рассмотрим самую простую такую ситуацию.

Теорема 1.5. *Система линейных однородных уравнений, в которой уравнений меньше, чем неизвестных, имеет ненулевое решение.*

Надо только объяснить употребляемые термины. *Линейное уравнение* — это уравнение вида

$$a_1x_1 + \ldots + a_nx_n = b.$$

Здесь $x_1, \ldots, x_n$ — *переменные* (или, как ещё говорят, *неизвестные*), а $a_1, \ldots, a_n$ — числа, называемые *коэффициентами*; число b называется *свободным членом*. Если свободный член равен нулю, то уравнение называется *однородным*. Набор значений переменных называется *решением* уравнения, если при этих значениях уравнение обращается в равенство. Если есть несколько уравнений, или, как говорят, *система уравнений*, то её решением называют набор значений, при которых все уравнения обращаются в равенства. Например, набор из нулей является решением любого однородного уравнения (и любой системы однородных уравнений). Теорема говорит о ситуации, когда есть и другие решения (ненулевые — не все переменные равны нулю, хотя некоторые переменные и могут).

В формулировке теоремы есть одна тонкость: как мы считаем, в уравнении $0x_1 + 7x_2 = 0$ сколько переменных — одна или две? Мы будем считать, что две, пусть одна из них входит с нулевым коэффициентом.

Если не бояться формул, то теорему можно сформулировать так: если $m < n$, то для любых чисел $a_{11}, a_{12} \ldots, a_{1n}, a_{21}, \ldots, a_{2n}, \ldots, a_{m1}, \ldots, a_{mn}$ найдутся числа

$x_1, \ldots, x_n$, не все одновременно равные нулю, для которых

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = 0, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = 0, \\ \qquad\qquad\qquad\qquad\qquad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = 0. \end{cases}$$

Полезно ещё уточнить, что m и n тут целые положительные числа (тогда не надо уточнять, как относится к системе из нуля уравнений или с нулём неизвестных, хотя это и можно было бы сделать), а коэффициенты и неизвестные — действительные числа. (Можно было бы взять и рациональные, или комплексные, или вообще из любого поля, если знать, что это такое, но рассуждение от этого тоже не изменится, так что будем считать числа действительными.)

Докажем эту теорему индукцией по числу уравнений. План действий такой. Возьмём какую-то переменную. Если она вообще не входит ни в одно уравнение (коэффициенты нулевые), то доказывать нечего, можно взять ненулевое значение для этой переменной и нулевые значения для остальных. Если она входит в какое-то уравнение, то её можно выразить из этого уравнения и подставить это выражение в остальные уравнения, при этом число переменных и уравнений уменьшится на 1 и можно воспользоваться предположением индукции.

Теперь подробно. Базис индукции: одно уравнение с $n > 1$ переменными. Тут есть два варианта.

(а) Все коэффициенты уравнения нулевые:

$$0 \cdot x_1 + 0 \cdot x_2 + \cdots + 0 \cdot x_n = 0.$$

Любой набор значений переменных является решением такого уравнения. Утверждение теоремы в этом случае выполняется.

(б) В уравнении

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = 0.$$

есть хотя бы один ненулевой коэффициент. Без ограничения общности можно считать, что он первый, то есть что $a_1 \neq 0$ (перенумеруем переменные).

Выберем какие-нибудь ненулевые значения для переменных x_2, …, x_n, скажем,

$$x_2 = \ldots = x_n = 1.$$

Значение x_1 выразим из уравнения:

$$x_1 = \frac{1}{a_1}\left(-a_2 - a_3 - \ldots - a_n\right).$$

Получили ненулевой набор значений переменных, на котором уравнение обращается в истинное равенство:

$$a_1 \cdot \frac{1}{a_1}\left(-a_2 - a_3 - \cdots - a_n\right) + a_2 \cdot 1 + \ldots + a_n \cdot 1 = \\ = -a_2 - a_3 - \ldots - a_n + a_2 + \ldots + a_n = 0.$$

Поэтому утверждение теоремы в этом случае выполняется. (Понятно, где использовано условие на число переменных? Что нарушится, если переменная только одна?)

Шаг индукции. Мы проводим этот шаг по числу уравнений, поэтому надо предположить, что утверждение выполняется для систем из m уравнений с $n > m$ переменными, и доказать, что оно выполняется и для систем из $(m+1)$ уравнений с $n > m+1$ переменными.

Пусть дана такая система с $(m+1)$ уравнениями. Возьмём первое уравнение

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = 0.$$

Тут тоже есть два случая.

(а) Если все его коэффициенты нулевые, то любой набор значений переменных обращает это уравнение в равенство. Поэтому можно взять ненулевое решение системы из оставшихся m уравнений с $n > m+1 > m$ переменными, которое существует в силу индуктивного предположения.

(б) Не все коэффициенты первого уравнения нулевые. Без ограничения общности можно считать, что $a_{11} \neq 0$ (перенумеруем переменные). Тогда первое уравнение равносильно уравнению

$$x_1 = -\frac{1}{a_{11}}\left(a_{12}x_2 + a_{13}x_2 + \cdots + a_{1n}x_n\right). \qquad (*)$$

Из этого уравнения значение переменной x_1 однозначно выражается через значения остальных переменных. Подставляя это выражение в уравнение с номером $i > 1$, получаем

$$\begin{aligned} a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = \\ = -\frac{a_{i1}}{a_{11}}\left(a_{12}x_2 + a_{13}x_2 + \cdots + a_{1n}x_n\right) + a_{i2}x_2 + \cdots + a_{in}x_n = \\ = \left(a_{i2} - \frac{a_{i1}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{in} - \frac{a_{i1}}{a_{11}}a_{1n}\right)x_n = a'_{i2}x_2 + \cdots + a'_{in}x_n = 0. \end{aligned}$$

(Здесь через a'_{ij} обозначены выражения в соответствующих скобках.) При условии первого уравнения эти преобразования сохраняют равносильность, поэтому исходная система равносильна уравнению $(*)$, к которому добавлена система уравнений с m уравнениями и $(n-1)$ переменными

$$\begin{cases} a'_{22}x_2 + \cdots + a'_{2n}x_n = 0, \\ a'_{32}x_2 + \cdots + a'_{3n}x_n = 0, \\ \qquad\qquad \cdots \\ a'_{(m+1)2}x_2 + \cdots + a'_{(m+1)n}x_n = 0. \end{cases} \qquad (**)$$

Пока что мы не изменили число переменных и уравнений, но разделили систему на части: есть одно уравнение, выражающее x_1 через остальные переменные, и система $(**)$ из m уравнений с $(n-1)$ переменных. У нас было $n > m+1$, поэтому $n-1 > m$, и

к системе (∗∗) можно применить предположение индукции есть ненулевое решение. У неё есть ненулевое решение (набор значений переменных $x_2, \ldots, x_n$, причём не все эти значения равны нулю). Добавим к нему значение переменной x_1 согласно (∗). Получим ненулевое решение исходной системы.

Задача 1.14. Пусть в матрице (прямоугольной таблице) из целых чисел больше столбцов, чем строк. Тогда можно вычеркнуть некоторые (но не все) столбцы таким образом, чтобы в оставшейся матрице сумма чисел в любой строке была чётной.

Указание. Эта задача является вариантом основного результата раздела для случая поля из двух элементов, и можно рассуждать аналогично. Можно рассуждать и иначе, применив принцип Дирихле ко всем суммам множеств столбцов.

1.10 Коды Грея

Есть такая игра, когда в столбик пишутся слова, и в каждом следущем можно изменить одну букву:

НОРА
КОРА
КОРТ
КАРТ
...

Коды Грея — это то же самое, только мы

- пишем не русские слова, а любые комбинации букв;
- берём не обязательно русские буквы, а любой набор символов (*алфавит*);
- (главное) требуем, чтобы любая комбинация данной длины встречалась по разу и чтобы круг замкнулся, то есть от последнего слова можно было перейти к первому, тоже изменив одну букву.

Пример 1.15 (Двоичные слова)**.** Например, можно рассматривать *двоичные слова*, то есть слова в алфавите $\{0, 1\}$. Если брать слова длины два, то их четыре:

$$00,\ 01,\ 10,\ 11.$$

Чтобы выполнить требования, достаточно их расположить в таком порядке:

00
01
11
10

Заметьте, что от каждого слова к следующему мы переходим, меняя только одну букву, и от последнего слова так можно перейти к первому.

Геометрически можно представлять себе эти четыре двоичных слова как четыре вершины квадрата, заданные коодинатами — и ясно, что код Грея соответствует обходу всех вершин по одному разу с возвращением в начало.

Аналогичным образом можно расположить восемь двоичных слов длины 3, для этого их удобно представлять себе в виде вершин куба (рис. 1.8). Можно сделать

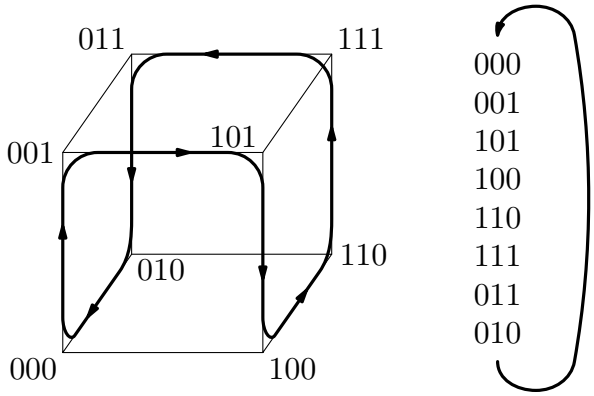


Рис. 1.8: Код Грея для двоичных слов длины 3

это и для чётырёхбуквенных двоичных слов, если уметь рисовать четырёхмерный куб.

Оказывается, что такое возможно для слов произвольной длины n — и не обязательно двоичных, но мы рассмотрим только двоичный случай, поскольку доказательство для него немного проще.

Теорема 1.6. *Можно написать все двоичные слова длины n в таком порядке, чтобы любые два соседних слова, а также первое и последнее слово, различались только в одной позиции.*

Доказательство проведём индукцией по n. Мы уже видели, как это сделать для $n = 2$ и $n = 3$, а для $n = 1$ это очевидно (любой из порядков $0, 1$ и $1, 0$ годится). Поэтому надо провести лишь шаг индукции.

Пусть слова длины n (их ровно 2^n, так как каждый новый бит увеличивает количество слов вдвое) уже расположены в нужном порядке:

$$x_1,\ x_2,\ \ldots, x_N$$

где $N = 2^n$. «Нужный порядок» тут означает, что x_i и x_{i+1} (а также x_1 и x_N) отличаются только в одной позиции. Как из этого списка получить список всех слов длины $n+1$? Можно к каждому слову приписать сначала нуль, а потом единицу (два варианта для последнего бита комбинируется с N вариантами для предыдущих). Получится

$$x_10,\ x_11,\ x_20,\ x_21,\ x_30,\ x_31, \ldots,\ x_N0,\ x_N1.$$

Но этот порядок нам не годится: хотя на первом шаге всё хорошо, x_10 и x_11 отличаются ровно в одной позиции (последней), на втором шаге x_11 и x_20 отличаются в двух позициях: в последней и там, где отличались x_1 от x_2. (Аналогичная проблема будет и при переходе от x_N1 к x_10.)

И что же делать? Возможно, вы уже догадались: надо каждый второй раз добавлять сначала 1, а потом 0, то есть использовать порядок

$$x_10,\ x_11,\ x_21,\ x_20,\ x_30,\ x_31,\ x_41,\ x_40,\ldots,\ x_N1,\ x_N0.$$

В последней паре сначала будет 1, а потом 0, поскольку $N = 2^n$ чётно, так что всё корректно замыкается по циклу.

Можно действовать и иначе: сначала ко всем добавить нуль, а потом единицу, идя в обратном порядке:

$$x_10,\ x_20,\ x_30,\ 0,\ldots,\ x_N0,\ x_N1,\ldots,\ x_31,\ x_21,\ x_11.$$

И здесь тоже замыкается правильно. Шаг индукции проведён (и даже двумя способами).

Символически разница между этими двумя способами изображена на рисунке (вертикальная координата соответствует изменению последнего бита, а движение по кривой — изменению первых N битов), но если эта абстрактная живопись воспринимается как бессмысленная мазня, то и ладно — нужный порядок формально описан выше.

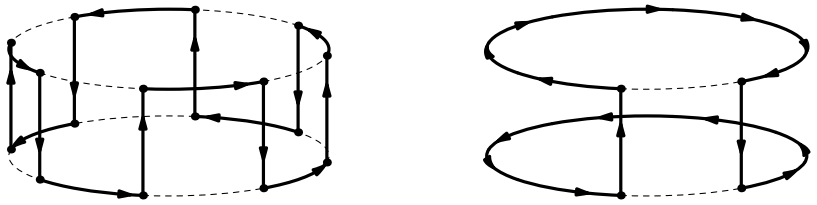


Рис. 1.9: Два варианта индуктивного перехода при построении кодов Грея

.

Задача 1.16. Докажите аналогичный результат для произвольного алфавита.

Указание. При первом способе будет трудность, если размер алфавита нечётный, но при втором её можно обойти: у нас будет не два уровня, а несколько, и можно перескакивать с уровня на уровень в разных местах цикла.

1.11 Теорема Холла о представителях

Мы старались выбрать разнообразные примеры рассуждений по индукции, но все они были достаточно простыми. Но в заключение надо всё-таки привести пример

более сложного рассуждения, в качестве которого мы выбрали знаменитую «теорему Холла о представителях». Она играет важную роль в теории графов, но её можно переформулировать и ничего не говоря о графах, почти как олимпиадную задачу для школьников, и сейчас мы приведём утверждение и доказательство этой теоремы в таких терминах.

Пусть в школе имеется несколько кружков (на разные темы). Администрация хочет назначить в каждом кружке старосту из числа участников этого кружка. При этом нельзя, чтобы один и тот же школьник был старостой сразу в нескольких кружках. Ясно, что это не всегда возможно: скажем, если школьник Петя ходит в два кружка, а больше никто в эти кружки не ходит, то требования невыполнимы — он не может быть одновременно старостой в обоих, а других школьников там нет. Другой (ещё более вырожденный) случай — когда есть один кружок, в который вообще никто не ходит. Вообще, если имеется k кружков, в которые всего ходят меньше k школьников (если собрать все кружки в одном помещении, в нём будет меньше k человек), то задача неразрешима — кандидатов меньше, чем должностей. Оказывается, что это единственное препятствие.

Теорема 1.7 (Теорема Холла о представителях)**.** *Если для любых k кружков общее число школьников, которые ходят хотя бы в один из них, не меньше k, то назначение старост возможно.*

«Представители» — это более научный термин для старост. По-английски это утверждение называют Hall's marriage theorem (попробуйте догадаться, почему) — а доказал её английский математик Филипп Холл в 1935 году.

Доказательство. Индукция по общему числу кружков. Если кружок один, то всё понятно (назначаем старостой любого его участника). Пусть кружков n, и для меньшего количества кружков утверждение теоремы верно. Попробуем волюнтаристский подход: выберем какой-то кружок, назначим там старосту произвольно — пусть это будет Лена, — и предложим остальным $n-1$ кружкам после этого самим разобраться со своими назначениями — разумеется, уже не назначая Лену старостой.

Если им это удастся, то всё хорошо. Если же не удастся, то мы знаем (индуктивное предположение) причину: есть некоторые s кружков, у которых вместе меньше s школьников (не считая Лены). А раньше? Раньше — с Леной — у них было *ровно* s школьников (больше быть не могло, раз стало строго меньше — а меньше быть не могло по условию). При этом $s > 0$ (поскольку меньше нуля школьников быть не может) и $s < n$ (потому что один кружок мы отбросили).

Итак, достаточно рассмотреть случай, когда *некоторые s кружков в объединении включают ровно s школьников, причём* $0 < s < n$. Выберем s кружков с таким свойством и назовём их «особыми». То есть в s особых кружков ходят ровно s школьников. Будем теперь решать задачу отдельно для особых и неособых кружков. И тех, и других меньше n, так что можно воспользоваться предположением индукции. Сначала сделаем это для особых — назначим им старост любым разрешённым способом (таковой существует по предположению индукции). После этого с неособыми кружками сложнее: мы не можем просто так воспользоваться предположением

индукции, так как s участников особых кружков уже назначили старостами и их назначать нельзя. Поступим так: всех участников особых кружков исключим из всех неособых кружков и проверим, что после этого всё равно можно воспользоваться индуктивным предположением. Для этого надо убедиться, что в любые t неособых кружков ходят не меньше t школьников, не посещающих особые кружки. В самом деле, если бы их было меньше t, то вместе с s особыми кружками мы получили бы $s+t$ кружков, в которые ходит меньше $s+t$ школьников, что противоречит предположению теоремы. Шаг индукции завершён. □

Это доказательство, хотя и не очень сложное, производит странное впечатление волшебства (или жульничества, если выражаться менее деликатно) — вроде мы ничего интересного не делаем, скорее переливаем из пустого в порожнее, а почему-то в итоге всё получается. Это бывает с индуктивными рассуждениями, и иногда можно придумать более наглядное (хотя, возможно, и более длинное) доказательство, которое лучше объясняет, «почему» теорема верна. В случае с теоремой Холла о представителях такое доказательство получается, если выводить эту теорему из общего утверждения о потоках в сетях (перевозке грузов по сети дорог с ограниченной пропускной способностью), но мы сейчас про это говорить не будем — мы хотели проиллюстрировать возможности индуктивных рассуждений.

1.12 Задачи для самостоятельного решения

17. Докажите, что для любого целого положительного n выполняется

a) $1+3+5+\cdots+(2n-1)=n^2$;

b) $1\cdot 2^1+2\cdot 2^2+3\cdot 2^3+\cdots+n\cdot 2^n=(n-1)\cdot 2^{n+1}+2$;

c) $1+\dfrac{1}{2}+\dfrac{1}{3}+\cdots+\dfrac{1}{2^n}>\dfrac{n}{2}+1.$

18. Докажите равенства

a) $1\cdot(n-1)+2\cdot(n-2)+\cdots+(n-1)\cdot 1=\dfrac{(n-1)n(n+1)}{6}$;

b) $\cos x+\cos 2x+\cdots+\cos nx=\dfrac{\sin(n+\frac{1}{2})x}{2\sin\frac{x}{2}}-\dfrac{1}{2}$.

19. Докажите неравенства

a) $\sqrt{1}+\sqrt{2}+\cdots+\sqrt{n}>\dfrac{2}{3}\cdot n\sqrt{n}$;

b) $\dfrac{1}{2}\cdot\dfrac{3}{4}\cdot\dfrac{5}{6}\cdot\ldots\cdot\dfrac{2n-1}{2n}<\dfrac{1}{\sqrt{n}}$.

20. Докажите, что 1 можно представить в виде суммы **a)** 2017; **b)** 2018 различных обыкновенных дробей с числителем 1 и положительным знаменателем.

21. В зачете участвовало несколько студентов и преподавателей. Известно, что в комнату, где происходил зачет, каждый участник зачета вошел лишь однажды и что каждый преподаватель поговорил с каждым студентом. Докажите, что в какой-то момент зачета в комнате присутствовали либо все студенты (и, может быть, кто-то из преподавателей), либо все преподаватели (и, может быть, кто-то из студентов).

22. В прямоугольнике $3\times n$ стоят фишки трех цветов, по n штук каждого цвета.

Докажите, что можно переставить фишки в каждой строке так, чтобы в каждом столбце были фишки всех цветов.

23. Из целых чисел от 1 до $2n$ выбрано $n+1$ число. Докажите, что среди выбранных чисел найдутся два, одно из которых делится на другое.

24. На доске написаны сто цифр — нули и единицы (в любой комбинации). Разрешается выполнять два действия:

1. заменять первую цифру (нуль на единицу и наоборот);
2. заменять цифру, стоящую после первой единицы.

Докажите, что после нескольких таких замен можно получить любую комбинацию из 100 нулей и единиц.

25. На краю пустыни имеется большой запас бензина и машина, которая при полной заправке может проехать 50 километров. Имеются (в неограниченном количестве) канистры, в которые можно сливать бензин из бензобака машины и оставлять на хранение (в любой точке пустыни). Доказать, что машина может проехать любое расстояние. (Канистры с бензином возить не разрешается, пустые можно возить в любом количестве.)

26. На кольцевой дороге стоит некоторое количество одинаковых автомобилей. Суммарное количество бензина в их бензобаках достаточно, чтобы один автомобиль мог совершить полный круг. Докажите, что найдется автомобиль, который, начав двигаться против часовой стрелки и забирая бензин по ходу движения у стоящих на дороге автомобилей, сможет совершить полный круг.

27. **a)** Докажите, что любой квадрат $2^n \times 2^n$, из которого вырезана угловая клетка, можно разрезать на уголки из трех клеток.

b) Докажите, что на уголки можно разрезать любой квадрат $2^n \times 2^n$, из которого вырезана любая (не обязательно угловая) клетка.

28*. Целые положительные числа a_1, a_2, …, a_n таковы, что $a_k \leqslant k$ и сумма всех этих чисел четна и равна $2S$. Докажите, что эти числа можно разбить на две группы, сумма по каждой из которых равна S.

29*. Лабиринтом называется клетчатый квадрат 10×10, некоторые пары соседних узлов в котором соединены отрезком — «стеной» — таким образом, что переходя из клетки в соседнюю по стороне клетку и не проходя через стены, можно посетить все клетки квадрата. Границу квадрата будем также считать обнесенной стеной. В некоторой клетке некоторого лабиринта стоит робот. Он понимает 4 команды — Л, П, В, Н, по которым соответственно идет влево, вправо, вверх и вниз, а если перед ним «стена», то стоит на месте. Как написать программу для робота, выполняя которую он обойдет все клетки независимо от лабиринта и от своего начального положения?

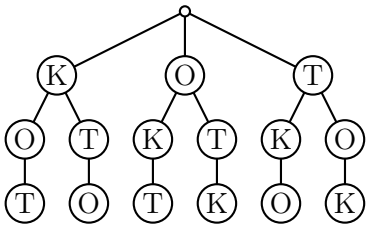
Лекция 2

Подсчёты

Сколько можно составить слов, переставляя буквы в слове КОТ? Нетрудно выписать все такие слова:

КОТ КТО ОКТ ОТК ТКО ТОК

(говоря о словах, мы не имеем в виду что-то осмысленное — просто цепочки букв). Конечно, возникает вопрос, почему мы ничего не пропустили — но это легко сообразить. Сначала написаны два варианта с буквой К на первом месте (две другие могут стоять в том или другом порядке), потом два с буквой О, потом с буквой Т, всего 6 вариантов.



Если бы слова были длиннее, то вручную их выписать было бы труднее, понадобилась бы компьютерная программа (не такая уж простая, кстати), а для совсем длинного слова и программа бы не помогла: вариантов было бы больше, чем можно перечислить в обозримое время. Но можно *подсчитать число вариантов, не перечисляя их все*, и в этой лекции мы будем как раз изучать разные способы такого подсчёта. Научное название для такого рода вопросов — «перечислительная комбинаторика», и это большой раздел (дискретной) математики с очень нетривиальными результатами и методами; мы разберём только самые простые. Начнём мы с совсем тривиальных вещей.

2.1 Правило суммы

Сколько существует четырёхзначных чисел? Это числа от 1000 (предыдущее — трёхзначное 999) до 9999 (дальше идёт пятизначное 10000). Несложно сообразить,

что их 9000. В самом деле, из списка

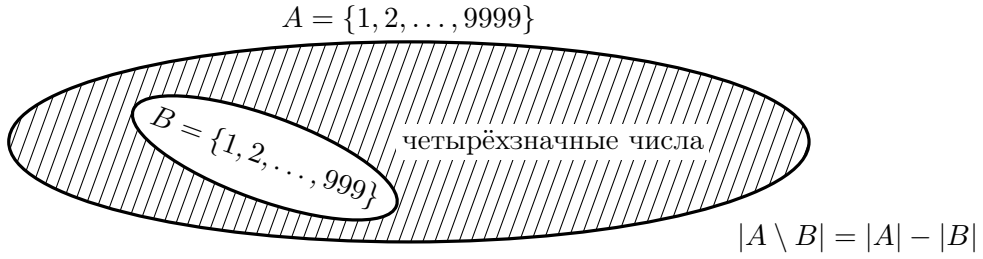
$$1, 2, 3, 4, 5, \ldots, 9999$$

в котором 9999 чисел (один, два, три,. . . , 9999), мы должны вычеркнуть числа

$$1, 2, 3, 4, 5, \ldots, 999$$

(первые 999), остаётся $9999 - 999 = 9000$.

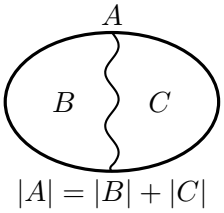
Это рассуждение — очень простой вариант более общего рассуждения, которое будет использоваться дальше очень часто. Мы рассмотрели *множество*[1] A, состоящее из чисел $1, 2, \ldots, 9999$, и его *подмножество* B, состоящее из чисел $1, 2, \ldots, 999$, и затем подсчитали, сколько чисел входит в их *разность* $A \setminus B$, множество всех четырёхзначных чисел. Это количество равно разности количеств чисел в A и в B, см. рисунок.



Мы обозначаем количество чисел в множестве X через $|X|$.

> **Правило суммы.** *Если какое-то множество A разделено на две части B и C, не имеющие общих элементов, то $|A| = |B| + |C|$.*

Другими словами, если надо подсчитать количество объектов какого-то вида, и эти объекты можно поделить на непересекающиеся типы, то общее количество объектов равно сумме количеств объектов каждого типа.



Философы могут спросить: как доказать это утверждение? или это не теорема, а аксиома, и доказывать её не надо? Не углубляясь в основания математики, можно ответить им, что это и не теорема, и не аксиома, а *определение* сложения: когда в начальной школе учили складывать числа с помощью «счётных палочек», то предлагали отсчитать три палочки, потом отдельно четыре палочки, а потом подсчитать их все и получить семь палочек — это и значит, что четыре плюс три равно семи.

[1] Подробно множества и операции с ними обсуждаются в лекции 5.

Задача 2.1. Какое наименьшее четырёхзначное число делится на 7? какое наибольшее? Сколько четырёхзначных чисел делятся на 7?

Задача 2.2. Сколько четырёхзначных чисел делятся на 123?

Задача 2.3. Сколько четырёхзначных чисел *не* делятся на 123?

Задача 2.4. Каких четырёхзначных чисел больше — чётных или нечётных? Почему?

В правиле суммы важно, что

- всякий элемент множества A входит либо в B, либо в C (мы никого не пропустили);
- в B и C нет общих элементов.

Если в классе m мальчиков и n девочек, то всего в нём $m+n$ школьников. Но если в классе m знающих английский язык и n знающих немецкий, то отсюда не следует, что всего там $m+n$ человек: во-первых, кто-то может не знать ни того, ни другого (и мы его не посчитаем в $m+n$), во-вторых, кто-то может знать оба языка (и мы его посчитаем дважды). Если мы хотим получить общее количество людей, знающих хотя бы один язык (из этих двух), то надо сложить число знающих английский и число знающих немецкий *и вычесть число посчитанных дважды*, то есть число знающих оба языка.

Другой пример: посчитаем, *сколько чисел от* 1 *до* 1000 *делятся на* 2 *или на* 3 (то есть делятся хотя бы на одно из этих двух чисел). Подсчитать чётные (делящиеся на 2) просто: числа

$$1,2,\ 3,4,\ 5,6,\ \ldots,\ 999,1000$$

делятся на пары (нечётное, чётное), и этих пар $500 = 1000/2$, то есть чётных чисел 500. Немного сложнее подсчитать делящиеся на 3; если делить на тройки, то одно число останется:

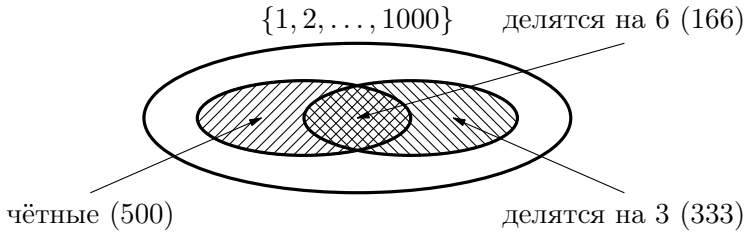
$$1,2,3,\ 4,5,6,\ 7,8,9,\ \ldots,\ 997,998,999,\ 1000.$$

Будет $333 = 999/3$ полные тройки и ещё одно число (не делящееся на 3), то есть всего есть 333 числа, делящиеся на 3. Сколько же чисел, делящихся на два или на три? Если мы сложим 500 и 333, то некоторые числа мы подсчитаем дважды. Это числа, которые делятся и на 2, и на 3, то есть делятся на 6, их в каждой шестёрке

$$1,2,3,4,5,6,\ 7,8,9,10,11,12,\ \ldots,\ 991,992,993,994,995,996,\ 997,998,999,1000.$$

по одному (а в последней группе из четырёх чисел нет вовсе).[2] Делим 1000 на 6 с остатком: $1000 = 166\times6+4$, видим, что шестёрок будет 166. Теперь можно посчитать, сколько чисел делятся на 2 или на 3:

[2]Бдительный читатель спросит здесь, почему свойства «делиться на 2 и делиться на 3» и «делиться на 6» равносильны. Тут можно сослаться на единственность разложения на множители (см. лекцию 4) или просто рассмотреть всевозможные остатки при делении на 6.



$$|A \cup B| = |A| + |B| - |A \cap B|$$

Если обозначить множество чётных чисел до 1000 за A, делящихся на 3 — за B, то можно записать наш подсчёт так:

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Здесь $A \cup B$ — это *объединение множеств* A *и* B (всё, что входит туда или сюда), $A \cap B$ — их *пересечение* (всё, что входит и туда, и сюда), а $|X|$ — число элементов в множестве X.

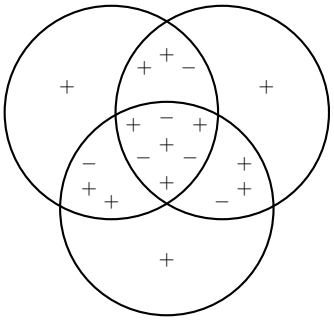
Задача 2.5. Сколько чисел от 1 до 1000 не делятся ни на 2, ни на 3?

Задача 2.6. Доля чисел, не делящихся ни на 2, ни на 3 в предыдущей задаче, близка к 1/3. Как это объяснить, глядя на шестёрки чисел?

Аналогичные подсчёты возможны и для большего числа множеств. Скажем, пусть в классе кто-то знает немецкий, кто-то английский, кто-то французский, и мы хотим подсчитать, сколько человек знают хотя бы один из этих трёх языков. Можно сложить все три числа, но это будет явно с избытком: тех, кто знают два языка из трёх, мы посчитаем дважды. Чтобы скомпенсировать это, можно вычесть три пересечения (знающих английский и немецкий, знающих английский и французский, и знающих немецкий и французский). Тогда получится правильно? Тоже нет: человека, знающего три языка, мы сначала три раза посчитали, а потом три раза вычли — и в итоге ни разу не посчитали, значит, его надо добавить. В итоге получаем такую формулу:

$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap C| - |B \cap C| + |A \cap B \cap C|.$$

Её называют *формулой включений-исключений* для трёх множеств — сначала включаем все множества, потом исключаем попарные пересечения, потом снова включаем пересечение всех трёх. По существу мы её уже доказали, убедившись, что в правой части любой элемент (входящий ровно в одно из трёх множеств, входящий ровно в два из трёх множеств, и входящий во все три) посчитан правильно (по разу).



Для каждого слагаемого $\pm|X|$ в формуле мы ставим знак слагаемого в те части картинки, где есть элементы из X. Правда, там виден только результат, а не процесс, но можете проделать это сами и убедиться, что в каждой части плюсов на один больше, чем минусов.

Задача 2.7. Сколько чисел от 1 до 1000 делятся хотя бы на одно из чисел $2, 3, 5$?

Задача 2.8. Как могла бы выглядеть аналогичная формула для четырёх множеств? Как её можно было бы доказать?

Мы ещё вернёмся к формуле включений-исключений для произвольного числа множеств и докажем её (даже разными способами).

Задача 2.9. Два колокола начали бить одновременно. Удары одного следуют через 2 секунды, а другого — через три секунды. Сколько ударов слышно в минуту, если одновременные удары двух колоколов считать за один?

В музыке такое встречается, когда в одном голосе идут триоли, а в другом обычные восьмые (скажем). Это не так просто сыграть без подготовки. Попробуйте двумя руками стучать по столу: левой на три счёта, правой — на два (каждая рука — как метроном, но частоты отличаются в полтора раза). Получилось? Если да, можно попробовать отношения $3:4$ или $3:5$.

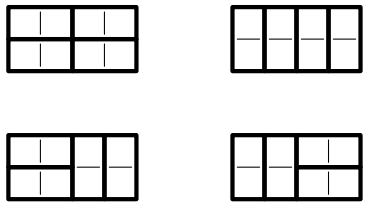
Задача 2.10. Тот же вопрос, если есть ещё и третий колокол, который бьёт раз в 5 секунд (и начал бить в то же время).

2.2 Рекуррентное соотношение: пример

Если нужно подсчитать какие-то предметы (в жизни или в математике), можно разбить их на группы, подсчитать отдельно в каждой группе и потом полученные числа сложить (правило суммы). Собственно, мы с этого начинали, переставляя буквы в слове КОТ: все варианты делятся на три группы (начиная с К, начиная с О, начиная с Т), а в каждой группе по два слова (две оставшиеся буквы идут в том или другом порядке). Всего получается $2 + 2 + 2 = 3 \times 2 = 6$ вариантов.

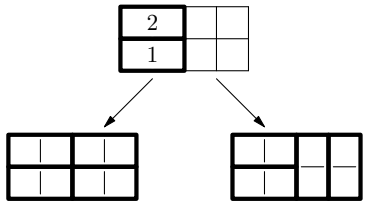
Тот же самый принцип действует и в более сложных случаях. Пусть *мы хотим замостить прямоугольник* 2×4 *доминошками* 1×2. *Сколькими способами это*

можно сделать? Можно класть их все горизонтально, или все вертикально, или часть положить горизонтально, а часть вертикально. На рисунке показано четыре варианта, но все ли это варианты? не пропустили ли мы чего-нибудь?

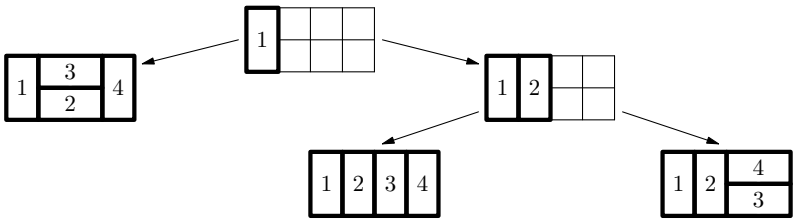


Задача 2.11. Укажите пропущенный вариант.

А теперь? ну хорошо, есть пятый вариант (с краёв вертикально, в середине горизонтально) — но, может, ещё что-то пропущено? Как это узнать? Давайте рассуждать логически, как советовал профессор Стравинский Ивану Бездомному у Булгакова. Посмотрим на левое нижнее поле ($a1$, сказали бы шахматисты). Оно должно быть покрыто доминошкой, которая может быть (а) горизонтальной или (б) вертикальной. Если она горизонтальна (плитка 1), то над ней неизбежно должна быть вторая горизонтальная (плитка 2). Остаётся замостить квадрат 2×2, и это можно сделать двумя способами.



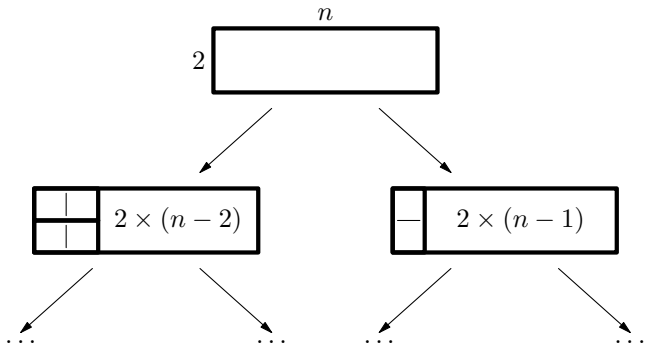
Остался случай (б), когда первая доминошка вертикальна. Тогда остаётся замостить прямоугольник 3×2. Тут снова приходится разбирать случаи.



Левый нижний угол может быть покрыт горизонтальной доминошкой 2 (левый вариант на картинке), тогда над ней будет однозначно 3, а справа 4. А может быть покрыт вертикальной: тогда остаётся квадрат 2×2, который можно замостить двумя способами.

Вот теперь мы не только нашли все возможные способы, но и убедились, что других нет: в группе (а) есть два варианта, в группе (б) есть три варианта, а всего пять вариантов.

Аналогичное рассуждение можно применить и к полю размера $2 \times n$.



Если левая нижняя клетка покрыта горизонтальной доминошкой, то над ней тоже горизонтальная, остаётся покрыть прямоугольник $2 \times (n-2)$. А если левая нижняя клетка покрытая вертикальной доминошкой, то остаётся покрыть прямоугольник $2 \times (n-1)$. То есть все способы — обозначим их число через $T(n)$ — делятся на две группы. В первой группе $T(n-2)$ способов, а во второй $T(n-1)$, итого будет

$$T(n) = T(n-2) + T(n-1).$$

Мы начали с прямоугольника 2×2, у которого два варианта. Если угодно, можно сказать, что $T(2) = T(0) + T(1)$, а прямоугольники 2×0 и 2×1 можно замостить единственным способом. Для прямоугольника 2×3 есть $T(1) + T(2) = 2 + 1 = 3$ варианта, для прямоугольника 2×4 есть $T(2) + T(3)$ варианта, и так далее. Разбиение $T(4)$ на $T(2) + T(3)$ мы как раз и обсуждали подробно, и подсчитали, что $T(4) = 5$. Далее $T(5) = T(3) + T(4) = 3 + 5 = 8$, $T(6) = T(4) + T(5) = 5 + 8 = 13$ и так далее.

Задача 2.12. Нарисуйте все 8 вариантов замощения для прямоугольника 2×5. Как они делятся на группы по 3 и 5?

Мы получили, как говорят, *рекуррентную формулу* для $T(n)$. Слово «рекуррентный» происходит от латинского recurrere (возвращаться, идти назад) и означает, что очередное значение $T(n)$ вычисляется через предыдущие значения. По этой формуле можно легко подсчитать $T(n)$ и дальше, для $n = 7, 8, 9, \ldots$; эту последовательность $1, 1, 2, 3, 5, 8, 13, \ldots$ называют *числами Фибоначчи*, по имени итальянского математика XIII века, хотя вроде бы она была и раньше известна в Индии. (Фибоначчи объяснял её на примере размножающихся кроликов, но весь этот «оживляж», кажется, только запутывает дело.) Для чисел Фибоначчи есть странная формула

$$T(n-1) = \frac{\varphi^n - \psi^n}{\sqrt{5}}, \quad \text{где} \quad \varphi = \frac{1+\sqrt{5}}{2}, \quad \psi = \frac{1-\sqrt{5}}{2}.$$

Задача 2.13. Проверьте, что по этой формуле при $n = 1$ и при $n = 2$ получится 1.

Задача 2.14. Проверьте, что в последовательности, построенной по этой формуле, каждое число равно сумме двух предыдущих. (Указание: числа φ и ψ являются корнями уравнения $1 + x = x^2$.)

Задача 2.15. Докажите по индукции эту формулу для последовательности Фибоначчи.

Задача 2.16. Проверьте, что можно найти числа Фибоначчи, просто округлив $\varphi^n/\sqrt{5}$ до ближайшего целого числа. Объясните, почему так получается.

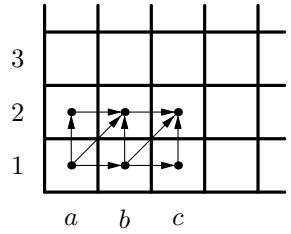
С точки зрения практического вычисления, непонятно, помогает ли эта формула или мешает. С одной стороны, вместо n сложений (если вычислять их подряд) нужно вычислять n-е степени, а это программисты умеют делать за $O(\log n)$ действий, возводя в квадрат и потом перемножая нужные степени двойки, то есть число операций сильно уменьшается. С другой стороны, операции нужно производить с действительными числами, и заранее неясно, с какой точностью, так что возникают дополнительные сложности. Но можно соединить достоинства этих двух способов и ограничиться $O(\log n)$ действиями с целыми числами, но для этого надо знать линейную алгебру и вычислять степени матрицы $\begin{pmatrix}1 & 1\\ 0 & 1\end{pmatrix}$.

Задача 2.17. (Для знакомых с линейной алгеброй) Как это сделать?

Задача 2.18. Требуется написать строку из n букв А и Б, причём есть такое ограничение: две буквы А не должны идти подряд. Сколькими способами это можно сделать? (Указание: поделим все варианты на две группы — если первая буква А и если первая буква Б.)

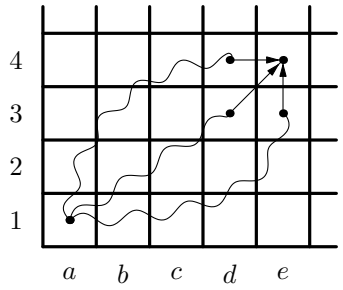
2.3 Рекуррентное соотношение: число путей

Сейчас мы применим тот же метод деления на части и составления рекуррентного соотношения в другой задаче. Она тоже будет на шахматной доске. Представим себе, что на поле $a1$ стоит шахматный король, которому разрешено двигаться только вправо, вверх и по диагонали вправо-вверх. Он может попасть на любое поле, причём не одним способом. Например, на поле $c2$ он может попасть пятью способами.



Задача 2.19. Покажите эти пять способов на картинке. (Они пересекают прямую между первой и второй горизонталью в пяти разных местах.)

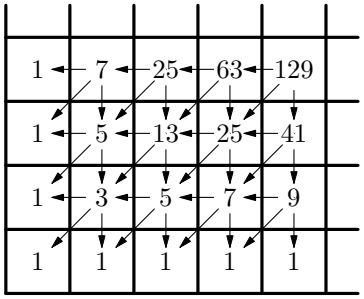
Если речь пойдёт о более далёком поле, скажем, $e4$, то вариантов путей будет много, и перечислять их долго. *Как подсчитать количество вариантов, не перечисляя их*? Давайте пойдём с конца. Откуда король мог попасть на поле $e4$?



Есть три возможных поля: $d4$, $d3$ и $e3$. Соответственно все возможные пути делятся на три (непересекающиеся) группы, и надо сложить количества путей в каждой из этих групп. То есть мы свели задачу к такой же задаче для других клеток:

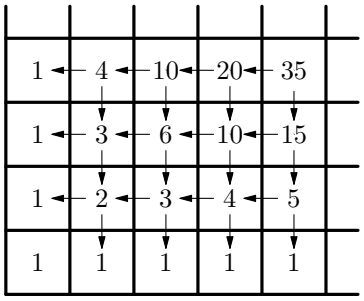
$$T(e4) = T(d4) + T(d3) + T(e3),$$

где $T(X)$ обозначает число способов попасть в клетку X. Аналогичное соотношение можно написать для всех других клеток, кроме клеток на нижней или верхней границе, когда вариант только один (идти вдоль границы — если мы отойдём, то не сможем вернуться). Теперь, пользуясь этим соотношением, мы можем вычислить ответы для всех клеток.



Каждое число, кроме крайних единиц, равно сумме трёх чисел: слева, снизу и слева-снизу (по рекуррентной формуле), и после нескольких сложений получаем ответ: в клетку $e4$ можно попасть 129 способами.

Если мы запретим королю ходить по диагонали, а разрешим ходить только на клетку вправо или вверх (отчего он немного станет ладьёй), то способов станет, естественно, меньше. И рекуррентная формула будет другой: чтобы попасть на клетку $e4$ можно только с $e3$ или $d4$, групп не три, а две. Соответственно число вариантов в каждой клетке равно сумме чисел под ним и слева от него:



Для клетки $e4$ получается 35 вариантов. Знающие люди сразу узна́ют в этой картинке *треугольник Паскаля*, только повёрнутый, и могут получить тот же ответ 35 по формуле

$$\binom{7}{3} = \frac{7!}{4!3!} = \frac{7 \cdot 6 \cdot 5}{1 \cdot 2 \cdot 3} = 35,$$

но и безо всех этих знаний заполнить таблицу можно совсем быстро и прийти к тому же ответу.

Мы ещё вернёмся к этой задаче, но сейчас рассмотрим несколько совсем простых рекуррентных соотношений.

2.4 Слова и правило произведения

Будем рассматривать пятизначные числа, в которых встречаются только цифры 1 *и* 2. Наименьшее из них 11111, наибольшее 22222, но идут они с большими промежутками, так что их на самом деле совсем немного. *Сколько*?

На первом месте может стоять 1 или 2, так что все такие числа делятся на две группы. Сколько чисел в каждой группе? нам остаётся дописать четыре цифры, так что надо подсчитать количество четырёхзначных чисел из цифр 1 и 2. Получаем, что пятизначных чисел вдвое больше, чем четырёхзначных: $T(5) = 2T(4)$, если за $T(n)$ обозначить количество n-значных чисел, составленных из цифр 1 и 2. Другое объяснение: из каждого четырёхзначного числа можно получить два пятизначных, приписав к нему единицу или двойку. Значит, пятизначных чисел вдвое больше.

По тем же причинам $T(4) = 2T(3)$ и так далее:

$$T(5) = 2T(4) = 4T(3) = 8T(2) = 16T(1) = 16 \cdot 2 = 32.$$

Мы использовали здесь, что $T(1) = 2$, поскольку однозначных чисел ровно два (числа 1 и 2). Можно было бы формально написать $T(1) = 2T(0) = 2 \cdot 1 = 2$, считая, что нульзначные числа не содержат ни одной цифры, тут выбора нет, и вариант только один. Так или иначе, $T(5) = 32 = 2^5$.

Задача 2.20. А чему равна сумма этих 32 чисел? (Чтобы её найти, не обязательно их все выписывать и складывать!)

Точно так же можно подсчитать, скажем, *количество семизначных чисел, в которых все цифры нечётны.* На первом месте может стоять любая из пяти нечётных цифр $1, 3, 5, 7, 9$, так что все они делятся на пять групп (в зависимости от первой цифры). Сколько чисел в каждой группе? Столько, сколько есть шестизначных чисел из нечётных цифр, и так далее. Получаем рекуррентное соотношение $T(n) = 5T(n-1)$ с начальным условием $T(1) = 5$, откуда

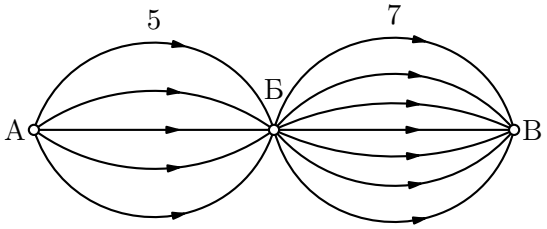
$$T(7) = 5T(6) = 5^2T(5) = 5^3T(4) = 5^4T(3) = 5^5T(2) = 5^6T(1) = 5^6 \cdot 5 = 5^7.$$

Задача 2.21. Найдите сумму всех этих чисел. (Теперь выписывать и складывать их явно слишком долго.)

В общей форме это утверждение можно сформулировать так. Пусть есть какой-то *алфавит*, содержащий n символов (*букв* этого алфавита, как говорят), и мы составляем последовательности (цепочки) из k букв этого алфавита. Среди математиков их принято называть *словами длины* k в этом алфавите, хотя, конечно, никакого смысла в этих «словах» не ищут. Программисты же (и в последнее время не только они) говорят «строки длины k», видимо, переводя английское слово *string*. (Иногда его даже не переводят и говорят о «стрингах длины k», а способы быстрой обработки слов называют «стрингологией», от английского жаргонного слова stringology — но это уже, пожалуй, лингвистический экстремизм.)

Так вот, мы по существу доказали, разбирая наши примеры, что *число различных слов длины k в алфавите из n символов равно n^k*. Формально это легко доказывается индукцией по k. При $k = 1$ получаем n слов длины n, то есть n букв (база индукции). Слова длины k делятся на k групп по первой букве, в каждой группе к этой первой букве дописывают любое слово из $k-1$ букв, поэтому в группе n^{k-1} слов (предположение индукции), а всего $n \cdot n^{k-1} = n^k$, что и требовалось доказать.

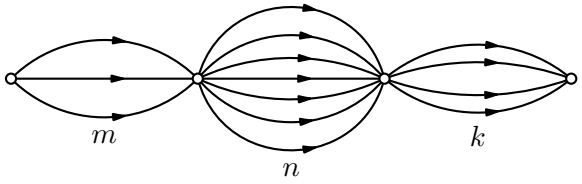
Ту же самую формулу можно объяснить немного другим образом. Представим себе, что *из города А в город Б ведут* 5 (*непересекающихся*) *дорог, а из города Б в город В ведут* 7 (*непересекающихся*) *дорог. Сколькими способами можно проехать из А до В в два приёма* (*сначала Б, и потом в В*)?



Ясно, что есть 35 вариантов: каждая из пяти возможностей на первом шаге может комбинироваться с каждой из семи возможностей на втором. (Если следовать нашей схеме, то все варианты делятся на пять групп, по своему первому шагу, и в каждой группе семь вариантов — по второму.)

Аналогичное утверждение верно и для большего числа этапов: если на первом шаге можно выбрать любой из m способов, на втором шаге (независимо от того, что

выбрано на первом) можно выбрать любой из n способов, а на третьем — любой из k способов, то всего путей будет mnk.



Формально говоря, его тоже можно доказывать индукцией по числу этапов, но это и так понятно: если у матери m дочерей, у каждой дочери по n дочерей, а у каждой из последних по k дочерей, то всего будет mnk правнучек. На генеалогическом дереве (по женской линии) у корня будет m дочерей, у каждой из них по n дочерей, у которых по k дочерей — всего mnk листьев (вершин верхнего уровня). Ясно, что коэффициенты ветвления на всех уровнях перемножаются (на каждом следующем уровне больше вершин во столько раз, каков коэффициент ветвления).

Приведённые выше примеры иллюстрируют «правило произведения».

> **Правило произведения.** *Если объект интересующего нас вида строится в несколько шагов* $(1, 2, \ldots, k)$*, и на каждом шаге есть выбор из какого-то числа вариантов* $(m_1, m_2, \ldots, m_k)$*, причём количество выборов на каждом шаге не зависит от сделанных ранее выборов, то общее количество объектов* n *равно произведению количеств вариантов выбора для каждого из шагов:* $n = m_1 \times m_2 \times \ldots \times m_k$.

Задача 2.22. В каждый из пяти дней недели один из 20 школьников класса объявляется дежурным. Сколькими способами можно составить таблицу дежурств на неделю? (На каждый день дежурным могут назначить любого школьника, независимо от того, дежурил он уже или нет.)

Задача 2.23. В некоторой стране[3] автомобильные номера состоят из трех цифр в начале и трёх букв в конце, при этом можно использовать любые цифры от 0 до 9 и любые английские буквы от A до Z (их 26). Какое максимальное число различных автомобильных номеров можно составить таким образом?

С другими применениями этой формулы хорошо знакомы программисты: скажем, существует $256 = 2^8$ различных байтов (последовательностей из восьми битов, или слов длины 8 в двоичном алфавите $\{0, 1\}$).

[3] В России автомобильные номера образца 1993 года тоже содержат три буквы и три цифры, но разрешается использовать лишь 12 букв, похожих на латинские, набор из трёх нулей не допускается. Ещё в правой части номера указывается код региона (обычно две цифры). Количество этих кодов постепенно увеличивается, когда старых номеров не хватает, и даже появляются трёхзначные коды регионов. Так что жизнь сложнее нашей формулы.

Задача 2.24. Шахматный король, который умеет ходить только вправо и вверх на одну клетку, находится на поле $a1$ и должен попасть на диагональ доски, идущую из $a8$ в $h1$. Сколько ходов он для этого должен сделать? Сколькими способами он может это сделать?

Можно подсчитать и количество различных подмножеств у множества из n элементов. Допустим, *в классе n школьников, и некоторые из них пошли сегодня вечером на концерт. Сколько может быть вариантов* (кто пошёл, а кто не пошёл)? Здесь подмножество состоит из тех школьников, которые были на концерте. Возможно, что никого не было — тогда получается, как говорят, *пустое* множество. Допускается и вариант, когда были все n школьников (само множество тоже считают его подмножеством).

Представим себе, что мы составляем отчёт о посещении концерта, поставив рядом с каждой из n фамилий плюс или минус (был или не был). Получится слово длины n в алфавите из двух символов $\{+, -\}$, и таких слов, как мы знаем, 2^n.

Другое объяснение: выберем одного из n школьников, скажем, Васю, и все варианты посещения концерта поделятся на две категории: те, где Вася был на концерте, и те, где он не был. А в каждой группе столько вариантов, сколько подмножеств у множества с $n-1$ элементами, так что с каждым новым школьником число вариантов увеличивается вдвое.

Или так: у школьников по очереди спрашивают, пойдут ли они на концерт. Получается n этапов, на каждом из них выбор из двух вариантов, по правилу произведения всего 2^n вариантов.

Задача 2.25. Каких подмножеств у множества $\{a, b, c, d, e, f\}$ больше — содержащих букву a или не содержащих? Каких подмножеств больше у того же множества — из чётного числа элементов или из нечётного числа элементов?

Задача 2.26. Будем рассматривать расстановки слонов на шахматной доске, при которых они не бьют друг друга. (Учитываем расстановки с любым количеством слонов на доске, в том числе и равным нулю). Покажите, что число расстановок является точным квадратом (квадратом целого числа).

2.5 Выбор с ограничениями

Применять формулу из предыдущего раздела нужно аккуратно. Скажем, в самом начале мы составляли слова длины 3 из букв К, О, Т, но получили не $3^3 = 27$ слов, по нашему теперешнему подсчёту, а только 6. В чём тут дело? Или ещё: подсчитаем четырёхзначные числа: четыре места, десять цифр, по формуле их будет $10^4 = 10000$, а у нас получалось 9000. Что не так?

Дело в том, что в обоих случаях мы считаем лишнее: нас интересуют только комбинации с некоторыми ограничениями. Составляя слова из К,О,Т, мы разрешали только переставлять буквы, то есть каждую букву можно было использовать по разу (не больше и не меньше). В четырёхзначных числах на первом месте не может

стоять нуль (тогда будет трёх- или меньше-значное число). Однако и в том, и в другом случае можно получить правильный ответ по формуле произведения.

Для четырёхзначных чисел: на первом месте может стоять любая из 9 цифр $1, 2, \ldots, 9$ (кроме нуля), на втором, третьем и четвёртом — любая из десяти цифр, поэтому в произведении получаем как раз $9 \cdot 10 \cdot 10 \cdot 10 = 9000$. Аналогично с котом: на первом месте может стоять любая из трёх букв К,О и Т. После того как первая буква выбрана, на втором месте может стоять любая из двух оставшихся букв — ну, а на третьем месте уже никакого выбора не остаётся, ставим ту букву, что не использована. Всего получаем как раз $3 \cdot 2 \cdot 1 = 6$.

Задача 2.27. Забор состоит из 100 вертикальных досок (самая левая, потом вторая слева, потом третья, и так далее до сотой). Маша хочет покрасить каждую доску забора в какой-то цвет. У неё есть три разных краски, и при этом она не хочет красить соседние доски в один и тот же цвет (иначе границы не видно). Сколько вариантов раскраски есть?

Задача 2.28. Сколько существует четырёхзначных чисел, в которые входит (хотя бы раз) цифра 7? (Указание. Эту задачу можно решать, разбивая числа на группы в зависимости от такого, на каких местах появляется семёрка. Но проще подсчитать четырёхзначные числа, в которые цифра 7 *не* входит.)

Важный случай выбора с ограничениями — это перестановки. Мы уже рассматривали все перестановки в слове из трёх букв, и насчитали шесть таких перестановок. А что будет для слова из n букв? *Сколько слов можно получить из него перестановками букв*? (Все буквы в слове считаем разными.)

Первую букву можно выбрать n способами. После этого (для каждого из n вариантов) вторую букву можно выбрать $n-1$ способами, взяв любую из $n-1$ оставшихся букв. На третьем шаге у нас $n-2$ способов, и так далее до последнего шага, когда уже выбора нет (осталась ровно одна буква). Всего (формально говоря, по индукции) получается

$$n(n-1)(n-2)(n-3) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$

способов, для этого числа есть специальное обозначение $n!$ (читается «эн факториал»).

Задача 2.29. Что больше: 100! или 100^{100}?

Эту задачу можно решить комбинаторно, посчитав, сколько всего есть слов длины 100 в 100-буквенном алфавите и сколько их них не содержат повторений букв (и потому каждую букву содержат по одному разу).

Более общая ситуация, чем перестановки — это выбор без повторений. Пусть есть какие-то n предметов, и нужно составить упорядоченный список, взяв k из них. Например, *пусть в соревновании участвуют* 20 *спортсменов, и разыгрываются три медали — золотая, серебряная и бронзовая.* В нашей терминологии это значит, что $n = 20$, а $k = 3$. *Сколько различных возможных исходов у такого соревнования*?

Победителем может стать любой из 20 человек. После того как победитель выбран, остаётся 19 кандидатов на второе место, а после того как один из них выбран, 18 кандидатов на третье место. Всего, таким образом, $20 \cdot 19 \cdot 18$ вариантов.

В общем случае надо перемножить k сомножителей $n(n-1)(n-2) \cdot \ldots \cdot (n-k+1)$ (проверьте, что последний из них написан правильно и получается именно k сомножителей).

В терминах слов последнее утверждение можно сформулировать так: *количество k-буквенных слов в n-буквенном алфавите, не содержащих повторяющихся букв, равно* $n(n-1)(n-2)\cdot\ldots\cdot(n-k+1)$. Это число иногда обозначают $(n)_k$ по аналогии со степенью n^k (которая получится, если не запрещать повторения букв).

Можно ещё записать, что

$$(n)_k = \frac{n!}{(n-k)!}$$

(в этой дроби последние $n-k$ множителей числителя сокращаются со знаменателем, и остаётся ровно то, что мы написали).

Мы предполагали, определяя число $(n)_k$, что $k \leqslant n$. Как вы думаете, как разумно определить $(n)_k$ при $k > n$? По формуле надо умножать числа n, $n-1$, и так далее до 1, потом дойдет дело до нуля (ведь $k > n$), и даже до отрицательных чисел. Но в любом случае в произведении будет нуль (один из сомножителей равен нулю). И с комбинаторным смыслом это тоже согласовано: нельзя составить слова без повторений, у которых длина больше числа букв в алфавите. Значит, количество таких слов равно нулю.

2.6 Подсчёты с кратностью

Вернёмся опять к перестановкам букв в слове КОТ. Что будет, если вместо слова КОТ взять, скажем, слово ТОТ? Тогда получится только три слова: ОТТ, ТОТ и ТТО. И понятно, почему их стало меньше: потому что в слове ТОТ есть две одинаковые буквы. Если в словах

КОТ КТО ОКТ ОТК ТКО ТОК

заменить К на Т, то получится список

ТОТ ТТО ОТТ ОТТ ТТО ТОТ,

в котором каждое слово входит два раза (перестановка букв К и Т давала новое слово, а перестановка двух букв Т — нет), и разных слов только три.

Вот ещё пример аналогичной ситуации. Предположим, что *в турнире* 10 *команд каждая сыграла по одному разу с каждой. Сколько было игр*? Будем считать это число так: каждая команда сыграла девять игр (с каждой из 9 остальных), команд 10, получается $10 \cdot 9 = 90$ игр. Правильно это?

Нет: скажем, для трёх команд подобный подсчёт даёт $3 \times 2 = 6$ игр, в то время как реально игры будет только три (отдыхать может любая из трёх команд). В

чём тут ошибка? Дело в том, что игру между любыми двумя командами А и Б мы посчитали дважды: один раз — когда мы считали игры команды А, и второй раз — когда считали игры команды Б. Значит, полученный результат вдвое больше истинного числа игр, и в случае 10 команд игр было не $10 \cdot 9 = 90$, а $10 \cdot 9/2 = 45$.

Задача 2.30. Будем считать иначе: когда первая команда отыграет свои $(n-1)$ игр, останется та же задача (подсчёт общего числа игр) для $n-1$ команд. Эта рекуррентная формула даёт ответ $(n-1) + (n-2) + \ldots + 3 + 2 + 1 + 0$ (когда останется одна команда, число игр равно нулю). Совпадает ли этот ответ с полученным нами раньше?

Задача 2.31. Сколько диагоналей в выпуклом n-угольнике? (Указание: из каждой вершины выходят $n-3$ диагонали во все места, кроме самой вершины и двух её соседей.)

Задача 2.32. Сколько есть двухэлементных подмножеств в множестве из n элементов? Почему эта задача повторяет задачу о количестве игр в турнире n команд?

Приведём ещё несколько примеров такого рода.

Вспомним задачу о соревновании, в котором из 20 участников определяются три победителя (золотой, серебряный и бронзовый призёр). Как мы посчитали, есть $20 \times 19 \times 18$ возможных исходов такого соревнования. *Что изменится, если в соревновании по-прежнему три победителя, но они никак не ранжированы, просто выбрано три человека из двадцати*? Тогда в списке из $20 \times 19 \times 18$ исходов каждый вариант будет встречаться 6 раз (ведь если уже фиксированы три победителя, то распределить между ними золотую, серебряную и бронзовую медаль можно $3! = 6$ способами, это мы тоже знаем). Значит, чтобы подсчитать количество возможных исходов в новом типе соревнования, старый ответ нужно поделить на $6 = 3!$, получится

$$\frac{20 \cdot 19 \cdot 18}{1 \cdot 2 \cdot 3} = 1140 \text{ исходов.}$$

Задача 2.33. Сколько существует 3-элементных подмножеств у 20-элементного множества?

Задача 2.34. В классе из 20 человек нужно выбрать старосту и двух его помощников. Сколькими различными способами это можно сделать? (Каким будет это число, если у старосты есть первый помощник и второй помощник, и как оно изменится, если перестать их ранжировать?)

Задача 2.35. Семь человек, среди которых есть Аня, Беня и Ваня, становятся в очередь. Сколькими способами они могут это сделать? В какой доле из них Аня стоит ближе к началу, чем Беня? В какой доле из них Аня стоит дальше от начала, чем Беня, но ближе, чем Ваня? В какой доле из них Аня стоит ближе к началу, чем Беня и чем Ваня?

Другой пример. Пусть *есть n людей, которые хотят образовать хоровод* (встать в круг и кружиться). *Сколькими способами они могут это сделать*? Разница между хороводом, и, скажем, очередью в том, что варианты, отличающиеся лишь поворотом круга, считаются за один (когда они начнут кружиться, уже никто не вспомнит, с чего началось).

Тот же приём позволяет получить ответ. Если не разрешать хороводу кружиться, а просто нарисовать n точек в вершинах правильного n-угольника, и считать, сколькими способами можно расставить в них n человек, то получится, как мы знаем, $n!$ способов. Если вспомнить о вращении, то получатся группы из n вариантов, отличающиеся лишь поворотом (из одного способа можно получить n, рассмотрев все повороты), поэтому этих групп будет $n!/n = (n-1)!$.

Задача 2.36. Как получить тот же самый ответ, описывая хоровод с точки зрения одного из участников?

С последней задачей связан любопытный парадокс (или, точнее сказать, софизм). Представим себе *круг, разбитый на 7 равных секторов. Каждый из секторов нужно раскрасить в красный или синий цвет. При этом раскраски, отличающиеся поворотом круга, считаем за одинаковые. Сколькими способами можно это сделать*?

Действуем как раньше: если круг не вращать, то всего есть 7 секторов, для каждого из них надо решить, в какой из двух цветов красить, получаем по правилу произведения $2^7 = 128$ вариантов. Повороты круга, как мы видели, делят их на группы по 7, так что групп будет $128/7 = 18\frac{2}{7}$. Это, конечно, ерунда — групп должно быть целое число. Где мы ошиблись? Постарайтесь догадаться, не читая объяснения в следующем абзаце.

* * *

На самом деле не все группы будут по 7: если все сектора круга красные (или синие), то вращение ничего не меняет. То есть есть две группы, состоящие из единственной раскраски. Остальные группы, как можно проверить (тут важно, что число 7 простое), действительно состоят из 7 элементов, их $(128-2)/7 = 18$, добавляя две особые группы, получаем ответ: 20 вариантов.

Задача 2.37*. Почему важно, что число n простое? Пройдёт ли это рассуждение, скажем, для круга из четырёх секторов? Покажите, что при простом n число $2^n - 2$ делится на n, и вообще $a^n - a$ делится на n при любом целом a. (Это утверждение эквивалентно *малой теореме Ферма*, о которой пойдёт речь в главе о целых числах. Можно сказать, таким образом, что эта задача даёт комбинаторное доказательство этой теоремы.)

Задача 2.38. Сколько различных слов можно получить, переставляя буквы в слове КАНАТ?

Задача 2.39. Сколько различных слов можно получить, переставляя буквы в слове КОКОН?

Задача 2.40. Сколько различных слов можно получить, переставляя буквы в слове МАТЕМАТИКА?

2.7 Подмножества и числа сочетаний

Сейчас мы используем метод предыдущего раздела, чтобы получить формулу для числа k-элементных подмножеств n-элементного множества. Представим себе, что *в классе из n учеников нужно сформировать спортивную команду из k учеников* (внутри команды все равны, никаких ролей нет). *Сколькими способами это можно сделать*?

Один способ подсчёта такой. Будем выбирать игроков команды по очереди. Первого можно выбрать n способами, второго $n-1$ способами (годятся все, кроме первого), третьего $n-2$ способами, всего

$$(n)_k = n(n-1)(n-2)\cdot\ldots\cdot(n-k+1)$$

способами (см. выше). Но так мы подсчитали не число возможных команд, а число возможных упорядоченных списков команд (известно, какой игрок первый, какой второй, и так далее). Если не обращать внимание на номера игроков, то много разных списков соответствуют одной команде — они получатся, если игроков переставлять в списке, а это можно сделать $k!$ способами. Значит, число групп равно $(n)_k/k!$; поскольку $(n)_k = n!/(n-k)!$, то можно переписать ответ в более симметричной форме:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Это число (число способов выбрать k элементов из n, без учёта порядка, или число k-элементных подмножеств n-элементного множества, если говорить научно) традиционно называют *числом сочетаний из n по k* и обозначают $\binom{n}{k}$ (мы использовали это обозначение в левой части формулы) или C_n^k. Первый вариант сейчас используется всё чаще, но во многих русских (и французских) книгах используется обозначение C_n^k.

Интересным следствием этой формулы является такое наблюдение: произведение любых подряд идущих k натуральных чисел делится на $k!$. В самом деле, это произведение равно $(n)_k$, где n — последнее из этих чисел, и по формуле $(n)_k/k!$ должно получиться целое число.

При $n=2$ получается, что произведение любых двух соседних натуральных чисел чётно (это не удивительно, поскольку одно из них нечётно, а второе чётно). При $n=3$ получается, что произведение любых трёх соседних натуральных чисел делится на $3! = 6$. Это тоже не так удивительно: ровно одно из этих чисел делится на 3, поэтому произведение делится на 3, а также оно делится на 2 (для этого достаточно даже двух чисел), значит, оно делится на 6. Дальше объяснения становятся сложнее.

Задача 2.41. Чисто арифметически (не ссылаясь на число сочетаний) докажите, что произведение любых четырёх подряд идущих чисел делится на 24.

Другой способ подсчёта чисел сочетаний — использование рекуррентной формулы. Допустим, мы хотим найти $\binom{n}{k}$, количество способов выбрать команду из k человек, если в классе всего n человек. Выберем какого-нибудь школьника, пусть его зовут Петя, и все способы разделим на две группы: когда Петя входит в команду и когда Петя не входит в команду.

Если мы решили, что Петя не входит в команду, то наша задача состоит в том, чтобы выбрать k человек из оставшихся $n-1$ человек. Это можно сделать $\binom{n-1}{k}$ способами (в наших обозначениях). С другой стороны, если мы решили, что Петя входит в команду, то остаётся выбрать $k-1$ человек из $n-1$ человек (один из k уже есть). По правилу суммы получаем рекуррентное соотношение

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Правда, чтобы это рассуждение и формула были законными, нужно, чтобы $0 < k < n$, иначе в правой части будет бессмыслица (при $k < 0$ или $k > n$ непонятно, как понимать $\binom{n}{k}$).

Эту формулу полезно проиллюстрировать на картинке, расположив числа сочетаний в виде треугольника:

$$\begin{array}{ccccccccccc}
 & & & & \binom{0}{0} & & & & & & \\
 & & & \binom{1}{0} & & \binom{1}{1} & & & & & \\
 & & \binom{2}{0} & & \binom{2}{1} & & \binom{2}{2} & & & & \\
 & \binom{3}{0} & & \binom{3}{1} & & \binom{3}{2} & & \binom{3}{3} & & & \\
\binom{4}{0} & & \binom{4}{1} & & \binom{4}{2} & & \binom{4}{3} & & \binom{4}{4} & & \\
 & \binom{5}{1} & & \binom{5}{2} & & \binom{5}{3} & & \binom{5}{4} & & \binom{5}{5} &
\end{array}$$

Наша рекуррентная формула означает, что каждое число в этом треугольнике получается сложением двух чисел в предыдущей строке (сочетания из $n-1\ldots$): слева над ним ($\ldots$ по $k-1$) и справа над ним ($\ldots$ по k). Числа с краю таблицы равны единице: если никого выбирать в команду не нужно ($k=0$) или если нужно выбрать всех ($k=n$), какие уж тут варианты. По этим правилам таблицу легко заполнить:

$$\begin{array}{ccccccccccc}
 & & & & 1 & & & & & & \\
 & & & 1 & & 1 & & & & & \\
 & & 1 & & 2 & & 1 & & & & \\
 & 1 & & 3 & & 3 & & 1 & & & \\
1 & & 4 & & 6 & & 4 & & 1 & & \\
 & 5 & & 10 & & 10 & & 5 & & 1 &
\end{array}$$

Этот треугольник называют *треугольником Паскаля*, в честь Блеза Паскаля, французского математика и философа XVII века (хотя, как считают историки, Паскаль не был первым — этот треугольник упоминался и раньше[4]).

[4]В частности, говорят, что его знал Омар Хайам, которого также считают автором четверости-

Теперь, помимо формулы с факториалами, у нас есть и другой способ находить числа сочетаний. Для заполнения таблицы он явно лучше (на каждый элемент уходит одно сложение). Если же нам надо вычислить какое-то одно число в этой таблице, то по сравнению с формулой операций больше (пропорционально n^2, а не n), но каждая из них проще (сложение вместо умножений и делений).

Можно окружить этот треугольник слева и справа нулями, тогда рекуррентная формула будет верной и для крайних чисел (и вообще для всех чисел: добавленные нули тоже будут ей подчиняться).

Задача 2.42. Сколько есть двоичных слов длины n, содержащих k единиц?

Задача 2.43. Слово состоит из n букв А и k букв Б. Сколько различных слов можно составить, переставляя буквы в этом слове?

Задача 2.44. Сколько существует слов длины 10 в алфавите $\{А, Б, В\}$, содержащих ровно 4 буквы А? (Сначала надо выбрать, где будут буквы А, а потом посмотреть, сколькими способами можно выбрать остальные буквы.)

Задача 2.45. Сколько существует четырёхзначных чисел, в которых цифры идут в убывающем порядке?

2.8 Ещё о числах сочетаний

Числа сочетаний обладают разными интересными свойствами. Доказывая эти свойства, мы можем пользоваться и определением (как числа способов), и формулой с факториалами, и свойством треугольника Паскаля. Часто одно и то же свойство можно доказывать по-разному.

2.8.1 Симметрия

Треугольник Паскаля симметричен относительно вертикальной оси: если читать каждую строку справа налево, получится то же самое. Другими словами,

$$\binom{n}{k} = \binom{n}{n-k} \quad \text{при } k = 0, 1, 2, \ldots, n$$

Доказательство с факториалами. В самом деле,

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}.$$

ший-рубаи. Правда, в них вроде бы ничего о числах сочетаний не говорится — напротив, одно из них, в переводе Германа Плисецкого, звучит так: «*Я спросил у мудрейшего: что ты извлёк // Из своих манускриптов? Мудрейший изрёк:// Счастлив тот, кто в объятьях красавицы нежной // По ночам от премудрости книжной далёк*». Однако Омар Хайам — настоящий математик, более известный на Востоке как учёный, чем как поэт. Например, он графически решал кубические уравнения, пересекая параболу и гиперболу, — задолго до Декарта.

Если заменить в этой формуле k на $n-k$, то $(n-k)$ превратится в $n-(n-k)=k$, так что два сомножителя в знаменателе просто поменяются местами. Например,

$$\binom{20}{3} = \frac{20!}{3!17!}, \quad \binom{20}{17} = \frac{20!}{17!3!}.$$

Доказательство с треугольником. Проведём доказательство индукцией по количеству первых строк треугольника. Первые несколько строк симметричны. Достаточно доказать, таким образом, что симметрия сохраняется при переходе к следующей строке. Это следует из того, что и правило симметрично: оба крайних числа равны единице, а между ними каждое равно сумме чисел слева-над и справа-над. Если мы будем следовать этому правилу и выписывать треугольник на стеклянной доске, то стоящий сзади доски коллега с нами согласится, просто для него числа «слева-над» и «справа-над» поменяются местами.[5]

Комбинаторное доказательство. Что такое, скажем, $\binom{19}{9}$? Это количество способов, которым в классе из 19 человек можно выбрать команду из 9 человек. А $\binom{19}{10}$ — это число способов выбрать команду из 10 человек. Но и в том, и в другом случае мы просто делим школьников на две команды по 9 и 10 человек, так что речь идёт об одном и том же числе.

Более формально можно сказать так. Пусть есть n-элементное множество A. Для каждого его k-элементного подмножества B рассмотрим множество из остальных $n-k$ элементов, обозначаемое $A \setminus B$ (те элементы A, которые не входят в B). Таким образом возникает взаимно однозначное соответствие между k-элементными подмножествами A и $(n-k)$-элементными подмножествами A. Значит, тех и других одинаковое количество.

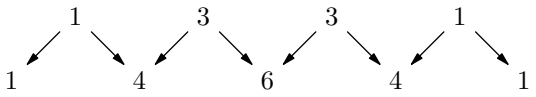
2.8.2 Сумма чисел в строке

Сумма всех чисел в n-й строке треугольника Паскаля равна 2^n:

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{n-1} + \binom{n}{n} = 2^n.$$

Например, $1+2+1=4=2^2$, $1+3+3+1=8=2^3$ и так далее.

Доказательство по индукции. Достаточно показать, что в каждой следующей строке сумма вдвое больше предыдущей. Раскроем каждое число как сумму двух чисел предыдущей строки. Скажем, для четвёртой строки:



Если сгруппировать числа по-другому, то видно, что каждое число предыдущей строки, включая крайние единицы, входит дважды. Поэтому сумма следующей строки будет вдвое больше, чем предыдущей.

[5] Не надо только понимать эту метафору слишком буквально: на самом деле с другой стороны цифры выглядят не так.

Комбинаторное доказательство. Число $\binom{n}{k}$ равно числу k-элементных подмножеств n-элементного множества. Когда мы складываем эти числа при всех k (и фиксированном n), мы получаем число всех подмножеств n-элементного множества, а оно равно 2^n, как мы знаем (для каждого из n элементов есть два варианта: включать его или нет).

Можно пересказать это в терминах двоичных слов. Скажем, при $n = 3$ мы считаем двоичные слова длины 3 по группам: сначала из одних нулей, потом с одной единицей, потом с двумя, и наконец, с тремя:

$$\underbrace{000}_{1} \quad \underbrace{001\ 010\ 100}_{3} \quad \underbrace{110\ 101\ 011}_{3} \quad \underbrace{111}_{1}$$

Ещё одно доказательство мы легко получим в следующем разделе по формуле бинома Ньютона.

2.8.3 Знакочередующаяся сумма

Если складывать числа в строке треугольника Паскаля с чередующимися знаками, то получится нуль:

$$\begin{aligned} 1 - 1 &= 0 \\ 1 - 2 + 1 &= 0 \\ 1 - 3 + 3 - 1 &= 0 \\ 1 - 4 + 6 - 4 + 1 &= 0 \\ 1 - 5 + 10 - 10 + 5 - 1 &= 0 \\ &\dots \end{aligned}$$

или, в общем виде,

$$\sum_{k=0}^{n} (-1)^k \binom{n}{k} = 0 \quad \text{при } n > 0$$

(множитель $(-1)^k$ как раз и обеспечивает знакочередование: он равен то $+1$, то -1).

Почему это так? Если приглядеться, то в нечётных строках всё понятно: равные числа сокращаются, и ничего не остаётся. Но и в чётных строках, уже более загадочным образом, получается нуль. Почему?

Посмотрим, скажем, как из третьей строки получается четвёртая. При этом нам даже неважно, какие числа стоят в третьей строке, так что заменим их буквами:

$$\begin{array}{ccccccc} & a & & b & & c & & d \\ a & & a+b & & b+c & & c+d & & d \end{array}$$

Теперь в знакопеременной сумме четвёртой строки всё сокращается:

$$a - (a + b) + (b + c) - (c + d) + d = 0.$$

Так и в общем случае: каждое число в n-й строке войдёт в два соседних числа в следующей, $(n + 1)$-й строке, и эти соседние числа будут с разными знаками, так что всё сократится.

Контрольный вопрос 2.46. Где в этом рассуждении использовано, что $n > 0$? (При $n = 0$ получается ложное равенство $1 = \binom{0}{0} = 0$.)

Можно доказать равенство нулю знакопеременной суммы и комбинаторно. Перенесём члены с минусами в другую часть. Нам надо доказать теперь, что

$$\binom{n}{1} + \binom{n}{3} + \binom{n}{5} + \ldots = \binom{n}{0} + \binom{n}{2} + \binom{n}{4} + \ldots$$

Другими словами, надо доказать, что у n-элементного множества столько же подмножеств нечётного размера (левая часть), что и чётного (правая). Почему это так? Выберем в нашем n-элементном множестве какой-то элемент, и поделим все подмножества на пары, отличающиеся только добавлением-удалением этого элемента. (Если подмножества представлять как битовые слова, то мы выбираем какую-то позицию и спариваем две строки, различающиеся в этой позиции.) Подмножества, образующие пару, отличаются по количеству элементов на один (тот самый выбранный), поэтому в одном из них чётное число элементов, а в другом нечётное число элементов. Дальше всё понятно (если в мешке белые и чёрные зёрна слиплись в бело-чёрные пары, то белых и чёрных зёрен поровну).

2.8.4 Снова о включениях и исключениях

Общая формула включений и исключений:

$$\begin{aligned}|A_1 \cup A_2 \cup \ldots \cup A_n| = |A_1| + \ldots + |A_n| - \\ - |A_1 \cap A_2| - |A_1 \cap A_3| - \ldots \\ + |A_1 \cap A_2 \cap A_3| + |A_1 \cap A_2 \cap A_4| + \ldots \\ \ldots \\ - (-1)^n |A_1 \cap A_2 \cap \ldots \cap A_n|\end{aligned}$$

Чтобы найти общее количество элементов в n пересекающихся множествах, мы складываем размеры всех этих множеств, затем вычитаем размеры попарных пересечений, потом прибавлям размеры тройных, и так далее с чередованием знаков до пересечения всех множеств. Знак перед последним членом зависит от чётности n, отсюда и множитель $(-1)^n$.

Нам надо проверить (как мы делали при $n = 2$ и $n = 3$), что каждый потенциальный кандидат учтён ровно один раз (после всех сокращений). Пусть кандидат входит в k из n множеств. Тогда он войдёт k раз, когда мы будем складывать размеры множеств. Потом мы его вычтем столько раз, во сколько пар множеств он входит. Другими словами, мы разными способами выбираем из содержащих его k множеств пару. Таких способов $\binom{k}{2}$. Затем мы учитываем кандидата по разу для каждой тройки содержащих его множеств, таких троек $\binom{k}{3}$. В итоге он будет учтён

$$\binom{k}{1} - \binom{k}{2} + \binom{k}{3} - \binom{k}{4} + \ldots$$

Это и есть знакопеременная сумма, которая равна нулю, только без первого члена и с обратным знаком, поэтому эта сумма равна единице, что нам и требовалось.

Например, если выбранный элемент входит в 4 множества, то сначала он будет посчитан 4 раза с каждым из них, потом вычтен 6 раз с каждой парой (таких пар $\binom{4}{2} = 6$), потом добавлен 4 раза обратно с каждой тройкой, и наконец вычтен при вычитании пересечения четырёх множеств. Всего получится

$$4 - 6 + 4 - 1 = 1$$

(если перенести левую часть в правую, это и будет нулевая знакопеременная сумма четвёртой строки треугольника Паскаля).

Задача 2.47. Имеется слово из n различных букв. Сколько слов можно получить перестановками этих букв, если требуется, чтобы никакая буква не осталась на своём месте? (Примените формулу включений и исключений, взяв в качестве A_i множество слов, где i-я буква остаётся на своём месте).

Если вы знакомы с математическим анализом, покажите, что доля таких слов (среди всех $n!$ слов) близка к $1/e$, где $e = 2{,}71828\ldots$ — основание натуральных логарифмов.

2.8.5 Пути, подмножества, слова

Мы рассматривали задачу, в которой надо было попасть в данную клетку шахматной доски, начав с нижней левой клетки и делая шаги вверх и вправо. Для числа способов (в том варианте, когда по диагонали ходить нельзя) получалась та же рекуррентная формула, что и для треугольника Паскаля (только повёрнутая). Отсюда уже следует, что число способов в этой задаче равно соответствующему числу сочетаний.

Но как это объяснить комбинаторно, не используя рекуррентного соотношения и рассуждения по индукции? Это тоже несложно. Будем записывать последовательность ходов как слово в алфавите {П, В}: ход **в**право буквой П, а ход на**в**ерх буквой В. Скажем, чтобы попасть из $a1$ в $e4$, надо сделать четыре хода вправо и три хода вверх. Это соответствие взаимно однозначно: любое слово из четырёх букв П и трёх букв В описывает движение из $a1$ в $e4$, надо просто следовать инструкциям, закодированным буквами.

Следовательно,

маршрутов из $a1$ в $e4$ столько же, сколько слов из четырёх букв П и трёх букв В.

А сколько таких слов? Можно сказать, что у нас есть слово длины 7, содержащее четыре буквы П (и остальные В). Чтобы задать такое слово, надо указать, в каких четырёх (из семи) позициях стоят буквы П. Другими словами, надо указать четырёхэлементное подмножество семиэлементного множества. А это можно сделать $\binom{7}{4}$ способами. Значит, и маршрутов из $a1$ в $e4$ будет столько же. То же самое, конечно, будет и для любой другой клетки доски: число маршрутов в эту клетку равно числу сочетаний $\binom{n+m}{n}$, где n и m — числа ходов вправо и вверх.

В нашем рассуждении есть некоторая несимметрия: почему мы говорили о позициях букв П, а не букв В? Можно было бы и наоборот, тогда получались бы трёхэлементные подмножества семиэлементного множества, но их столько же (как раз об этом соответствии между множеством и его дополнением мы говорили выше).

Задача 2.48. Сколько различных слов можно составить из 10 букв А и 15 букв Б, если требуется, чтобы буквы А не шли подряд (а всегда были бы разделены хотя бы одной буквой Б)? (Ответ: $\binom{16}{10}$.)

Зная, что число сочетаний равно количеству слов с данным числом букв В и П, можно обобщить задачу и спросить: пусть у нас есть три буквы, а не две, сколько тогда будет слов с данными количествами букв? Сколько, скажем, есть *десятизначных чисел, в которых две тройки, три пятёрки и пять двоек*? (Здесь вместо букв у нас цифры, но какая разница.) Другая переформулировка того же вопроса: сколько различных слов можно получить, переставляя буквы в слове ААББББВВВВВ?

Для этого количества тоже есть обозначение, аналогичное числам сочетаний: $\binom{10}{2,3,5}$. Сверху стоит общее число букв, а снизу через запятую идут количества букв всех трёх сортов. (Сумма чисел снизу должна быть равна числу сверху, иначе это обозначение не имеет смысла.) Оказывается, что для этого числа тоже есть формула с факториалами:

$$\binom{10}{2,3,5} = \frac{10!}{2!3!5!}$$

(проверьте, что при двух числах в нижней строке эта новая формула даёт то же, что и раньше). Как это доказать?

Временно снабдим буквы в слове ААББББВВВВВ нижними индексами, чтобы их можно было различать, получится слово $А_1А_2Б_1Б_2Б_3В_1В_2В_3В_4В_5$ В нём уже все десять букв различны, поэтому можно выписать 10! перестановок этих букв. Все эти перестановки будут разными, если учитывать индексы у букв, но после стирания индексов получатся группы из одинаковых слов (которые раньше отличались только индексами).

Сколько слов в такой группе? Они все получаются, если по-разному расставлять индексы. Сколькими способами это можно сделать? В слове две буквы А, и нужно одну из них назвать $А_1$, а другую $А_2$, это можно сделать двумя способами. Букв Б у нас три, и есть $6 = 3!$ способов снабдить их индексами. Для букв В получается 5! вариантов. По правилу произведения количество способов расставить индексы в данном слове равно 2!3!5!, то есть в каждой группе по 2!3!5! слов с индексами. Соответственно, число групп равно 10!/(2!3!5!), как мы и говорили.

Ясно, что конкретный выбор чисел $10, 2, 3, 5$ не играет роли, аналогичная формула верна для других чисел и для другого количества букв:

$$\binom{m+n+\ldots+k}{m,n,\ldots,k} = \frac{(n+m+\ldots+k)!}{m!n!\cdot\ldots\cdot k!}.$$

Задача 2.49. Имеется колода из 36 различных карт. Эти карты раздают игрокам, по 9 карт каждому. Сколькими способами это можно сделать? (Способы считаются

одинаковыми, если в них любой из четверых игроков получил одни и те же карты — но, возможно, в разном порядке.)

2.8.6 Соседние числа в строке

Глядя на треугольник Паскаля, мы видим, что числа сначала растут (до середины), а потом начинают симметрично убывать — график имеет горб в середине.[6] Как это доказать?

Для этого вычислим отношение соседних чисел в строке. Это можно сделать, исходя из формулы

$$\binom{n}{k} = \frac{n(n-1)\cdot\ldots\cdot(n-k+1)}{1\cdot 2\cdot\ldots\cdot k}.$$

Что происходит с этой дробью, когда k увеличивается на 1? При этом в числителе появляется новый множитель $(n-k)$, а в знаменателе новый множитель $k+1$, так что

$$\binom{n}{k+1} \Big/ \binom{n}{k} = \frac{n-k}{k+1}$$

Это частное больше 1, если $n-k > k+1$, или $k+(k+1) < n$, то есть если середина между k и $k+1$ левее середины строки $n/2$. Так что при чётном n есть максимальное среднее число $\binom{n}{n/2}$, а при нечётном n есть два максимальных числа $\binom{n}{(n-1)/2}$ и $\binom{n}{(n+1)/2}$.

Интересно, что отношение соседних чисел можно вычислить, и не применяя формулы для чисел сочетаний. Мы хотим сравнить число k-элементных и $(k+1)$-элементных подмножеств n-элементного множества. Нарисуем картинку (двудольный граф): слева изобразим точками все k-элементные подмножества, а справа все $(k+1)$-элементные подмножества. Нас интересует соотношение количества точек слева и справа.

Чтобы найти его, соединим линиями точку X слева и точку Y справа, если $X \subset Y$, то есть Y получается из X добавлением одного элемента (ну а X получается из Y удалением одного элемента). Посмотрим, сколько линий выходит из одной точки слева и справа. К данному множеству из k элементов можно присоединить любую из $n-k$ оставшихся точек, поэтому слева из каждой точки выходит $n-k$ линий. Из данного множества с $k+1$ элементами можно удалить любой из них, поэтому справа из каждой точки выходит $k+1$ линий. Это определяет соотношение между левой и правой частями: если каждый мальчик в классе поздравил двух девочек, а каждая девочка получила поздравление от трёх мальчиков, значит, в классе в полтора раза больше мальчиков, чем девочек. Чтобы это увидеть, достаточно подсчитать по отдельности число левых концов линий и правых концов линий (даже если два левых или два правых конца совпадают как точки на плоскости, мы их

[6] Этот горб называют биномиальным распределением, он при разных n имеет примерно один и тот же вид и близок к масштабированному графику $y = e^{-x^2}$, называемому *нормальным распределением*. Факт этой близости составляет простейший вариант *центральной предельной теоремы* из теории вероятностей.

считаем разными *концами линий*). Эти числа равны, отсюда получаем пропорцию между количествами точек.

Рассматривая этот граф и применяя теорему Холла о представителях, можно вывести интересное следствие:

Задача 2.50. Покажите, что при $k < n/2$ можно к каждому k-элементному подмножеству n-элементного множества так добавить по элементу, что все получившиеся $(k+1)$-элементные множества будут разные. При $k > n/2$ из каждого k-элементного множества можно так выбросить по элементу, что все полученные $(k-1)$-элементные множества будут разные.

Задача 2.51. (Продолжение) Покажите, что все подмножества $2n$-элементного множества можно разбить на $\binom{2n}{n}$ цепочек так, что в каждой цепочке соседние множества отличаются добавлением/удалением элемента.

Задача 2.52. (Продолжение) Покажите, что если в некотором семействе подмножеств множества из $2n$ элементов любые два элемента несравнимы по включению, то есть ни один не является подмножеством другого, то в этом семействе не больше $\binom{2n}{n}$ множеств.

2.8.7 Монеты и перегородки

Есть ещё одна задача, где естественным (но не вполне очевидным) образом возникают числа сочетаний. *Сколько решений имеет уравнение* $x_1 + x_2 + \ldots + x_k = n$ *в целых неотрицательных числах*? Если это недостаточно наглядно, можно спросить иначе: есть k различных человек и n одинаковых монет; сколькими способами можно раздать этим людям эти монеты? Каждый такой способ определяется тем, сколько монет получил каждый (но какие именно монеты — не учитывается, все монеты одинаковые). Это целые неотрицательные числа (допускаются варианты, где некоторые получили по нулю монет).

Задача 2.53. А если бы все монеты были разными и мы учитывали не только, сколько монет досталось, но и то, какие достались — тогда сколько было бы способов? (Ответ: n^k.)

Математики бы сказали, что мы ищем число целых точек в симплексе $x_1 + x_2 + \ldots + x_k = n$, $x_i \geqslant 0$. При $k = 3$ это получается треугольник (и такие числа называются «треугольными»), при $k = 4$ это тетраэдр и так далее.

Как найти это число? Представим себе, что наши n монет разложены в ряд. Прежде чем раздавать эти монеты, давайте разделим их на k групп перегородками, и договоримся, кому идёт самая левая группа, кому вторая слева и т. д. Заметим, что мы допускаем случай, когда две перегородки оказываются рядом — это значит просто, что человеку, которому была назначена группа между ними, не повезло и в этом раскладе ему ни одной монеты не достанется.

$$\begin{array}{ccccccccccccc} x_1 & + & x_2 & + & x_3 & + & x_4 & + & x_5 & + & x_6 & + & x_7 & = & 7 \\ 0 & & 2 & & 2 & & 0 & & 3 & & 0 & & 0 & & \end{array}$$

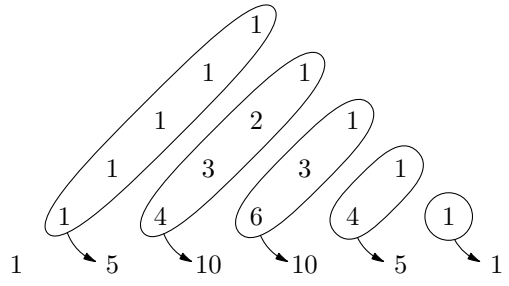
Каждому варианту раздачи (каждому решению уравнения в неотрицательных числах) соответствует последовательность из n монет и $k-1$ перегородок. (Число перегородок на единицу меньше числа группы: первая группа стоит слева от первой перегородки, а последняя — справа от последней.) Наоборот, каждой последовательности из n монет и $k-1$ перегородок соответствует некоторый способ раздачи монет. Поэтому надо подсчитать число способов расставить перегородки.

А это совсем просто — каждый такой способ можно рассматривать как слово в алфавите $\{\text{монета}, \text{перегородка}\}$, содержащее n монет и $k-1$ перегородок. Это количество, как мы знаем, равно $\binom{n+k-1}{n}$ (или $\binom{n+k-1}{k-1}$).

Мы искали решения уравнения в неотрицательных целых числах. А что будет, если искать лишь *решения уравнения в положительных целых числах*, то есть рассматривать условия $x_1 + \ldots + x_k = n$, $x_i > 0$? Легко сообразить, что получится примерно та же самая задача, только симплекс немного уменьшится. В терминах монет мы теперь требуем, чтобы каждый человек получил хотя бы одну монету — так выдадим каждому по монете заранее, до основной раздачи. Тут никакого выбора нет, а после этого остаётся раздать $n-k$ монет по-старому. Получаем ответ $\binom{n-1}{n-k}$ или $\binom{n-1}{k-1}$.

В терминах уравнений: мы вводим новые переменные $y_i = x_i - 1$. Тогда $y_1 + \ldots + y_k = n - k$ и $y_i \geqslant 0$, так что задача сводится к предыдущей с заменой n на $n-k$.

Задача 2.54. Для числа решений уравнения $x_1 + \ldots + x_k = n$ в неотрицательных целых числах можно написать рекуррентную формулу: последняя переменная может быть равна $0, 1, 2, \ldots, n$, поэтому искомое число равно сумме чисел решений такого же уравнения с $k-1$ переменными и числами $n, n-1, n-2, \ldots, 0$ в правой части. Покажите, что в терминах чисел сочетаний получается такое тождество: сумма чисел на каждой диагонали (см. рисунок) равна числу, стоящему снизу справа под ней. Как объяснить это тождество непосредственно, не вспоминая про уравнения?



2.9 Бином Ньютона и производящие функции

Одно из открытий в перечислительной комбинаторике состоит в том, что «считать нужно не числами, а многочленами» (быть может, многочленами бесконечной степени — степенными рядами). Это значит, что из последовательности чисел $a_0, a_1, a_2 \ldots$ нужно собрать один математический объект — многочлен $a_0 + a_1 x + a_2 x^2 + \ldots$ и работать с ним.

Это звучит заманчиво, но едва ли понятно, и мы начнём издалека — с бинома Ньютона, названного в честь того самого Ньютона, который изобрёл параллельно с Лейбницем математический анализ, открыл законы механики и обосновал закон всемирного тяготения, изучал преломление света в призме, наблюдал кольца Ньютона при контакте выпуклого стекла с плоским, занимался алхимией, выслеживал фальшивомонетчиков, будучи начальником монетного двора, и многое другое, вот только «бином Ньютона» был известен до него (в частности, Паскалю), а Ньютон обобщил его на случай нецелых показателей и бесконечных рядов.

Хорошо известны формулы раскрытия скобок

$$(a+b)^2 = a^2 + 2ab + b^2$$
$$(a+b)^3 = a^3 + 3a^2 b + 3ab^2 + b^3$$

(а если и неизвестны, их легко проверить). Оказывается, что есть *формула бинома* для любой степени:

$$(a+b)^n = \binom{n}{0} a^n + \binom{n}{1} a^{n-1} b + \ldots + \binom{n}{k} a^{n-k} b^k + \ldots + \binom{n}{n} b^n$$

Или, с употреблением знака суммы:

$$(a+b)^n = \sum_k \binom{n}{k} a^{n-k} b^k = \sum_k \frac{n!}{k!(n-k)!} a^{n-k} b^k.$$

Собственно говоря, сразу ясно, что после раскрытия скобок в $(a+b)(a+b)\ldots(a+b)$ (n раз) получатся произведения n букв (сколько-то a, остальные b), и вопрос только в том, какие будут коэффициенты при этих произведениях (сколько подобных членов). Так вот, формула бинома Ньютона и говорит, какие это будут коэффициенты: это числа сочетаний, и они написаны в n-й строке треугольника Паскаля. Поэтому числа сочетаний также называют *биномиальными коэффициентами*.

Это можно доказать разными способами. Если мы раскроем скобки в $(a+b)^n$ и не будем приводить подобные члены, переставляя сомножители, то получится 2^n слагаемых — всевозможные слова длины n из букв a и b. (В каждой скобке можно взять a или b = на каждом месте в слове может стоять a или b.) Если сгруппировать члены по степеням, то соберутся все слова, в которых данное число букв a и данное число букв b. А сколько их, мы знаем — это как раз числа сочетаний. Слова, в которых $n-k$ букв a и k букв b, войдут в количестве $\binom{n}{k}$ штук в слагаемое $\binom{n}{k} a^{n-k} b^k$, как и предсказывает бином Ньютона.

Другое доказательство — по индукции. Допустим, у нас уже есть формула

$$(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3,$$

и мы хотим из неё получить формулу для $(a+b)^4$. Для этого надо домножить правую часть на $a+b$:

$$\begin{aligned}(a+b)^4 = (a+b)^3(a+b) &= (a^3 + 3a^2b + 3ab^2 + b^3)(a+b) = \\ &= (a^3 + 3a^2b + 3ab^2 + b^3)a + \\ &\qquad + (a^3 + 3a^2b + 3ab^2 + b^3)b = \\ &= a^4 + 3a^3b + 3a^2b^2 + ab^3 + \\ &\qquad + a^3b + 3a^2b^2 + 3ab^3 + b^4 = \\ &= a^4 + 4a^3b + 6a^2b^2 + 3ab^3 + b^4.\end{aligned}$$

Видно, что коэффициент в новой формуле получается как раз из двух соседних коэффициентов старой ($3+1=4$, $3+3=6$ и так далее), как в треугольнике Паскаля, и из третьей строки треугольника получается четвёртая. Ясно, что так же будет и дальше.

Что даёт нам эта формула? Сразу же можно получить несколько забавных следствий. Давайте подставим $a=b=1$. Тогда слева получится 2^n, а справа — сумма всех биномиальных коэффициентов, то есть всех чисел в n-ой строке треугольника Паскаля. Это равенство мы уже доказывали (двумя способами — комбинаторно и по индукции), а теперь мы его вывели как простое следствие формулы бинома.

Можно подставить $a=1$, $b=-1$. Тогда слева получится нуль, а справа получится знакопеременная сумма биномиальных коэффициентов (с плюсом будут те, при которых число b возводится в чётную степень, то есть каждый второй). И это мы уже доказывали.

Чтобы получить что-то новое, подставим $a=1, b=2$. Получится, что

$$3^n = \binom{n}{0} + \binom{n}{1}2 + \binom{n}{2}2^2 + \binom{n}{3}2^3 + \ldots \binom{n}{n}2^n.$$

Вроде такого у нас не было — можно радоваться, что бином Ньютона мы доказали не зря. Впрочем, скептики скажут, что это можно доказать и комбинаторно. Давайте подсчитаем, сколько можно составить n-буквенных слов в алфавите $\{a,b,c\}$. Их, конечно, 3^n, но будем считать по-другому, отдельно рассматривая слова с разным количеством букв a. Тех, где n букв a, всего одно. Освободим одно место для букв b/c. Это можно сделать $\binom{n}{1}$ способами, а после этого есть два варианта вписать на это место b или c. Если для букв b/c оставить два места, то это можно сделать $\binom{n}{2}$ способами, а потом на эти места вписать буквы $2^2=4$ способами. И так далее — как раз в соответствии с правой частью формулы.

Можно использовать бином Ньютона и более изысканным образом. Для начала оставим только одну переменную, положив $a=1$, а вместо b будем писать x, так

привычнее. Получится такая формула:

$$(1+x)^n = \sum_k \binom{n}{k} x^k = 1 + nx + \frac{n(n-1)}{2}x^2 + \frac{n(n-1)(n-2)}{1\cdot 2\cdot 3}x^3 + \ldots$$

Имея эту формулу, мы можем с ней делать разные странные вещи. Например, можно её продифференцировать (почему бы и нет? это просто равенство двух многочленов). Получится

$$n(1+x)^{n-1} = \sum_k k\binom{n}{k} x^{k-1}.$$

А вот теперь подставим единицу, получится ещё одна формула, которой у нас не было:

$$\sum_k k\binom{n}{k} = n2^{n-1}.$$

Задача 2.55. Дайте комбинаторное доказательство этого тождества, начав так: пусть мы хотим выбрать в классе из n человек команду из k человек с капитаном. Мы можем сначала выбрать капитана, а потом решать, кого ещё мы включим в команду, а кого нет, а можем сначала выбрать команду...

Вот ещё одно тождество с числами сочетаний, которое сразу следует из бинома Ньютона. Начнём с очевидного равенства

$$(1+x)^k(1+x)^l = (1+x)^{k+l},$$

в котором левая и правая части — равные многочлены. Значит, как известно из алгебры, у них равные коэффициенты. Коэффициент при x^n равен $\binom{k+l}{n}$, согласно биному Ньютона. А в левой части два многочлена перемножаются, и потому коэффициент при x^n складывается из нескольких произведений (когда степени в сумме дают n). Должно получиться одно и то же, и поэтому мы заключаем, что

$$\binom{k+l}{n} = \binom{k}{0}\binom{l}{n} + \binom{k}{1}\binom{l}{n-1} + \binom{k}{2}\binom{l}{n-2} + \ldots + \binom{k}{n}\binom{l}{0}.$$

Впрочем, и комбинаторно это совсем просто. Пусть множество состоит из двух частей: в одной k элементов, в другой l. Чтобы выбрать n элементов из этого множества, нужно выбрать сколько-то, пусть i, элементов из первой части и $n-i$ элементов из второй. При данном i это можно сделать $\binom{k}{i}\binom{l}{n-i}$ способами, остаётся сложить ответы для разных i и получить то самое тождество.

Ещё одно применение бинома Ньютона: убедимся, что *при больши́х n величина* 1.01^n *больше, чем* n. В самом деле, разложим $(1+0.01)^n$ по формуле бинома, там все члены положительны, так что можно оставить только второй член:

$$(1+0.01)^n > \frac{n(n-1)}{2}\cdot 0.0001$$

При больши́х n (точно: когда $(n-1)/2\cdot 0.0001 > 1$) будет $1.01^n > n$.

Задача 2.56. Докажите, что $1.01^n > n^2$ при всех достаточно больши́х n.

До сих пор мы имели дело с конечными суммами (многочленами). Но и для бесконечной последовательности $a_0, a_1, a_2, a_3, \ldots$ можно написать выражение

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots,$$

которое называется её *производящей функцией*. Это выражение можно рассматривать по-разному, или просто как «формальный ряд», с которым можно обращаться по формальным правилам, или в самом деле как функцию действительного или комплексного переменного (и тогда, если ряд бесконечный, возникает вопрос о сходимости). На первый взгляд это странно (какая разница, говорить ли просто о последовательности чисел, или о последовательности коэффициентов многочлена или ряда), но дело в том, что алгебраические операции приобретают комбинаторный смысл или наоборот.

Например, определение чисел Фибоначчи ($F_0 = 1$, $F_1 = 1$, $F_n = F_{n-1} + F_{n-2}$ при $n \geqslant 2$) в переводе на язык рядов говорит, что ряд

$$\varphi(x) = F_0 + F_1 x + F_2 x^2 + F_3 x^3 + \ldots = 1 + x + 2x^2 + 3x^3 + 5x^4 + 8x^5 + \ldots$$

удовлетворяет уравнению

$$\varphi(x) = x\varphi(x) + x^2\varphi(x) + 1$$

(при складывании последовательности коэффициентов, сдвинутой вправо на 1 и на 2, получается исходная последовательность, кроме первого члена). Действуя по формальным алгебраическим правилам, можно заключить, что

$$\varphi(x) = \frac{1}{1 - x - x^2},$$

затем разложить знаменатель на множители и представить дробь в виде суммы, заметив, что

$$\frac{1}{(x-a)(x-b)} = \frac{1}{a-b}\left(\frac{1}{x-a} - \frac{1}{x-b}\right),$$

а потом снова перейти к рядам, используя формулу для суммы бесконечной геометрической прогрессии

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots.$$

Это даст формулу для чисел Фибоначчи (с. 50) — и все эти действия можно надлежащим образом узаконить.

Ещё несколько примеров: в произведении

$$(1 + x + x^2 + x^3 + x^4 + \ldots)(1 + x^2 + x^4 + x^6 + x^8 + \ldots)(1 + x^5 + x^{10} + x^{15} + \ldots)$$

коэффициент при x^n равен числу способов заплатить n рублей, имея монеты в 1, 2 и 5 рублей: он складывается из произведений x^k из первой скобки, x^{2l} из второй

скобки и x^{5m} из третьей, и каждое такое произведение соответствует варианту с k рублёвыми, l двухрублёвыми и m пятирублёвыми монетами. А если мы перемножим

$$(1+x)(1+x^2)(1+x^3)(1+x^4)(1+x^5)\dots,$$

то это будет соответствовать ситуации, когда у нас есть монеты всех достоинств (рубль, два, три,...), но по одной штуке, то есть коэффициент при x^n будет равен *количеству разбиений числа n в сумму различных слагаемых*. Теперь в этом произведении можно сгруппировать члены: сначала выделить степени двойки

$$(1+x)(1+x^2)(1+x^4)(1+x^8)\dots,$$

затем степени вида $3\cdot 2^k$, то есть

$$(1+x^3)(1+x^6)(1+x^{12})(1+x^{24})\dots,$$

затем степени вида $5\cdot 2^k$, то есть

$$(1+x^5)(1+x^{10})(1+x^{20})(1+x^{40})\dots,$$

и так далее для всех нечётных чисел (ведь каждое целое положительное число однозначно представляется в виде $u2^v$ с нечётным u). Если раскрыть скобки в первом произведении, то получится просто

$$1+x+x^2+x^3+x^4+x^5+\dots,$$

поскольку каждое число однозначно представляется в виде суммы степеней двойки (двоичная система). Во втором произведении то же самое, только x надо заменить на x^3, получится

$$1+x^3+x^6+x^9+x^{12}+\dots;$$

третье даст

$$1+x^5+x^{10}+x^{15}+x^{20}+\dots,$$

а вместе получится произведение

$$(1+x+x^2+x^3+x^4+x^5+\dots)(1+x^3+x^6+x^9+x^{12}+\dots)(1+x^5+x^{10}+x^{15}+x^{20}+\dots)\cdot\dots,$$

в котором, если приглядеться, коэффициент при x^n равен *количеству разбиений n в сумму нечётных слагаемых* (не обязательно различных). Таким образом, наши действия (если считать их законными) доказывают, что *количество разбиений числа n в сумму нечётных слагаемых равно количеству его разбиений в сумму различных слагаемых*. Попробуем, скажем, число 10. Вот его разбиения в сумму нечётных слагаемых:

$$\begin{aligned}
&9+1,\\
&7+3, 7+1+1+1,\\
&5+5, 5+3+1+1, 5+1+1+1+1+1,\\
&3+3+3+1, 3+3+1+1+1+1, 3+1+1+1+1+1+1+1,\\
&1+1+1+1+1+1+1+1+1+1.
\end{aligned}$$

(в каждом разбиении слагаемые мы записываем от больших к меньшим). А вот разбиения в сумму различных слагаемых:

$$\begin{aligned}
&10,\\
&9+1,\\
&8+2,\\
&7+3, 7+2+1,\\
&6+4, 6+3+1,\\
&5+4+1, 5+3+2,\\
&4+3+2+1.
\end{aligned}$$

Видно, что разбиения совершенно разные, но чудесным образом их количество оказывается одинаковым — и это чудо объясняется волшебными манипуляциями с производящими функциями (которые вполне законны, если разобраться). В данном случае механизм волшебства не так сложен, и это рассуждение можно проследить шаг за шагом и превратить в комбинаторное доказательство.

Задача 2.57. Сделайте это и укажите взаимно однозначное соответствие между разбиениями n в сумму нечётных слагаемых и в сумму различных слагаемых. (Ответ: в разбиении на различные слагаемые нужно каждое чётное число заменять на две его половины, пока не останутся одни нечётные. В другую сторону нужно заменять два одинаковых слагаемых на их сумму.)

Вернёмся к Ньютону: он понял, что формула бинома верна для дробных показателей степени, но с двумя оговорками. Во-первых, для дробных показателей степени числа сочетаний не имеют комбинаторного смысла, и надо просто пользоваться формулой в том виде, в котором она имеет смысл:

$$(1+x)^\alpha = 1 + \alpha x + \frac{\alpha(\alpha-1)}{1\cdot 2}x^2 + \frac{\alpha(\alpha-1)(\alpha-2)}{1\cdot 2\cdot 3}x^3 + \ldots$$

(вместо n мы написали α в знак того, что теперь n не обязательно целое). Во-вторых, при нецелом α получается бесконечно много членов (потому что множители в числителе не обращаются в нуль, как раньше), и не очень понятно, что эта бесконечная сумма означает.

Здесь мы уже переходим в область математического анализа, но скажем коротко о некоторых вариантах такого понимания. Во-первых, при $|x|<1$ можно понимать сумму как предел (берём всё больше и больше членов, и смотрим, к чему это сходится), и этот предел будет правильным. Во вторых, можно написать приближённую формулу с конечным числом членов, например,

$$(1+x)^\alpha \approx 1 + \alpha x.$$

Про эту формулу можно доказать, что при малых x ошибка (разница между левой и правой частью) мала, и не просто мала, а мала даже по сравнению с x. В

учебниках математического анализа пишут $(1+x)^\alpha = 1 + \alpha x + o(x)$. Если взять следующий член, то ошибка будет $o(x^2)$. Для доказательства нужно использовать формулу Тейлора (с остаточным членом в форме Пеано); другой вариант формулы Тейлора (с остаточным членом в форме Лагранжа) позволяет доказать формулу бинома с рядом.

Задача 2.58. Проверьте, что для $\alpha = 1/2$ формула бинома Ньютона даёт

$$\sqrt{1+x} = 1 + \frac{x}{2} - \frac{x^2}{8} + \frac{x^3}{16} + \ldots + \frac{(-1)^n(2n)!}{(1-2n)(n!)^2 4^n} x^n + \ldots$$

(Указание: произведение нечётных чисел $1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-1)$, иногда обозначаемое $(2n-1)!!$, можно вычислить домножив его на $2 \cdot 4 \cdot 6 \cdot \ldots \cdot (2n) = 2^n n!$.)

В частности, получается хорошо известная приближённая формула $\sqrt{1+x} \approx 1 + \frac{x}{2}$ (попробуйте вычислить корень из 1.000001 на калькуляторе).

Вот ещё одно применение биномиальных коэффициентов с нецелыми числами. В формуле бинома Ньютона для произвольного α мы рассматривали выражение $\binom{\alpha}{k}$ при произвольном α, определив его как

$$\frac{\alpha(\alpha-1)\cdot\ldots\cdot\ldots\cdot(\alpha-k+1)}{1\cdot 2\cdot\ldots\cdot k}.$$

Заметим, что при фиксированном k это выражение (как функция α) является многочленом. Обозначим его через $P_k(\alpha)$. Степень этого многочлена равна k, и он принимает целые значения в целых точках (в положительных точках это числа сочетаний $\binom{\alpha}{k}$, в отрицательных — тоже числа сочетаний, но другие и со знаком: $(-1)^k\binom{-\alpha+k-1}{k}$).

Складывая многочлены P_k при различных k с разными целыми коэффициентами, мы получаем многочлен, который в целых точках принимает целые значения. Оказывается, что верно и обратное: *любой многочлен от одной переменной, который во всех целых точках принимает целые значения, можно представить как сумму многочленов P_k с целыми коэффициентами.*

Чтобы доказать этого, введём такое определение: для любого многочлена $P(x)$ рассмотрим его «дискретную производную» $Q(x) = P(x+1) - P(x)$. (Если рассматривать только значения в целых точках, то дискретная производная — это последовательность разностей между соседними членами.) Например, для многочлена x^2 дискретная производная равна $2x+1$ (разности между точными квадратами образуют арифметическую прогрессию), а для многочлена x^3 равна $3x^2+3x+1$ (разложение по биному без первого члена, аналогично и для x^n). Степень дискретной производной на единицу меньше степени многочлена (почему?). Если дискретная производная равна нулю, то многочлен — константа.

Ключевое наблюдение: *дискретная производная многочлена P_{n+1} равна P_n*, то есть

$$P_n(\alpha) = P_{n+1}(\alpha+1) - P_{n+1}(\alpha).$$

В самом деле, при целых α это превращается в

$$\binom{\alpha}{n} = \binom{\alpha+1}{n+1} - \binom{\alpha}{n+1},$$

(рекуррентное соотношение для чисел сочетаний), а из алгебры известно, что если многочлены совпадают в бесконечном числе точек (в данном случае в целых α, начиная с $n+1$) , то они равны.

Теперь можно рассуждать по индукции. Пусть некий многочлен P принимает в целых точках целые значения. Если его степень равна нулю, то он константа, и всё ясно. Если его степень больше нуля, то его дискретная производная P' имеет меньшую степень и по предположению индукции представляется в виде целочисленной комбинации наших многочленов. Заменив каждый многочлен P_i в этой комбинации на его «дискретную первообразную» P_{i+1}, мы получим многочлен, у которого дискретная производная такая же, как у P, и потому у их разности дискретная производная равна нулю, значит, эта разность константа (и целая, потому что это разность двух многочленов с целыми значениями). Остаётся только добавить эту константу к целочисленной комбинации и получить P.

Из этого наблюдения, в частности, следует интересный факт о суммах степеней. Как мы видели в главе 1, сумма первых n чисел является многочленом второй степени от n, а сумма квадратов первых n чисел является многочленом третьей степени от n. Теперь можно доказать общий факт.

Теорема 2.1. *Для любого d сумма $1^d + 2^d + \cdots + n^d$ является многочленом от n степени не выше $d+1$.*

Для этого нам нужно ещё одно равенство, которое легко доказывается как по индукции, так и комбинаторными рассуждениями или вычислениями с производящими функциями.

Задача 2.59. Докажите, что

$$\sum_{j=0}^{n} P_k(j) = \sum_{j=0}^{n} \binom{j}{k} = \binom{n+1}{k+1} \tag{2.1}$$

С точки зрения производящих функций равенство (2.1) содержится в равенстве

$$\sum_{j=0}^{n} (1+t)^j = \frac{(1+t)^{n+1}-1}{t}$$

(сумма геометрической прогрессии).

Этот факт можно доказать и комбинаторно (см. задачу 2.54).

Теперь перейдём к доказательству теоремы 2.1. Выразим n^d как многочлен от n через многочлены $P_k(n)$:

$$n^d = \sum_{i=0}^{d} a_{d,i} P_i(n).$$

Тогда для суммы d-х степеней получаем

$$\sum_{j=0}^{n} j^d = \sum_{j=0}^{n}\sum_{i=0}^{d} a_{d,i}P_i(j) = \sum_{i=0}^{d} a_{d,i}\sum_{j=0}^{n}\binom{j}{i} = \sum_{i=0}^{d} a_{d,i}\binom{n+1}{i+1} = \sum_{i=0}^{d} a_{d,i}P_{i+1}(n+1).$$

В предпоследнем равенстве использовано равенство (2.1). Последнее выражение как функция от n является суммой многочленов степени не выше $d+1$, что и требовалось доказать.

2.10 Числа Каталана

2.10.1 Правильные последовательности скобок

Будем составлять из символов «(» и «)» последовательности правильно вложенных скобок. «Правильная вложенность» означает, что последовательности `((()))`, `(()())()`, `()()()` допускаются, а, скажем, последовательности `)(` или `())(` — нет. В правильной последовательности всегда равное число открывающих и закрывающих скобок, но это только необходимое, но не достаточное условие (как показывают два последних примера).

Интуитивно довольно понятно, что мы имеем в виду, говоря о «правильной вложенности» скобок, но как это объяснить формально? Одна из возможностей — дать такое индуктивное определение:

- пустая последовательность (длины 0) правильна;
- если A — правильная последовательность, то `(`A`)` — правильная последовательность;
- если A, B — правильные последовательности, то их конкатенация AB (приписываем B к A справа) — правильная последовательность.

Надо добавить ещё, что других правильных последовательностей нет, кроме тех, которые можно построить по этим правилам: правильные последовательности — это те, которые можно получить из пустой с помощью операций заключения в скобки и конкатенации. Можно сказать, что множество правильных последовательностей — это минимальное множество, обладающее этими тремя свойствами. (Заметим, что оно не единственно — скажем, множество всех последовательностей тоже ими обладает.)

Программисты записывают это определение так:

$$\langle\texttt{ппп}\rangle ::= \quad | \ \texttt{(}\langle\texttt{ппп}\rangle\texttt{)} \ | \ \langle\texttt{ппп}\rangle\langle\texttt{ппп}\rangle$$

Здесь $\langle\texttt{ппп}\rangle$ — сокращение для «правильно построенная последовательность», а черта «|» разделяет разные варианты: пустую последовательность, правильную последовательность в скобках, наконец, конкатенацию двух правильно построенных последовательностей. Такая запись называется *нормальной формой Бэкуса – Наура*

(Backus–Naur Form, BNF). Иногда то же самое записывают иначе: говорят, что есть *контекстно-свободная грамматика* с правилами

$$\langle \text{пппп} \rangle \to$$
$$\langle \text{пппп} \rangle \to \texttt{(}\langle \text{пппп} \rangle\texttt{)}$$
$$\langle \text{пппп} \rangle \to \langle \text{пппп} \rangle \langle \text{пппп} \rangle$$

которые означают то же самое.

Так или иначе, мы определили понятие правильной последовательности скобок, и теперь можем доказать следующий факт.

Теорема 2.2. *Последовательность правильно построена тогда и только тогда, когда в ней поровну скобок, а в любом её начале не меньше левых, чем правых.*

Это кажется достаточно очевидным, но давайте это аккуратно докажем.

Доказательство. Для удобства введём временное название для последовательностей, обладающих указанным свойством (про количество скобок): будем называть их, скажем, *квазиправильными*. Нам надо показать, что правильные и квазиправильные — это одно и то же. Сначала покажем, что всякая правильная последовательность квазиправильна. Это доказывается «индукцией по построению»[7]: надо доказать, что

- пустая последовательность квазиправильна;
- если A квазиправильна, то $\texttt{(}A\texttt{)}$ квазиправильна;
- если A и B квазиправильны, то их конкатенация AB квазиправильна

Первое очевидно. Второе: начало последовательности $\texttt{(}A\texttt{)}$ либо пусто, либо состоит из открывающей скобки и начала последовательности A, либо есть вся последовательность $\texttt{(}A\texttt{)}$. В первом и третьем случаях скобок поровну, во втором добавление левой скобки только улучшает дело (используем квазиправильность A). Аналогично для конкатенации: её начало может быть либо началом A, либо всей последовательностью A плюс начало B, и в обоих случаях всё очевидно.

В обратную сторону: почему всякая квазиправильная последовательность правильна? Будем идти слева направо, смотря за превышением левых скобок над правыми. (По предположению мы знаем, что оно всё время неотрицательно, а в конце равно нулю.) Обращается ли это превышение в нуль где-то в середине? Если нет, и превышение всегда по крайней мере 1, то выделим первую и последнюю скобки. Первая будет открывающей (иначе условие нарушится сразу же), последняя закрывающая (иначе условие нарушится на предпоследнем шаге). Между ними будет квазиправильная последовательность (потому что отбрасывание первой скобки

[7] Формально говоря, тем самым мы докажем, что множество квазиправильных последовательностей удовлетворяет трём свойствам из определения правильных, а множество правильных последовательностей было минимально среди таких.

уменьшает превышение ровно на 1, и оно останется неотрицательным. По индукции заключаем, что эта внутренняя последовательность правильна, а значит, и наша последовательность правильна.

Другой случай: когда превышение обратится в ноль где-то внутри последовательности. Разрежем её в этом месте. Получатся две квазиправильные (проверьте!) последовательности меньшей длины, значит (предположение индукции) они правильны, значит, наша последовательность получена конкатенацией правильных, и потому правильна. □

После всей этой подготовки мы можем наконец поставить вопрос: *сколько существует правильных последовательностей из n левых и n правых скобок*?

Задача 2.60. Убедитесь, что в таблице ничего не пропущено, и выпишите аналогичный список для $n = 4$.

$n = 0$	1	
$n = 1$	1	`()`
$n = 2$	2	`(())`, `()()`
$n = 3$	5	`((()))`, `(()())`, `(())()`, `()(())`, `()()()`

(Указание: их будет 14.)

Уже в этой задаче видно, что подсчёт правильных скобочных выражений — дело не такое простое (попробуйте написать программу, которая перечисляет их все для произвольного n). Их количество называют *n-м числом Каталана* (в честь бельгийского математика XIX века) и обозначают C_n. Эти числа удивительным образом появляются в самых разных комбинаторных задачах, и для них есть явная формула с красивыми доказательствами.

2.10.2 Рекуррентная формула

Как считать правильные скобочные выражения? Раз какого-то простого способа сразу не видно, давайте попробуем найти рекуррентную формулу. Её можно составить по грамматике. Сначала мы сделаем это неправильно (попробуйте заметить ошибку до того, как она будет указана), а потом исправим.

Итак, откуда берутся правильные выражения длины $2n$? Из двух источников: можно взять правильное выражение длины $2n-2$ и заключить его в скобки, а также можно соединить правильные выражения длин $2k$ и $2l$, где $k + l = n$ и $0 < k < n$. Первым способом получается C_{n-1} выражений; выражений второго типа будет $C_k C_l$ (при данных k и l), и потом это надо сложить. Отсюда получаем формулу:

$$C_n = C_{n-1} + \sum_{k=1}^{n-1} C_k C_{n-k}$$

Начальное условие $C_0 = 1$ (пустое выражение). Дальше $C_1 = 1$ (сумма не содержит ни одного слагаемого), потом $C_2 = 1 + C_1 C_1 = 2$, потом $C_3 = 2 + C_1 C_2 + C_2 C_1 = 6$

— и тут мы замечаем, что получилось не 5, как должно быть, а больше, мы что-то лишнее посчитали. Что?

* * *

На самом деле мы не то чтобы посчитали лишнее, а посчитали одно и то же выражение дважды: ведь `()()()` можно получить как конкатенацию `()()` и `()` а можно наоборот, и получится одно и то же. Как говорят, наша грамматика *неоднозначна*. Специалисты по формальным языкам сказали бы, что одно и то же выражение `()()()` имеет два «дерева разбора» в этой грамматике.

Чтобы исправить дело, мы используем другую грамматику для правильных выражений. (Это обычное дело при построении компиляторов: если грамматика из описания языка программирования не обладает хорошими свойствами, её преобразуют в другую, иногда это помогает.)

Вот эта однозначная грамматика в форме Бэкуса–Наура:

$$\langle \texttt{ппп} \rangle ::= \quad || \ \texttt{(}\langle \texttt{ппп} \rangle\texttt{)}\langle \texttt{ппп} \rangle$$

Надо только понять, почему она задаёт тот же самый язык (то же множество правильных выражений) и почему она однозначна.

Заметим, что по этой грамматике получаются только правильные выражения, так как действие $A, B \mapsto \texttt{(}A\texttt{)}B$ состоит из двух разрешённых (окружение скобками и конкатенация). Почему так можно получить все правильные выражения? Это можно доказать индукцией по длине выражения. Выражение длины 0 (нет скобок) стоит в правой части правила грамматики (там, где в формуле пустое место перед ||). Это база индукции. Индуктивный переход можно доказать, вспомнив рассуждение с разностью левых и правых скобок. Предположим, что все правильные выражения длины меньше n получаются по этой грамматике. Рассмотрим правильное выражение длины n. Первая скобка в нём левая, после неё разность левых и правых скобок равна 1. Посмотрим на то место, где разность впервые обращается в нуль. (Возможно, это конец слова, если раньше такого не случилось.) Перед ним стоит правая скобка, и внутри этой пары скобок разность 1 или больше. Значит, там стоит правильное выражение (теорема 2.2). Оставшаяся часть также является правильным выражением, поскольку отрезанное начало на разность не влияет и можно снова воспользоваться теоремой 2.2. Итак мы представили выражение в виде $\texttt{(}A\texttt{)}B$, где A, B — меньшие правильные выражения. Применив к ним предположение индукции, получаем, что рассматриваемое выражение получается по данной грамматике.

Почему грамматика однозначна? Потому что место разреза однозначно определяется разностью левых и правых скобок: до этого места она положительна, а здесь обращается в нуль.

Теперь, глядя на эту грамматику, можно написать правильную формулу:

$$C_n = \sum_{k+l=n-1, k,l\geqslant 0} C_k C_l$$

и по-прежнему $C_0 = 1$. По этой формуле $C_1 = C_0C_0 = 1$, $C_2 = C_0C_1 + C_1C_0 = 2$, $C_3 = C_0C_2 + C_1C_1 + C_2C_0 = 5$, так что в этот раз всё идёт как надо.

Используя эту рекуррентную формулу, можно легко вычислять числа Каталана. Но удивительным образом для них есть более явная формула:

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Скажем, для C_3: эта формула даёт

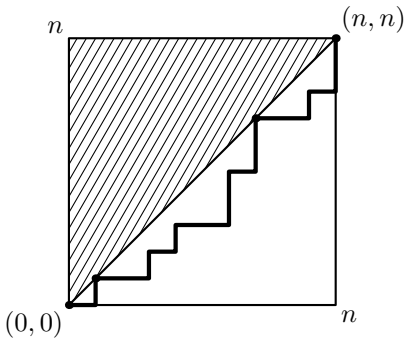
$$\frac{1}{4}\binom{6}{3} = \frac{6 \cdot 5 \cdot 4}{(1 \cdot 2 \cdot 3) \cdot 4} = 5,$$

всё в порядке.

У этой формулы есть множество доказательств, но ни одно из них, пожалуй, нельзя назвать простым и естественным. Тем не менее они красивы и поучительны, так что мы приведём несколько из них.

2.10.3 Вычисление с помощью отражений

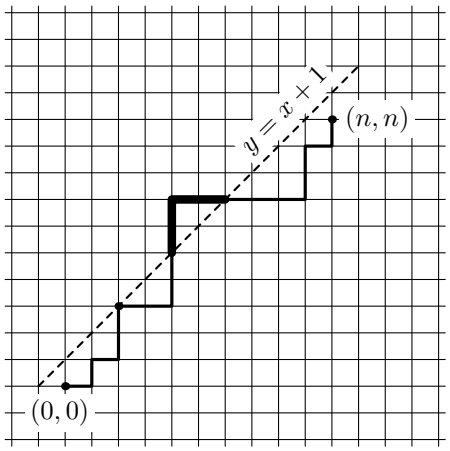
Удобно представлять себе последовательности скобок как ломаные. Начав с $(0, 0)$, мы изображаем открывающую скобку как шаг вправо на 1, а закрывающую — как шаг вверх. (Конечно, можно было бы наоборот.) Если у нас n левых и n правых скобок, то мы заканчиваем движение в точке (n, n). Осталось понять, что означает правильность скобочного выражения. Как мы знаем, она равносильна тому, что в любом начальном отрезке выражения (=пути) левых скобок (=шагов вправа) не меньше, чем правых. Геометрически это значит, что мы никогда не попадаем выше отрезка $y = x$, соединяющего $(0, 0)$ с (n, n). Другими словами, нам надо узнать, *сколько есть путей* $(0, 0)$ *в* (n, n)*, не попадающих внутрь верхней половины квадрата.* (Говоря о путях, мы имеем в виду последовательность единичных шагов вправо и вверх; можно было бы говорить о «монотонных путях», имея в виду, что влево и вниз идти нельзя.)



Предполагаемый ответ (который нам надо доказать) гласит, что они составляют $1/(n+1)$-ю долю общего числа путей (которое, как мы знаем, равно $\binom{2n}{n}$). Можно

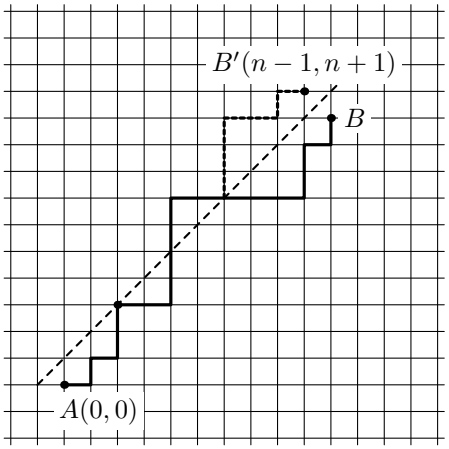
сказать, что если мы рассмотрим случайный путь, то скорее всего (с вероятностью $1-1/(n+1)$) попадём внутрь верхнего треугольника, но всё же с некоторой небольшой вероятностью $(1/(n+1)$ не выйдем из нижнего. (На границу заходить разрешается). Специалисты по теории вероятностей сказали бы, что *если в конце серии из* $2n$ *игр в орлянку мы остались при своих, то вероятность того, что мы ни разу не уходили в минус, равна* $1/(n+1)$.

Но как это доказать? Будем считать плохие пути — те, которые поднимаются выше прямой $y = x$, или, другими словами, доходят до прямой $y = x+1$ (пересекают её или касаются). На рисунке показана эта прямая и один плохой путь: сначала он её касается, а потом даже пересекает (эти участки нарисованы жирнее).



Есть такая задача: даны две точки A и B с одной стороны прямой реки, надо найти кратчайший путь из A в B с заходом на реку («за водой»). Она решается так: участок пути от набора воды до точки B мы отражаем симметрично относительно реки, получается путь из A в симметричную точку B'. Он имеет ту же длину, и чтобы сделать эту длину минимальной, надо идти по отрезку AB' (который заведомо пересекает реку, так как A и B' по разные стороны).

К чему все эти разговоры? Сделаем примерно то же самое в нашей задаче. А именно, для каждого плохого пути посмотрим на последний момент, когда он попал на запрещённую прямую, и участок пути после этого момента отразим симметрично относительно этой запрещённой прямой:



Получится путь из точки $A(0,0)$ в $B'(n-1,n+1)$; он тоже будет монотонным, так как шаги вправо после симметрии становятся шагами вверх и наоборот. Теперь ключевое наблюдение: *это преобразование задаёт взаимно однозначное соответствие между плохими путями и всеми путями из A в B'*. В самом деле, для нового пути последний момент пересечения с прямой тот же самый, что и для старого, поэтому старый путь можно получить из нового, отразив симметрично участок после этого момента. (Кстати, это останется верным, если мы изменим рассуждение и будем рассматривать первый момент пересечения с запрещённой прямой, а не последний.) И любой путь из A в B' волей-неволей пересекает прямую, так что получатся они все.

Остаётся подсчитать: плохих путей $\binom{2n}{n-1}$, а всего путей $\binom{2n}{n}$, так что надо вычислить разность

$$\binom{2n}{n} - \binom{2n}{n-1} = \frac{(2n)!}{n!n!} - \frac{(2n)!}{(n-1)!(n+1)!} = \frac{(2n)!}{n!n!}\left(1 - \frac{n}{n+1}\right) = \frac{1}{n+1}\binom{2n}{n}$$

(Заметим в скобках, что это вычисление, независимо от всяких чисел Каталана, показывает, что средний элемент в $2n$-й строке треугольника Паскаля делится на $n+1$, что не сразу очевидно.)

2.10.4 Вычисление с производящей функцией

Ну хорошо, мы нашли изящный приём, позволяющий доказать формулу для чисел Каталана. Но есть ли какой-то более систематический подход, позволяющий получать подобные формулы? Оказывается, что производящие функции тут могут помочь. Вот соответствующее рассуждение (в котором все действия на самом деле могут быть узаконены).

Начнём с того, что поймём, что означает рекуррентное соотношение

$$C_n = \sum_{k+l=n-1, k,l\geqslant 0} C_k C_l \quad \text{при } n \geqslant 1$$

в терминах производящей функции, то есть ряда

$$c(x) = C_0 + C_1 x + C_2 x^2 + C_3 x^3 + \ldots = 1 + x + 2x^2 + 5x^3 + \ldots$$

Правая часть будет коэффициентом при x^{n-1} в произведении двух копий этого ряда, то есть в $c^2(x)$. Соответственно, чтобы получить ряд для $c(x)$, мы должны домножить это произведение на x и добавить свободный член 1:

$$c(x) = xc^2(x) + 1.$$

Это можно рассматривать как квадратное уравнение

$$xc^2(x) - c(x) + 1 = 0$$

с неизвестным значением $c(x)$ и известными коэффициентами x, -1 и 1, и написать решение по обычной формуле для квадратного уравнения:

$$c(x) = \frac{1 \pm \sqrt{1-4x}}{2x}$$

Вот только взять плюс или минус? Если взять плюс, то при $x \to 0$ правая часть стремится к бесконечности (числитель стремится к 2, а знаменатель стремится к нулю), а это нехорошо, так как $c(x)$ обращается в 1 при $x = 0$. Поэтому возьмём минус:

$$c(x) = \frac{1 - \sqrt{1-4x}}{2x}$$

Проверим, что поведение около нуля выглядит правдоподобно: $\sqrt{1-4x} \approx 1-2x$ при $x \approx 0$, так что действительно в частном получается примерно 1.

Ну и чего мы добились? мы искали $c(x)$, считая его рядом, а вместо этого получили формулу с квадратным корнем — что это даёт? На самом деле осталось совсем немного: вспомним, что бином Ньютона для показателя степени $1/2$ как раз и представляет корень в виде ряда (с. 78). Правда, в формуле для $(1+x)^{1/2}$ надо заменить x на $-4x$, как у нас теперь. Получится

$$\sqrt{1-4x} = 1 - \frac{4x}{2} - \frac{(4x)^2}{8} - \frac{(4x)^3}{16} - \ldots - \frac{(2n)!}{(2n-1)(n!)^2 4^n}(4x)^n + \ldots$$

(теперь все члены одного знака, поскольку ещё одно чередование возникает из-за минуса в $-4x$; в общей формуле поэтому в знаменателе надо изменить знак и написать $2n-1$ вместо $1-2n$). Выпишем явно первые несколько членов и сократим четвёрки в числителе и знаменателе:

$$\sqrt{1-4x} = 1 - 2x - 2x^2 - 4x^3 - \ldots - \frac{(2n)!}{(2n-1)(n!)^2}x^n + \ldots$$

Теперь по формуле корней надо вычесть это из единицы и разность поделить на $2x$. Получится

$$\frac{1-\sqrt{1-4x}}{2x} = 1 + x - x^2 + \ldots + \frac{(2n)!}{2(2n-1)(n!)^2}x^{n-1} + \ldots$$

Число Каталана C_n должно быть коэффициентом при x^n. Видно, что начинается всё правильно (с $1, 1, 2, \ldots$), а в общей формуле надо сдвинуть n на единицу:

$$C_n = \frac{(2n+2)!}{2(2n+1)((n+1)!)^2} = \frac{(2n+1)(2n+2)}{2(2n+1)(n+1)(n+1)} \cdot \frac{(2n)!}{n!n!} = \frac{1}{n+1}\binom{2n}{n},$$

что и требовалось доказать. (Последние слова двусмысленны — строго говоря, мы не доказали того, что требовалось, пока не узаконили все наши действия: операции с бесконечными рядами и применение формулы бинома для показателя $1/2$. Это можно сделать, но требует разных сведений из алгебры и анализа.)

2.10.5 Вычисление с теорией вероятностей и поворотами

В нашем первом (комбинаторном) вычислении мы считали плохие пути. Сейчас мы попробуем подсчитать хорошие, используя терминологию теории вероятностей.[8] Для начала будем представлять себе скобочное выражение как дорогу, в которой есть участки подъёма и спуска (на одну и ту же величину). Левой скобке соответствует подъём, а правой спуск. Последовательностям скобок длины $2n$ соответствуют дороги из $2n$ участков. Равенство тех и других скобок означает, что дорога кончается на той же высоте, что и началась, то есть что её можно считать кольцевой с отмеченным началом (старт и финиш одновременно). Будем также считать, что в проекции дорога имеет вид окружности, а участки спуска и подъёма проектируются в дуги этой окружности одинаковой угловой величины π/n. Правильность означает, что дорога нигде не опускается ниже своей начальной точки, и вероятность этого события для случайно взятой дороги (все последовательности считаем равновероятными) равна $1/(n+1)$ — это мы хотели бы доказать.

Ничего не изменится, если мы сдвинем точку старта на какой-то фиксированный угол, кратный π/n, у каждой дороги: распределение вероятностей на дорогах (с выделенной точкой старта) останется тем же. Поэтому и при случайном независимом выборе дороги и сдвига вероятность останется той же. Теперь заметим, что если мы выбираем для данной кольцевой дороги точку старта, то *последовательность подъёмов и спусков будет правильной в том и только том случае, когда мы начинаем с самой низкой точки дороги.* Если бы таких точек всегда была одна, то вероятность равнялась бы $1/(2n)$, но их больше, так что это рассуждение не проходит. (Зато, если знать ответ, оно показывает, что у случайной дороги в среднем $2n/(n+1)$ точек минимума, что совсем не очевидно).

Что же делать? Оказывается, что помогает следующий довольно странный трюк. Давайте ко всем последовательностям из n участков вверх и вниз добавим вначале участок вверх. Получится, что искомое число равно *числу последовательностей из $n+1$ шагов вверх и n шагов вниз, у которых все точки, кроме начальной, строго выше начальной.* (Тут надо заметить, что сформулированное условие гарантирует, что первый шаг будет вверх, так что лишних мы не получим.)

[8] Если вы с этим не сталкивались, то можно прочитать рассуждение просто как правдоподобную байку, но оно на самом деле вполне корректно. В конце раздела мы приводим краткий перевод рассуждения на комбинаторный язык.

Правда, теперь все наши разговоры о кольцевой дороге утрачивают силу: у нас есть $n+1$ участков, где подъём (скажем, на 1) и n участков спуска на 1, так что в итоге после $2n+1$ участков мы оказываемся на 1 выше, чем начинали. Сделаем такую, на первый взгляд нелепую, вещь: добавим на каждом шаге спуск $1/(2n+1)$, тогда он как раз скомпенсирует разность между числом подъёмов и спусков. Мы получаем случайную кольцевую дорогу из $n+1$ подъёмов на $1-1/(2n+1)$ и n спусков на $1+1/(2n+1)$. Нам надо узнать, какова вероятность того, что мы будем всегда оставаться выше начальной точки — с учётом добавочного спуска или без, это всё равно, так как вначале подъёмы и спуски у нас были целыми числами, и добавка, не достигающая 1 на неполном круге, не может полностью уничтожить даже минимальный возможный подъём на 1. Таким образом, вопрос можно сформулировать так: какова вероятность того, что начало окажется минимумом такой кольцевой дороги? Опять же можно добавить случайный поворот, и заметить, что теперь у каждой дороги минимум единствен, так как благодаря добавке все высоты различаются (по модулю 1, а значит, и вообще). Значит, эта вероятность равна $1/(2n+1)$, а число хороших дорог (равное числу Каталана, как мы видели) есть

$$\frac{1}{2n+1}\binom{2n+1}{n} = \frac{(2n+1)!}{(2n+1)(n+1)!n!} = \frac{1}{n+1}\binom{2n}{n}.$$

Рассуждения со ссылками на теорию вероятностей — дело тонкое, в них легко не заметить какой-то ошибки. Поэтому, чтобы снять с себя подозрения, кратко перескажем то же рассуждение в комбинаторных терминах. Запишем все последовательности из $n+1$ открывающих скобок и n закрывающих в столбик. Рядом с этим столбиком напишем такой же, но только каждую последовательность циклически сдвинем на 1, затем ещё один, с циклическим сдвигом на 2, и так далее. Получим прямоугольную таблицу высоты $\binom{2n+1}{n}$ и ширины $2n+1$. В каждом столбце все комбинации встречаются по разу, значит во всей таблице каждая комбинация встречается $2n+1$ раз, и все они появляются одинаково часто. Благодаря этому доля правильных комбинаций (последовательностей, в которых любое непустое начало содержит больше открывающих скобок, чем закрывающих) во всей таблице такая же, как в первом (или любом другом) столбце. Но во всей таблице эта доля равна $1/(2n+1)$, поскольку наше рассуждение с дорогами, спусками, подъёмами и коррекциями показывает, что в каждой строке ровно одна правильная комбинация.

2.10.6 Доказательство по индукции с дробными параметрами

Если формулу для чисел Каталана мы уже знаем (пусть даже без доказательства), то возникает естественная идея: доказать её по индукции. Другими словами, мы можем определить числа

$$\tilde{C}_n = \frac{1}{n+1}\cdot\frac{2n!}{n!\,n!}$$

безотносительно к их комбинаторному смыслу, и доказать, что для них верна рекуррентная формула

$$\tilde{C}_n = \sum_{k+l=n-1, k,l\geqslant 0} \tilde{C}_k \tilde{C}_l$$

(мы пишем знак тильды, временно различая «комбинаторные» числа Каталана, и полученные по формуле). Эта формула — некоторое тождество для чисел сочетаний, но как его доказать? В него входят только средние числа в разных строках треугольника, поэтому не очень понятно, как бы это можно было комбинаторно истолковать.

Тут помогают дробные аргументы. Несложным вычислением с факториалами можно убедиться, что

$$\binom{1/2}{n} = -\frac{(-1)^n}{2^{2n-1}} \tilde{C}_{n-1}$$

(в левой части возникает произведение всех нечётных чисел $1 \cdot 3 \cdot 5 \cdot \ldots \cdot (2n-3)$, иногда обозначаемое $(2n-3)!!$; чтобы выразить его через обычные факториалы, надо домножить на произведение чётных чисел $(2n-2)!! = 2 \cdot 4 \cdot 6 \cdot \ldots \cdot (2n-2)$, равное $2^{n-1}(n-1)!$, и получить обычный факториал).

Правда, при $n = 0$ правая часть не имеет смысла, и чтобы придать ей смысл, надо положить $C_{-1} = -1/2$. Теперь мы можем вспомнить тождество

$$\binom{k+l}{n} = \binom{k}{0}\binom{l}{n} + \binom{k}{1}\binom{l}{n-1} + \binom{k}{2}\binom{l}{n-2} + \ldots + \binom{k}{n}\binom{l}{0},$$

которое у нас было для натуральных k и l (выбирая n элементов из $k+l$, надо взять сколько-то из первых k и оставшиеся из остальных l). Но это равенство двух многочленов от k и l, и из алгебры известно, что если оно верно для всех целых чисел, то оно всегда верно. После этого запишем его для $k = l = 1/2$, правая часть обратится в нуль, и получится тождество

$$\tilde{C}_{-1}\tilde{C}_{n-1} + \tilde{C}_0\tilde{C}_{n-2} + \tilde{C}_1\tilde{C}_{n-3} + \ldots + \tilde{C}_{n-2}\tilde{C}_0 + \tilde{C}_{n-1}\tilde{C}_{-1} = 0.$$

Вспоминая о соглашении $C_{-1} = -1/2$, мы получаем искомое рекуррентное соотношение (только для меньшего на единицу значения n).

2.10.7 Неассоциативные произведения, триангуляции и стековый калькулятор

Мы уже описали четыре способа получения формулы для чисел Каталана — но почему именно они нас интересуют? Дело в том, что эти числа удивительным образом появляются во многих комбинаторных задачах, и мы сейчас рассмотрим два таких примера.

Пусть у нас есть m сомножителей. Вычисляя их произведение, мы можем выполнять действия в разном порядке. Обычно этот порядок указывают скобками. Например, для трёх сомножителей возможны порядка $(ab)c$ и $a(bc)$ — можно сначала

перемножить a и b, а потом умножить на c, а можно умножить a на предварительно вычисленное произведение bc. Окончательный результат получится один и тот же (это свойство называют *ассоциативностью*). Для четырёх сомножителей вариантов (если не разрешать переставлять сомножители местами) будет 5:

$$((ab)c)d,\ (a(bc))d,\ (ab)(cd),\ a((bc)d),\ a(b(cd)).$$

Задача 2.61. Выпишите все варианты для пяти сомножителей (каким-то систематическим способом, чтобы ни один не пропустить) и убедитесь, что их будет 14.

Сколько вариантов возможно для m сомножителей? Наверное, вы уже догадались, что ответом к этой задаче (обозначим его P_m) будет число Каталана, только со сдвигом на единицу: $P_m = C_{m-1}$. Для нескольких первых значений m мы это видим своими глазами, но почему это будет верно и дальше?

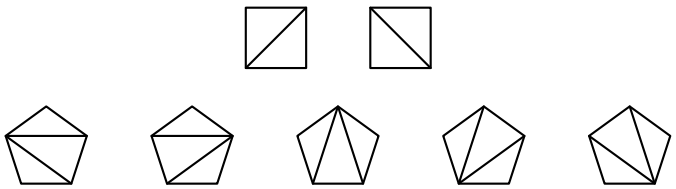
Самый простой способ это доказать — проверить, что для новой последовательности (с учётом сдвига нумерации на единицу) выполняется то же самое рекуррентное соотношение. В самом деле, пусть у нас есть m сомножителей. Посмотрим на последнюю операцию (умножение, которое даёт окончательный ответ). Она может быть в разных местах — после первого сомножителя, после второго сомножителя и так далее. Соответственно по правилу суммы надо посчитать количество вариантов каждого типа и их сложить. Для каждого варианта мы по правилу произведения перемножаем число способов записать выражение слева и справа от точки деления. Скажем, для произведения P_6 мы получаем формулу

$$P_6 = P_1P_5 + P_2P_4 + P_3P_3 + P_4P_2 + P_5P_1$$

(например, член P_3P_3 соответствует вариантам, когда на последнем шаге мы перемножаем произведение первых трёх сомножителей и последних трёх, то есть точка деления посередине). Если переписать это с уменьшенными на единицу индексами, то получится в точности рекуррентное соотношение для чисел Каталана, что и завершает доказательство.

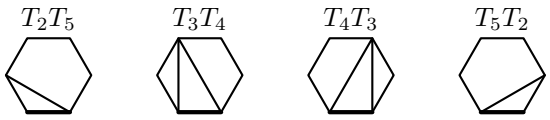
Вот ещё одна похожая задача. Рассмотрим выпуклый m-угольник, и будем разрезать его непересекающимися диагоналями (общие концы разрешаются) на треугольные части. Можно сразу же заметить (сумма углов), что этих частей будет $m-2$, и что понадобится $m-3$ диагонали, как их ни проводи (каждая новая диагональ добавляет часть). Вопрос: *сколькими способами можно выполнить такую триангуляцию*? Обозначим их число через T_n.

Для треугольника способ один, для четырёхугольника, очевидно, два (можно разрезать по любой из диагоналей). Для пятиугольников способов пять (и все они сводятся к тому, что надо из одной вершины провести все диагонали).



Задача 2.62. Выпишите все варианты для шестиугольника (каким-нибудь систематическим образом, чтобы ни одного не пропустить) и посчитайте их (должно получиться 14).

Как вы догадываетесь, мы хотим теперь доказать, что $T_m = C_{m-2}$, и понятно как: получив рекуррентную формулу для числа триангуляций. Вот как это делается. Пусть у нас есть, скажем, шестиугольник. Выберем и зафиксируем какую-то его сторону, назовём её основанием. К этому основанию должен примыкать какой-то треугольник. Его третьей вершиной может быть любая из четырёх оставшихся вершин. После того как мы выбрали положение третьей вершины, мы независимо триангулируем оставшиеся части. (В двух крайних случаях с одной стороны остаётся «двуугольник», и выбора нет, так что мы полагаем $T_2 = 1$.)

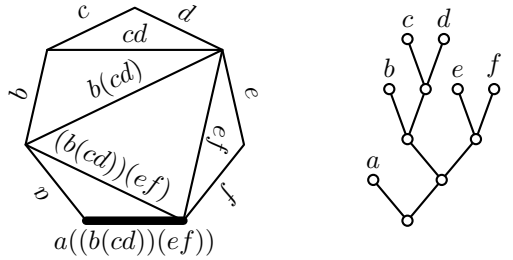


Получаем формулу

$$T_6 = T_2T_5 + T_3T_4 + T_4T_3 + T_5T_2$$

которая опять отличается от каталановской сдвигом индексов (как раз на 2).

Сравнивая две эти задачи, заключаем, что количество триангуляций $(m+1)$-угольника равно количеству способов расстановки скобок в неассоциативном произведении m элементов. Этому равенству можно дать и комбинаторное объяснение, установив взаимно однозначное соответствие между теми и другими. Для этого на всех сторонах $(m+1)$-угольника, кроме его основания, напишем буквы (переменные, которые надо перемножать, слева направо). Пусть дана некоторая триангуляция этого $(m+1)$-угольника. Тогда на всех диагоналях можно написать промежуточные результаты в процессе умножения, как на рисунке:



Также этой триангуляции и этому произведению можно поставить в соответствие двоичное дерево с корнем, листьями которого являются переменные (как показано справа на том же рисунке), поэтому количество таких деревьев с m вершинами тоже равно C_{m-1}.

Наконец, если уж мы заговорили о комбинаторных доказательствах (их называют ещё «биективными», поскольку взаимно однозначные соответствия называют

биекциями), то возникает вопрос: *можно ли поставить в соответствие каждому правильному скобочному выражению* (с которых мы начинали, из n открывающихся и n закрывающихся скобок) *некоторый порядок действий в произведении* $n+1$ *элементов*? Мы ведь уже знаем, что количество тех и других одинаковое.

Первая идея, которая приходит в голову: может быть, просто стереть все буквы в произведении и оставить одни скобки? получится ведь правильное скобочное выражение. Беда в том, что это никак не будет взаимно однозначным соответствием: без букв в выражениях $a(bc)$ и $(ab)c$ останется одно и то же. Тем не менее такое соответствие можно явно описать, и оно имеет программистский смысл — но скобки в правильном скобочном выражении будут иметь другой смысл, это операции загрузки и умножения в стековом калькуляторе для выражений. Вот что это значит.

Есть такая «обратная польская запись» для арифметических выражений, где знак операции ставится после операндов. При этом скобки не нужны: скажем, произведение $(ab)c$ записывается как $ab{\times}c{\times}$, а произведение $a(bc)$ записывается как $abc{\times}{\times}$. Достоинство такой записи (из-за чего она использовалась в некоторых программируемых калькуляторах) состоит в том, что записанное таким образом выражение легко вычислять слева направо. Видя букву, мы помещаем соответствующую переменную в стек, а видя операцию ($\times$), мы заменяем два верхних элемента стека на их произведение (тем самым уменьшая число элементов на единицу). Вот как это получается, скажем, для выражения $((ab)(cd))$, которое в польской записи имеет вид $ab{\times}cd{\times}{\times}$; вершина стека справа:

	`a`
a	
	`b`
a, b	
	$\times$
ab	
	`c`
ab, c	
	`d`
ab, c, d	
	$\times$
ab, cd	
	$\times$
$(ab)(cd)$	

В правом столбце сверху вниз написаны символы выражения в обратной польской записи, а слева содержимое стека после соответствующего действия.

Названия переменных среди команд идут в том же порядке, что и в исходном произведении, поэтому они излишни, можно писать одну и ту же букву, скажем, v в смысле «очередная переменная». Тогда расстановке скобок, указывающей порядок умножений в произведении m сомножителей, будет соответствовать последовательность из m букв v (соответствующих сомножителям) и $m-1$ знаков $\times$ (каждое

умножение уменьшает число сомножителей на 1). В этой последовательности первые две буквы должны быть v (потому что умножение возможно, только если в стеке как минимум два операнда), а потом в любом начальном отрезке количество букв v должно быть как минимум на одну больше, чем количество знаков умножения (иначе стек опустеет: буквы его удлиняют, а умножения укорачивают). Приходим — если отбросить первую букву v — как раз к правильным скобочным выражениям, где v играет роль левой скобки, а $\times$ — правой.

Конечно, это всё надо аккуратно определить и доказать (мы не сделали ни того, ни другого — попробуйте на досуге), но хотя бы само соответствие мы описали достаточно явно.

2.11 Что дальше?

На этом мы заканчиваем знакомство с перечислительной комбинаторикой — но сама она, конечно, не заканчивается. Чему мы научились — и чему ещё предстоит научиться?

Мы убедились, что во многих случаях можно посчитать число объектов с данным свойством, не пересчитывая эти объекты. Иногда ответ даётся явной формулой; иногда такую явную формулу найти не удаётся, но можно составить рекуррентное соотношение и с его помощью вычислить искомое количество.[9] (Программистам этот приём известен под названием «динамическое программирование».)

Пример с числами Каталана показывает, что при доказательстве комбинаторных тождеств иногда нужны не только остроумные соображения (скажем, идея отражения путей), но и разные сведения из других разделов математики (разложение в ряд, формула Тейлора и многое другое — у нас были только самые простые примеры). Если вам это понравилось, много других интересных вещей вы найдёте в книге «Конкретная математика» [7] (Р. Грэхем, Д. Кнут и О. Паташник; один из авторов — тот самый Дональд Кнут, который написал знаменитый многотомник «Искусство программирования» [10] и не менее знаменитую программу TEX, с помощью которой сейчас набираются практически все книги по математике и программированию, включая эту) и в книге «Лекции о производящих функциях» [15] (С. К. Ландо).

[9] Можно спросить, всегда ли для такого рода задач (подсчитать количество объектов с данным просто проверяемым свойством) существует быстрый алгоритм. Этот вопрос естественно формулируется в терминах теории сложности вычислений, но ответ на него неизвестен. Он является некоторым аналогом известной проблемы о равенстве классов P и NP.

Лекция 3

Графы

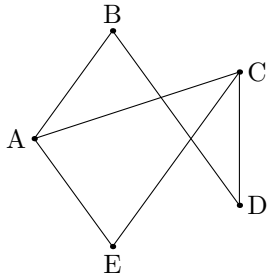
Чтобы решить какую-то задачу, часто бывает полезно нарисовать картинку, иллюстрирующую её условие. В этой главе мы рассмотрим один вид таких картинок: «графы». Граф — это набор точек («вершин»), соединённых линиями («рёбрами»). При этом важно, какие точки соединены, а как именно это ребро нарисовано, не имеет значения.

Прежде чем давать точные определения соответствующих понятий, мы разберём несколько задач, в которых подобные картинки помогают.

3.1 Примеры

3.1.1 Граф авиарейсов

Задача 3.1. Представим себе страну, в которой есть пять городов A, B, C, D, E, между которыми летают самолёты. Есть шесть рейсов: A–B, A–C, A–E, B–D, C–D, C–E (каждый рейс в обе стороны). Можно ли долететь из города A в город D прямым рейсом? с одной пересадкой? с двумя пересадками? Сколькими способами?



Это совсем простая задача: чтобы её решить, достаточно нарисовать картинку. Сразу видно, что прямого рейса нет, с одной пересадкой есть два способа A–B–D и A–C–D, а с двумя пересадками есть единственный вариант A–E–C–D.

Ту же картинку можно использовать, чтобы ответить на более сложный вопрос.

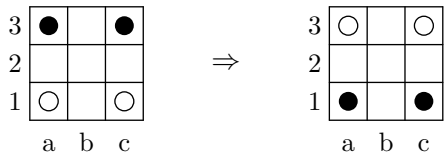
Задача 3.2. Докажите, что после отмены любого из шести рейсов оставшиеся пять позволяют добраться из любого города в любой («не теряется связность», как говорят).

Решение. Можно по очереди вычеркнуть каждый из рейсов и убедиться в требуемом. Если хотеть сказать короче, можно заметить, что можно составить два кольца («цикла») A–B–D–C–A и A–C–E. Всякий рейс входит в одно из колец; если он отменится, остальные рейсы кольца могут его заменить (пусть и с пересадками). □

3.1.2 Перестановка коней

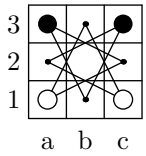
Следующая задача часто разбирается на математических кружках для школьников, так что, возможно, вы её уже видели.

Задача 3.3. На доске 3×3 в углах a1, c1, a3, c3 поставлены белые и чёрные кони (белые снизу, чёрные сверху). Можно ли их поменять местами, чтобы чёрные оказались внизу, а белые наверху?

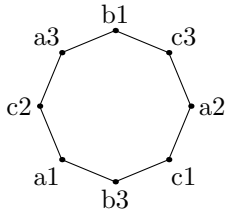


(По шахматным правилам конь ходит буквой «Г», на две клетки вперёд и на одну вбок, и не может стать на уже занятую клетку.)

Решение. Давайте отметим на доске, из каких клеток можно попасть в какие ходом коня. (Из центральной клетки никуда пойти нельзя, оставаясь на доске).



Картина станет яснее, если мы «распутаем» эти линии, составив схему возможных движений с точки зрения коня. Например, из a1 конь может пойти в b3 и c2, из b3 можно пойти обратно в a1 и дальше в c1, и так далее.



Теперь видно, что «топологически» (если не обращать внимание на географическое положение, а только на возможности ходов) кони могут двигаться по кругу, как в хороводе. Чтобы поменять чёрных и белых местами, достаточно в этом круге сделать полоборота против часовой стрелки. Сначала каждый конь делает по одному шагу:

$$\mathrm{a1} \to \mathrm{b3},\ \mathrm{c1} \to \mathrm{a2},\ \mathrm{c3} \to \mathrm{b1},\ \mathrm{a3} \to \mathrm{c2}.$$

Контрольный вопрос 3.4. Нарисуйте (на шахматной доске) положения коней в этот момент.

Затем кони делают ещё шаг в том же направлении по кругу:

$$\mathrm{b3} \to \mathrm{c1},\ \mathrm{a2} \to \mathrm{c3},\ \mathrm{b1} \to \mathrm{a3},\ \mathrm{c2} \to \mathrm{a3}.$$

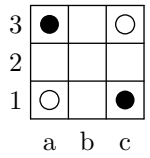
Контрольный вопрос 3.5. Убедитесь, что кони снова в углах шахматной доски, но цвета ещё не те.

Чтобы прийти к искомому положению, достаточно повторить такой сдвиг по кругу ещё два раза. Задача решена. □

Задача 3.6. Сколько всего ходов сделали кони в нашем решении? Можно ли обойтись меньшим числом ходов?

Решение. На первый вопрос ответить легко: каждый конь сделал четыре хода, поэтому всего понадобилось 16 ходов. Второй вопрос немного сложнее. Заметим, что порядок коней по кругу не меняется при движении (как для машин на круговом шоссе, где нет возможности обгона). Значит, белый конь a1 должен попасть на c3, а белый конь c1 должен попасть на a3, а не наоборот, и каждый из них должен сделать по четыре хода (ему надо попасть в противоположную точку кольца). Аналогично для чёрных коней, и потому меньше 16 ходов не получится. □

Задача 3.7. Можно ли поменять местами одного чёрного и одного белого коня, получив показанную на рисунке позицию? (Начальная позиция та же, что и в предыдущих задачах.)

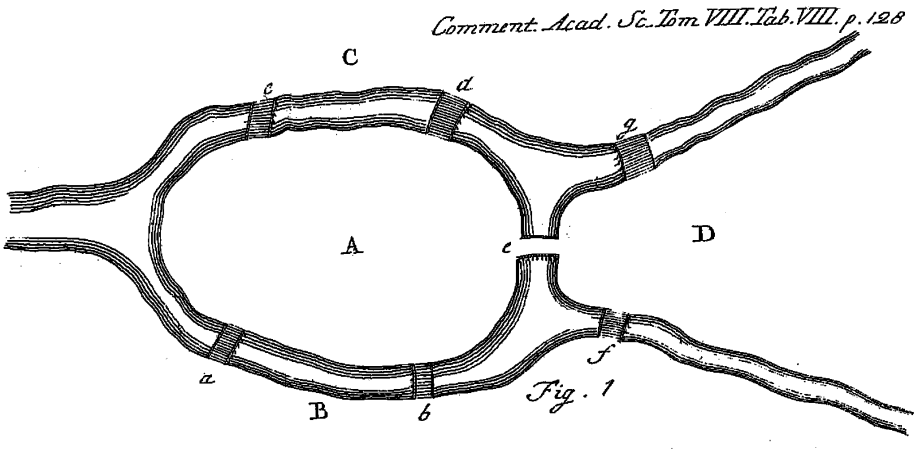


Решение. Нет — в терминах кольца это означает, что один конь должен обогнать другого, а это невозможно (чёрные кони были рядом, и останутся рядом, аналогично и для белых). □

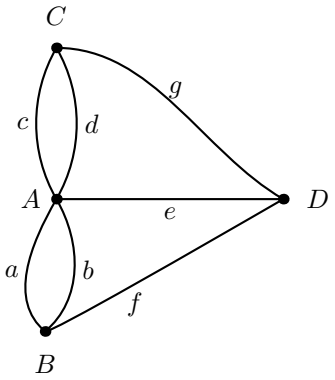
Задача 3.8. Найдите ответы на те же вопросы для коней, стоящих в углах доски 4×3.

3.1.3 Эйлер и мосты в Кёнигсберге

Следующая задача была разобрана в статье великого математика (работавшего в Петербурге) Леонарда Эйлера. Эту его статью считают первой работой по топологии. Эйлера спросили, можно ли совершить прогулку по Кёнигсбергу,[1] пройдя по каждому мосту по одному разу и вернувшись в исходную точку. На рисунке (взятом из статьи Эйлера; её можно найти, например, в https://math.dartmouth.edu/~euler/docs/originals/E053.pdf) мосты обозначены буквами a, b, c, d, e, f, g.



Как решить задачу Эйлера? Заметим, что ходьба по суше никак не ограничивается — и поэтому каждый участок суши можно «стянуть в точку». На рисунке Эйлер отметил четыре таких участка A, B, C, D. Изобразим их точками и нарисуем (в виде линий) соединяющие их мосты, сохраняя обозначения Эйлера.



Теперь задачу Эйлера можно переформулировать так: можно ли пройти по каждой линии один раз и вернуться в исходную точку? Можно ли нарисовать эту кар-

[1] Сейчас (2015) этот город называется Калининград и находится на территории Российской Федерации — но некоторые из эйлеровских мостов разрушены, и построены новые. В этом городе родился, жил, работал и умер философ Кант — в честь которого назван тамошний университет.

тинку карандашом, не отрывая карандаша от бумаги, не проходя второй раз по уже нарисованным линиям и в конце вернувшись в исходную точку?

Глядя на картинку, легко понять, что это невозможно и в чём состоит препятствие. Посмотрим, например, на вершину D. В неё входят три линии. Обходя наш маршрут по циклу, мы должны пройти по каждой из этих трёх линий по одному разу. Но, войдя в точку по одной линии, мы должны выйти по другой, и снова войдя по третьей, мы уже не сможем выйти (если не хотим проходить по какому-то мосту второй раз). Другими словами, необходимое условие разрешимости задачи обхода: в каждую точку входит чётное число линий (иначе они не поделятся на пары вход – выход). И в задаче Эйлера оно не выполнено.

Контрольный вопрос 3.9. Можно ли в этом рассуждении использовать вместо D другую точку? (Ответ: да, годится любая точка (в точках B и C тоже сходятся по три линии; в точке A их пять, но пять — тоже нечётное число.)

Задача 3.10. Можно ли пройти по каждому мосту по разу, если не требовать, чтобы маршрут был замкнут (начало и конец могут быть в разных местах)?

Решение. Нет: при этом нечётное число линий может быть в начале и конце пути, а во всех остальных вершинах должно быть по-прежнему чётное. А у нас во все четыре вершины входит нечётное число линий. □

Задача 3.11. Какое минимальное число мостов в задаче Эйлера нужно закрыть для прохода, чтобы для оставшихся мостов задача стала разрешимой?

Задача 3.12. Как нарисовать распечатанный конверт, не отрывая карандаша от бумаги (левый рисунок)? Почему нельзя сделать того же для запечатанного конверта (правый рисунок)?

3.1.4 Рукопожатия

Задача 3.13. Перед началом встречи некоторые её участники пожали руки другим. Докажите, что количество участников, сделавших *нечётное* число рукопожатий, *чётно*.

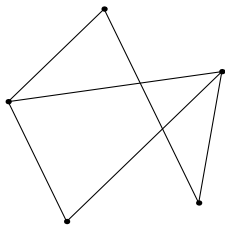
(Рукопожатие симметрично, то есть учитывается для обоих участников.)

Решение. Вначале все участники сделали 0 рукопожатий, поэтому «нечётных участников» (тех, кто сделали нечётное число рукопожатий) в этот момент нет. Их количество равно нулю и потому чётно. (Нуль делится на 2 без остатка и является чётным числом.)

Что происходит при каждом рукопожатии? Количество «нечётных» либо уменьшается на 2 (два «нечётных» участника, пожав друг другу руки, становятся «чётными»), либо увеличивается на 2 (два «чётных» участника, пожав друг другу руки,

становятся нечётными), либо остаётся прежним (если «чётный» участник жмёт руку «нечётному», то становится «нечётным», а другой — «чётным»). Задача решена. □

Но можно решать задачу и иначе. Изобразим участников точками, а рукопожатия — линиями. (На плоскости линии могут пересекаться, но эти пересечения нас не интересуют).



Например, на нашем рисунке изображено 5 участников после 6 рукопожатий.

Контрольный вопрос 3.14. Кто из участников сделал больше всего рукопожатий? Сколько? (Ответ. Число рукопожатий, сделанных участником — это число линий, выходящих из изображающей его точки. Максимальное такое число равно 3, и таких наиболее общительных участников на рисунке двое.)

Подсчитаем для каждого участника число сделанных им рукопожатий. (На языке теории графов — подсчитаем для каждой вершины графа её степень, число выходящих из неё рёбер.) Все эти числа сложим. Что получится, как вы думаете?

Оказывается, *получится удвоенное число рукопожатий*. В самом деле, для каждой точки выпишем все линии, которые в неё входят. Число линий в этом общем списке будет равно интересующей нас сумме, а каждая линия будет учтена дважды (для двух её концов).

Почему наше наблюдение позволяет решить задачу? Почему, *если сумма нескольких целых чисел чётна, то число нечётных слагаемых чётно*? Понятное дело: чётные слагаемые вообще на чётность суммы не влияют, а каждое нечётное слагаемое меняет чётность суммы, значит, их должно быть чётное число.

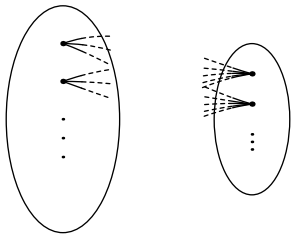
Задача 3.15. Какое минимальное число дополнительных рукопожатий нужно сделать в описанной ситуации, чтобы все участники сделали по одинаковому числу рукопожатий?

3.1.5 Задачи и решения

Задача 3.16. На контрольной каждый из 20 школьников решил ровно 3 задачи, а каждую задачу решило ровно 5 человек. Сколько было задач?

Решение. Представим себе, что школьники писали решения задач на отдельных листках бумаги (каждое решение занимает ровно один листок). Тогда всего было $20 \times 3 = 60$ листков. С другой стороны, если разложить листки по задачам, то на каждую задачу придётся 5 листков, поэтому задач $60/5 = 12$. □

Понятно ли, почему эта задача похожа не предыдущую? Нарисуем картинку, в которой школьники и задачи изображены точками, а решения изображены линиями (один конец — кто решил, другой — что решил). Чтобы не путаться, нарисуем школьников слева, а задачи справа. (Научно это называется «двудольный граф» со школьниками в левой доле и с задачами в правой).



На этом языке условие говорит, что в левой доле 20 точек («вершин»), и из каждой выходит по три линии («каждая вершина слева имеет степень 3»). А в правой доле неизвестно сколько точек, но в каждую приходит по 5 линий (от тех школьников, кто эту задачу решил). Сколько точек в правой доле? Решая задачу, мы подсчитали число линий двумя способами: считая их левые концы и их правые концы. Первый подсчёт даёт сумму степеней вершин слева, то есть $20 \times 3 = 60$, а второй даёт $5 \times$ (число задач). Поэтому задач $60/5 = 12$. Мы использовали такое соображение: сумма степеней левых вершин равна сумме степеней правых вершин (и равна числу рёбер).

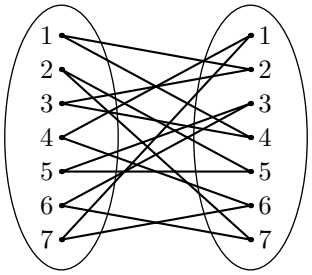
Есть ли какая-то польза от такой переформулировки? Вроде бы мы просто повторили то же самое рассуждение. Но в таком виде его может быть легче себе представить и легче применить похожие идеи в других задачах. Мы сейчас приведём такой пример — но задача эта более сложная, и строгое рассуждение в ней совсем не так просто придумать и понять (так что этот раздел можно пропустить, если не получается разобраться).

3.1.6 Разбор контрольной*

Задача 3.17. На контрольной каждый из N школьников решил ровно 2 задачи, а каждую задачу решило ровно 2 школьника. Докажите, что можно так организовать разбор задач, чтобы каждый школьник выступил по одному разу, рассказав одну из решённых им задач, и чтобы каждая задача была рассказана по одному разу.

Рассуждение из предыдущей задачи (всего $2N$ листков с решениями, по два для каждой задачи) показывает, что задач тоже было N. Видно, что по количеству всё сходится: задач столько же, сколько школьников. Но это ещё не значит, конечно, что можно организовать разбор.

Чтобы доказать, что это всегда возможно, нарисуем двудольный граф, о котором мы говорили. Слева и справа в нём по N вершин (слева школьники, справа задачи), и из каждой вершины (и слева, и справа) ведут по два ребра. Вот пример такого графа при $N = 7$.



Слева изображены школьники, справа — задачи. Например, школьник номер 1 решил задачи 2 и 4, а школьник номер 5 решил задачи 3 и 5.

Контрольный вопрос 3.18. Кто из школьников на рисунке решил задачу 4? Можно ли найти двух школьников, которые решили одинаковый набор задач? (Ответ: задачу 4 решили школьники 1 и 3. Школьники 1 и 3 решили одни и те же задачи 2 и 4; школьники 4 и 7 решили одни и те же задачи 1 и 7.)

Задача 3.19. Покажите, как для этого конкретного случая можно организовать разбор (каждый из семи школьников рассказывает одну из решённых им задач, и в итоге все задачи рассказаны).

Решение. Например, годится такой вариант: $1 \to 2$, $2 \to 5$, $3 \to 4$, $4 \to 6$, $5 \to 3$, $6 \to 7$, $7 \to 1$, где запись $i \to j$ означает, что школьник i рассказывает задачу j. □

Задача 3.20. Придумайте другой вариант разбора. Сколько таких вариантов существует?

Ну хорошо, для этого конкретного случая мы нашли искомое — но почему это возможно всегда? Сейчас мы это докажем — правда, не совсем строго. Представим себе рёбра графа в виде верёвок, соединяющих гвозди-вершины. В каждой вершине сходятся (по условию) две верёвки. Всего будет $2N$ верёвок, соединяющих N вершин слева и N вершин справа. Давайте в каждой вершине свяжем концы верёвок друг с другом, отвязав от гвоздя и зацепив полученную петлю за тот же гвоздь. Свободных концов верёвок не остаётся, так что должны получиться (зацепленные друг за друга — это нам не важно) верёвочные кольца, надетые на гвозди.

Задача 3.21. Сколько колец какой длины получится на приведённом нами рисунке с 7 школьниками и 7 задачами?

Решение. Получатся кольца $1 \to 2 \leftarrow 3 \to 4 \leftarrow 1 \to \ldots$, $2 \to 5 \leftarrow 5 \to 3 \leftarrow 6 \to 7 \leftarrow 2 \to \ldots$, $4 \to 1 \leftarrow 7 \to 6 \leftarrow 4 \to \ldots$ из четырёх, шести и четырёх верёвок соответственно. □

Задача 3.22. Почему каждое кольцо обязательно состоит из чётного числа верёвок?

Решение. В кольце чередуются вершины слева (школьники) и справа (задачи), поэтому нужно чётное число рёбер, чтобы вернуться в исходную долю. □

То же рассуждение показывает, что в каждое кольцо входят поровну школьников и задач – они чередуются по схеме

$$Ш_1 - З_1 - Ш_2 - З_2 - Ш_k - З_k - Ш_1 \text{ (замыкая цикл)}$$

(Если в хороводе соседями мальчиков бывают только девочки и наоборот, то в нём поровну тех и других.)

Видно, что если в кольце оставить только каждую вторую связь, то получится как раз возможная схема разбора задач: школьник $Ш_1$ разбирает задачу $З_1$,..., школьник $Ш_k$ разбирает задачу $З_k$.

Контрольный вопрос 3.23. Какой второй вариант организации разбора для того же кольца? (Ответ: первую задачу разбирает второй школьник,..., $(k-1)$-ю задачу разбирает k-й школьник, k-ю задачу разбирает первый школьник.)

Объединяя такие разборы для всех колец, получаем схему разбора всех задач всеми школьниками. В самом деле, каждый школьник входит ровно в одно кольцо — в его вершине сходились две верёвки, которые и вошли в это кольцо. Аналогично для задач.

Задача 3.24. Покажите, что число вариантов организации разбора всегда является степенью двойки. Какое максимальное число вариантов может быть, когда школьников и задач по N? (Ответ: $2^{\lceil N/2 \rceil}$.)

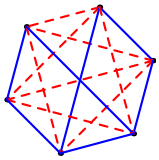
А что было бы, если бы каждый школьник решил ровно три задачи, и каждую задачу бы решило ровно три школьника? Оказывается, по-прежнему гарантирована возможность организовать разбор, но доказать это уже сильно сложнее. Один из способов состоит в использовании так называемой «теоремы Холла о представителях», см. теорему 1.7 в первой главе.

3.1.7 Знакомые и незнакомые

Задача 3.25. В компании из 6 человек некоторые знакомы друг с другом (это симметрично: если А знаком с Б, то и Б знаком с А). Докажите, что можно выбрать трёх человек, которые либо попарно знакомы (все три пары), либо попарно незнакомы (все три пары).

Как изобразить условие этой задачи? Соединим знакомых линией одного цвета (скажем, синей), а незнакомых линией другого (скажем, красной — на рисунке красные линии пунктирные). Получится граф из шести вершин, в котором каждая вершина соединена с каждой ребром. Такой граф называют «полным графом на шести вершинах». Его можно нарисовать в виде шестиугольника, в котором помимо сторон проведены все диагонали, и каждое ребро (рёбра = стороны и диагонали) либо красное, либо синее. (Пересечения диагоналей внутри шестиугольника не считаются вершинами, как обычно.)

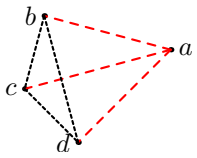
Цвета рёбер изображают ситуацию в задаче — а доказать надо, если говорить в этих терминах, что в таком графе обязательно найдётся либо целиком красный треугольник, либо целиком синий треугольник.



Контрольный вопрос 3.26. Найдите на рисунке одноцветный треугольник. Сколько их? (Ответ: синих треугольников нет, а красных треугольников треугольников два.)

Контрольный вопрос 3.27. Измените цвет одного ребра так, чтобы остался только один красный треугольник. Можно ли избежать при этом появления синего треугольника? (Ответ: можно перекрасить любое из шести рёбер двух красных треугольников, но тогда обязательно появится синий треугольник, иногда даже два.)

Осталось решить нашу задачу и доказать, что одноцветный треугольник обязательно найдётся. Как это сделать? Выберем произвольную вершину a. Она соединена с пятью другими вершинами пятью линиями. Эти линии двух цветов, поэтому есть как минимум три линии одного цвета (если линий каждого цвета было бы только две, то всего было бы 4, а не 5). Посмотрим на это подробнее: пусть из вершины a ведут три одинаковых (скажем, красных) ребра в какие-то три вершины b, c и d.



Понятно, как теперь завершить решение задачи? Достаточно посмотреть на цвета трёх рёбер в треугольнике b–c–d. Если среди них есть хотя бы одно красное, то оно образует красный треугольник с вершиной a. Если же все три ребра синие, так вот он и есть — синий треугольник. Утверждение доказано.

Что мы выиграли, говоря о красных и синих рёбрах вместо знакомых и незнакомых? Всего лишь одну (но важную) вещь: рассуждение теперь симметрично относительно замены цветов, что в первоначальной формулировке совсем не так явно видно (и потому рассуждение нужно было бы повторять дважды: когда у человека есть трое знакомых среди оставшихся и когда есть трое незнакомых).

Контрольный вопрос 3.28. Повторите рассуждение, избегая упоминаний рёбер и их цветов, а говоря только о знакомых и незнакомых.

Приведённое утверждение — лишь одно из серии утверждений, который называют утверждениями «рамсеевского типа» (в честь Ф. П. Рамсея, 1903–1930). Среди них есть и сложные теоремы, и открытые проблемы, но мы приведём только несколько (не самых сложных) примеров.

Задача 3.29. Остаётся ли утверждение задачи верным, если в группе не шесть человек, а только пять?

Решение. Нет. Можно, например, нарисовать синий пятиугольник с красными диагоналями: ни сам пятиугольник, ни красная звезда внутри него не содержат треугольников. □

Задача 3.30*. Докажите, что в группе из десяти человек можно выбрать либо четверых попарно знакомых (всего шесть знакомств между ними), либо трёх попарно незнакомых.

В терминах графов: в полном графе из десяти вершин с красными и синими рёбрами можно найти либо полный синий граф с четырьмя вершинами (=синий четырёхугольник с диагоналями), либо полный красный граф с тремя вершинами (красный треугольник).

Решение. Выберем одну вершину a и посмотрим на 9 рёбер, из неё выходящих. Там есть либо 6 синих, либо 4 красных, иначе всего было бы только $5 + 3 = 8$ вершин вместо 9. (Теперь у нас уже цвета не симметричны, поэтому и границы нужны разные, но схема рассуждения та же.)

Пусть есть 6 синих. Посмотрим (временно) только на них и соединяющие их рёбра. Мы уже знаем, что там можно найти либо синий, либо красный треугольник. И то, и другое нам годится: красный треугольник сам по себе хорош, а синий треугольник вместе с ведущими из a синими рёбрами даёт синий четырёхугольник с диагоналями.

Теперь второй случай: пусть есть 4 красных. Если эти четыре точки соединяет хотя бы одно красное ребро, то есть красный треугольник, если же нет — то есть синий четырёхугольник с диагоналями. Задача решена. □

Задача 3.31*. Пусть $R(m, n)$ — наименьшее число людей в группе, которое гарантирует наличие m попарно знакомых или n попарно незнакомых. Эти числа определены при целых положительных m, n, при этом считаем $R(m, 1) = R(1, n) = 1$ (любой человек считается за одноэлементную группу попарно знакомых, а также попарно незнакомых). Рассмотренные нами задачи показывают, что $R(3, 3) = 6$ и что $R(4, 3) \leqslant 10$ (предыдущая задача).

Чему равно $R(2, 2)$? (Ответ: 2)

Чему равно $R(2, 3)$? (Ответ: 3)

Чему равно $R(2, m)$? (Ответ: m).

Докажите, что $R(m, n) \leqslant R(m - 1, n) + R(m, n - 1)$.

3.2 Неориентированные графы

Пора уже дать точное определения графа, не ограничиваясь примерами и неформальными разговорами о картинках. Это особенно важно, потому что в разных задачах приходится использовать графы разных видов (с разными определениями), и надо внимательно следить за тем, какое именно определение мы используем.

Начнём с того, что называется неориентированными графами. (Более точное название было бы «неориентированные графы без петель и кратных рёбер», но мы будем опускать это уточнение.)

3.2.1 Определение

Чтобы задать *неориентированный граф*, нужно:

- Указать конечное множество V, элементы которого называются *вершинами* графа. (Другими словами, нужно перечислить вершины графа, при этом порядок перечисления не важен.)
- Для каждой пары различных вершин v и v' из V указать, соединены они ребром или нет.

Например, граф из раздела 3.1.1 имеет пять вершин A, B, C, D, E. Это записывают так: $V = \{A, B, C, D, E\}$ (порядок перечисления мог бы быть и другим). Рёбрами этого графа являются пары вершин (a, b), (a, c), (a, e), (b, d), (c, d), (c, e). Здесь речь идёт о *неупорядоченных* парах, то есть мы не различаем, скажем, (a, b) и (b, a). Обозначая через E множество рёбер, можно написать, что

$$E = \{(a, b), (a, c), (a, e), (b, d), (c, d), (c, e)\}.$$

И здесь порядок перечисления рёбер (и порядок указания концов рёбер) не имеет значения.

Традиция использовать букву V для вершин и E для рёбер происходит от английских слов «vertex» (вершина) и «edge» (ребро).

При компьютерном представлении графа с n вершинами можно как-то занумеровать его вершины числами от 0 до $n - 1$, а биты, кодирующие наличие рёбер, записать в таблицу $edge[i, j : 0..n - 1]$: `Boolean`. Если $edge[i, j]$ истинно, то в графе есть ребро между вершинами i и j. Эта таблица должна быть симметричной: $edge[i, j] = edge[j, i]$, поскольку тут речь идёт об одном и том же ребре.[2] Ещё можно заметить, что значения $edge[i, i]$ не соответствуют никаким рёбрам, так как мы не разрешаем «петли» — рёбра из вершины в саму себя. На практике бывает удобно считать, что эти места таблицы содержат значение `false`.[3] Слова о том, что «в графе нет кратных рёбер» означают, что не может быть нескольких рёбер, соединяющих одну и ту же пару вершин.[4]

Задача 3.32. При больших n имеет смысл хранить информацию о рёбрах графа более экономно. Предложите такой способ и оцените, во сколько раз тут выигрыш.

[2] Напомним, что в этом разделе мы рассматриваем неориентированные графы. Для ориентированных графов всё не так, см. ниже раздел 3.3.

[3] Но иногда удобнее другое соглашение: все значения $edge[i, i]$ равны `true`. Так бывает, когда мы интересуемся достижимостью одних вершин из других.

[4] Можно было бы это разрешить, и тогда в таблице вместо булевых значений надо было бы хранить неотрицательные целые числа, «кратности» рёбер. Иногда это полезно, но мы такие графы рассматривать не будем.

(Указание. Если хранить только элементы таблицы над диагональю, скажем, в одномерном массиве типа `Boolean`, можно вдвое уменьшить расход памяти. Правда, чтение тогда будет более дорогой операцией.)

На математическом языке говорят, что *графом* называется пара (V, E), где V — некоторое конечное множество, элементы которого называются *вершинами* графа, а E — конечное множество, элементы которого называются *рёбрами* и являются неупорядоченными парами вершин (двухэлементными подмножествами множества V).

3.2.2 Соседи. Степени вершин

Вершины $v, v' \in V$ называются *соседями*, если в графе есть соединяющее их ребро (другими словами, если $edge[v, v'] = \texttt{true}$). Напомним, что мы не разрешаем петли, поэтому сама вершина *не* является своим соседом.

Число соседей вершины (другими словами, число выходящих из неё рёбер) называется *степенью* этой вершины. В нашем компьютерном представлении, если принять $\texttt{false} = 0$ и $\texttt{true} = 1$ и складывать значения как целые числа, степень вершины v можно записать как $\sum_{w \in V} edge[v, w]$. Здесь существенно, что таблица симметрична и что диагональ $edge[v, v]$ мы считаем нулевой.

Теперь мы уже можем точно сформулировать утверждение о сумме степеней.

Теорема 3.1. *В неориентированном графе сумма степеней всех вершин равна удвоенному числу рёбер.*

(Отсюда следует, в частности, что эта сумма чётна — и потому количество вершин нечётной степени чётно, как мы видели.)

Доказательство. Будем складывать все элементы таблицы $edge[i, j]$. В строке i сумма элементов таблицы равна степени вершины i. (Напомним, что мы считаем $edge[i, i] = 0$.) Значит, сумма всех чисел в таблице равна сумме степеней вершин. С другой стороны, каждое ребро графа вносит вклад 2: ребро между i и j встречается как $edge[i, j]$ и $edge[j, i]$. Значит, сумма всех чисел в таблице равна удвоенному числу рёбер. □

Контрольный вопрос 3.33. Чему равна сумма чисел в *столбце* таблицы $edge$? (Ответ: тоже степени соответствующей вершины, таблица симметрична.)

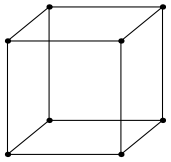
Пример 3.34. *Граф–путь* P_n имеет n вершин $v_1, \ldots v_n$. Рёбрами связаны пары вершин v_i и v_{i+1} $(1 \leqslant n - 1)$. Таким образом, в графе–пути $n - 1$ ребро. Говорят, что длина пути равна $n - 1$.

Контрольный вопрос 3.35. Какие степени вершин в графе P_n?

Пример 3.36. *Граф–цикл* C_n имеет n вершин $v_1, \ldots v_n$. Рёбрами связаны пары вершин v_i и v_{i+1} $(1 \leqslant n - 1)$ и пара вершин v_n и v_1. Таким образом, в графе–цикле n рёбер. Говорят, что длина цикла равна n.

Задача 3.37. Нарисуйте граф из 8 вершин, в котором степень каждой вершины равна 3.

Решение. Степень всех вершин одинакова, поэтому естественно стараться выбрать граф каким-нибудь симметричным. Можно сообразить, что годится каркас куба: вершины графа — это вершины куба, а рёбра графа — рёбра куба (отрезки, соединяющие соседние вершины).



На этом рисунке пересечения рёбер вне вершин куба, как всегда, не считаются. Но этих пересечений можно было бы и избежать. □

Можно рассмотреть аналогичные графы («булевы кубы») больших размерностей.

Вершинами *булева куба* размерности n являются битовые строки длины n, а соседними считаются вершины, отличающиеся ровно в одной позиции (один из битов изменяется).

Контрольный вопрос 3.38. Сколько вершин и рёбер в этом графе? Сколько соседей у каждой вершины? (Ответ: 2^n вершин, у каждой n соседей, всего рёбер $2^n n/2$.)

В *полном графе с n вершинами* любая пара вершин соединена ребром.

Контрольный вопрос 3.39. Сколько рёбер в полном графе? (Ответ: степень каждой вершины $n-1$, всего рёбер $n(n-1)/2$.)

Контрольный вопрос 3.40. Приведите пример графа с n вершинами и n рёбрами, в котором каждая вершина имеет степень 2. (Ответ: вершины и стороны графа-цикла.)

Задача 3.41*. Как нарисовать куб на плоскости, чтобы рёбра его не пересекались? Почему это можно сделать для любого выпуклого многогранника?

Задача 3.42*. Нарисуйте граф из 12 вершин, каждая из которых имеет степень 5. (Указание. Какие вы знаете правильные многогранники?)

Задача 3.43*. В предыдущих задачах, имея выпуклый многогранник, мы строили граф, в котором вершинами и рёбрами были как раз вершины и рёбра многогранника. Но можно построить и граф, в котором вершинам будут соответствовать *грани* многогранника, а рёбра — по-прежнему рёбра многогранника. Как?

Задача 3.44*. Бывает ли выпуклый многогранник, в котором все его 15 граней — треугольники?

Задача 3.45. Нарисуйте граф из 9 вершин, в котором степень каждой вершины равна 3.

Решение. Тут есть подвох: такого графа не существует. В самом деле, по доказанному в нём должно было бы быть $3 \times 9/2 = 13{,}5$ рёбер, а их может быть лишь целое число. □

На кружках для школьников эту задачу часто излагают так: в группе из 9 человек каждый человек имеет трёх друзей. Покажите, что либо кто-то учитывает самого себя в числе друзей, либо дружба не всегда взаимна.

Задача 3.46. При каких n существует граф из n вершин, в котором каждая вершина имеет степень 3?

Решение. Мы уже знаем, что при нечётных n это невозможно. Кроме того, это невозможно и при $n = 2$ (почему)? При чётных n, начиная с 4, такой граф можно построить. Достаточно расположить вершины по кругу в вершинах правильного многоугольника, соединив каждую вершину с двумя соседними по кругу, а также с диаметрально противоположной, получится три ребра. □

Контрольный вопрос 3.47. Где мы использовали, что n чётно?

Задача 3.48. Докажите, что в любом графе с более чем одной вершиной есть две вершины одинаковой степени.

Решение. Пусть в графе N вершин. Их степени могут быть числами от 0 до $N-1$. Таких чисел как раз N. Значит, если степени всех вершин разные, то использованы все варианты от 0 до $N-1$. Но степени 0 и $N-1$ не могут встречаться одновременно: первое означает, что из какой-то вершины не выходит ни одного ребра, второе — что какая-то другая вершина соединена рёбрами во всеми остальными. Но тогда спрашивается, соединены ли эти две вершины ребром? □

Контрольный вопрос 3.49. Где в этом решении используется, что $N > 1$? (Ответ: в противном случае получаются не две вершины, а одна.)

Задача 3.50. В любой момент футбольного турнира, проходящего в один круг — каждая команда играет с каждой по одному разу — есть две команды, сыгравшие одинаковое число матчей.

Решение. В каждый момент турнира можно составить граф, вершинами которого являются команды, а рёбрами — уже сыгранные матчи. Степень вершины (команды) тогда равна числу сыгранных ей матчей, и можно применить предыдущую задачу. □

Задача 3.51. Где в этом рассуждение использовано, что турнир в один круг? Верно ли то же самое утверждение — в любой момент турнира найдутся две команды, сыгравшие одинаковое число матчей, — для турнира в два круга? (Каждая команда играет с каждой по два раза равно, расписание турнира произвольное.)

Задача 3.52. Докажите, что в выпуклом многограннике всегда найдутся две многоугольные грани с одинаковым числом сторон.

Задача 3.53*. Как надо было бы определить граф с петлями и кратными рёбрами, и степени его вершин, чтобы сумма степеней вершин по-прежнему равнялась удвоенному числу рёбер?

3.2.3 Связные компоненты

Вспомним задачу об авиарейсах. В ней вершины графа — это города (точнее было бы говорить об аэропортах), а ребро между вершинами v и v' означает наличие авиасообщения между v и v' (двустороннего). В таком графе про любые два города возникает вопрос: можно ли добраться из одного в другой, пусть с пересадками. Все города (вершины) разбиваются на «связные компоненты»: если два города в одной связной компоненте, то добраться можно, а если в разных, то нельзя.[5]

Теперь хотелось бы определить понятие связности графа (и связной компоненты) более формально. Ничего сложного, неожиданного или красивого при этом не будет, но этим формальным языком надо уметь пользоваться.

Формальное определение

Будем называть *путём* в графе последовательность вершин $v_1, v_2, v_3, \ldots, v_k$, в которой стоящие рядом члены (вершины v_i и v_{i+1} при всех i) соединены ребром. Вершина v_1 называется *началом* пути, вершина v_k — его *концом*. *Длиной* пути будем называть число рёбер, то есть $k-1$. (Предупреждение: длина пути на единицу меньше числа вершин в нём!). Мы будем разрешать также и пути длины 0, то есть последовательности из одной вершины. У такого пути начало совпадает с концом. Рёбер в таком пути нет, но вершина (одна) есть.

Вершины v и v' называются *связанными*, если существует путь с началом в v и концом v'. Граф называется *связным*, если любые две его вершины связаны.

Контрольный вопрос 3.54. Связана ли вершина с самой собой? (Ответ: да, поскольку мы разрешаем пути длины 0.)

Бдительные читатели, наверно, уже заметили, что наша терминология (связанные вершины) симметрична: мы неявно подразумеваем, что *если есть путь с началом v и концом v', то есть и путь с началом v' и концом v*. Почему это действительно так? Достаточно «обратить» путь, то есть записать его вершины в обратном порядке. Обращение пути $v_1, \ldots, v_k$ даёт путь $v_k, \ldots, v_1$. Это по-прежнему путь (граф неориентированный), и начало и конец в нём поменялись местами.

Ещё одно важное свойство называется *транзитивностью* отношения связанности: *если вершина u связана с вершиной v, а вершина v связана с вершиной w, то вершина u связана с вершиной w*. Свойство это выглядит очевидным (если из

[5] Похожие вопросы возникают и для рельсового транспорта: например, в московской трамвайной сети приходится перевозить трамваи из одной связной компоненты в другую автотранспортом.

u можно добраться в v, пусть даже с пересадками, и из v можно добраться в w, то из u можно добраться в w), но если хотеть изложить его доказательство более формально, это можно сделать так.

По предположению вершина u связана с v: есть путь $s_1, \ldots, s_k$, в котором $s_1 = u$, а $s_k = v$. Есть и путь $t_1, \ldots, t_l$, в котором $t_1 = v$, а $t_l = w$. Их можно соединить в один путь

$$s_1, s_2, \ldots, s_{k-1}, s_k = t_1, t_2, \ldots, t_{l-1}, t_l,$$

поскольку конец первого пути совпадает с началом второго. Ясно, что это снова путь (любая пара рядом стоящих членов была в одном из путей, так что соединена ребром по предположению). Начало этого пути $s_1 = u$, а конец $t_l = w$, так что вершины u и w связаны.

Теперь можно точно сформулировать обсуждавшееся свойство графов:

Теорема 3.2. *Вершины неориентированного графа можно разбить на непересекающиеся группы, называемые* связными компонентами*, при этом:*

- *каждая вершина графа попадает ровно в одну группу;*
- *любые две вершины из одной группы связаны;*
- *любые две вершины из двух разных групп не связаны.*

Такая компонента может быть одна — это значит, что любые две вершины связаны (и, как мы уже говорили, в этом случае граф называют связным).

Доказательство. Для каждой вершины v составим группу из всех связанных с ней вершин. Обозначим эту группу $C(v)$ — от слова «connected» (*англ.* связаны). Наблюдение: для любых двух вершин v и w группы $C(v)$ и $C(w)$ либо совпадают (содержат одни и те же вершины), либо не пересекаются. В самом деле, если v и w связаны, то по транзитивности всякая вершина, связанная с одной, связана и с другой, так что $C(v) = C(w)$. Если же v и w не связаны, то докажем, что $C(v)$ и $C(w)$ не пересекаются (не могут иметь общих вершин). В самом деле, если u было бы общей вершиной ($u \in C(v) \cap C(w)$), то v и u связаны (по определению $C(w)$), а также w и u связаны (по определению $C(w)$). Транзитивность гарантирует, что v связано с w, вопреки предположению.

Мы проверили, что получилось разбиение на непересекающиеся группы. Каждая вершина v попадает в группу $C(v)$. Вершины из одной группы связаны: если $v, w \in C(u)$, то v и w связаны с u, и по транзитивности v и w связаны друг с другом. (Мы повторяем уже использованное рассуждение). Вершины из разных групп не связаны: если вершина $p \in C(u)$ связана с вершиной $q \in C(v)$, то получается цепочка связанных вершин: u с p, затем p с q, наконец, q с v. Дважды применяя транзитивность, заключаем, что u и v связаны и потому $C(u)$ и $C(v)$ — не разные группы, а одна и та же. Теорема доказана. □

Контрольный вопрос 3.55. К каким тройкам вершин мы применяли транзитивность в первый и второй раз? (Ответ: к u, p, q и к u, q, v.)

Подобные формальные рассуждения, конечно, удручают — мы долго и скучно доказываем нечто совершенно понятное (по крайней мере в ситуации с рейсами). Тем не менее это надо уметь — в более сложных случаях это позволит избежать ошибок.

Задача 3.56*. Рассмотрим граф, вершинами которого являются трёхбуквенные слова русского языка, а рёбра соединяют два слова, которые отличаются в одной букве (получаются одно из другого заменой одной буквы). Находятся ли слова ТОК и СЫЧ в одной связной компоненте?

Нижняя оценка для числа связных компонент

Теперь мы всё-таки сформулируем и докажем какой-то менее очевидный (хотя и простой) результат.

Теорема 3.3. *Если граф имеет V вершин и E рёбер, то в нём не меньше $V - E$ связных компонент. В частности, если граф связен, то $E \geqslant V - 1$.*

(Часто буквами V и E обозначают не только множества вершин и рёбер, но и количества тех и других. Мы тоже так будем делать, если нет оснований опасаться путаницы.)

Контрольный вопрос 3.57. Что утверждает теорема при $E = 0$? (Ответ: что в графе без рёбер не меньше V связных компонент — а именно, каждая вершина образует свою компоненту.)

Контрольный вопрос 3.58. Почему второе утверждение следует из первого (и мы имеем право сказать «в частности»)? (Ответ: в связном графе одна компонента, и теорема даёт $V - E \leqslant 1$, то есть $V - 1 \leqslant E$.)

Доказательство теоремы 3.3. Коротко говоря, добавление одного ребра уменьшает число связных компонент, первоначально равное V, не более чем на 1 (уменьшение происходит, если ребро соединяет разные компоненты).

Более формально это доказательство можно оформить как индукцию по числу рёбер.

Базис индукции: в графе нет рёбер, и теорема утверждает очевидную вещь: в таком графе число связных компонент не меньше числа вершин (на самом деле тут равенство).

Шаг индукции. Пусть имеется граф G с V вершинами и $E > 0$ рёбрами. Временно удалим одно ребро. Получится граф G' с V вершинами и $E - 1$ рёбрами. В нём, по предположению индукции, не менее $V - E + 1$ компонент. Нам надо доказать, что возвращение удалённого ребра уменьшает число компонент не более чем на 1. Пусть это удалённое ребро e соединяло вершины v и w.

Случай 1. В графе G' вершины v и w лежали в одной компоненте. Тогда в G будут те же компоненты, что и в G'. В самом деле, любой путь в G' будет путём в G, и остаётся проверить, что если вершины связаны (соединены путём) в G, то они

связаны и в G'. Соединяющий их путь может не использовать ребра e, и тогда он является путём в G'. Если же он использует ребро e, то вместо этого ребра нужно вставить путь из v в w, который по предположению имелся в G'.

Случай 2. В графе G' вершины v и w лежали в разных компонентах. Теперь (в G) они соединены ребром, то есть попали в одну компоненту. Покажем, что все остальные компоненты, кроме этих двух, остались без изменений. В самом деле, если вершина z в графе G' лежала в какой-то третьей компоненте, то ни v, ни w из ней доступны не были — а значит, появление ребра v–w не увеличило множество доступных из z вершин (прежде чем в первый раз воспользоваться ребром v–w, нам надо дойти либо в v, либо в w, а это невозможно).

На этом доказательство шага индукции (и всей теоремы) завершается. □

Применения нижней оценки*

Что даёт нижняя оценка на число связных компонент? Можно, конечно, сказать, что для связи между n городами нужно как минимум $n-1$ авиарейсов — это просто бытовая переформулировка доказанного. Но есть и более интересные применения. Сейчас мы рассмотрим два таких примера.

Пример 1. *Самый тяжёлый камень.*[6] Есть n различных по весу камней. Эксперт, знающий веса камней, выбрал самый тяжёлый камень и хочет доказать суду, что он действительно самый тяжёлый. Для этого он в присутствии суда выполняет взвешивания на чашечных весах без гирь. На каждую чашку этих весов помещается по камню, и весы показывают, какой камень тяжелее. Какое минимальное число взвешиваний придётся сделать эксперту?

Ясное дело, что достаточно $n-1$ взвешиваний: эксперт может публично сравнить известный ему самый тяжёлый камень со всеми остальными — их как раз ровно $n-1$. Это не единственный возможный способ: например, эксперт может заранее упорядочить камни по весам $a_1 < a_2 < \ldots < a_n$ и продемонстрировать взвешивания $a_1 < a_2$, $a_2 < a_3$,…, $a_{n-1} < a_n$. После этого порядок камней станет ясен не только эксперту, но и суду. Второй способ требует тоже $n-1$ сравнений. Но нельзя ли обойтись меньшим числом сравнений?

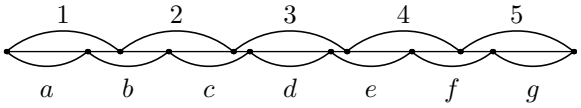
Наша оценка на число связных компонент позволяет доказать, что нельзя. В самом деле, представим себе неориентированный граф, где вершины — это камни, а рёбра — это выполненные экспертом сравнения камней. Заметим, что мы даже не интересуемся тем, каковы были результаты сравнения, а просто фиксируем факт его проведения. Теперь ключевое соображение: если сравнений было меньше $n-1$, то этот граф не связен, в нём несколько связных компонент. В этом случае по результатам сравнений нельзя судить о том, какой камень самый тяжёлый. Понятно ли, почему?

В самом деле, поскольку взвешивания происходят только внутри связных компонент, то изменение весов всех камней внутри одной компоненты на одно и то же число (добавление константы) не изменит результатов взвешиваний ни в этой ком-

[6] См. также более формальное обсуждение этой задачи и подобных ей в гл. 12.

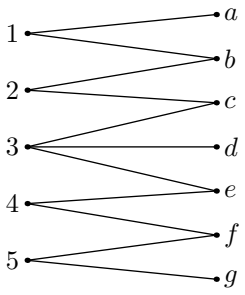
поненте, ни (тем более) в других. Значит, суд увидит то же самое, что было бы и без изменения. А между тем такое изменение может увеличить вес камней в любой из компонент настолько, что самый тяжёлый камень окажется там — так что результаты показанных суду взвешиваний ничего не доказывают.

Пример 2. *Деление торта.* Допустим, что нам надо заранее разрезать торт на несколько кусков, готовясь к приходу гостей — так, чтобы его можно было раздать поровну m людям, а также чтобы его можно было (с теми же разрезами, только перегруппировав куски) раздать поровну n людям. Какое минимальное число кусков понадобится? Пусть, скажем, $m = 5$ и $n = 7$; сколько надо кусков, чтобы можно было раздать торт поровну и пяти, и семи людям? Наивный способ состоит в том, чтобы разделить торт на части по 1/35 и потом группировать куски по 5 и по 7. Но легко понять, что это не оптимально (с точки зрения числа кусков) — представим себе торт отрезком и наметим точки разреза на 5 и на 7 равных частей. В первом случае будет 4 точки разреза, во втором 6, всего 10 точек разреза, то есть 11 кусков. Сделав все эти разрезы заранее, мы решим задачу, то есть 11 кусков достаточно. (В общем случае достаточно $m + n - 1$ кусков.)



Как доказать, что это оптимальный способ, то есть что 10 кусков недостаточно? Можно разбирать разные варианты и случаи и это установить, но есть такое рассуждение (которое легко обобщается на произвольные взаимно простые m и n).

Нарисуем схему раздачи в виде графа. Слева изобразим 5 гостей одного варианта, назовём их 1, 2, 3, 4, 5. Справа изобразим 7 гостей другого, назовём их a, b, c, d, e, f. Куски изобразим рёбрами, соединяющими тех, кому они попали при том и другом варианте. Получится граф, рёбра которого соответствуют кускам: ребро соединяет тех людей (в первом и втором варианте), которым соответствующий кусок достанется.[7] Вот граф, изображающий ту же самую раздачу, что на предыдущем рисунке, но, разумеется, возможны и другие варианты раздач.



[7]Тут на самом деле есть проблема: вообще говоря, могут быть два куска, которые попадают одним и тем же людям в обоих вариантах, чего в графе быть не может. Но в этом случае эти два куска можно объединить в один, уменьшив число кусков.

Нам надо доказать, что в этом графе не менее 11 рёбер. Заметим, что в нём 12 вершин. Значит, если рёбер меньше 11, то граф будет несвязен (вот где нужна предыдущая теорема). Связная компонента состоит из некоторых вершин; разобьём и рёбра в соответствии с тем, в какую компоненту они попали (концы любого ребра попадают в одну компоненту, так как они связаны). Таким образом, куски разбиваются на несколько групп. Спросим себя, какой может быть суммарный вес кусков в одной из групп. Все эти куски в каждом из вариантов идут каким-то гостям, и больше эти гости ничего не получают (иначе эти новые куски вошли бы в ту же связную компоненту). Значит, общий вес кусков в группе должен быть кратен $1/5$ торта и $1/7$ торта одновременно, а это невозможно, кроме того случая, когда общий вес равен 1, то есть группа только одна. Получаем искомое противоречие.

Нахождение связных компонент: on-line алгоритм

В качестве отступления разберём красивый алгоритм нахождения связных компонент в графе. Представим себе граф с n вершинами, пронумерованными от 0 до $n-1$. Изначально в этом графе нет рёбер. Их постепенно добавляют, сообщая нам, какие две вершины i и j связывает очередное ребро. Помимо этого, нас время от времени спрашивают, связана ли такая-то вершина с такой-то (находятся ли они в одной компоненте на данный момент). В каком виде нам надо хранить информацию о текущем состоянии графа, чтобы обрабатывать запросы этих двух видов по возможности быстро? (Попытайтесь сами придумать какой-то способ, прежде чем читать дальнейшее.)

Другими словами, мы должны написать модуль, реализующий такие функции:

- *Initialize*(n : `integer`): создать граф с n вершинами без рёбер;
- *AddEdge*(i, j : `integer`): добавить ребро, соединяющее вершины i и j.
- *Connected*(i, j : `integer`) : `Boolean`: связаны ли вершины i и j в текущем графе (находятся ли они в одной связной компоненте).

Сначала реализуем совсем простую идею. Вспомним доказательство нижней оценки для числа компонент. Нам надо при каждом добавлении ребра узнавать, лежат ли концы его ребра в одной компоненте. Если лежат, то добавление этого ребра ничего не меняет; если нет, то надо слить две компоненты в одну.

Договоримся, что в каждый момент в каждой компоненте выбрана одна из вершин, которую мы будем называть *представителем* этой компоненты. В остальных вершинах компоненты хранится ссылка на этот самый представитель — или, по крайней мере, ссылка на вершину, где есть ссылка на этот представитель, или на вершину, где есть ссылка на вершину, где есть ссылка на него, и т.д. Более точно, мы заводим массив $ref[0..n-1]$ с таким свойством: если i — представитель какой-то компоненты, то $ref[i] = i$, а если нет, то цепочка

$$i, ref[i], ref[ref[i]], \ldots$$

рано или поздно стабилизируется на представителе связной компоненты, в которую попадает i.

Этот инвариант несложно поддерживать и использовать. При инициализации мы полагаем $ref[i] = i$: каждая вершина образует отдельную компоненту и является её представителем.

В любой момент можно найти представитель данной вершины, пройдя по цепочке:

Representative (i : `integer`) :

`while` $ref[i] \neq i$: $i \leftarrow ref[i]$

`return` i

Попадание в одну компоненту означает совпадение представителей:

Connected(i, j) : `return` (*Representative*$[i]$ = *Representative*$[j]$)

Соединение двух компонент в одну теперь можно реализовать так:

AddEdge(i, j) :

`if not` *Connected*(i, j) : $ref[$*Representative*$(i)] \leftarrow$ *Representative*(j)

Представителем объединённой компоненты становится представитель второй из объединяемых, а в цепочках для первой компоненты добавляется ещё один член. (Можно было бы сделать и наоборот, взяв представителя из первой компоненты.)

Этот алгоритм можно сильно ускорить с помощью двух оптимизаций. Первая состоит в том, что, проходя по цепочке, её можно оптимизировать, записав везде в *ref* соответствующего представителя. Это проще написать в рекурсивной форме:

Representative (i : `integer`) :

`if` $ref[i] = i$: `return` i;

`if` $ref[i] \neq i$: $rep \leftarrow$ *Representative*$(ref[i])$; $ref[i] \leftarrow rep$; `return` rep

Вторая оптимизация, про которую мы подробно говорить не будем, состоит в том, что мы стараемся использовать в качестве представителя объединения «более авторитетного» из двух представителей — того, у кого выше специальный параметр, называемый «рангом»; при таком присоединении ранг этого более авторитетного представителя не меняется (а ранг другого более не имеет значения). При равенстве рангов мы берём любого из двух, и его ранг увеличиваем на 1.

Подробнее про этот алгоритм и оценку времени его работы можно прочесть в главе 22 книги «Построение и анализ алгоритмов» [11] (Кормен, Лейзерсон, Ривест, М.: МЦНМО, 2000), или в главе 5 книги «Алгоритмы» [8] (Дасгупта, Пападимитриу, Вазирани, М.: МЦНМО, 2014). Можно также найти много информации в интернете по ключевым словам “disjoint-set data structure” или “Union–Find data structure”.

3.2.4 Расстояния. Простые пути

Обычно, покупая билет на самолёт, мы при прочих равных стараемся выбрать маршрут с наименьшим числом пересадок. Аналогичным образом, имея вершины u и v в графе, можно искать путь минимальной длины с началом в u и концом в v. Длина этого пути называется *расстоянием* от u до v в графе. Будем обозначать его $d(u, v)$.

Это определение расстояния имеет смысл, если вообще есть путь из u в v, то есть если u и v связаны. Если же они лежат в разных компонентах, то удобно считать $d(u, v) = +\infty$.

Педантичные читатели заметили бы, что мы не обосновали, почему кратчайший путь существует (если вершины u и v связаны). Педантичные авторы ответили бы, что расстояния представляют собой натуральные числа, и любое множество натуральных чисел имеет наименьший элемент.

Контрольный вопрос 3.59. Чему равно расстояние от u до u? (Ответ: нулю — мы считаем, что пути, состоящие из единственной вершины, имеют нулевую длину.)

Контрольный вопрос 3.60. Для каких пар вершин u, v расстояние $d(u, v)$ равно 1? (Ответ: для вершин, соединённых ребром.)

Контрольный вопрос 3.61. Закончите предложение: «$d(u, v) \geqslant k$ означает, что в графе нет пути...». (Ответ: «...из u в v, имеющего длину меньше k».)

Задача 3.62. Докажите, что $d(u, v) = d(v, u)$.

Решение. Обращая путь из u в v, получим путь той же длины из v в u. Поэтому $d(v, u) \leqslant d(u, v)$. Меняя обозначения, замечаем, что $d(u, v) \leqslant d(v, u)$, поэтому $d(u, v) = d(v, u)$. □

Задача 3.63. Докажите *неравенство треугольника* для расстояний в графе:

$$d(u, w) \leqslant d(u, v) + d(v, u).$$

(Название объясняется тем, что для точек плоскости аналогичное неравенство говорит, что сторона uw треугольника uvw не превосходит суммы двух его других сторон uv и vw.)

Решение. Пусть $d(u, v) = k$ и $d(v, w) = l$. Тогда существует путь из u в v длины k и путь из v в w длины l. Соединяя их, получаем, что существует путь длины $k + l$ из u в w, так что $d(u, v) \leqslant k + l = d(u, v) + d(v, w)$. □

Контрольный вопрос 3.64. Можно ли утверждать в последнем рассуждении, что $d(u, w) = k+l$? (Ответ: нет, построенный путь из u в w не обязан быть кратчайшим.)

Задача 3.65*. Докажите, что для любых трёх вершин u, v, w выполнено неравенство $|d(u, v) - d(u, w)| \leqslant d(v, w)$.

Может быть, вы слышали от знакомых математиков про «число Эрдёша» — названное по имени американского математика венгерского происхождения, доказавшего много разных вещей, в том числе про графы, и имевшего много работ в соавторстве. Рассмотрим граф, вершинами которого являются авторы математических работ. Авторы u и v соединены в этом графе ребром, если у них есть совместная публикация (работа, где оба они являются авторами). Получается некоторый большой (но конечный) граф, и Эрдёш является одной из его вершин. Расстояние до Эрдёша в этом графе (длина кратчайшего пути, то есть кратчайшей цепочки, в которой рядом стоят соавторы) называют «числом Эрдёша» автора.

Другой похожий пример — «теория шести рукопожатий». В ней говорится о графе, вершинами которого являются живущие сейчас люди, а ребро $u - v$ означает, что u и v знакомы (= пожимали друг другу руки). Теория эта предполагает, что расстояние между любыми двумя вершинами в этом графе не превышает 6. Вряд ли она верна уж совсем буквально (наверно, бывают отшельники, ни с кем не знакомые?), но говорят, что это довольно близко к действительности.

Вообще для связного графа определяют его *диаметр* как наибольшее расстояние между какими-то его двумя вершинами. Теория шести рукопожатий утверждает, таким образом, что диаметр графа знакомств не превышает 6.

Все эти разговоры не выглядят серьёзно (и не претендуют на серьёзность), но статистический анализ больших графов, возникающих в жизни (скажем, графов знакомств в социальных сетях) — это популярная в наше время тема для статистических исследований и построения каких-то вероятностных моделей формирования такого рода графов.

Алгоритмы вычисления расстояний (и отыскания кратчайших путей) в графах — важный раздел алгоритмической теории графов; важно это и на практике, где обычно приписывают каждому ребру графа некоторую длину и длиной пути считают не число рёбер, а сумму длин этих рёбер. С такими алгоритмами сталкивались все, кто видели работу навигатора для автомобиля, определяющего кратчайший маршрут к цели. Можно только поразиться, до чего дошла наука о быстрых алгоритмах на графах — сравнительно слабые процессоры в смартфонах без особенного труда отыскивают кратчайшие пути на картах с огромным числом населённых пунктов и дорог. Но это отдельный разговор.

Понятие расстояния позволяет легко доказать следующее утверждение.

Теорема 3.4. *Если вершины u и v в графе связаны, то существует соединяющий их путь, в котором все вершины различны (нет повторяющихся вершин).*

Доказательство. Рассмотрим кратчайший путь из u в v. Если бы в него дважды входила некоторая вершина w, то участок между этими вхождениями можно было бы выбросить, и получился бы более короткий путь из u в v, вопреки предположению. □

Пути без повторяющихся вершин называют иногда *простыми путями*. (К сожалению, терминология тут не вполне установилась.)

Контрольный вопрос 3.66. Приведите пример графа и простого пути в нём, не являющегося кратчайшим. (Ответ: возьмём граф в форме кольца и простой путь, занимающий больше половины этого кольца.)

Задача 3.67. Путь a_1–a_2–…–a_n является кратчайшим путём между a_1 и a_n. Докажите, что любой его участок a_k–a_{k+1}–…–a_l является кратчайшим путём между a_k и a_l.

Решение. Если этот участок — не кратчайший, то в исходном пути его можно заменить на более короткий, а это невозможно (исходный путь по предположению был кратчайшим). □

Контрольный вопрос 3.68. Расстояние $d(u, v)$ между вершинами u и v в графе G равно 7. Докажите, что найдётся вершина w, для которой $d(u, w) = 4$ и $d(v, w) = 3$.

Задача 3.69. Имеется связный граф. Докажите, что в нём можно выбрать одну из вершин так, чтобы после её удаления вместе со всеми ведущими из неё рёбрами останется связный граф.

Решение. Выберем произвольную вершину u, назовём её «началом». Теперь выберем наиболее удалённую от начала вершину v (одну из таких, если их несколько). Докажем, что после удаления вершины v и её рёбер граф останется связным, а именно, что любая оставшаяся вершина w по прежнему связана с началом u. В самом деле, кратчайший путь из u в w в исходном графе не мог проходить через v, потому что в этом случае w было бы строго дальше от u, чем v, а v была одной из наиболее далёких от начала вершин. □

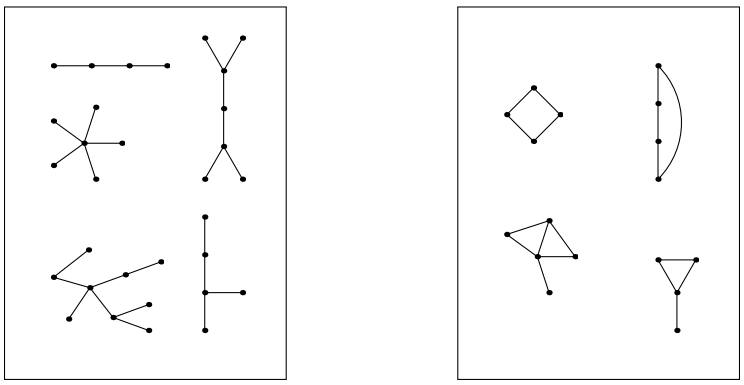
Задача 3.70. Вершинами булева куба, как мы знаем, являются битовые строки длины n, а соседями являются строки, отличающиеся в одной позиции. Чему равно расстояние между двумя произвольными вершинами? Чему равен диаметр этого графа? Сколько существует кратчайших путей между вершинами $00\ldots0$ и $11\ldots1$?

Решение. По определению расстояние — это минимальная длина пути по рёбрам, то есть минимальное число замен битов, позволяющих получить одну строку из другой. Оно равно числу различающихся битов и максимально для строки и её побитового отрицания, так что диаметр равен n. Кратчайший путь из $00\ldots0$ в $11\ldots1$ изменяет все биты в каком-то порядке (никакой бит не меняется дважды), и существует $n!$ таких порядков. □

3.2.5 Деревья

Есть такая игра, называется «распознавание образов» или «машинное обучение» — человеку (или программе) дают два набора картинок, и требуется сформулировать правило, которое отличает объекты этих двух групп. Обычно при этом часть картинок при обучении не показывают, приберегая для «экзамена», где проверяется, насколько правило, сформулированное на основе учебных картинок, хорошо работает для экзаменационных.

Попробуйте сделать нечто подобное: не читая текста дальше, сформулируйте, чем отличаются графы левой и правой групп. (Все графы связны, так что мы не разделяем на рисунке их друг от друга — и так понятно, где кончается один и начинается другой.)



Картинки слева называются *деревьями*.

Контрольный вопрос 3.71. Расклассифицируем несколько первых букв алфавита на деревья (Г, Е, Ж) и не деревья (А, Б, В, Д).

Что будет с остальными буквами алфавита в этой классификации? (Буквы Ё, Й и Ы не связны, так что их мы не рассматриваем.) (Ответ: налево пойдут буквы З, И, К, Л, М, Н, П, С, Т, У, Х, Ц, Ч, Ш, Щ, Э, направо пойдут буквы О, Р, Ф, Ъ, Ь, Ю, Я.)

Наверно, вы уже придумали, как объяснить разницу между деревьями и недеревьями. Вот несколько вариантов описания свойства, отличающего графы слева от графов справа:

(1) связный граф, где нельзя удалить ни одного ребра без нарушения связности;

(2) связный граф, где число рёбер на единицу меньше числа вершин;

(3) связный граф, где для любых двух вершин u, v существует единственный простой путь из u в v;

(4) связный граф, где нет простых циклов длины больше 2.

В последнем пункте под *простым циклом* мы понимаем путь $a_1, a_2, \ldots, a_n, a_1$ (начало совпадает с концом), в котором все вершины $a_1, \ldots, a_n$ различны.

Теорема 3.5. *Все четыре указанных свойства равносильны: граф, обладающий одним из них, обладает и всеми остальными.*

Тем самым эти свойства определяют разными способами один и тот же класс графов; графы этого класса называются *деревьями*.

Доказательство. Начнём с очевидных наблюдений.

(2) ⇒ (1). Если в графе число рёбер на единицу меньше числа вершин, то после удаления ребра их будет на 2 меньше, а мы уже знаем, что в этом случае связность нарушится.

(1) ⇒ (2). Пусть дан связный граф с n вершинами, в котором ни одного ребра нельзя удалить без нарушения связности. Как мы доказывали, что в нём не меньше $n-1$ рёбер? Мы представляли себе, что рёбра добавляются по очереди, и смотрели, как меняется число связных компонент. Если добавленное ребро соединяло вершины, которые уже и так связаны, число компонент не менялось; в противном случае оно уменьшалось на единицу. Теперь первый случай невозможен: в нём добавленное ребро соединяет то, что соединено и так (другими рёбрами), поэтому оно лишнее и в окончательном графе (его можно удалить без нарушения связности, вопреки предположению). Поэтому к моменту, когда останется одна компонента, было добавлено как раз $V-1$ рёбер, где V — число вершин.

Итак, (1) и (2) равносильны. Теперь докажем, что из этих свойств следует (3), то есть единственность простых путей. Здесь снова нужно будет смотреть за процессом добавления рёбер — мы знаем, что добавляемое ребро всегда соединяет две разные компоненты — и по индукции доказывать, что нет двух простых путей с одинаковыми началом и концом.

Итак, предположим, что это свойство выполнено для графа из нескольких связных компонент, и мы добавляем туда ребро p–q, соединяя две различные компоненты P и Q (содержащие p и q соответственно). Нам надо доказать, что и для нового графа это свойство выполнено. Пусть это не так, и есть два простых пути с одинаковым началом u и одинаковым концом v. Как минимум один из этих путей должен включать новое ребро p–q, поскольку до его добавления свойство единственности выполнялось (предположение индукции). Ребро это может входить только один раз, поскольку путь простой. Значит, начало пути лежит в одной из компонент P или Q, а конец в другой (остальные рёбра старые и не переводят в другую компоненту). Пусть, например, u лежит в P, а v лежит в Q. Тогда путь состоит из трёх частей: простой путь от u до p, ребро p–q, и простой путь от v до q. Посмотрим теперь на другой путь из u в v и убедимся, что он совпадает с первым. Он тоже должен использовать ребро p–q, поскольку перед добавлением этого ребра вершины u и v лежали в разных компонентах. Тогда по тем же причинам он разбивается на часть от u до p, ребро p–q, и часть от q до v. Остаётся воспользоваться предположением индукции (единственность пути в старом графе от u до p, и от q до v) и увидеть, что два рассматриваемых пути совпадают.

Из (3) легко следует (4). В самом деле, если есть простой цикл a_1–a_2–…–a_n–a_1, где $n>2$ и все $a_1,\ldots,a_n$ различны, то есть два простых пути из a_1 в a_n, а именно путь a_1–a_2–…–a_n и путь из одного ребра a_1–a_n. (Понятно, почему существенно условие $n>2$?)

Наконец, надо убедиться, что из (4) следует (1) и (2). Снова нужно вспомнить процесс слияния компонент при добавлении рёбер. Если (1) и (2) неверны, то в какой-то момент добавляется ребро, соединяющее уже связанные друг с другом вершины. Эти вершины связаны простым путём (как мы доказали), и вместе с добавленным ребром получится цикл. При этом длина цикла по крайней мере 3, потому что имевшийся простой путь был с пересадками (ребра-то раньше не было).

Теорема об эквивалентности четырёх свойств доказана. □

Задача 3.72. Покажите, что из любого связного графа можно удалить часть рёбер таким образом, чтобы оставшийся граф был деревом.

Решение. Здесь удобно пользоваться определением (2): если связный граф ещё не дерево, то есть ребро, которое можно удалить без нарушения связности. Если то, что останется — снова не дерево, удалим ещё одно ребро без нарушения связности, и так далее, пока не останется дерево. □

Остовным деревом в графе называется дерево, которое получается удалением части рёбер. В предыдущей задаче утверждается, что в любом связном графе есть остовное дерево. Остовных деревьев может быть много. Например, в полном графе на n вершинах есть n^{n-2} остовных дерева. Мы не приводим доказательство этого факта.

Задача 3.73. Покажите, что в дереве из более чем одной вершины всегда есть вершина степени 1.

Решение. Вершин степени нуль в этом дереве нет, иначе оно не будет связным. Если все вершины степени 2 или более, то сумма степеней не меньше удвоенного числа вершин, так что число рёбер (равное, как мы знаем, половине суммы степеней) не меньше числа вершин, а в дереве оно меньше (на единицу). □

Задача 3.74*. Покажите, что если граф является деревом, то его вершины можно так разместить на плоскости, чтобы соединяющие их рёбра (отрезки прямых) не пересекались. (Указание: можно использовать предыдущую задачу.)

Задача 3.75*. Докажите, что если в графе нет простых циклов длины 3 или больше, то можно добавить к нему рёбра так, чтобы получилось дерево.

Задача 3.76*. Связность и деревья можно описать в терминах линейной алгебры. Для данного графа G с множеством вершин V рассмотрим векторное пространство $\mathbb{F}_2^V$ над полем $\mathbb{F}_2 = \{0,1\}$ из двух элементов, состоящее из формальных комбинаций вершин с коэффициентами 0/1 (другими словами, его элементы — это функции $V \to \mathbb{F}_2$, или битовые строки длины $|V|$). Каждому ребру $e = (v, v')$ поставим в соответствие сумму его концов $v + v'$, то есть функцию, равную 1 на v и v' и 0 в остальных местах. Все рёбра лежат в подпространстве Z, образованном строками с нулевой (=чётной) суммой коэффициентов.

1. Покажите, что граф связен тогда и только тогда, когда векторы, соответствующие его рёбрам, порождают всё подпространство Z.

2. Покажите, что граф не имеет циклов тогда и только тогда, когда векторы, соответствующие рёбрам, линейно независимы.

(Указание. Если граф не связен, то рёбра лежат в меньшем подпространстве, соответствующем нулевой сумме во всех связных компонентах. Если вершины u и v связаны, то $u+v$ лежит в подпространстве, порождаемом рёбрами. Сумма рёбер, соответствующих простому циклу, равна нулю. Если цикла нет, то удаление ребра u–v разводит вершины u и v в разные связные компоненты, и для всех остальных рёбер сумма коэффициентов в каждой компоненте равна нулю, в отличие от выбранного ребра u–v, так что это ребро не выражается через остальные.)

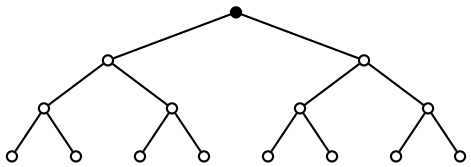
3.2.6 Полное бинарное дерево

Рассмотрим граф, вершины которого являются двоичными словами длины $\leqslant n$, а рёбра имеют вид $\{u, u0\}$ или $\{u, u1\}$. Такой граф является деревом. Действительно, он связный: от любого слова можно перейти к любому, стирая последний символ или дописывая символ к концу слова.

Посчитаем количество рёбер. Для этого удобно ввести на рёбрах ориентацию. Концы ребра — это два слова. Будем считать более короткое из них началом ребра, а более длинное — концом ребра. Начал рёбер столько же, сколько всего рёбер. Это количество равно удвоенному числу двоичных слов длины меньше n (каждое такое слово является началом двух рёбер), т.е.

$$2 \cdot (1 + 2 + \cdots + 2^{n-1}) = 2^{n+1} - 2.$$

Количество вершин в таком графе равно количеству слов длины не больше n, т.е. $2^{n+1} - 1$. Таким образом, этот граф является деревом. Его полное название: *полное бинарное дерево глубины n*. На рисунке изображено полное бинарное дерево глубины 3. (Подумайте, каким вершинам какие слова соответствуют.)



На это рисунке выделена одна из вершин. Это пустое слово.

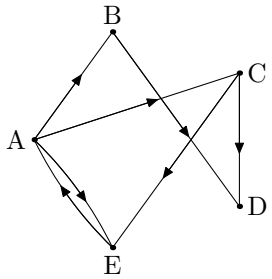
Если в дереве выделена особая вершина, она обычно называется *корнем дерева*. В этом случае вершины степени 1 (*висячие вершины*) называют *листьями дерева*. (С единственным исключением: корень тоже может иметь степень 1, но его листом не считают.)

Количество листьев в полном бинарном дереве глубины n равно 2^n — это количество двоичных слов длины n.

3.3 Ориентированные графы

3.3.1 Определение

Часто полезно рассматривать графы, в которых рёбра имеют ориентацию (тогда их изображают линиями со стрелками, а не просто линиями). Например, если в разделе 3.1.1 допустить односторонние рейсы (самолёт возвращается без пассажиров или летит в какое-то другое место), то такой односторонний рейс будет изображаться линией со стрелкой — а двусторонний рейс можно будет изобразить двумя линиями со стрелками туда и сюда. Изучая, можно ли теперь из города X попасть в город Y, мы должны будем искать путь из X в Y по стрелкам. Это отношение достижимости уже не обязано быть симметричным: можно случиться так, что из X в Y проехать можно, а обратно — нет.



Скажем, в ориентированном графе на рисунке видно, что из A можно попасть в D (с пересадками), а обратно проехать нельзя никак.

Формально говоря, мы рассматриваем *ориентированные графы*,[8] в которых для каждой пары различных вершин v, v' известно, есть ли ребро из v в v'. Про такое ребро говорят, что v является его *началом*, а v' — его *концом*. При этом возможно, что ребро из v в v' есть, а ребра из v' в v нет (скажем, на рисунке есть ребро AB, но нет ребра BA). Может быть и так, что рёбра есть в обе стороны (скажем, на рисунке есть ребро AE и есть ребро EA) — или что ни в одну сторону нет рёбер (скажем, на рисунке нет ни ребра AD, ни ребра DA). Но «кратные рёбра» (несколько рёбер с общими началом и концом) и «петли» (рёбра, ведущие из вершины в неё же) по-прежнему мы не разрешаем.

Занумеровав вершины ориентированного графа числами от 0 до $n-1$, можно закодировать информацию о её рёбрах в виде таблицы $edge[i, j]$, в которой i, j принимают значения в интервале от 0 до $n-1$, и $edge[i, j]$ истинно, когда есть ребро из i в j (и ложно в противном случае). Теперь уже не требуется, чтобы таблица была симметрична. Как и раньше, значения $edge[i, i]$ не имеют смысла, и удобно договориться, что в таблице $edge[i, i] =$ `false`.[9]

На математическом языке говорят, что *ориентированным графом* называется пара (V, E), где V — некоторое конечное множество, элементы которого называются

[8] Иногда для краткости говорят даже «орграфы» (по-английски ‘directed graphs’ сокращают до ‘digraphs’).

[9] Как и в случае неориентированных графов, при анализе достижимости вершин удобнее считать, что $edge[i, i] =$ `true`.

вершинами графа, а E — конечное множество, элементы которого называются *рёбрами* и являются упорядоченными парами вершин (другими словами, $E \subset V \times V$). При этом, согласно сказанному выше, мы не разрешаем пары (v, v) в качестве рёбер.

Неориентированные графы (те, которые мы рассматривали раньше) можно отождествить с ориентированными графами, в которых для каждого ребра существует и обратное.

3.3.2 Степени вершин

В неориентированных графах мы определяли степень как число рёбер, выходящих из этой вершины. Теперь надо различать выходящие и входящие рёбра, поэтому вместо одного числа получаются два. Их называют *исходящей степенью* и *входящей степенью*.

Раньше у нас была теорема о том, что сумма степеней вершин равна удвоенному числу рёбер. Как её надо переделать для случая ориентированных графов? Наверное, вы уже догадались, как:

Теорема 3.6. *Сумма исходящих степеней всех вершин равна сумме входящих степеней всех вершин: обе суммы равны числу рёбер графа.*

Доказательство. Каждое ребро имеет одно начало (выходит из какой-то вершины) и поэтому учитывается по разу, когда мы складываем исходящие степени всех вершин. Аналогично для концов рёбер. □

3.3.3 Пути и достижимость

Определяя пути в ориентированных графах, мы должны учитывать направления рёбер (идти только по стрелкам на рисунках, а не против). *Путём* в ориентированном графе будет последовательность вершин $v_1, v_2, \ldots, v_k$, в которой соседние члены соединены ребром в нужную сторону, то есть в графе есть ребро с началом v_i и концом v_{i+1} (при всех $i = 1, 2, \ldots, k-1$). Длиной пути по-прежнему считают число рёбер, то есть $k - 1$. Путь из одной вершины имеет длину 0. Вершину v_1 называют началом пути, а вершину v_k — его концом.

В отличие от неориентированных графов, пути обращать нельзя (точнее, не всегда можно обращать), и вместо определения связанных вершин мы должны дать другое. Говорят, что вершина v *достижима* из вершины u, если существует путь с началом u и концом v. В частности, каждая вершина достижима сама из себя, так как мы разрешаем пути длины 0. Это отношение уже не обязано быть симметрично: возможно, что вершина v достижима из вершины u, но не наоборот. Но оно по-прежнему транзитивно: если вершина v достижима из u, а w достижима из v, то w достижима из u (соединяем пути).

Как и раньше, *простой путь* в ориентированном графе — это путь, не проходящий дважды через одну вершину. Достижимость, как и раньше, можно определять с помощью простых путей: если v достижима из u, то есть простой путь из u в v.

В самом деле, если какая-то вершина встречается дважды, то участок пути между её вхождениями можно вырезать, получив более короткий путь, так что путь минимальной длины всегда будет простым (а он существует).

В неориентированном графе мы определяли расстояние между связанными вершинами как длину кратчайшего пути. В ориентированном графе, если вершина v достижима из u, можно тоже рассмотреть длину кратчайшего пути из u в v. Но называть это «расстоянием» не стоит: обычно от расстояния требуют симметричности, а здесь она очевидным образом нарушается. (Если по кольцевой дороге автобус ходит в одну сторону, и мы проехали свою остановку, то нам придётся почти что объехать кольцо, чтобы вернуться.)

Контрольный вопрос 3.77. Выполнено ли для этого «несимметричного расстояния» $d(u, v)$ свойство транзитивности

$$d(u, w) \leqslant d(u, v) + d(v, w)?$$

3.3.4 Достижимость и разрезы

Сейчас мы на примере достижимости попытаемся проиллюстрировать важную идею, которую называют *двойственностью*. Представьте себе, что вам на экзамене дали граф, указали две вершины s и t и просят доказать, что вершина t достижима из вершины s. Что вы напишете в качестве доказательства? Согласно определению, для доказательства достаточно предъявить путь из s в t. Этот путь можно выбрать простым, и тогда его длина меньше числа вершин (и предъявить его вполне реально — раз уж в условии хватило места на граф, в решении хватит места на путь).

Теперь более сложный вопрос: пусть вам надо доказать, что вершина t *недостижима* из вершины s, то есть что такого пути нет. Что же тогда можно предъявить в качестве доказательства?[10] Можно, конечно, перечислить все пути из вершины s, длина которых меньше числа вершин графа, и убедиться, что ни один из них не кончается в t. В этом случае, как мы знаем, и более длинных путей из s в t нет. Но путей даже и ограниченной длины может быть очень много (как говорят, их число может экспоненциально расти с ростом графа). Нельзя ли предъявить какое-то более короткое доказательство?

Таким доказательством может служить то, что называют *разрезом* графа. Представим себе, что все вершины графа разбиты на две категории (каждая вершина отнесена ровно к одной из двух), названные S и T. При этом вершина s отнесена к категории S, а вершина t отнесена к категории T. Такое деление называют *разрезом* графа (можно представить себе, что бумажку, где граф был нарисован, аккуратно разрезали на две части). Если при этом оказалось, что *в графе нет ни одного ребра, ведущего из S в T*, то ясно, что t недостижима из s. (Чтобы из страны можно было

[10] Рассказывают, что много лет назад (при советской власти) при посадке в поезд с доской для сёрфинга начальник поезда потребовал справку, что эта доска *не принадлежит* никакой государственной организации. Но задумался и отстал, когда его спросили, какой организацией должна быть выдана справка.

улететь, нужно иметь хотя бы один рейс изнутри наружу.) Заметим, что указать разрез не так сложно (у каждой вершины надо написать, в S она или в T) — и после этого можно убедиться, проверив все рёбра по очереди, что ни одно из них не ведёт из S в T. Так что никаких объектов экспоненциального размера в таком доказательстве недостижимости не появляется. Следующая «теорема двойственности» говорит, что это универсальный способ доказательства.

Теорема 3.7. *Если в ориентированном графе вершина t недостижима из s, то существует разрез (S, T), это устанавливающий ($s \in S$, $t \in T$, и из S в T не ведёт ни одного ребра).*

Доказательство. Пусть S — множество всех вершин, достижимых из s (соответственно, T — множество всех вершин, недостижимых из s). Тогда $s \in S$ по определению, $t \in T$ по условию теоремы, и остаётся проверить, что нет рёбер из достижимых вершин (S) в недостижимые (T). Но это очевидно: если из достижимой (из s) вершины куда-то ведёт ребро, то и конец этого ребра достижим (из s), надо просто добавить это ребро в конец пути. □

Формально говоря, из этого доказательства не видно, как реально найти это самое множество всех достижимых из s вершин (за разумное время — не перебирая все пути). Но по существу из него можно извлечь алгоритм:

$S \leftarrow \{s\}$

while есть ребро, ведущее из S вовне S **do**

    добавить конец этого ребра к S

end

Контрольный вопрос 3.78. Для программистов: что надо добавить в этот алгоритм, чтобы в случае достижимости найти путь из s в t? Для опытных программистов: как реализовать этот алгоритм так, чтобы время его работы было пропорционально числу рёбер (если граф представлен списками исходящих рёбер для каждой вершины)?

Контрольный вопрос 3.79. Доказательство теоремы несимметрично: вершина s в нём играет более центральную роль, чем вершина t. Но можно сделать и наоборот, начав с вершины t и посмотрев, из каких вершин она достижима и из каких нет. Может ли при этом получиться другой разрез?

Как говорят в теоретической информатике, мы сначала доказали, что достижимость лежит в классе NP (есть короткие доказательства достижимости), потом что недостижимость лежит в классе NP (есть короткие доказательства недостижимости), и наконец заметили, что достижимость лежит в классе P, объяснив, как найти короткое доказательство того или другого.

3.3.5 Компоненты сильной связности и ациклические графы

Отношение достижимости несимметрично, но можно определить симметричное отношение «достижимости в обе стороны». Будем говорить, что вершина u *сильно связана* с вершиной v, если v достижима из u и наоборот, то есть если есть путь из u в v, а также путь из v в u.

Теорема 3.8. *Вершины ориентированного графа можно разбить на непересекающиеся группы, называемые* сильно связными компонентами*, при этом:*

- *каждая вершина графа попадает ровно в одну группу;*
- *любые две вершины из одной группы сильно связаны (есть пути в обе стороны);*
- *любые две вершины из двух разных групп не являются сильно связанными (в одну из сторон — или даже в обе — пути нет).*

Для неориентированных графов аналогичное утверждение было доказано в разделе 3.2.3.

Доказательство. Повторяет рассуждение для неориентированных графов: для каждой вершины v мы составляем группу $C(v)$ из всех сильно связанных с ней вершин и доказываем, что получается искомое разбиение. □

На самом деле это общее рассуждение: свойства рефлексивности, симметричности и транзитивности для отношения (в данном случае для отношения сильной связанности) гарантируют возможность разбиения на классы. Такие отношения называют *отношениями эквивалентности*.

Два крайних случая:

(1) в графе любые две вершины сильно связаны (из любой вершины можно попасть в любую, сильно связанная компонента только одна). Такие графы называют *сильно связными*.

(2) никакие две различные вершины не являются сильно связанными (для любой пары хотя бы в одну сторону пути нет, сильно связные компоненты одноэлементные). Такие графы называются *ациклическими* — название объясняется следующей теоремой.

Теорема 3.9. *Следующие свойства ориентированного графа равносильны:*

- *Каждая сильно связная компонента состоит из одной вершины.*
- *В графе нет циклов (путей вида $a_1, \ldots, a_n, a_{n+1}$, где начало совпадает с концом, то есть $a_1 = a_{n+1}$; при $n = 0$ путь из единственной вершины, имеющий нулевую длину, циклом не считается).*
- *Вершины графа можно пронумеровать натуральными числами таким образом, чтобы все рёбра вели «вверх»: из вершины с меньшим номером в вершину с большим.*

Третье условие удобно представлять себе так: если выписать вершины в порядке номеров, то рёбра графа будут идти слева направо.

Доказательство. Начнём с очевидных утверждений. Если в графе есть цикл с $n > 1$ вершинами, то вершины этого цикла сильно связаны (из любой можно попасть в любую по циклу), так что из первого свойства следует второе. Наоборот, если различные вершины a, b сильно связаны, то существуют пути из a в b и из b в a, и из этих двух путей можно составить цикл. Наконец, если возможна нумерация вершин, при которой рёбра ведут «вверх», то цикла нет: вдоль него номера вершин должны расти, и вернуться в начало мы не сможем.

Осталось доказать единственное нетривиальное утверждение: если в ориентированном графе нет циклов, то его вершины можно пронумеровать требуемым способом.

Лемма 3.10. *В ориентированном графе без циклов есть вершина, из которой не выходит ни одного ребра.*

Доказательство леммы. Пусть это не так и из любой вершины выходит хоть одно ребро. Возьмём какую-то вершину a_1, из неё идёт ребро в какую-то другую вершину a_2, из неё идёт ребро ещё куда-то, и так далее. Поскольку граф конечен, то рано или поздно мы придём второй раз в какую-то вершину a_i, где уже были, и участок пути после a_i свернётся в цикл — вопреки предположению, что циклов нет. □

Теперь можно закончить доказательство теоремы. Выберем вершину, из которой не ведёт ни одного ребра. Ей можно без опаски присвоить номер, бóльший всех остальных номеров. Поэтому рассуждаем по индукции: удалив эту вершину (и все входящие в неё рёбра) из графа, получим граф без циклов. (Циклы в нём были бы циклами и в исходном графе.) Пронумеруем его вершины от 1 до какого-то числа n (число оставшихся вершин) с соблюдением условия (рёбра ведут из меньшего к большему), и после этого вернём выброшенную вершину под номером $n + 1$. (Базис индукции — граф с одной вершиной — очевиден.) □

Эта теорему можно объяснить так. Представим себе, что есть какие-то бумаги (справки, разрешения и пр.), которые надо получать в определённом порядке: налоговая инспекция требует счёта в банке, банк требует регистрации где-нибудь ещё, и пр. Эту зависимость можно представить в виде графа: вершины это нужные бумаги, а ребро $u \to v$ показывает зависимость v от u (чтобы получить v, надо уже иметь u).[11] Приведённое нами доказательство возможности нумерации тоже легко объяснить в этих терминах: если нет ни одной бумаги, которую можно получить просто так (не имея каких-то других), и из каждой инстанции нас посылают в какую-то другую, то рано или поздно мы совершим цикл и придём в инстанцию, где уже были. Если же такая простая для получения бумага есть, то начнём с того, что получим её — и сведём задачу к меньшей (которую, по предположении индукции,

[11] Те счастливые люди, которые никогда не собирали документов, могут использовать другую метафору: ребро (носок → ботинок) означает, что ботинок нужно надевать, когда носок уже надет.

можно считать уже решённой). Пронумеровав все остальные бумаги, добавим нашу простую бумагу в конец.

Два заключительных замечания:

(1) Доказанную теорему тоже можно воспринимать как утверждение типа двойственности: если цикл есть, то можно его предъявить, а если цикла нет, то в качестве доказательства того, что его быть не может, можно предъявить нумерацию.

(2) Доказательство теоремы по существу даёт алгоритм поиска цикла или нумерации (что найдётся): сначала вызываем процедуру, соответствующую лемме, которая либо находит цикл, либо находит вершину, из которой не выходят рёбра. В первом случае задача решена, во втором случае мы рекурсивно вызываем тот же алгоритм для остальных вершин. Этот алгоритм не очень долгий (время работы полиномиально зависит от числа вершин), но не оптимальный: если нам нужно искать такую нумерацию на практике (это называют «топологической сортировкой»), то есть более эффективные алгоритмы (время их работы пропорционально числу рёбер, если граф представлен списками рёбер).

3.3.6 Графы преобразований

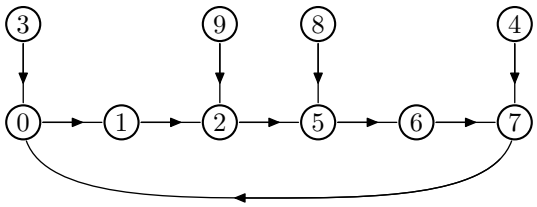
Рассмотрим для примера такую задачу: с числом 1 тысячу раз проделали операцию «возведение в квадрат и прибавление единицы»: после первой операции получилось 2, после второй 5, после третьей 26 и так далее. Какая будет последняя цифра у получившегося после тысячи операций числа?

Формально: $a_0 = 1$, $a_{n+1} = a_n^2 + 1$; надо найти последнюю цифру a_{1000}.

Хотя на вид задача выглядит пугающе — число астрономически большое — на самом деле она совсем не сложная. Для начала заметим, что последняя цифра квадрата числа определяется последней цифрой самого числа, предыдущих цифр знать не надо. (Почему? вспомните, как происходит умножение в столбик.) То же самое и с прибавлением единицы. Соответственно, можно сказать, какая будет последняя цифра числа $a^2 + 1$ в зависимости от последней цифры числа a:

a	0	1	2	3	4	5	6	7	8	9
a^2+1	1	2	5	0	7	6	7	0	5	2

Для удобства представим информацию в этой таблице в виде ориентированного графа: первая колонка изображается стрелкой $0 \to 1$, вторая — стрелкой $1 \to 2$, третья — стрелкой $2 \to 5$ и так далее.



Теперь задачу можно переформулировать так: *выйдя из точки* 1*, мы прошли тысячу раз по стрелкам, куда мы пришли*? В этом виде её решить совсем просто: через

шесть шагов мы возвращаемся в точку 1, значит, через $996 = 166 \cdot 6$ шагов мы тоже вернёмся в точку 1, останется четыре шага, которые приведут нас в точку 7. Ответ: число a_{1000} оканчивается на 7.

Контрольный вопрос 3.80. На какую последнюю цифру оканчивается число a_{2000}? На какую последнюю цифру оканчивается b_{1000}, если $b_0 = 4$, а $b_{n+1} = b_n^2 + 1$? На какую последнюю цифру оканчивается c_{1000}, если $c_0 = 1$, $c_{n+1} = c_n^2 + 2$?

В этом примере из каждой вершины выходила ровно одна стрелка,[12] поэтому движение по стрелкам было предопределено (нет выбора, куда идти). Начав двигаться из некоторой точки, мы в какой-то момент попадаем в уже пройденную точку (граф конечен), и после этого зацикливаемся. Это позволяет легко рассчитать, в какой точке мы окажемся после данного числа шагов.

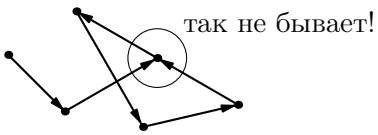
Задача 3.81. Предположим, что последовательность чисел задана своим первым членом a_0 и соотношением $a_{n+1} = f(a_n)$, где f — некоторая функция (определённая на всех числах). Покажите, что либо все члены последовательности различны, либо она периодична: после некоторого начала (предпериода) числа начинают повторяться (период).

Задача 3.82*. (Продолжение предыдущей) Покажите, что второй случай имеет место тогда и только тогда, когда $a_{2n} = a_n$ при некотором n.

Бывает и так, что не только из каждой вершины выходит одна стрелка, но и в каждую вершину входит ровно одна стрелка. Другими словами, входящая и исходящая степени каждой вершины равны 1. Примером такого графа является *цикл* $a \to b \to c \to \ldots \to p \to q \to a$. Такие (ориентированные) графы называются графами перестановок.[13]

Теорема 3.11. *Граф перестановки разбивается на непересекающиеся (не имеющие общих вершин и рёбер) циклы.*

Доказательство совсем простое: возьмём какую-то вершину и пойдём по стрелкам. Рано или поздно мы впервые попадём в уже посещённую вершину. Эта вершина должна быть начальной, иначе в одну вершину ведут две стрелки, что противоречит предположению.



Тем самым возникает цикл, и из его вершин больше никаких стрелок не выходит и в них не входит. Если в цикл вошли не все вершины графа, повторяем рассуждение с остатком, пока граф не разобьётся на циклы.

[12] Такие графы называют графами функций (соответствующая функция отображает начало стрелки в её конец, так что стрелка направлена от аргумента к значению).

[13] Соответствующая графу функция представляет собой взаимно однозначное отображение множества всех его вершин в себя, или, как говорят, *перестановку* множества вершин.

Задача 3.83. Имеется n людей, которые хотят обменяться квартирами (каждый хочет переехать в квартиру кого-то другого, при этом в каждую квартиру хочет переехать только один человек). За один день несколько пар жильцов могут поменяться квартирами (в каждой паре жилец переезжает на место другого). Докажите, что весь переезд можно провести за два дня (в конце первого дня все жильцы размещены в каких-то квартирах — возможно, не в тех, откуда они выезжают и не в тех, куда они переезжают).

Задача 3.84. (Для тех, кто видел кубик Рубика.) Если повернуть одну (скажем, правую) грань на 90° по часовой стрелке, и сделать так четыре раза, то кубик вернётся в исходное положение. Назовём это действие A. Аналогичным образом, если повернуть верхнюю грань на 90° по часовой стрелке (назовём это преобразование B), то кубик тоже вернётся в исходное положение. А теперь будем чередовать A и B, выполняя их в последовательности $ABABAB\ldots$. Будет ли момент, когда кубик вернётся в исходное положение?

3.4 Эйлеровы циклы

3.4.1 Определение

В этом разделе мы вернёмся к уже обсуждавшимся мостам в Кёнигсберге (см. раздел 3.1.3) и сформулируем и докажем общее утверждение. Оно существует в двух вариантах: для неориентированных и ориентированных графов.

Мы уже говорили, что *циклом* в графе называется путь, в котором начало и конец совпадают, то есть последовательность вершин $a_1, a_2, \ldots, a_n$, в которой (a_i, a_{i+1}) является ребром графа (при всех $i = 1, 2 \ldots, n-1$) и начало совпадает с концом: $a_1 = a_n$.[14]

Это определение относится и к неориентированным, и к ориентированным графам. Во втором случае имеется в виду, что ребро (a_i, a_{i+1}) ведёт из a_i в a_{i+1}, то есть a_i является началом, а a_{i+1} —концом ребра.

Цикл называется *эйлеровым*, если он проходит по всем рёбрам графа по одному разу (любое ребро входит в цикл, и никакое ребро не входит дважды). Для неориентированных графов мы имели в виду именно это, когда просили нарисовать граф не отрывая карандаша от бумаги и вернуться в исходную точку (в разделе 3.1.3).

3.4.2 Критерий существования

Теперь мы можем сформулировать обещанный критерий, сначала для неориентированных графов, потом для ориентированных.

Теорема 3.12. *Неориентированный граф без вершин нулевой степени содержит эйлеров цикл тогда и только тогда, когда он связен и степени всех вершин чётны.*

[14] Можно не выделять в цикле точку начала и конца, то есть считать, скажем, $a \to b \to c \to a$ и $b \to c \to a \to b$ одним и тем же циклом. Мы будем придерживаться исходного определения, с выделенным началом и концом, если противное не оговорено явно.

Теорема 3.13. *Ориентированный граф без вершин нулевой степени (в которые не входит и из которых не выходит рёбер) содержит эйлеров цикл тогда и только тогда, когда он сильно связен и у любой вершины входящая степень равна исходящей.*

Оговорка про вершины нулевой степени необходима: они нарушают связность графа, но никак не мешают эйлерову циклу (их можно удалить, и с точки зрения поиска эйлерова цикла это ничего не меняет).[15]

Доказательство. Будем доказывать параллельно оба варианта теоремы. Пусть сначала эйлеров цикл есть. Тогда он проходит через все вершины (поскольку они имеют ненулевую степень), и по нему можно дойти от любой вершины до любой. Значит, граф связен (сильно связен в ориентированном случае).

Теперь про степени. Возьмём какую-то вершину v, пусть она встречается в цикле k раз. Идя по циклу, мы приходим в неё k раз и уходим k раз, значит, использовали k входящих и k исходящих рёбер. При этом, раз цикл эйлеров, других рёбер у этой вершины нет, так что в ориентированном графе её входящая и исходящая степени равны k, а в неориентированном графе её степень равна $2k$. Таким образом, в одну сторону критерий доказан.

Рассуждение в обратную сторону чуть сложнее. Будем рассматривать пути, которые не проходят дважды по одному ребру. (Таков, например, путь из одного ребра.) Выберем среди них самый длинный путь

$$a_1 \to a_2 \to a_3 \to \ldots \to a_{n-1} \to a_n$$

и покажем, что он является искомым циклом, то есть что $a_1 = a_n$ и что он содержит все рёбра.

В самом деле, если он самый длинный, то добавить к нему ребро $a_n \to a_{n+1}$ уже нельзя, то есть все выходящие из a_n рёбра уже использованы. Это возможно, лишь если $a_1 = a_n$. В самом деле, если вершина a_n встречалась только внутри пути (пусть она входит k раз внутри пути и ещё раз в конце пути), то мы использовали $k + 1$ входящих рёбер и k выходящих, и больше выходящих нет. Это противоречит равенству входящей и исходящей степени (в ориентированном случае) или чётности степени (в неориентированном случае).

Итак, мы имеем цикл, и осталось доказать, что в него входят все рёбра. В самом деле, если во всех вершинах цикла использованы все рёбра, то из вершин этого цикла нельзя попасть в вершины, не принадлежащие циклу, то есть использованы все вершины (мы предполагаем, что граф связен или сильно связен) и, следовательно, все рёбра. С другой стороны, если из какой-то вершины a_i выходит ребро $a_i \to v$, то путь можно удлинить до

$$a_i \to a_{i+1} \to \ldots \to a_n = a_1 \to a_2 \to \ldots \to a_i \to v,$$

[15] Педантичные читатели заметили бы, что мы не оговорили особый случай графа (ориентированного графа) совсем без вершин. В нём нет вершин нулевой степени. Он связен (соответственно сильно связен), и можно считать, что в нём есть пустой эйлеров цикл — ну, или отдельно потребовать, чтобы в графе были вершины.

вопреки нашему выбору (самого длинного пути). Аналогично можно получить противоречие и для входящего ребра $v \to a_i$, добавив его в начало. (А можно заметить, что если есть неиспользованное входящее ребро, то есть и неиспользованное выходящее.) Это рассуждение было для ориентированного случая, но в неориентированном всё аналогично. Теорема доказана. □

Помимо эйлеровых циклов, можно рассматривать *эйлеровы пути* — пути в графе, которые проходят один раз по каждому ребру. (Для неориентированных графов: рисуем картинку, не отрывая карандаша от бумаги, но не обязаны вернуться в исходную точку.) Для них тоже есть критерий: в неориентированном случае нужно, чтобы граф был связен и было не более двух вершин нечётной степени.

Контрольный вопрос 3.85. Может ли такая вершина быть только одна?

Задача 3.86. Как сформулировать аналогичный критерий существования эйлеровых путей в ориентированном графе?

3.4.3 Последовательности де Брёйна

Вот любопытное применение этой теоремы. Пусть мы хотим написать цифры по кругу — скажем, по окружности круглого стола. Идя вокруг стола, мы читаем двузначные числа (образованные парами соседних цифр). Требуется, чтобы каждое число от 00 до 99 встретилось при этом ровно по одному разу.

Попробуем доказать, что наше требование выполнимо (предупреждение: в этом рассуждении есть ошибка, попробуйте её заметить, не читая следующего абзаца). Рассмотрим ориентированный граф с вершинами $0, 1, 2, \ldots, 9$, в котором есть все возможные рёбра (любые две вершины соединены рёбрами в обе стороны). Он, очевидно, сильно связен, и из любой вершины входит столько же рёбер, сколько и выходит. Значит, по доказанной теореме в нём есть эйлеров цикл, и это как раз то, что нам надо: написав вершины этого цикла по окружности, мы прочтём вдоль неё каждое двузначное число по одному разу.

Заметили ошибку? В нашем определении графа не разрешались петли, ведущие из вершины в себя. Поэтому числа $00, 11, \ldots, 99$ по кругу не встретятся. Но их можно добавить: для каждой цифры нужно выбрать какое-то место, где её удвоить.

С точки зрения задачи об эйлеровом цикле петли (рёбра из вершины в неё же) и кратные рёбра (несколько рёбер с общими началами и концами) вполне имеют смысл. Можно было бы изменить определение графа, разрешив их.[16] Например, можно считать, что граф задаётся множествами E («вершин») и V («рёбер») и двумя отображениями множества E в V, которые ставят в соответствие каждому ребру его начало и конец. (Как тогда следует определить путь?) После этого можно доказать критерий эйлеровости для таких графов, и обойтись без сделанной нами оговорки, но мы эту тему развивать не будем.

[16] Собственно говоря, граф мостов в Кёнигсберге как раз содержит рёбра с общими концами.

Задача 3.87. Докажите, что по кругу можно написать тысячу цифр так, чтобы вдоль круга можно было бы прочесть все трёхзначные числа от 000 до 999 по одному разу.

Из этой задачи следует, что если для открытия кодового замка нужно нажать три цифры кода в правильном порядке (и не важно, какие цифры нажимались до этого), то его можно открыть, сделав не более чем 1002 нажатий. (Почему нельзя обойтись меньшим числом?)

Построенные таким образом последовательности называются последовательностями де Брёйна (1918–2012), хотя частный случай последовательностей из двух элементов был рассмотрен ещё в XIX веке. Мы привели этот пример, поскольку в нём простой критерий существования эйлерова цикла при правильном взгляде на вещи сразу же даёт требуемый результат — совсем не очевидный, если не догадаться про интерпретацию в терминах графов.

3.4.4 Гамильтоновы циклы

В этом разделе мы ничего не будем доказывать, а только дадим определение: *гамильтонов цикл* — это цикл, который проходит через каждую вершину ровно по одному разу. (Начало и конец в цикле a_1–a_2–…–a_n мы считаем за один раз. Аналогично определяется и *гамильтонов путь* — путь, в котором каждая вершина встречается ровно один раз (тут уже начало и конец должны быть разными вершинами и считаются каждый за себя).

Хотя определение и похоже, но ситуация с поиском гамильтоновых путей (циклов) принципиально сложнее, чем с поисками эйлеровых: тут уже нет простого критерия или алгоритма проверки (есть ли гамильтонов путь в данном графе). Более того, есть некоторые причины считать, что эффективного (быстро работающего) алгоритма проверки нет. Как говорят, эта задача является NP-полной, и хотя пока и не доказано, что для таких задач нет быстрых алгоритмов, но мало кто на это надеется — «too good to be true», как говорится. Впрочем, их отсутствие, может быть, и к лучшему — если такие алгоритмы обнаружатся, то нынешним системам криптографии (используемым в банковском деле, в протоколах https, ssl и других), придёт конец.

3.5 Двудольные графы

3.5.1 Определение

Двудольным графом называется неориентированный граф, в котором вершины заранее разделены на две *доли* — *левые* и *правые*, и все рёбра соединяют вершины из разных долей (нет рёбер, соединяющих вершины одной доли). Другими словами, чтобы задать двудольный граф, надо указать два конечных множества L (левую долю) и R (правую долю) и указать, какие вершины левой доли соединены с какими вершинами правой доли. В программе такой граф представляется булевым

массивом $edge[0..l-1, 0..r-1]$, где l, r — количества вершин в левой и правой доле; мы считаем, что левые вершины пронумерованы от 0 до $l-1$, а правые от 0 до $r-1$. При этом $edge[i][j]$ истинно, если i-я вершина левой доли соединена с j-й вершиной правой доли. Наглядно можно представлять себе прямоугольную таблицу $l \times r$, в которой в некоторых клетках стоят плюсы (означающие наличие ребра). Наконец, можно определить двудольный граф формально как тройку (L, R, E), где L и R — конечные множества, а $E \subset L \times R$ — некоторое множество упорядоченных пар (первые элементы в L, вторые в R).[17]

В жизни встречаются разные ситуации, которые можно изобразить двудольными графами. Например, в левой доле могут быть учёные, а в правой статьи, и наличие ребра l–r означает, что учёный l является одним из авторов статьи r. Или в левой доле могут быть студенты, а в правой — учебные курсы, и ребро означает, что студент прослушал курс. Или слева могут быть кавалеры, а справа дамы, и ребро означает, что они танцевали друг с другом. Наконец, можно вспомнить о теореме Холла и рассмотреть двудольный граф, в котором в левой доле стоят школьные кружки, в правой — школьники, и ребро означает участие школьника в кружке.

3.5.2 Двудольные графы и раскраска в два цвета

Мы предполагали, что при задании двудольного графа указано, какие вершины в левой доле, а какие в правой. Но можно действовать в другом порядке: взять произвольный неориентированный граф и спросить себя, можно ли так разделить его вершины на левую и правую долю, чтобы все рёбра соединяли вершины разных долей. Если обозначать это деление цветом: можно ли так раскрасить вершины графа в два цвета, чтобы концы любого ребра были разного цвета?

Если требуется доказать, что такое возможно, достаточно предъявить раскраску. А что надо предъявить, чтобы доказать, что такое невозможно? Наверно, вы уже догадались, но вот формальное утверждение.

Теорема 3.14. *Раскраска описанного типа возможна когда и только тогда, когда в графе нет циклов нечётной длины.*

Напомним, что циклом называется последовательность вершин $a_1, \ldots, a_n$, в которой вершины a_i и a_{i+1} соединены ребром, и $a_1 = a_n$. Длиной такого цикла является число рёбер, то есть $n-1$. (Не удивляйтесь, что рёбер в цикле меньше, чем вершин — это только кажется, потому что a_1 и a_n одна и та же вершина).

Доказательство. Если в графе есть цикл нечётной длины, то его нельзя раскрасить: соседние вершины должны быть противоположных цветов, и дойдя до конца, мы получим противоречие. (Попробуйте раскрасить вершины треугольника так, чтобы концы любого ребра были разного цвета!)

[17] В этом определении возникает вопрос, могут ли множества L и R иметь общие элементы. В принципе это можно и разрешить, но тогда, говоря о вершине, надо всегда уточнять, рассматриваем ли мы её как вершину левой доли или как вершину правой. Поэтому обычно предполагают, что L и R не имеют общих элементов.

Чтобы доказать обратное, предположим, что циклов нечётной длины нет. Выберем некоторую вершину a и решим, что она белая. (Это не ограничивает общности — всегда можно поменять цвета местами.) Для любой другой вершины b посмотрим, сколько рёбер в пути от a к b.

Лемма 3.15. *Если есть два пути из a в b, то либо в обоих чётное число рёбер, либо в обоих нечётное.*

Доказательство леммы. Если есть путь $a \to b$ с чётным числом рёбер, а также другой путь $a \to b$ с нечётным числом рёбер, то есть цикл с нечётным числом рёбер. А именно, пойдём из a в b по первому пути и вернёмся по второму. Это противоречит предположению. □

Таким образом, мы поделили вершины графа на два типа: соединённые с a путями чётной длины и путями нечётной длины. Если какая-то вершина соединена с a путями чётной длины, то её соседи соединены путями нечётной длины (один такой путь — через соседа — заведомо есть, а тогда и все пути имеют нечётную длину). Требуемая раскраска построена.

Заметили пробел в рассуждении? На самом деле мы раскрасили не весь граф, а только связную компоненту вершины a. Но это легко исправить: каждую связную компоненту можно раскрашивать независимо, так как рёбер между ними нет. □

Контрольный вопрос 3.88. Сколькими способами можно раскрасить в два цвета (с соблюдением требований) граф, в котором k связных компонент? (Ответ: 2^k.)

Замечание для программистов. Если попытаться реализовать это доказательство как алгоритм раскраски, получится сложно. Проще совместить поиск раскраски с поиском связных компонент и сразу же их раскрашивать: когда связная компонента растёт за счёт добавления соседа, цвет добавленной вершины определён однозначно.

Задача 3.89. Можно ли в критерии раскрашиваемости графа запретить лишь *простые* циклы нечётной длины?

Задача 3.90. Как мы уже не раз говорили, *булев куб* размерности n — это неориентированный граф, вершинами которого являются двоичные слова длины n, а рёбра соединяют слова, отличающиеся в одной позиции. Всегда ли такой граф можно раскрасить в два цвета?

Задача 3.91. Всякое ли дерево можно раскрасить в два цвета?

3.5.3 Степени вершин

Для вершин двудольного графа можно определить *степени* обычным образом — как число выходящих рёбер (или, что то же самое, как число соседей в другой доле). Скажем, в примере с авторами и статьями степень автора — это число публикаций (важный параметр — им пытаются измерять научную значимость автора!). А степень статьи — это число её авторов.

Контрольный вопрос 3.92. Каков житейский смысл степеней в других приведённых нами примерах?

Как и раньше, есть простой факт о сумме степеней.

Теорема 3.16. *Сумма степеней всех вершин левой доли равна сумме степеней всех вершин правой доли, и равна числу рёбер в графе.*

Контрольный вопрос 3.93. Почему это верно?

Часто иллюстрируют эту теорему так: если на контрольной было 20 задач, каждую задачу решили три школьника, и каждый школьник решил две задачи, то сколько было школьников?

Контрольный вопрос 3.94. Где здесь граф, степени, и сколько-таки было школьников?

3.5.4 Паросочетания

Задачу, решённую в разделе 3.1.6, теперь можно переформулировать так: имеется двудольный граф, при этом степени всех вершин (и слева, и справа) равны 2. Тогда можно удалить часть рёбер так, чтобы степени всех вершин стали равны 1. (Такой граф задаёт схему разбора задач.)

В главе про индукцию мы обсуждали задачу о назначении старост школьных кружков (см. теорему Холла на с. 41). Эту задачу можно переформулировать так. Пусть есть двудольный граф, причём если мы возьмём любые k вершин из левой части, то у них всего не меньше k соседей (внимание: мы не складываем их степени, а смотрим, сколько вершин в правой части являются соседями одной из выбранных в левой части вершин). Тогда можно удалить часть рёбер графа так, чтобы у всех вершин слева степени стали равны 1.

Обе эти задачи можно сформулировать в терминах «паросочетаний».[18] Мы называем *паросочетанием* двудольный граф, у которого степени всех вершин не больше 1. Такой граф устанавливает некоторое взаимно однозначное соответствие между частью вершин левой доли и частью вершин правой доли. Размер паросочетания не превосходит степени любой из долей (очевидно), и часто задача состоит в том, чтобы найти в графе паросочетание максимального размера. Лучшее, на что мы можем надеяться — достигнуть размера меньшей из долей, и в упомянутых выше задачах мы доказывали, что это возможно. Сформулируем ещё раз теорему Холла, используя введённую терминологию.

Теорема 3.17 (Теорема Холла)**.** *Пусть в двудольном графе любой набор из k левых вершин имеет не менее k соседей справа. Тогда в этом графе существует паросочетание, включающее все вершины левой доли.*

[18] Это традиционный термин, хотя звучит довольно странно. По-английски используется слово *matching*.

Соседи набора — соседи хотя бы одной из его вершин; «в графе есть паросочетание» — можно получить это паросочетание, удалив часть вершин и рёбер графа.

Контрольный вопрос 3.95. Почему верно обратное к теореме Холла утверждение (если есть паросочетание со всеми левыми вершинами, то любой набор с k вершинами имеет не менее k соседей)? [Указание: возьмите соседей из паросочетания.]

Задача 3.96. Рассмотрим граф с N вершинами слева и справа и будем искать в нём паросочетание размера N. Можно применить теорему Холла «слева-направо» и «справа-налево»: во втором случае требуется, чтобы любой набор из k вершин справа имел не менее k соседей слева. Как понять, не доказывая теорему Холла, что эти два условия эквивалентны?

В заключение этого раздела объясним, почему первая из рассмотренных задач (если все вершины графа имеют степень 2, то в нём есть паросочетание, включающее все вершины) следует из теоремы Холла. Мы уже знаем, что в таком графе поровну вершин слева и справа (половина числа рёбер), и надо доказать лишь, что любой набор из k вершин слева имеет не менее k соседей справа. Из вершин этого набора выходит $2k$ рёбер, и все эти $2k$ рёбер ведут в соседей. Поскольку в каждого соседа ведёт не более двух рёбер, то соседей не меньше k.

Преимущество этого рассуждения перед тем, которое у нас было раньше, в разделе 3.1.6: его легко обобщить на случай, когда каждый школьник решил 3 задачи, а каждую задачу решили 3 школьника. Тут уже граф не разбивается на циклы и как быть без теоремы Холла, непонятно. А с теоремой Холла всё так же: набор из k вершин имеет не менее k соседей, иначе выходящим из него $3k$ рёбрам некуда будет приткнуться (больше чем по три им собираться не разрешено). То же самое и для любого другого числа вместо 3.

Когда-то это утверждение предлагалось на школьных олимпиадах с таким «оживляжем»: каждый из 100 завов соединён телефонными проводами с 15 замами, при этом каждый из 100 замов соединён с 15 завами; надо доказать, что можно обрезать часть проводов так, чтобы каждый зав остался соединённым ровно с одним замом.

Задача 3.97. Покажите, что двудольный граф с долями размера n, в котором все вершины имеют степень k, можно разбить на k паросочетаний размера n. (Другими словами, можно раскрасить все рёбра в k цветов, и рёбра каждого цвета будут паросочетанием.)

Задача 3.98*. В квадратной таблице $n \times n$ стоят неотрицательные числа, причём суммы чисел в каждой строке и в каждом столбце равны 1.[19] Докажите, что можно поставить n ладей, не бьющих друг друга, на клетки с положительными числами.

Задача 3.99*. В каждой клетке прямоугольной таблицы $m \times n$ стоит по гному. Они хотят произвести перестановку (каждый гном перемещается в выбранную им клетку, причём ни одну клетку не выбрало несколько гномов). Докажите, что это всегда

[19] В теории вероятностей такие таблицы называются *дважды стохастическими матрицами.*

можно сделать в три дня, причём в первый день все гномы остаются с своих столбцах (перемещаются лишь по вертикали), второй день — в своих строках, а третий день — снова в столбцах.

3.6 Клики и независимые множества

Кликой называется такое подмножество вершин графа, каждая пара которых связана ребром.

Независимым множеством называется такое подмножество вершин графа, никакая пара которых не связана ребром.

Задача 3.100. Найдите максимальный размер клики и максимальный размер независимого множества в графе–пути P_n.

Решение. Будем считать, что вершины графа–пути занумерованы числами от 1 до n так, что рёбра соединяют пары вершин с номерами i, $i+1$. Если разность номеров двух вершин больше 1, ребра между ними нет. Поэтому клики размера 3 в графе–пути нет: из любых трёх целых чисел хотя бы одна пара отличается хотя бы на 2.

С другой стороны, любая пара вершин, соединённых ребром, образует клику размера 2. Поэтому максимальный размер клики в графе P_n равен 2 при $n \geqslant 2$. (Оставшийся случай графа–пути P_1 разберите самостоятельно.)

С независимыми множествами чуть сложнее. Во-первых, ответ зависит от чётности n. Для чётных n максимальный размер независимого множества равен $n/2$. Например, независимое множество образуют вершины с чётными номерами 2, 4, …, n. Для нечётных n максимальный размер независимого множества равен $(n+1)/2$. Независимое множество такого размера образуют вершины с нечётными номерами 1, 3, …, n.

Чтобы объединить эти два ответа в один, можно использовать функцию $\lceil x \rceil$, которая равна наименьшему целому числу, которое не меньше x. В обоих предыдущих примерах размер независимого множества равен $\lceil n/2 \rceil$. (Проверьте!)

Пока мы лишь привели примеры независимых множеств достаточно большого размера. Теперь нужно доказать, что бóльших независимых множеств в графе–пути нет. Пусть вершины с номерами $i_1 < i_2 < \cdots < i_k$ образуют независимое множество в графе–пути. По определению это означает, что между этими вершинами нет ребра, то есть $i_{j+1} - i_j > 1$. Значит, между каждой парой вершин независимого множества есть ещё хотя бы одна. А всего вершин в графе–пути n. Поэтому

$$2k - 1 = k + (k-1) \leqslant n, \quad \text{что равносильно } k \leqslant (n+1)/2.$$

Проверьте, что последнее неравенство означает, что $k \leqslant \lceil n/2 \rceil$ (не забудьте, что размер независимого множества — целое число). □

Вершинным покрытием называется такое множество вершин S, что для любого ребра хотя бы один из концов лежит в S.

Задача 3.101. Докажите, что S — вершинное покрытие тогда и только тогда, когда $V \setminus S$ — независимое множество.

С независимыми множествами и кликами связана одна из самых интересных теорем в комбинаторике — теорема Рамсея. В первом разделе этой главы мы уже её обсуждали, говоря о попарно знакомых и незнакомых людях.

Теорема 3.18 (теорема Рамсея)**.** *Для любых k, n найдётся такое число $R(k, n)$, что в любом графе на $R(k, n)$ вершинах есть или клика размера k, или независимое множество размера n.*

Ясно, что если утверждение теоремы справедливо для графа на R вершинах, то оно справедливо и для графов с $N > R$ вершинами. Поэтому обычно под $R(k, n)$ понимают *число Рамсея* —- минимальное количество вершин, для которого справедлива теорема.

Доказательство. Мы перескажем рассуждение, намеченное в первом разделе этой главы, для общего случая. Для этого применим индукцию.

Будем доказывать индукцией по s, что для любой пары чисел k, n такой, что $k + n = s$ справедливо утверждение теоремы.

База индукции $s = 2$ очевидна: $2 = 1 + 1$ — это единственный способ разложить число 2 в сумму целых положительных слагаемых, а одна вершина является одновременно и кликой, и независимым множеством.

Теперь докажем индуктивный переход. Предположим, что утверждение выполнено для всех пар (k, n) таких, что $k + n \leqslant s$.

Докажем его для пары (k, n) такой, что $k + n = s + 1$. По индуктивному предположению утверждение теоремы выполнено для пар $(k - 1, n)$ и $(k, n - 1)$.

Рассмотрим граф на $R(k - 1, n) + R(k, n - 1)$ вершине и возьмём какую-то вершину v этого графа. Пусть степень вершины v равна d.

Вершин в графе за исключением вершины v ровно $R(k - 1, n) + R(k, n - 1) - 1$ штук. Поэтому либо у вершины v есть $R(k - 1, n)$ соседей, то есть $d \geqslant R(k - 1, n)$, либо $d < R(k - 1, n)$ и тогда $R(k - 1, n) + R(k, n - 1) - 1 - d \geqslant R(k, n - 1)$, так что есть $R(k, n - 1)$ вершин, не связанных с v ребром.

Оба случая рассматриваются аналогично.

Первый случай. В индуцированном соседями вершины v подграфе по предположению найдётся или клика размера $k - 1$, или независимое множество размера n. В первом варианте добавление вершины v даёт клику в исходном графе размера k, во втором варианте в исходном графе есть независимое множество размера n.

Второй случай. В индуцированном несоседями вершины v подграфе по предположению найдётся или клика размера k, или независимое множество размера $n - 1$. В первом варианте в исходной графе есть клика размера k, а во втором добавление вершины v даёт независимое множество размера n в исходном графе.

Итак, мы доказали утверждение теоремы и для произвольной пары (k, n), для которой $k + n = s + 1$. Индуктивный переход доказан, и теорема следует из принципа математической индукции. □

Из доказательства теоремы получается также неравенство для чисел Рамсея

$$R(k,n) \leqslant R(k-1,n) + R(k,n-1),$$

которое напоминает рекуррентное соотношение для биномиальных коэффициентов. Поскольку $R(1,n) = R(k,1) = 1$, то легко убедиться, что

$$R(k,n) \leqslant \binom{k+n-2}{k-1}.$$

Лекция 4

Арифметика остатков

4.1 Чётные и нечётные числа

Все знают, что числа бывают чётные и нечётные. Чётные делятся на 2 без остатка, а нечётные дают остаток 1. Другими словами, чётные числа имеют вид $2k$ для целых k, а нечётные $2k+1$, тоже при целых k. Скажем, число 0 чётное ($0 = 2 \cdot 0$), а число -3 нечётное ($-3 = 2 \cdot (-2) + 1$).

Несложно проверить, что сумма двух чётных или двух нечётных чисел чётна, а сумма чётного и нечётного числа нечётна:

$+$	Ч	Н
Ч	Ч	Н
Н	Н	Ч

$\times$	Ч	Н
Ч	Ч	Ч
Н	Ч	Н

Например, если мы складываем чётное и нечётное число, то получаем $2k + (2l + 1) = 2(k + l) + 1$, то есть нечётное число. А если мы умножаем два нечётных числа, то получаем

$$(2k + 1)(2l + 1) = 4kl + 2k + 2l + 1 = 2(2kl + k + l) + 1,$$

то есть нечётное число.

Среди целых чисел чётные и нечётные встречаются одинаково часто: если мы возьмём большой интервал подряд идущих чисел, то количество чётных и нечётных в этом интервале будет почти одинаково (отличаться максимум на 1).

Задача 4.1. Докажите это утверждение.

Если заменить в таблицах буквы Ч и Н на 0 и 1, взяв нуль и единицу в качестве представителей классов чётных и нечётных чисел, то получатся почти что обычные таблицы сложения и умножения:

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Разница с обычными только в одном месте: $1 + 1 = 0$.

4.2 Деление на 3 и остатки

Попробуем составить аналогичные таблицы сложения и умножения для чисел, делящихся и не делящихся на 3. Тут сразу же возникает проблема: мы не знаем, что сказать про сумму двух чисел, не делящихся на 3. Она может делиться на 3 (например, $1 + 5 = 6$), а может и не делиться (например, $2 + 5 = 7$). Дело в том, что не делящиеся на 3 числа могут быть двух видов: одни дают остаток 1 (имеют вид $3k + 1$ при целом k), а другие дают остаток 2 (имеют вид $3k + 2$).

Задача 4.2. К какому типу относится число 1000? А число -1? Найдите соответствующие значения k.

Задача 4.3. Покажите, что все три типа чисел (делящиеся на 3, дающие остаток 1 и дающие остаток 2) встречаются одинаково часто: в большом отрезке подряд идущих чисел их количества отличаются максимум на 1.

Теперь можно составить аналогичную таблицу сложения и умножения остатков по модулю 3:

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Каждую клеточку в этой таблице несложно проверить. Например,

$$(3k + 1) + (3l + 2) = 3(k + l) + 3 = 3(k + l + 1) + 0$$

и

$$(3k + 2)(3l + 2) = 9kl + 6k + 6l + 4 = 3(3kl + 2k + 2l + 1) + 1.$$

Аналогичные таблицы можно составить не только для деления на 2 и 3, но и для любого числа. Но сначала мы дадим более формальные определения.

4.3 Деление с остатком

Говорят, что целое число a делится на целое число b, если $a = bk$ для некоторого целого числа k. В этом случае говорят также «a кратно b», и «b является делителем числа a».

В этом определении можно было бы сказать: «если частное a/b целое», но этим бы исключался случай $b = 0$, который формально допустим по нашему определению. Правда, особого смысла в нём всё равно нет: единственное число, которое делится на 0, это число 0. Определение допускает также отрицательные a и b: скажем, число -6 делится на -2 (а также и на 2), всего у него 8 делителей, если считать и положительные, и отрицательные.

Задача 4.4. Что это за делители?

Впрочем, обычно, говоря о количестве делителей у положительного целого числа, имеют в виду только положительные делители (считая единицу и само число).

Задача 4.5. Сколько (положительных) делителей у числа $30 = 2 \cdot 3 \cdot 5$? у числа $210 = 2 \cdot 3 \cdot 5 \cdot 7$?

Задача 4.6. Придумайте (положительное целое) число, у которого было бы ровно 6 (положительных) делителей. Тот же вопрос для 7 делителей.

Задача 4.7. Говорить о кратных можно не только для целых чисел, но и для отрезков: один отрезок кратен другому, если второй укладывается в первом целое число раз, то есть если отношение (длина первого)/(длина второго) целое. Докажите, что два отрезка имеют *общую меру* (отрезок, которому они оба кратны) тогда и только тогда, когда они имеют общее кратное.

Задача 4.8. Найдите наименьшее общее кратное отрезков длиной $15/6$ и $21/10$.

Задача 4.9. Могут ли два отрезка не иметь общего кратного?

Если два числа a и b делятся на третье число c, то и их сумма $a + b$ и разность $a - b$ делятся на это c. В самом деле, если $a = kc$ и $b = lc$, то $a + b = (k + l)c$ и $a - b = (k - l)c$.

Для делимости произведения достаточно делимости одного из сомножителей: если a делится на c, то и ab делится на c, каково бы ни было (целое) b. В самом деле, если $a = kc$, то $ab = k(bc) = (kb)c$.

Задача 4.10. Можно ли утверждать обратное: если произведение ab делится на c, то хотя бы один из сомножителей a и b делится на c? (Указание: делится ли 2×9 на 6?)

Множество целых чисел называют *идеалом*, если вместе с любыми двумя числами оно содержит их сумму и разность, и вместе с любым числом оно содержит все его кратные. Используя эту терминологию, можно сказать, что для любого c множество всех кратных числа c является идеалом.

Задача 4.11. Докажите, что произведение любых трёх последовательных целых чисел делится на 3.

Задача 4.12. Докажите, что число $a^3 - a$ делится на 3 при любом целом a.

Задача 4.13. Докажите, что сумма $84 + 85 + 86 + 87 + 88 + 89 + 90$ делится на 7 и на 87.

Задача 4.14. Найдите трёхзначный и семизначный делители числа 103103103.

Задача 4.15. Какие из следующих утверждений верны: (1) если a делится на c, а b не делится на c, то $a + b$ не делится на c; (2) если a не делится на c и b не делится на c, то $a + b$ не делится на c; (3) если a не делится на c и b не делится на c, то ab не делится на c? Докажите верные и приведите контрпримеры к неверным.

Задача 4.16. Известно, что a, b, c, d — положительные целые числа, и $ab = cd$. Докажите, что если a делится на c, то d делится на b.

Задача 4.17. Числа a и b целые, причём $2a + 3b$ делится на 7. Докажите, что $a + 5b$ также делится на 7.

Задача 4.18. Положительное целое число a чётно, но не делится на 4. Покажите, что количество (положительных) чётных делителей a равно количеству (положительных) нечётных делителей a.

Задача 4.19. Докажите, что произведение любых k подряд идущих целых чисел делится на $k! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot k$.

Теперь определим деление с остатком. Пусть b — целое положительное число. Деля на b с остатком, мы связываем предметы в пачки по b в каждой, пока это возможно: количество полных пачек называется *частным* (говорят ещё «неполное частное», чтобы отличать от частного как дроби), и сколько-то предметов останется, их количество и называют *остатком*.

Формально: разделить целое a на целое положительное b означает найти такое целое q (*частное*) и такое целое r (остаток), что

$$a = b \cdot q + r; \quad 0 \leqslant r < b.$$

Ограничения на r понятны: остаётся сколько-то предметов (возможно, ни одного), но меньше b, иначе возникла бы ещё одна целая пачка.

Теперь можно сформулировать теорему: *деление с остатком всегда возможно, притом единственным образом.*

Не очень понятно, что тут доказывать (неужели это не очевидно, если подумать о связывании предметов в пачки?). Тем не менее для педантов можно привести такое доказательство.

Единственность. Если $a = bq + r = bq' + r'$, то $r - r' = b(q' - q)$ и потому $r - r'$ делится на b. Но оба числа r, r' находятся в интервале $0, 1, \ldots, b - 1$, так что их разность (если из большего вычесть меньше) не больше $b - 1$, и может делиться на b, лишь если равна нулю. Поэтому $r = r'$, откуда и $q = q'$.

Существование можно доказать индукцией по a. Для $a = 0$ частное и остаток равны нулю: $0 = 0 \cdot b + 0$. Если $a = bq + r$, то $a + 1 = bq + (r + 1)$. При этом $r + 1 \leqslant b$, так как $r < b$. Если $r + 1 < b$, то для $a + 1$ получаем частное q и остаток $r + 1$. Если же $r + 1 = b$, то $a + 1 = bq + b = b(q + 1) + 0$, получаем частное $q + 1$ и остаток 0.

Программистам будет ближе другое доказательство существования:

```
q:=0; r:=a;
{a=bq+r; r>=0}
пока (r>=b):
   q:=q+1; r:=r-b;
{a=bq+r; r>=0; r<b}
```

Здесь равенство $a = bq + r$ и неравенство $r \geqslant 0$ являются, как говорят, *инвариантом цикла*, они выполняются после любого числа итераций. В самом деле, начальные значения $q = 0$ и $r = a$ удовлетворяют условию $a = qa + r$; итерация цикла происходит при $r \geqslant b$ и потому r после уменьшения на b остаётся неотрицательным, а условие $a = bq + r$ не нарушается, если увеличить q на единицу и одновременно уменьшить r на b. Выход из цикла происходит, когда условие нарушается, то есть $r < b$ (в дополнение к инварианту) — что и требуется определением остатка.

Внимательные читатели, наверно, уже заметили ошибку в рассуждении: оно годится лишь при $a \geqslant 0$, поскольку иначе инвариант $r \geqslant 0$ будет нарушен. Собственно, и индуктивное рассуждение годилось тоже только при $a \geqslant 0$. Так что для отрицательных a возможность деления с остатком надо доказывать отдельно. Пусть $a = -a'$, разделим a' на b с остатком: $a' = bq' + r'$. Если $r' = 0$, то $a = -a' = b(-q') + 0$, так что можно взять $q = -q'$ и $r = 0$. Если $r' > 0$, то можно записать

$$a = -a' = -bq' - r' = b(-q' - 1) + (b - r'),$$

так что можно взять $q = -q' - q$ и $r = b - r'$: поскольку мы предположили, что $0 < r' < b$, то $0 < b - r' < b$, и r попадает в нужный диапазон.

Аналогичная проблема возникает во многих языках программирования: целочисленное деление a на b даёт неполное частное в нашем смысле лишь при $a \geqslant 0$, а, скажем, значение -7 `div` 3 оказывается равным -2, а не -3, как требует наше определение. Тем не менее математически наше определение удобнее, так как при этом сохраняется общая формула для чисел с данным остатком: скажем, числа $3k+1$ при всех целых k при делении на 3 дают остаток 1 (и частное k).

Задача 4.20. Можно ли разрезать шахматную доску 8×8 на прямоугольники 3×1?

Задача 4.21. Найти число вида $100 * **$ (звёздочки обозначают некоторые цифры), которое делится на 547.

Задача 4.22. Книги на столе пытались связывать в пачки по 2, по 3, по 4 и по 5 книг, и каждый раз оставалась одна лишняя. Сколько книг было на столе? (Известно, что их было больше одной и не больше 100.)

Задача 4.23. Число a даёт остаток 6 при делении на 12. Может ли оно давать остаток 12 при делении на 20?

4.4 Сравнения по модулю

Если два числа a и b дают одинаковые остатки при делении на положительное число N, то говорят, что они *сравнимы* по модулю N, и пишут $a \equiv b \pmod N$.

Эквивалентное определение: a и b сравнимы по модулю N, если разность $a - b$ делится на N. (В самом деле, если они дают одинаковый остаток r, то $a = kN + r$, $b = lN + r$, и $a - b = kN - lN = (k - l)N$. Наоборот, если $a - b = mN$, и b даёт остаток r, то $b = lN + r$ и $a = (a - b) + b = mN + lN + r = (m + l)N + r$, то есть a даёт тот же остаток r.)

Можно сказать, что при данном N все целые числа разбиваются на N классов в зависимости от остатков по модулю N: два числа в одном классе сравнимы, а числа в разных классах — нет.

Задача 4.24. Докажите, что числа a^2 и b^2 дают одинаковые остатки при делении на $a-b$, если a и b — положительные целые числа, и $a > b$.

Важное свойство сравнений: чтобы узнать, в какой класс попадет сумма или произведение двух чисел, достаточно знать, в каком классе лежат слагаемые или сомножители: если одно из слагаемых (один из сомножителей) изменить на кратное N, то сумма (произведение) тоже изменится на кратное N.

В самом деле, если к одному из слагаемых прибавить kN, то к сумме тоже прибавится kN, аналогично для разности. С произведением: $(a+kN)b = ab+kbN \equiv ab \pmod N$.

Благодаря этому свойству в выражении, содержащем операции сложения и умножения (или возведение в целую степень, которое сводится к многократному умножению), можно заменять слагаемые или сомножители на сравнимые по модулю N — если результат нам важен лишь по модулю N. Например, можно найти $2^{100} \bmod 7$ (остаток от деления 2^{100} на 7): поскольку $2^3 = 8 \equiv 1 \pmod 7$, то $2^{99} \equiv (2^3)^{33} \equiv 1^{33} = 1 \pmod 7$, так что $2^{100} \equiv 2^{99} \cdot 2 \equiv 1 \cdot 2 = 2 \pmod 7$.

Задача 4.25. Какой остаток даёт число 100^{100} при делении на 99?

Задача 4.26. Найдите две последние цифры числа 99^{1000}.

Сравнения по модулю (пусть не под таким названием) часто встречаются в быту. Скажем, циферблат показывает количество прошедших часов по модулю 12, а также количество минут по модулю 60. Измеряя углы в градусах, мы фактически измеряем число градусов по модулю 360.

Задача 4.27. Как построить угол в 1°, если задан угол в 19°?

Последняя цифра (для положительного целого числа) сравнима с этим числом по модулю 10, а две последние цифры — по модулю 100. В музыке двенадцать полутонов составляют целую октаву, и потом названия нот (до, до диез и так далее) повторяются.

Известный признак делимости на 9 (число делится на 9 тогда и только тогда, когда его сумма цифр делится на 9) можно обобщить и сказать, что число и его сумма цифр сравнимы по модулю 9. Это легко следует из наших рассуждений. Скажем, для четырёхзначного числа:

$$\overline{abcd} = 1000a + 100b + 10c + d \equiv 1a + 1b + 1c + 1d = a + b + c + d \pmod 9,$$

поскольку $10 \equiv 1 \pmod 9$ и, следовательно, 10^2, 10^3,... все сравнимы с 1 по модулю 9.

Задача 4.28. Придумайте признак делимости на 11, использующий знакопеременную сумму цифр.

Если мы хотим представить себе наглядно числа по модулю N, можно вообразить кольцевое шоссе длиной в N километров: на нём километровые столбы будут $0, 1, 2, \ldots, N-1$, далее идёт столб N, который на том же месте, где 0, затем $N+1$ на том же месте, где 1, и так далее. Можно пойти в другую сторону и поставить столб -1 на том же месте, где $N-1$: это соответствует тому, что $(N-1) \equiv (-1) \pmod{N}$.

Для действительных (не целых) чисел равенство дробных частей можно назвать сравнением по модулю 1. Точнее говоря, целой частью числа x называют наибольшее целое число, не превосходящее x; целую часть обозначают обычно $\lfloor x \rfloor$. Разницу $x - \lfloor x \rfloor$ называют дробной частью и иногда обозначают $\{x\}$. Числа с одинаковой дробной частью отличаются на целое число единиц; можно сказать, что они «сравнимы по модулю 1». Для положительных чисел можно сказать (хотя лучше не делать этого на экзамене!), что целая часть — это число до десятичной запятой/точки, а дробная часть — число без целой части, начиная от десятичной запятой/точки. Но для отрицательных чисел это уже не так: $\lfloor -2.3 \rfloor = -3$ и $\{-2.3\} = 0.7$. (Впрочем, и для положительных чисел сказанное не вполне верно: $\lfloor 1.9999\ldots \rfloor = \lfloor 2 \rfloor = 2$.)

Задача 4.29. Нарисуйте на плоскости пары (x, y), для которых $\lfloor x \rfloor = \lfloor y \rfloor$. Тот же вопрос для условия $\{x\} = \{y\}$.

4.5 Таблицы сложения и умножения по модулю N

Мы уже видели таблицы сложения и умножения по модулю 2 и 3. Теперь мы знаем, что аналогичную таблицу можно составить по любому модулю. Например, по модулю 10 получатся таблицы сложения и умножения (рис. 4.1), которые получаются из обычных, если оставить только последнюю цифру.

Скажем, $7 \times 8 = 56$, поэтому в восьмой клетке седьмого ряда мы пишем $56 \bmod 10 = 6$.

Задача 4.30. Найдите последнюю ненулевую цифру числа $10! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot 10$. Тот же вопрос для числа 100!

Задача 4.31. На какие цифры могут оканчиваться точные квадраты (=квадраты целых чисел)?

Задача 4.32. В обычной алгебре уравнение $x^2 = x$ имеет только решения $x = 0$ и $x = 1$. Покажите, что по модулям 10, 100 и 1000 оно имеет ещё два решения. (Скажем, существуют два трёхзначных числа x, для которых $x^2 \equiv x \pmod{1000}$, то есть x^2 оканчивается на x.)

Имея такое «сложение» и «умножение», стоит задуматься о том, верны ли для них знакомые из школьного курса алгебры правила. Скажем, будет ли сложение коммутативно ($a + b = b + a$)? Это можно проверить по таблице: она симметрична относительно главной диагонали. Но можно и так сообразить: чтобы найти, скажем, число в седьмой клетке третьего ряда, надо сложить числа с остатками 3 и 7 и взять

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

×	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	5	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

Рис. 4.1: Таблицы сложения и умножения по модулю 10

остаток по модулю 10. А в третьей клетке седьмого ряда мы складываем числа с остатками 7 и 3. Поскольку обычное сложение коммутативно, то получится одно и то же.

Аналогичное рассуждение показывает, что и другие обычные свойства сложения и умножения сохраняются, если перейти к сложению и умножению по произвольному модулю N:

- $a + (b + c) = (a + b) + c$ (ассоциативность сложения);
- $ab = ba$ (коммутативность умножения)
- $a(bc) = (ab)c$ (ассоциативность умножения)
- $a(b + c) = ab + ac$ (дистрибутивность)

Числа 0 и 1 (точнее, классы чисел, дающие остаток 0 и 1 при делении на N) обладают обычными свойствами:

- $0 + a = 0$;
- $1 \cdot a = a$;
- $0 \cdot a = 0$.

Для каждого остатка a есть противоположный, который в сумме с a равен нулю (а именно, $N - a$ при $a \neq 0$, и нуль при $a = 0$). Его естественно обозначить через $-a$. Как обычно, противоположные остатки можно использовать при вычитании. В самом деле, по определению вычитания, $b - a$ это такое x, что $a + x = b$. В качестве такого x годится $b + (-a)$, потому что $(b + (-a)) + a = b + ((-a) + a) = b + 0 = b$ (пользуемся ассоциативностью, свойствами противоположного, свойствами нуля). Решение уравнения $a + x = b$ не только существует, но и единственно: прибавляя

$(-a)$ к обеим частям, получаем $(-a) + (a + x) = (-a) + b$, но левая часть равна $((-a) + a) + x = 0 + x = x$.

Так что с вычитанием проблем нет и всё как обычно. А вот с делением, то есть с решением уравнений $ax = b$, всё сложнее. Решая такое уравнение, мы должны в a-ой строке найти b, и посмотреть, в каком столбце это b окажется (то есть на что надо умножить a, чтобы получить b).

Задача 4.33. Разделить 3 на 7 по модулю 10, пользуясь таблицей.

При делении 3 на 7 проблем не возникло, потому что число 3 встречается в строке 7 ровно один раз. Вот если бы мы делили 3 на 6, то ничего бы не вышло: в строке 6 таблицы числа 3 нет (там стоят только чётные числа). А при делении 4 на 6 мы не знали бы, что выбрать: и $6 \cdot 4$, и $6 \cdot 9$ кончаются на 4, так что уравнение $6x = 4$ в остатках (или, как говорят ещё, «вычетах») по модулю 10 имеет два решения.

Чтобы в этом разобраться, нам понадобится понятие обратимого элемента в множестве остатков (вычетов).

4.6 Обратимые остатки по модулю N

Остаток (вычет) по модулю N называется *обратимым*, если в произведении с каким-то другим остатком он даёт 1. Другими словами, a обратим, если уравнение $ax = 1$ имеет решение, то есть если в строке a таблицы умножения встречается единица.

Задача 4.34. По таблице умножения по модулю 10 найдите все обратимые элементы.

На обратимые элементы можно делить: *если a обратим, то уравнение $ax = b \pmod N$ имеет единственное решение при любом b.*

В самом деле, обозначим через a^{-1} элемент, который в произведении с a даёт единицу. (Пока мы не знаем, что такой только один, так что обозначим какой-то.) Положим $x = a^{-1}b$. Тогда $ax = a(a^{-1}b) = (aa^{-1})b = 1 \cdot b = b$, так что одно решение уравнения $ax = b$ мы нашли. Оно будет единственным: если $ax = b$, то $a^{-1}(ax) = a^{-1}b$, но $a^{-1}(ax) = (a^{-1}a)x = 1 \cdot x = x$, так что для x есть только одна возможность.

Остаётся выяснить, какие элементы (остатки, вычеты) обратимы по модулю N. Решив задачу в начале этого раздела, мы знаем, что по модулю 10 это будут $1, 3, 7, 9$.

Задача 4.35. При любом $N \geqslant 3$ легко указать как минимум два обратимых элемента. Что это за элементы?

Ответ на вопрос о том, какие остатки обратимы по модулю N составляет первый существенный результат этого раздела.

Определение 4.1. *Взаимно простыми* называются числа, которые не имеют общего положительного делителя, не считая 1.

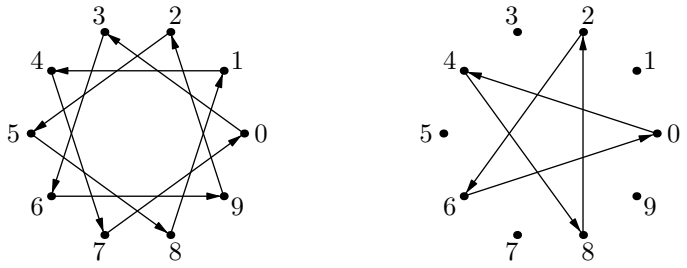
Теорема 4.2. *Обратимыми по модулю N являются те и только те остатки, которые взаимно просты с N.*

Например, по модулю 10 отпадают все чётные числа (у которых общий делитель 2), а также число 5 (общий делитель 5), остаются как раз $1, 3, 7, 9$.

В частности, если N простое (не разлагается в произведение меньших чисел), то общим делителем могут быть только 1 и N, так что все остатки, кроме нуля, взаимно просты с N и обратимы. Для простого модуля действуют все обычные правила сложения и умножения (как в обычной алгебре), и делить можно на всё кроме нуля. (Математики выражают это словами: вычеты по простому модулю p образуют *поле*.)

Прежде чем доказывать эту теорему, представим себе наглядно, что означает обратимость остатка a. Для этого вспомним, что остатки по модулю N можно расположить на круговом шоссе длины N, как автобусные остановки. Запустим маршрут автобуса, которые делает остановки каждые a километров: у него первая остановка будет в a, вторая в $a + a = 2a$, третья в $3a$ (всё по модулю N, естественно). Обратимость означает, что таким странным способом все остановки будут обслужены (в строке таблицы умножения встретятся все остатки).

Вот две картинки, показывающие, что по модулю 10 остаток 3 будет обратимым, а остаток 4 нет:



в первом случае мы проходим все остановки, а во втором не все (только чётные).

Кстати, из этой картинки хорошо видно, что если элемент обратим (мы попадаем в соседнюю остановку), то деление всегда возможно (мы попадём во все остановки). В самом деле, если через k шагов мы попали в соседнюю, то ещё через k мы попадём в следующую и так далее, пройдя через все остановки.

Теперь легко доказать простую часть утверждения: если N и шаг a имеют общий делитель d, то элемент a необратим (мы не попадём в соседнюю остановку). В самом деле, будем отмечать остановки через $d, 2d, 3d$ и так далее от начальной. Поскольку N делится на d, то мы дойдём до N-й (то есть начальной) остановки, пройдя весь круг. Если a кратно d, то a-автобус будет останавливаться только в выделенных остановках, и в соседнюю никогда не попадёт.

Осталось доказать сложную часть утверждения: если N и шаг a не имеют общих делителей, то все остановки будут обслужены. Тут полезно вспомнить о перестановках и отвечающих им ориентированных графах, см. раздел 3.3.6.

В нашем случае вершины графа расположены по кругу, как вершины правильного N-угольника. Стрелки (рёбра графа) соответствуют движению автобуса, проезжающего a перегонов до следующей остановки. Другими словами, из вершины x (остатка по модулю N) стрелка ведёт в вершину $x + a$. По построению из каждой

вершины выходит только одна стрелка, и входит тоже только одна, из вершины $x-a$ (однозначность вычитания). Нам надо доказать, что если a взаимно просто с N, то есть только один цикл, включающий все вершины.

Пусть цикл, начатый из вершины 0, включает только часть вершин. Посмотрим, в какие вершины он попадает. Пусть ближайшая из них (по кругу) имеет номер d. Ясно, что построение цикла можно начать с любой вершины (круг везде одинаков), поэтому если мы начнём с d, то следующая обслуженная вершина будет $2d$. Значит, в цикл входят вершины 0, потом (по кругу) d, потом $2d$ и так далее, пока мы не вернёмся обратно в начальную вершину 0. В результате мы пройдём полный круг в N вершин, двигаясь шагами по d, поэтому N кратно d. С другой стороны, вершина a обслужена (на первой же остановке автобуса), поэтому и a тоже кратно d. Получается, что d — общий делитель N и a. А мы предположили, что они взаимно просты, то есть $d = 1$, что означает, что все вершины (все остатки по модулю N) попадают в один цикл.

Задача 4.36. Покажите, что построенное в этом рассуждении число d делится на любой общий делитель чисел a и N и является их наибольшим общим делителем.

Последняя задача показывает, что уравнение $ax = b \pmod N$ разрешимо тогда и только тогда, когда b делится на НОД(a, N) (наибольший общий делитель чисел a и N).

Задача 4.37. Покажите, что в общем случае граф отображения $x \mapsto x + a$ состоит из НОД(a, N) циклов одинаковой длины $N/\,$НОД(a, N).

Если наше доказательство с циклами и многоугольниками, вписанными в окружность, кажется непонятным, ничего страшного: другое доказательство будет приведено в следующем разделе.

4.7 Обратимые элементы и диофантовы уравнения

Мы только что доказали, что всякий остаток a, взаимно простой с N, имеет обратный. Но как этот обратный найти? Можно пробовать все возможности по очереди. При небольших N это реально, но что делать, если, скажем $N = 5704689200685129054721$ (хотите верьте, хотите проверьте, но это простое число, один из делителей числа $2^{128} + 1$). Проверить столько вариантов даже и с помощью какого-нибудь суперкомпьютера не удастся. Нет ли какого-то более быстрого способа? (Научно говоря, нет ли полиномиального алгоритма — время работы которого было бы ограничено многочленом от числа цифр в N?)

Оказывается, что такой алгоритм есть, и по существу он был известен ещё Евклиду в древней Греции. Для начала переформулируем нашу задачу более симметричным образом. Мы хотим решить сравнение $ax \equiv b \pmod N$. Это сравнение означает, что разность $b-ax$ должна быть целым числом, кратным N, то есть должна равняться Ny при каком-то y. Таким образом, нам надо решить в целых числах уравнение $ax + Ny = b$. Здесь a, N, y — известные целые числа, и мы ищем целые x

и y, при которых левая часть равна правой. Найдя их, мы можем про y забыть, а x будет решением.

Уравнения, в которых нам нужны целые решения, называются *диофантовыми*, в честь другого древнего грека — Диофанта. (Он, правда, не такой древний, как Евклид, и жил уже после Р.Х.)

Таким образом, теорема об обратных элементах сводится к такому утверждению: *уравнение* $ax + Ny = b$ *имеет решение в целых числах тогда и только тогда, когда* b *кратно НОД*(a, N). Из него следует, что при взаимно простых a и N это уравнение имеет решение при любом b.

Чтобы сделать запись ещё более симметричной и забыть о неравноправии a и N в исходной постановке, заменим букву N на b, а b на c. Таким образом, нам надо доказать такое утверждение:

Теорема 4.3. *Пусть* a, b, c *— произвольные целые числа. Уравнение* $ax+by = c$ *имеет целочисленнное решение тогда и только тогда, когда число* c *кратно НОД*(a, b).

Опять в одну сторону это очевидно: если d есть (наибольший) общий делитель a и b, то $ax + by$ всегда делится на d, так что уравнение может иметь решение только при c, кратном d. Интересно обратное утверждение, и мы его докажем с помощью алгоритма Евклида, заодно показав, как можно искать это самое решение на практике.

4.8 Алгоритм Евклида

Давайте сначала посмотрим на этот алгоритм в том виде, как это было у Евклида в его учебнике геометрии («Начала»). Пусть нам надо найти общую меру двух отрезков, то есть третий отрезок, который укладывается целое число раз в первом и во втором. (Другими словами, мы хотим найти меру длины, при которой длины обоих данных нам отрезков будут целыми числами.)

Евклид предлагает делать это так: будем откладывать меньший отрезок m внутри большего M. Если нам повезёт и он уложится целое число раз, то меньший отрезок m и будет общей мерой. Если нет, то он уложится сколько-то раз и что-то (уже меньшее m) останется. Обозначим этот остаток за r. Теперь повторим эту процедуру с отрезками m и r, укладывая меньший из них (то есть r) в большем. Снова либо он уложится без остатка, либо получится остаток r', меньший r, и мы применяем алгоритм к r и r', и так далее. Алгоритм заканчивает свою работу, когда и если

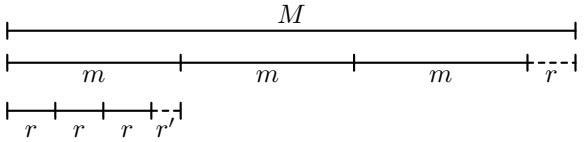
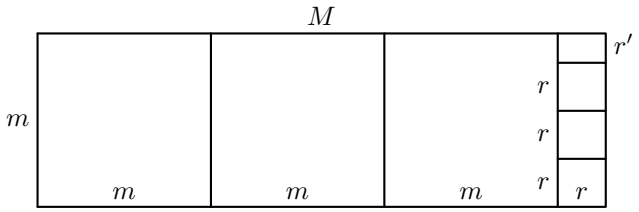


Рис. 4.2: Первые два шага алгоритма Евклида

меньший отрезок укладывается в большем без остатка.

Другой способ представить ту же самую процедуру получается, если считать отрезки сторонами прямоугольников: сначала есть прямоугольник $M \times m$, от которого отрезают квадраты $m \times m$, пока это возможно. Когда останется прямоугольник $r \times m$, в котором $r < m$, от него отрезают квадраты $r \times r$, остаётся прямоугольник $r \times r'$ с $r' < r$. Можно считать, что у нас есть автомат, который отрезает от прямоугольника квадрат со стороной, равной меньшей стороне прямоугольника (он сам разбирается, какая сторона меньше). Мы кромсаем прямоугольник на квадратные части, засовывая остаток снова и снова в этот автомат.



По существу это, конечно, тот же самый процесс, может быть, в немного более наглядной форме.

Основное свойство алгоритма Евклида теперь можно сформулировать так:

Теорема 4.4. *Если исходные отрезки имеют общую меру, то алгоритм заканчивает работу и последний отрезок (тот, что уложится целое число раз) будет наибольшей их общей мерой. Если же исходные отрезки не имеют общей меры, то алгоритм никогда не остановится.*

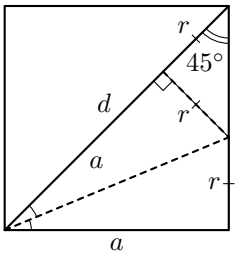
Доказательство. (1) Если исходные отрезки имеют общую меру d, примем её за единицу измерения. Тогда оба наших отрезка имеют целую длину, и мы делим одно целое число (M) на другое (m) с остатком r, потом делим m на r с остатком r' и так дальше, при этом $m > r > r' > \ldots$ (остаток меньше делителя) и все они целые, так что бесконечно это продолжаться не может. В терминах прямоугольников: нарисуем всё по клеточкам на клетчатой бумаге с шагом d, тогда и все разрезы пройдут по клеточкам, и возможных мест разреза конечное число.

(2) Теперь покажем, что если алгоритм заканчивает работу, то последний отрезок будет общей мерой. В самом деле, если принять его за единицу измерения, то предыдущий отрезок будет целым числом (ведь последний укладывается целое число раз), перед ним тоже будет целый отрезок (равный сумме нескольких целых чисел) и пр. В терминах квадратов: если принять самый маленький квадрат при разрезании за клетку на клетчатой бумаге, то все большие квадраты и составленные их них прямоугольники пойдут по линиям сетки, и их стороны будет целыми, то есть сторона маленького квадрата будет общей мерой. (Отсюда уже следует последнее утверждение: если общей меры нет, то алгоритм не заканчивает работу.)

(3) Осталось показать, что последний отрезок будет *наибольшей* общей мерой. Более того, он даже будет кратен любой другой общей мере d. Почему? Приняв d за единицу измерения, мы видим, что оба наших отрезка имеют целую длину, и все

последующие отрезки будут целыми. Значит, ответ алгоритма Евклида тоже целый (кратен d), как мы и утверждали.

Алгоритм Евклида можно использовать, чтобы показать, что сторона квадрата и его диагональ не имеют общей меры. На рисунке сторона бумажного квадрата загнута по биссектрисе так, чтобы она пошла по диагонали. Видно, что сторона a укладывается в диагонали d один раз и остаётся отрезок r. Этот же отрезок можно найти ещё в двух местах, и если его один раз (уже на следующем шаге алгоритма Евклида) отложить на стороне, то останется как раз диагональ квадрата со стороной r. Таким образом, через два шага мы приходим к тому же соотношению между диагональю квадрата и его стороной, и всё повторяется, так что алгоритм никогда не закончит работу.



Задача 4.38. Посмотрите внимательно на картинку и восстановите пропущенные геометрические детали.

Задача 4.39. Каково должно быть отношение сторон прямоугольника, чтобы после отрезания от него одного квадрата получался прямоугольник, подобный исходному? Покажите, что стороны такого прямоугольника несоизмеримы (=не имеют общей меры).

Это отношение называют *золотым сечением*; говорят, что оно встречается в красивых постройках и картинах.

Задача 4.40. Найдите целые положительные числа x, y, z, t, при которых

$$\frac{17}{10} = x + \cfrac{1}{y + \cfrac{1}{z + \cfrac{1}{t}}}$$

При чём тут алгоритм Евклида?

4.9 Алгоритм Евклида и диофантовы уравнения

Весь этот экскурс в историю нужен нам не сам по себе, а чтобы научиться решать уравнения $ax + by = c$ при c, кратном НОД(a, b). Ясно, что достаточно уметь это делать при $c = \text{НОД}(a, b)$, потом можно умножить на отношение $c/\,\text{НОД}(a, b)$, раз оно целое.

Мы можем найти $d = \text{НОД}(a, b)$, разрезая на квадраты прямоугольник $a \times b$, это будет сторона наименьшего из квадратов. Ключевое наблюдение: *все стороны квадратов, появляющиеся в ходе алгоритма, представляются в виде $ax + by$ с некоторыми целыми x и y.* Как говорят, они являются «целочисленными линейными комбинациями» a и b. Это же относится и к последнему отрезку, то есть d, и мы получаем искомое решение.

Почему они будут целочисленными линейными комбинациями? Пусть мы сначала делим a на b с остатком r, тогда $a = bq + r$, и $r = a - bq$ представлен такой комбинацией. Теперь мы делим b на r, получаем остаток r', то есть $b = q'r + r'$, и $r' = b - q'r$ есть целочисленная комбинация b и r. Вспомним, что само r есть комбинация a и b, получится

$$r' = b - q'r = b - q'(a - bq) = b - q'a + q'bq = (qq' + 1)b - q'a,$$

то есть r' тоже есть целочисленная комбинация a и b, и так далее.

Для программистов всё сказанное можно заменить алгоритмом:

```
{a,b - целые положительные числа}
делим a на b, получаем частное q и остаток r
M:=b; m:=r;
Ma:=0; Mb:=1; ma:=1; mb:=-q;
{M>m>=0; НОД(M,m)=НОД(a,b); M=Ma*a+Mb*b; m=ma*a+mb*b}
пока m>0:
    делим M на m, получаем частное q и остаток r
    M, Ma, Mb, m, ma, mb <- m, ma, mb, r, Ma-q*ma,Mb-q*mb
{m=0; НОД(a,b)=НОД(M,m)=НОД(M,0)=M=Ma*a+Mb*b}
ответ: M=НОД(a,b); (x,y)=(Ma,Mb) - решение ax+by=НОД(a,b)
```

Здесь стрелка `<-` означает одновременное присваивание: шесть переменных получают новые значения, указанные справа, при этом при вычислении выражений в правой части используются старые значения. Если, как это принято, выполнять присваивания последовательно, то надо написать

```
M_new:=m; Ma_new:=ma; Mb_new:=mb
m_new:=r; ma_new:=Ma-q*ma; mb_new:=Mb-q*mb
M:=M_new; Ma:=Ma_new; Mb:=Mb_new
m:=m_new; ma:=ma_new; mb:=mb_new
```

Этот алгоритм иногда называют «расширенным алгоритмом Евклида» (по-английски “extended Euclidean algorithm”). Что можно сказать о количестве операций в этом алгоритме? Можно заметить, что за два шага алгоритма Евклида больший отрезок (большее число для алгоритма с целыми числами) уменьшается по крайней мере вдвое. В самом деле, если меньший отрезок не превосходит половины большего, то это случится уже за один шаг; если же нет, то на первом шаге частное равно 1, а остаток не больше половины большего отрезка, так что за два шага это всё равно случится. Поэтому число шагов для n-битовых чисел равно $O(n)$. Каждый

шаг требует деления с остатком — если это делать «уголком», как в школе, то число действий будет $O(n^2)$, так что всего получается $O(n^3)$. Конечно, если следовать определению буквально и делить с помощью последовательного вычитания (как мы делали, доказывая существование частного и остатка), то будет плохо (число действий будет экспоненциальным).

Задача 4.41. Напишите алгоритм решения уравнения $ax + by = \text{НОД}(a, b)$, использующий три соотношения:

- $\text{НОД}(2a, 2b) = 2\,\text{НОД}(a, b)$;
- $\text{НОД}(2a, b) = \text{НОД}(a, b)$ при нечётном b;
- $\text{НОД}(a, b) = \text{НОД}(a - b, b)$, которое верно всегда, но нужно при нечётных a и b и $a \geqslant b$.

В каждом случае задачу решения уравнения можно рекурсивно свести к той же задаче для новой пары. Сколько действий требует этот вариант алгоритма Евклида?

Задача 4.42. Известно, что алгоритм Евклида (стандартный вариант, см. с. 157) для пары целых положительных чисел (a, b) требует n шагов. Каково наименьшее возможное значение a? (Считаем, что $a \geqslant b$.)

Если не интересоваться алгоритмической стороной дела, то можно доказать теорему о разрешимости уравнений $ax + by = c$ следующим образом. Напомним, что мы называли *идеалом* множество целых чисел, которое вместе с любыми двумя элементами содержит их сумму и разность, и вместе с любым элементом содержит все его кратные. В качестве примера идеала мы приводили множество I_c всех кратных некоторого числа c. Такие идеалы алгебраисты называют *главными*. Оказывается, что (для целых чисел) других и нет. В самом деле, пусть I — некоторый идеал. Если он состоит только из нуля, то $c = 0$. Если в нём есть ненулевые числа, возьмём минимальное по модулю число c. Будем считать, что оно положительно (перейдя к $-c$, если надо). Теперь ясно, что все числа в I кратны c: если какое-то $x \in I$ даёт ненулевой остаток r при делении на c, то $x = qc + r$ и $r = x - qc$ есть разность двух чисел из I и потому тоже принадлежит I, что противоречит минимальности c. Такое число c, при котором $I = I_c$, называется *образующей* идеала I.

Пусть теперь a, b — произвольные целые числа, а I состоит из всех целых чисел, которые можно представить в виде $ax + by$. Это будет идеал, и по доказанному он совпадает с некоторым I_c. Поскольку $a, b \in I$, то они кратны c, так что c будет общим делителем. С другой стороны, любой общий делитель d чисел a, b делит все элементы I, в том числе и c, так что c будет наибольшим общим делителем (и кратен любому делителю — это мы тоже доказали заодно). Наконец, c представим в виде $ax + by$ по построению I.

Задача 4.43. Покажите, что общие кратные двух чисел a и b образуют идеал, и выведите отсюда, что наименьшее общее кратное $\text{НОК}(a, b)$ является делителем любого общего кратного.

Задача 4.44. Покажите, что НОД$(a,b)\cdot$ НОК$(a,b)=ab$.

Задача 4.45. Покажите, что для данных целых a,b числа x, при которых ax кратно b, образуют идеал. Как найти его образующую, зная a, b и НОД(a,b)?

Задача 4.46. При каких целых a,b,c,d уравнение $ax+by+cz=d$ разрешимо в целых числах?

4.10 Однозначность разложения на множители

Теперь мы можем доказать теорему об однозначности разложения целых чисел на простые множители, которую иногда торжественно называют «основной теоремой арифметики». Напомним, что целое число $p>1$ называется *простым*, если оно не разлагается в произведение меньших чисел (то есть не имеет положительных делителей, кроме 1 и p).

Теорема 4.5. *Всякое целое положительное число, большее* 1*, разлагается на простые множители, причём единственным образом: любые два разложения отличаются только перестановкой сомножителей.*

(Можно сказать, что единица тоже разлагается в произведение нуля простых множителей — если принять, что произведение нуля сомножителей равно 1. Как вы думаете, кстати, что следует считать суммой нуля слагаемых?)

Трудность с этой теоремой в том, что она (видимо, со школы) кажется само собой разумеющейся, и непонятно, что тут доказывать. Тем не менее доказательство требует некоторых усилий (которые мы по большей части уже проделали).

Существование разложения совсем просто. Если данное число N простое, то получилось разложение из одного сомножителя. Если нет, то $N=ab$ для каких-то меньших a,b. Если a и b простые, то хорошо, если нет, то разложим их в произведение меньших и так далее до тех пор, пока дальше уже ничего не раскладывается, поскольку числа простые. (Формально говоря, мы рассуждаем по индукции и считаем, что для меньших чисел a и b существование разложения уже известно.)

Единственность разложения. Пусть некоторое число N имеет два разложения

$$N=p_1\cdot p_2\cdot\ldots\cdot p_n=q_1\cdot q_2\cdot\ldots\cdot q_m$$

(они могут отличаться и числом сомножителей, m не обязано равняться n). Мы хотим получить противоречие. Сократим на общие сомножители (если они есть). Если сократится не всё, то получим два разложения одного числа, не имеющих общих сомножителей

$$p_1\cdot p_2\cdot\ldots\cdot p_n=q_1\cdot q_2\cdot\ldots\cdot q_m.$$

Как говорят, «без ограничения общности» можно предположить, что общих сомножителей нет (сократив на них, если есть).

В чём тут противоречие? С одной стороны, левая часть делится на p_1 (можно было бы взять любой другой p_i, если там несколько сомножителей). А правая часть

равна произведению чисел, ни одно из которых не делится на p_1: они ведь простые и p_1 среди них по предположению нет. Осталось доказать, что такого не бывает, то есть доказать следующую лемму.

Лемма 4.6. *Если p — простое число, то произведение чисел, не делящихся на p, не может делиться на p.*

По существу мы уже это доказали: в терминах вычетов по модулю p нужно доказать, что произведение ненулевых вычетов не равно нулю. А мы знаем, что если $a \neq 0 \pmod p$, то a взаимно просто с p, поэтому a обратим и уравнение $ax = 0$ имеет единственное решение (нулевое).

Более подробно. Во-первых, достаточно доказать лемму для двух сомножителей. Если есть, скажем, три числа a, b, c, не делящиеся на p, то мы применяем лемму для двух сомножителей a и b и заключаем, что ab не делится на p. После этого уже можно применить лемму к двум сомножителями ab и c и заключить, что $(ab)c$ не делится на p. (Аналогично для любого числа сомножителей — формально говоря, мы используем индукцию по числу сомножителей.) Лемма доказана, и тем самым мы завершили доказательство теоремы 4.5.

Для двух сомножителей мы можем доказать более общий факт:

Лемма 4.7. *Если ab делится на n, и при этом a взаимно просто с n, то b делится на n.*

(Более общий он потому, что если a не делится на простое p, то a взаимно просто с p — других общих делителей быть не может.)

Доказательство. Если $\text{НОД}(a, n) = 1$, можно найти x, y, для которых $ax+ny = 1$. Умножим это равенство на b, получим, что

$$b = abx + ny.$$

Осталось заметить, что оба слагаемых в правой части делятся на n: по предположению ab делится на n, и очевидным образом ny делится на n. Значит, и сумма (то есть b) делится на n, что и требовалось доказать.

Задача 4.47. Как вычислить $\text{НОД}(a, b)$ и $\text{НОК}(a, b)$, зная разложения a и b на простые множители? Докажите, что $\text{НОД}(a, b) \cdot \text{НОК}(a, b) = ab$, используя (очевидное) равенство $\min(x, y) + \max(x, y) = x + y$.

Однозначность разложения на множители делает очевидными многие утверждения, например, такое: *если число a делится на b и на c, и при этом b и c взаимно просты, то a делится на bc.* В самом деле, если a делится на b, то разложение a на множители содержит все множители из разложения b (и ещё что-то), поскольку можно разложить по отдельности b и a/b. Аналогичным образом разложение a содержит все множители из разложения c. Но общих множителей у b и c нет, так что это разные множители, и в разложении a можно выделить разложение для bc.

Можно, конечно, обойтись и без теоремы об единственности разложения: поскольку b и c взаимно просты, то $bx + cy = 1$ при некоторых x и y, так что $a = a(bx + cy) = abx + acy$, и оба слагаемых abx и acy делятся на bc (почему?) — но это всё же выглядит более искусственно...

Задача 4.48. Докажите, что если a взаимно просто с числами b и c по отдельности, то оно взаимно просто с bc. (Можно воспользоваться разложением на множители, а можно и перемножить равенства $ax + by = 1$ и $au + cv = 1$.)

4.11 Китайская теорема об остатках

Как найти положительное целое число, которое даёт остаток 1 при делении на 23 и остаток 1 при делении на 37? Понятно как: надо взять число, делящееся и на 23, и на 37, например, $23 \cdot 37$, и прибавить единицу.

Немного сложнее найти положительное целое число, которое даёт остаток 22 при делении на 23 и 36 при делении на 37. Надо заметить, что если прибавить 1, то получится число, делящееся и на 23, и на 37, так что можно попробовать $23 \cdot 37 - 1$, и всё получится.

А что делать, если надо найти число, дающее (скажем) остаток 17 при делении на 23, и одновременно остаток 24 при делении на 37? И вообще, возможно ли это? Оказывается, что возможно, причём для любой комбинации остатков, поскольку 23 и 37 взаимно просты. Это гарантирует следующая теорема, традиционно называемая «китайской теоремой об остатках» (Chinese remainder theorem).

Теорема 4.8. *Пусть числа m и n взаимно просты, и пусть u и v — любые целые числа. Тогда можно найти число x, для которого $x \equiv u \pmod m$ и одновременно $x \equiv v \pmod n$.*

Прежде чем доказывать это, посмотрим на какой-нибудь пример. Числа 23 и 37 слишком большие, возьмём, скажем, 3 и 4 и составим таблицу:

x	0	1	2	3	4	5	6	7	8	9	10	11	
$x \bmod 3$	0	1	2	0	1	2	0	1	2	0	1	2	
$x \bmod 4$	0	1	2	3	0	1	2	3	0	1	2	3	

(Дальше можно уже не продолжать, поскольку 12 делится и на 3, и на 4, и всё повторится с начала.)

Так вот, китайская теорема учит, что в двух нижних строках встретятся все возможные комбинации остатков: любой из остатков $0, 1, 2$ комбинируется с любым из остатков $0, 1, 2, 3$ (всего как раз 12 вариантов: три варианта mod 3 комбинируются с четырьмя вариантами mod 4, так что каждая комбинация встречается ровно по одному разу.)

Эта таблица подсказывает доказательство китайской теоремы об остатках. Построим аналогичную таблицу, записав в первой строке числа $0, 1, 2, \ldots, mn - 1$, во второй строке их остатки при делении на m (получится $0, 1, 2, \ldots, m-1$, повторённое

n раз), а в третьей строке их остатки при делении на n (получится $0, 1, 2, \ldots, n-1$, повторённое m раз). Покажем, что все комбинации остатков (их будет mn) встретятся ровно по одному разу. Для этого заметим, что никакая комбинация не может повториться дважды: если числа u и v дают одинаковые остатки и при делении на m, и при делении на n, то их разность $u - v$ делится и на m, и на n. Как мы уже видели, это значит, что $u - v$ кратно mn (поскольку m и n взаимно просты), а у нас все остатки при делении на mn представлены по одному разу.

Осталось заметить, что если ни одна из mn комбинаций не повторяется в имеющихся mn столбцах, то придётся использовать все mn комбинаций. (Если N голубей разместить в N норах, причём в каждой норе не больше одного голубя, то все норы будут заполнены. Это называется по-английски pigeon-hole principle, а по-русски принципом Дирихле, поскольку Дирихле использовал это соображение, изучая приближения действительных чисел рациональными.)

Задача 4.49. Докажите, что если числа m, n, k попарно взаимно просты, то для любых u, v, w найдётся такое x, что

$$x \equiv u \pmod{m}; \quad x \equiv v \pmod{n}; \quad x \equiv w \pmod{k}.$$

Приведённое нами доказательство китайской теоремы об остатках не даёт алгоритма нахождения искомого x. Но легко понять, что эта задача сводится к уже известным. В самом деле, нам нужно найти x, для которого $x \equiv u \pmod{m}$ и $x \equiv v \pmod{n}$. Первое условие означает, что $x = u + tm$, и надо искать t. Второе условие, переписанное в терминах t, означает, что $u + tm \equiv v \pmod{n}$, то есть надо решить сравнение $tm \equiv -u \pmod{n}$, что мы уже умеем делать с помощью алгоритма Евклида (надо найти обратный для m в вычетах по модулю m, используя взаимную простоту m и n).

Заодно мы получили другое доказательство китайской теоремы об остатках.

4.12 Малая теорема Ферма

Этот раздел называется «малая теорема Ферма», хотя исторически это не вполне справедливо: как и в случае «большой», или «последней» теоремы Ферма (о том, что уравнение $x^n + y^n = z^n$ при $n > 2$ не имеет решений в целых положительных числах), сам Ферма не предложил доказательства — то ему полей не хватало (с большой теоремой Ферма), то он боялся надоесть адресату (je vous envoierois la démonstration, si je n'appréhendois d'être trop long). Но в этом случае не понадобилось ждать до конца XX века, доказательство вроде бы знал Лейбниц в том же XVII веке, а опубликовано оно было Эйлером в 1736 году, и вполне можно допустить, что это доказательство было известно Ферма.

Вот о каком утверждении идёт речь:

Теорема 4.9. *Если p — простое число, то*

$$a^{p-1} \equiv 1 \pmod{p}$$

при любом a, не делящемся на p.

Вот одно из доказательств (пожалуй, наиболее естественное). Рассмотрим все ненулевые остатки по модулю p. Их всего $p-1$: от единицы до $p-1$. Сделаем их вершинами графа, проведя стрелки из x в ax (умножение по модулю p). Разница с тем графом, что у нас уже был, двоякая: во-первых, мы рассматриваем не все p остатков, а только $p-1$ ненулевых остатков. Но главное, мы не прибавляем a, а умножаем на a: стрелка из x ведёт не в $x+a$, а в ax. Так что красивых картинок с правильными звёздчатыми многоугольниками, как раньше, не получится.

Но тем не менее из каждой вершины выходит одна стрелка (по построению) и в каждую вершину входит одна стрелка (поскольку a не делится на p, оно обратимо и уравнение $ax = b$ имеет единственное решение при любом b, то есть в b входит ровно одна стрелка). Значит, граф, как и раньше, разбивается, на циклы.

Что это за циклы? Начнём с какой-то вершины, например, с остатка 1, и будем идти по стрелкам, пока цикл не замкнётся:

$$1 \mapsto a \mapsto a^2 \mapsto a^{k-1} \mapsto a^k = 1$$

Здесь k — минимальная степень a, равная 1 (как мы знаем, цикл может замкнуться, только придя в начальную вершину). Если начать построение с какой-то вершины b вместо 1, то получится цикл того же размера:

$$b \mapsto ab \mapsto a^2b \mapsto a^{k-1}b \mapsto a^kb = 1b = b.$$

Почему? Ясно, что если $a^k = 1$, то и $a^kb = b$, так что после k шагов цикл замкнётся. Но надо понять, почему этот цикл не замкнётся раньше. В самом деле, если $a^lb = b$, то можно домножить на b^{-1} (ведь остаток b тоже обратим, и можно взять обратный к нему остаток b^{-1}), и получится $a^l = 1$, а мы предположили, что k минимальное.

Таким образом, $p-1$ ненулевых остатков разбиваются на циклы одинакового размера k, так что $p-1$ делится на k, то есть $p-1 = km$ при каком-то m. Тогда

$$a^{p-1} = a^{km} = (a^k)^m = 1^m = 1 \pmod p,$$

что и требовалось доказать.

Есть и другие доказательства, приведём два любопытных.

Первое доказательство мы уже обсуждали в главе про подчёты на с. 60. Достаточно доказать, что $a^p \equiv a \pmod p$, потом можно сократить на обратимый элемент a. Представим себе, что имеется a цветов, и ими нужно раскрасить круг из p равных секторов (каждый сектор в какой-то цвет), причём раскраски, отличающиеся лишь поворотом круга, считаются за одну. Сколькими способами это можно сделать? Можно все сектора красить в один цвет, это можно сделать a способами (по числу цветов). А можно использовать более одного цвета, на это остаётся $a^p - a$ способов, но они делятся на группы по p, отличающихся поворотами (каждую неоднородную раскраску можно повернуть p различными способами). Получается ответ $a + \frac{a^p-a}{p}$. Но число способов должно быть целым, так что $a^p - a$ делится на p.

Второе доказательство использует отображение остатков по модулю p в себя, заданное формулой $f(x) = x^p$. Нам надо доказать, что оно тождественное, то есть что $f(x) = x$ при всех x. Это вытекает из трёх его свойств:

- $f(0) = 0$;
- $f(1) = 1$;
- $f(x + y) = f(x) + f(y)$.

В самом деле, третье свойство можно по индукции распространить на любое число слагаемых, скажем,

$$f(x + y + z) = f((x + y) + z) = f(x + y) + f(z) = f(x) + f(y) + f(z)$$

и так далее, поэтому

$$f(x) = f(1 + 1 + \ldots + 1) = f(1) + f(1) + \ldots + f(1) = 1 + 1 + \ldots + 1 = x$$

(многоточия обозначают суммы из x слагаемых). Так что достаточно доказать эти три свойства. Первые два очевидны, а третье следует из бинома Ньютона: в разложении

$$(x + y)^p = x^p + \frac{p!}{1!(p-1)!}x^{p-1}y^p + \frac{p!}{2!(p-2)!}x^{p-2}y^2 + \ldots + \frac{p!}{(p-1)!1!}x^1y^{p-1} + y^p$$

все биномиальные коэффициенты, кроме двух крайних, делятся на p (у них есть p в числителе, и это простое число, которое не может сократиться с меньшими числами в знаменателе), так что по модулю p остаются только два крайних члена, что и требовалось.

Алгебраисты сказали бы, что эти три свойства верны в любом поле «характеристики p», где сумма p единиц равна нулю — их можно ещё дополнить четвёртым свойством $f(xy) = f(x)f(y)$. Но отображение f уже не будет (вообще говоря) тождественным; его называют «автоморфизмом Фробениуса» (слово «автоморфизм» означает, что выполнены эти четыре свойства, а Фробениус — немецкий математик, в честь которого он назван).

С точки зрения практиков, теорема Ферма может использоваться как способ убедиться, что число не простое. Скажем, можно вычислить $2^8 = 256 \equiv 4 \pmod 9$ и заключить отсюда, что число 9 не простое (если бы мы этого и так не знали). Это выглядит глупо, но на самом деле имеет вычислительный смысл: возвести какое-то a в степень $p - 1$ по модулю p не так сложно (надо вычислять $a, a^2, a^4, a^8, \ldots \bmod p$ последовательными возведениями в квадрат, а потом перемножить нужные из них, следуя двоичному разложению для $p - 1$), это требует полиномиального числа действий (от n для n-битового числа p), и если нам повезёт и получится не 1 по модулю p, то мы сможем убедиться, что число p составное. Можно назвать число a, для которого $a^{p-1} \not\equiv 1 \pmod p$, «свидетелем» того, что p составное; как мы обсудили, если такой свидетель известен, «проверить его показания» можно быстро.

Собственно говоря, проверка разложения на множители (перемножение сомножителей) тоже полиномиальна, но преимущество нового способа в том, что свидетелей обычно бывает много, и есть шанс наткнуться на них, взяв наугад число от 1 до $p-1$. (А на разложение на множители так просто не наткнуться; полиномиального алгоритма разложения на множители не известно, и для чисел из нескольких тысяч цифр искать разложение на множители никто не умеет.) Так что если вам нужно быстро найти какое-нибудь большое простое число, можно выбрать случайную последовательность цифр и проверить теорему Ферма для (скажем) $a = 2, 3, 5$; если она не выполнена, число не простое и надо взять другую случайную последовательность цифр, пока этот тест не будет пройден. С большой вероятностью этот способ даёт простые числа.

Задача 4.50. Мы предлагали проверять теорему Ферма для $2, 3, 5$, но почему-то пропустили 4. Как вы думаете, почему?

4.13 Функция Эйлера и теорема Эйлера

На самом деле приведённое доказательство малой теоремы Ферма имеет смысл не только для простых модулей. Пусть N — какое-то число, не обязательно простое. Мы можем по-прежнему рассмотреть остатки по модулю N, но взять только взаимно простые с N. (В алгебре это называют «мультипликативной группой кольца вычетов по модулю N».) Произведение двух таких остатков тоже взаимно просто с N, и все они обратимы, так что взяв какой-то остаток a среди них, мы можем построить ориентированный граф $x \mapsto xa$, и в нём из каждой вершины будет исходить одна стрелка и в каждую вершину будет входить одна стрелка. Этот граф разбивается на циклы, и по тем же причинам все циклы будут одинаковой длины. Эта длина видна на примере цикла, начинающегося с 1:

$$1 \mapsto a \mapsto a^2 \mapsto \ldots \mapsto a^k = 1,$$

где k — наименьшая степень a, равная 1 по модулю N. Значит, k является делителем числа остатков, взаимно простых с N, и мы получаем такой результат, называемый «теоремой Эйлера».

Теорема 4.10. *Пусть $N > 1$ — произвольное целое число, а $\varphi(N)$ — количество остатков среди $0, 1, \ldots, N-1$, взаимно простых с N. Пусть a — один из этих остатков. Тогда*

$$a^{\varphi(N)} \equiv 1 \pmod{N}.$$

Определённую таким образом функцию φ называют *функцией Эйлера* и традиционно обозначают буквой φ. Если число N простое, то $\varphi(N) = N - 1$, и теорема Эйлера превращается в малую теорему Ферма.

Как вычислить функцию Эйлера $\varphi(N)$, зная N? Это несложно, если мы дополнительно знаем разложение числа $N-1$ на простые множители (как мы уже говорили, разложение на множители вычислительно сложно), и делается с помощью таких свойств:

- $\varphi(p^n) = p^n(1 - 1/p) = p^{n-1}(p-1)$ для простого p;
- $\varphi(uv) = \varphi(u)\varphi(v)$, если u и v взаимно просты.

Первое свойство позволяет найти φ для степеней простых чисел, а второе свойство позволяет собрать из них любое число. Свойства эти нам по существу уже известны. Первое говорит, что доля чисел, не взаимно простых с p^n (то есть делящихся на p) составляет $1/p$ среди всех чисел, и их надо вычеркнуть. Второе легко доказать, сославшить на китайскую теорему об остатках и на то, что взаимная простота с uv означает одновременную взаимную простоту с u и v. В терминах теории вероятностей можно сказать, что определённые на вычетах по модулю uv функции «остаток по модулю u» и «остаток по модулю v» независимы и имеют равномерное распределение, и вероятность события «быть взаимно простым с uv» можно вычислять как произведение вероятностей событий «быть взаимно простым с u» и «быть взаимно простым с v».

Второе свойство называют *мультипликативностью* функции Эйлера. В теории чисел встречаются разные мультипликативные функции.

Задача 4.51. Покажите, что функция $d(n)$, равная числу положительных делителей числа n, мультипликативна.

Задача 4.52. Покажите, что функция $\sigma(n)$, равная сумме всех положительных делителей числа n, мультипликативна.

4.14 Что дальше?

Мы познакомились с базовыми свойствами сложения и умножения по модулю N, или, как говорят, *кольца вычетов по модулю* N, стараясь не использовать алгебраического языка. Есть и много других интересных (и не таких сложных) результатов. Скажем, есть такая теорема: при простом N найдётся остаток a по модулю N, при котором все элементы $1, a, a^2, \ldots, a^{N-2}$ различны (следующий элемент по теореме Ферма равен 1). Или другой результат: ровно половина всех ненулевых вычетов по простому модулю являются точными квадратами (квадратами других вычетов). В принципе и их можно было бы изложить, не используя языка алгебры (группы, подгруппы, кольца, гомоморфизмы, многочлены и пр.), но в этом нет смысла: пора с ним познакомиться. Это можно сделать по любому учебнику алгебры — есть старинный «Курс высшей алгебры» Куроша [14] (когда-то бывший стандартным мехматским учебником), или несколько более новый учебник Кострикина [12, 13] (тоже по материалам мехматского курса). Стандартные учебники, покрывающие существенно больший материал — классические книги ван дер Вардена [2] и Ленга [16], или более новые книги Винберга [5] и Городенцева [6].

Где все эти алгебраические премудрости используются «на практике»? Вот игрушечный пример: пусть вы хотите послать кому-то номер своей кредитной карточки, но боитесь, что ваше сообщение будет перехвачено. Тогда есть такой способ: каждую

цифру a_i этого номера можно представить в виде суммы по модулю 10 случайно выбранной цифры r_i и недостающей цифры $s_i = a_i - r_i$, и послать последовательности цифр r_i и s_i (той же длины, что и номер карточки) в двух разных электронных письмах. Получив оба этих письма, адресат легко восстановил $a_i = r_i + s_i$. Но если враг перехватит только одно из них, а второе нет — то у него не будет никакой информации о вашей карточке. В самом деле, сообщение r не содержит никакой информации (это просто случайная последовательность цифр), и сообщение s в отдельности тоже (потому что при данном a это сообщение может быть любым, и все они равновероятны).

Конечно, защита тут слабая — лишь на случай перехвата одного из писем. Но, скажем, если одно посылается по электронной почте, а другое — как сообщение в Skype, то враг должен получить доступ к обоим сервисам (они разные, и каналы передачи сообщений разные). Существуют и более надёжные методы передачи сообщений по открытой сети, прежде всего так называемая система RSA (она используется в банках, в протоколах https, ssh, цифровой подписи PGP и вообще на каждом шагу). И она тоже основана на арифметике остатков. Наших знаний достаточно, чтобы понять, почему она не искажает сообщения — но вот доказать, что её нельзя взломать, мы не можем, и никто пока не может — если кто-то вдруг научится быстро разлагать большие числа на множители, эта система, а вместе с ней и все банки, рухнет. Но для этого (быстрого разложения) нужны либо новые алгоритмы, либо квантовые компьютеры (которые, если вдруг их удастся построить, могут разлагать числа на множители — это обнаружил P. Shor). Другие системы шифрования основаны на более сложных алгебраических методах (эллиптические кривые) — так что знания по алгебре лишними не будут.

Вообще алгебраические методы незаменимы там, где нужно построить «регулярно устроенное» семейство объектов с хорошими свойствами. Попробуйте, например, найти в 7-элементном множестве как можно больше 3-элементных подмножеств, любые два из которых имеют общий элемент. (Сколько дней подряд можно назначать по три дежурных в походе из 7 человек, чтобы наряды на любые два дня имели ровно одного общего?) Не так просто — если не знать, что двумерные подпространства в трёхмерном пространстве над полем вычетов по модулю 2 содержат (не считая нуля) по три элемента и любые два из них пересекаются по одномерной прямой, содержащей ровно один ненулевой элемент.

Часть II

Основные конструкции

Лекция 5

Множества и логика

Мы уже не раз использовали понятие множества в предыдущих главах, с помощью него оказывается удобно формулировать разные утверждения и вести доказательства. Это понятие настолько важно в математике, что о множествах стоит поговорить отдельно, что мы и сделаем в этой главе.

Вообще, понятие множества лежит в основе современной формализации математики. Центральную роль в этом играет аксиоматика теории множеств Цермело-Френкеля. Стоит отметить, однако, что мы не будем излагать теорию множеств формально. Это большое и тяжелое дело. Наша цель более скромная — объяснить, как понятия множества, операций с множествами и свойств множеств используются для формулировки математических утверждений и описания математических рассуждений. Далее мы будем при необходимости свободно использовать теоретико-множественный язык.

5.1 Основные свойства множеств и операции с множествами

Неформально, множество — это совокупность каких-то элементов. Природа элементов неважна, как и возможные взаимоотношения между элементами. Единственное, что существенно для определения множества — это какие элементы в него входят, а какие — нет. При этом множество не может содержать часть элемента[1]: всякий элемент либо входит в множество, либо нет.

С примерами множеств мы уже сталкивались. Можно рассматривать множество трехзначных натуральных чисел, множество всевозможных натуральных чисел, множество десятичных цифр, множество ребер в каком-нибудь графе. Но на ряду с этими естественными множествами мы имеем право рассмотреть и какое-нибудь неестественное множество. Например, мы могли бы рассмотреть множество, состоящее из четных натуральных чисел, а также из всех белых медведей. Более того, в принципе элементами какого-нибудь множества могли бы быть другие множества, так что множества могут быть устроены довольно хитро. Наша цель в этой

[1]Однако дробным частям элементов тоже можно придать разумный смысл. Мы обсудим это при изучении теории вероятностей, см. лекцию 10.

главе, обсудить множества как *абстрактный* объект. То есть нас сейчас не интересует, из чего именно состоят множества. Все, что мы здесь докажем, должно быть верно для всех множеств, как бы они ни были устроены.

Множество полностью определяется своими элементами. Два множества A и B называются *равными*, если каждый элемент множества A является элементом множества B, а каждый элемент множества B является элементом множества A. Это определение равенства множеств является по сути самым важным из свойств множеств. Всюду дальше, когда мы будем говорить о равных множествах, мы будем иметь в виду это определение.

Среди множеств есть одно особенное множество — пустое, — которое не содержит никаких элементов. Пустое множество обозначается $\varnothing$.

Задача 5.1. Используя определение равенства множеств, аккуратно докажите единственность пустого множества (то есть, что любые два пустых множества равны между собой).[2]

Чтобы с множествами было удобно работать, нам нужно определить еще несколько ключевых понятий, связанных с множествами, и зафиксировать для них обозначения.

Говорят, что элемент x *принадлежит* множеству A, если он является его элементом. Стандартное обозначение этого утверждения[3] такое: $x \in A$. Соответственно, отрицание принадлежности, то есть «x не принадлежит множеству A», обозначается $x \notin A$.

Если удалить из множества часть элементов, то мы снова получим множество. Такие множества, называются *подмножествами* изначального множества. По определению множество A является подмножеством множества B, если каждый элемент множества A принадлежит множеству B. Обозначается это так: $A \subseteq B$ (словами: «A подмножество множества B» или «A включено в B»).

Равенство множеств и включение напоминают равенство и неравенства для чисел, а также достижимость в неориентированных и ориентированных графах, и обладают похожими свойствами. Для равенства это рефлексивность «$A = A$», симметричность «если $A = B$, то $B = A$» и транзитивность «если $A = B$, $B = C$, то $A = C$». Для включения множеств это рефлексивность «$A \subseteq A$», и транзитивность «если $A \subseteq B$, $B \subseteq C$, то $A \subseteq C$» и ещё одно свойство антисимметричности: «если $A \subseteq B$ и $B \subseteq A$, то $A = B$» (это сразу следует из определений — проверьте!). Эти свойства часто возникают в математике, мы подробнее поговорим о них в главах 7 (про отношения) и 9 (про частичные порядки).

Нам осталось разобраться с тем, как же нам задавать множества. Если элементов в множестве мало, его можно задать, просто перечислив все эти элементы. При

[2] В одном из учебников математики для младших школьников была задача «приведите несколько примеров пустых множеств». Вероятно, авторы имели в виду ответы типа «множество слонов в посудной лавке и множество чёрных кошек в пустой комнате», но математики сказали бы, что это два разных способа задачть *одно и то же* пустое множество.

[3] Это утверждение: элемент либо принадлежит множеству, либо нет; поэтому $x \in A$ либо истинно, либо ложно.

этом принято заключать список элементов в фигурные скобки. Хотя на письме мы вынуждены записывать множества в каком-то порядке, этот порядок не играет роли. Поэтому

$$\{0, 2, 4, 6, 8\} = \{4, 2, 0, 8, 6\}$$

и это просто разные обозначения множества чётных цифр.

Если элементов в множестве мало, то такой способ удобен, но если множество большое, то уже нет.

А что будет, если написать $\{0, 2, 2, 4, 6, 8, 8\}$? Ещё одно обозначение того же самого множества чётных цифр: элемент либо входит, либо нет, но не может входить дважды — так что будет то же самое множество (хотя в сбивающей с толку и провоцирующей на ошибки записи). Иногда говорят о *мультимножествах*, разрешая такие «кратные вхождения», но в этой книге речи о них не будет.

Пример 5.2. Множество простых десятизначных чисел содержит достаточно мало элементов, чтобы написать программу, которая перечислит все его элементы. Однако на бумаге такой список займет очень много места, так что пользоваться им уже неудобно.

Но с другой стороны, только что нам удалось очень коротко описать это множество (“Множество простых десятизначных чисел”). Именно так и выходят из таких ситуаций — множество можно задать, описав свойства, которым должны удовлетворять его элементы.

Бывают и такие множества, которые уже никак нельзя задать списком элементов просто потому, что элементов бесконечно много. Для самых важных из таких множеств есть общепринятые обозначения. Так, множество натуральных чисел обозначается через $\mathbb{N}$, множество целых чисел — через $\mathbb{Z}$, множество рациональных чисел — через $\mathbb{Q}$, а множество действительных чисел — через $\mathbb{R}$.

Из уже имеющегося множества можно выделить подмножество, указав некоторое свойство элементов: те элементы множества, которые обладают этим свойством, образуют подмножество. Например, чётные числа — это те целые числа, которые делятся на 2. Для определения таких множеств используется формальная запись, которая на примере чётных чисел выглядит так:

$$\{x \in \mathbb{Z} \mid x = 2y \text{ при некотором } y \in \mathbb{Z}\}$$
$$\text{или так} \tag{5.1}$$
$$\{x \in \mathbb{Z} : x = 2y \text{ при некотором } y \in \mathbb{Z}\}.$$

До вертикальной черты или двоеточия указывается, элементы какого множества будут в определяемом нами подмножестве, а после черты указывается свойство, которому должны удовлетворять элементы, чтобы попасть в подмножество. Выбор между вертикальной чертой и двоеточием осуществляется из эстетических соображений или из требований к однозначному пониманию формулы (и вертикальная черта, и двоеточие часто используются в математических формулах).

Если ясно, о каком объемлющем множестве идёт речь, его обозначение опускают. Скажем, если из контекста ясно, что мы говорим только о целых числах, допустимо определить натуральные числа[4] так:

$$\mathbb{N} = \{x \mid x \geqslant 0\}.$$

Есть ещё один удобный способ определять новые множества из уже имеющихся. Для этого к множествам можно применять *операции*.

На множествах определено большое количество операций, самые важные из которых мы сейчас перечислим.

Объединение множеств. Обозначение $A \cup B$. Это множество, состоящее в точности из тех элементов, которые принадлежат хотя бы одному из множеств A и B.

Пересечение множеств. Обозначение $A \cap B$. Это множество, состоящее в точности из тех элементов, которые принадлежат обоим множествам A и B.

Разность множеств. Обозначение $A \setminus B$. Это множество, состоящее в точности из тех элементов, которые принадлежат множеству A, но не принадлежат множеству B.

Симметрическая разность множеств. Обозначение $A \triangle B$. Это множество, состоящее в точности из тех элементов, которые принадлежат ровно одному из множеств: либо A, либо B.

Помимо словесных определений, приведённых выше, есть наглядный графический способ иллюстрировать операции с множествами: круги Венна (иногда говорят Эйлера–Венна). При этом способе множество изображается условным кругом (или другой геометрической фигурой) и предполагается, что внутренность круга изображает элементы множества.

На паре кругов легко изобразить объединение, пересечение, разность и симметрическую разность множеств, как это сделано на рисунке 5.1, — выделяя результат применения операции штриховкой или цветом.

Контрольный вопрос 5.3. Определите, на каких картинках рис. 5.1 какие операции изображены.

В заключение этого раздела мы обсудим более сложные и нетривиальные примеры множеств. Как мы уже упоминали, элементами множества могут быть другие множества.

Пример 5.4. Для начала рассмотрим такой пример множества, заданного перечислением элементов:

$$\{\varnothing, 1, \{1, \varnothing\}, \{1, 2\}\}.$$

Если мы аккуратно и формально прочитаем эту запись, то убедимся, что в этом множестве 4 элемента, один из которых — пустое множество $\varnothing$, ещё один — число 1 и есть также два элемента, которые сами по себе являются двухэлементными множествами: $\{1, \varnothing\}$ и $\{1, 2\}$, причем первое из них снова содержит в качестве элемента множество – пустое.

[4] Мы предпочитаем именно такое определение натуральных чисел, включающее 0.

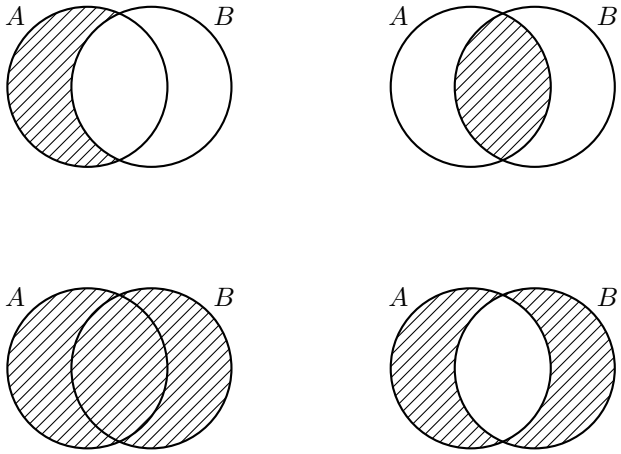


Рис. 5.1: Теоретико-множественные операции на кругах Венна

Пример 5.5. Вспомним, как мы определяли неориентированные графы. Для каждой пары вершин мы указывали, соединены ли они ребром. Пары вершин, соединённых ребром образуют множество, элементами которого являются 2-элементные подмножества вершин (попросту пары вершин).

Приходим к другому определению неориентированного графа (без петель и кратных рёбер) как пары множеств (V, E) (множества вершин и множества рёбер), в которой E является подмножеством множества, состоящего из всех пар вершин. Это определение общеупотребительно в книгах по теории графов.

Пример 5.6. Ещё один пример множества, элементами которого также являются множества, встретился нам в арифметике остатков. Вычет по модулю n — это множество целых чисел, имеющих одинаковый остаток при делении на n. Всего разных вычетов ровно n штук. Используя обозначения для задания множества списком элементов с некоторой вольностью, можно записать множество вычетов по модулю 3 как

$$\{\{\dots,-3,0,3,\dots\},\ \{\dots,-2,1,4,\dots\},\ \{\dots,-1,2,5,\dots\}\}.$$

Замечание 5.1. На всякий случай отметим ещё одну тонкость в построении множеств. Давайте «определим» множество A списком его элементов: $A = \{A\}$. То есть единственным элементом множества является оно само. Что же это за множество, которое является своим собственным элементом и притом единственным? Не вполне ясно, как бы оно могло выглядеть.

На самом деле, это достаточно опасная ситуация. Рассматривая такие множества, можно прийти к противоречию. В формальной теории множеств такая конструкция запрещена и никакое множество не является своим собственным элементом. Если говорить неформально, то при построении очередного множества разрешается в его определении ссылаться только на ранее построенные множества. Оказывается, таким способом можно изготовить все обсуждаемые нами множества, начав лишь с пустого.

Мы не будем на этом подробно останавливаться, но скажем лишь, что ключевую роль в этом играет следующая операция над множествами, которая и для нас будет важна.

Множество подмножеств. По любому множеству A определено множество его подмножеств, которое обозначается 2^A. Элементами этого множества являются в точности все подмножества множества A.

Например для множества $A = \{0, 1\}$ мы получим $2^A = \{\varnothing, \{0\}, \{1\}, \{0, 1\}\}$. А для множества $A = \varnothing$ получится $2^\varnothing = \{\varnothing\}$. Странное обозначение 2^A становится чуть более понятным, если сравнить число элементов в A и в 2^A в этих примерах.

Замечание 5.2. Можно заметить, что среди теоретико-множественных операций мы не назвали *дополнение*. Тому есть очень существенная причина.

Дополнение к множеству (обозначение $\bar{A}$) определяют обычно как те элементы, которые не входят в множество A. Если использовать такое определение буквально, то дополнение — не очень осмысленное понятие. Скажем, в дополнение к множеству $\{1\}$ нужно включить и число 2, и функцию синус, и полный граф на 140 вершинах.

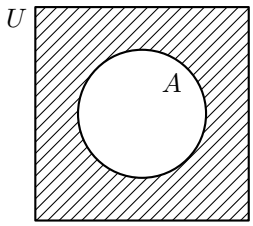


Рис. 5.2: Дополнение $\bar{A} = U \setminus A$ на рисунке заштриховано

Определение дополнения имеет смысл в более узкой ситуации, которая, однако, встречается очень часто. Предположим, мы рассуждаем о подмножествах некоторого множества U. В этом случае элементы множества U, которые не принадлежат некоторому подмножеству A, образуют множество, которое и называется дополнением $\bar{A}$ к множеству A. Определённое таким образом дополнение выражается через операцию разности множеств: $\bar{A} = U \setminus A$ (см. рис. 5.2). В этом случае дополнение является по сути сокращённым обозначением разности некоторого фиксированного в рассуждении множества U и подмножества этого множества. Именно так оно будет возникать в этой книге в дальнейшем.

5.2 Теоретико-множественные тождества

Может оказаться так, что применение теоретико-множественных операций в разном порядке даёт одно и то же множество, к каким бы множествам ни применять данные последовательности операций. В таких случаях говорят о теоретико-множественных тождествах.

Простейшие примеры тождеств

$$A \cup B = B \cup A;\ A \cap B = B \cap A;\ (A \cup B) \cup C = A \cup (B \cup C);\ (A \cap B) \cap C = A \cap (B \cap C).$$

Они очевидны из определения: достаточно пересказать определения словами и станет ясно, что левые и правые части равенств задают одно и то же.

И вообще, определения объединения и пересечения имеют смысл не только для двух (как в исходном определении) или трёх (как в этом примере) множеств, но и для произвольного *семейства* множеств.[5] Объединением семейства множеств является множество, состоящее в точности из элементов, которые входят хотя бы в одно множество семейства. Пересечением семейства множеств является множество, состоящее в точности из элементов, которые входят во все множества семейства.

Вот менее очевидный пример тождества:

$$(A \cap B) \setminus C = (A \setminus C) \cap B. \tag{5.2}$$

Нарисуем левую и правую часть на картинке кругов Венна, рис. 5.3. Видим, что множество получается одно и то же в обоих случаях.

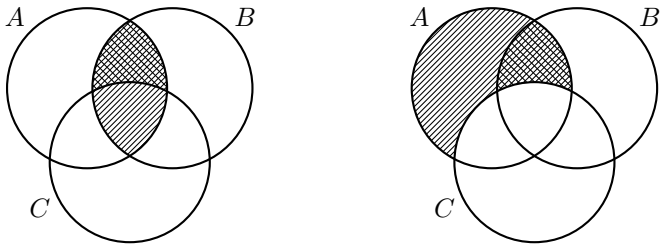


Рис. 5.3: На рисунках штриховкой вправо выделен результат первой операции, двойной штриховкой выделен результат второй операции

Почему рассуждение с картинкой корректно? Оно на самом деле скрывает под собой разбор возможных случаев. Каждый элемент может принадлежать или не принадлежать каждому из трёх множеств. Поэтому всего есть 8 вариантов, которые отмечены на рисунке 5.4. Варианты обозначены двоичными словами.

Цифра 0 на первом месте означает, что данная область обозначает элементы, которые не входят в множество A, цифра 1 означает вхождение в множество A. Вторая цифра аналогично означает принадлежность или непринадлежность множеству B; третья — множеству C.

Если известно, в какие множества из данных трёх входит элемент, то можно определить, входит ли он в пересечение, разность и т.д. каких-то из этих трёх мно-

[5] Слово «семейство» здесь и в аналогичных ситуациях является синонимом слова «множество» и используется для гладкости речи.

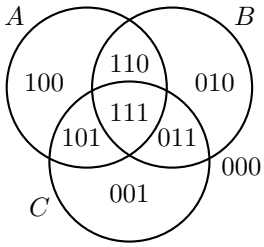


Рис. 5.4: 8 вариантов принадлежности или не принадлежности каждому из трёх множеств

жеств. Составим таблицу всех возможных вариантов:

$x \in A$	$x \in B$	$x \in C$	$x \in A \cap B$	$x \in (A \cap B) \setminus C$	$x \in A \setminus C$	$x \in (A \setminus C) \cap B$
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	1	0
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	0	0

Легко видеть, что пятый и седьмой столбцы таблицы совпадают. Это означает, что каждый элемент входит в множество, задавамое левой частью (5.2), тогда и только тогда, когда он входит в множество, задаваемое правой частью. По определению равенства множеств эти множества совпадают.

Аналогично можно доказывать и другие тождества.

Задача 5.7. Докажите следующие тождества, используя круги Венна и таблицы значений:

(a) $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$ (дистрибутивность объединения и пересечения);

(b) $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$ (дистрибутивность пересечения и объединения);

(c) $(A \triangle B) \triangle C = A \triangle (B \triangle C)$ (ассоциативность симметрической разности).

При большом количестве переменных-множеств круги Венна становятся не слишком наглядными, а таблицы значений становятся слишком длинными. Поэтому удобнее использовать свойства операций с множествами и выполнять равносильные преобразования аналогично тому, как мы это делаем в элементарной алгебре. На самом деле, операции с множествами тоже можно пересказать на алгебраическом языке, что мы и сделаем в следующих разделах.

5.3 Логические переменные, логические связки

В этом разделе мы обсудим связь теоретико-множественных операций и логики.

Начнём с примера. Как мы обсудили в прошлом разделе, значение каждой формулы с множествами задаётся таблицей, в которой для каждого набора вариантов вхождений элемента в множества, к которым применяются операции, указано, принадлежит ли данный элемент множеству-результату выполнения всех этих операций.

Мы записывали в таблицу значения 1 и 0. При этом подразумевалось, что 1 означает истинность утверждения вида «элемент x принадлежит множеству A», а 0 означает ложность такого утверждения.

Это соответствие между логическими значениями «истина», «ложь» и цифрами 1, 0 является стандартным и далее оно всюду подразумевается, если не оговорено что-то иное.

Посмотрим на таблицу, задающую пересечение множеств:

$x \in A$	$x \in B$	$x \in A \cap B$
0	0	0
0	1	0
1	0	0
1	1	1

В первых двух столбцах таблицы указаны возможные логические значения утверждений «элемент x принадлежит множеству A», «элемент x принадлежит множеству B». В третьей указаны логические значения для составного утверждения

«элемент x принадлежит множеству A» И «элемент x принадлежит множеству B».

Это утверждение состоит из двух частей, его истинность полностью определяется истинностью этих частей. Как именно, указано связующим словом «И».

Такие составные высказывания возникают очень часто. Они получаются применением *логических связок*. В примере мы использовали связку, которая называется *конъюнкция*. Мы видели, что конъюнкция соответствует пересечению множеств,

Есть и другие связки, они также соответствуют операциям и высказываниям о множествах.

Ниже в таблице приведены значения самых употребительных связок: конъюнкции $\wedge$, дизъюнкции $\vee$, суммы по модулю 2, она же XOR, она же «исключающее ИЛИ» (обозначение $\oplus$), равносильности $\equiv$ и импликации (логическое следование: «если A, то B», «из A следует B», обозначение $\to$).

x	y	$x \wedge y$	$x \vee y$	$x \to y$	$x \equiv y$	$x \oplus y$
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	0	0	1
1	1	1	1	1	1	0

Контрольный вопрос 5.8. Напишите таблицу значений для связки $y \to x$.

На примере мы убедились, что конъюнкция соответствует пересечению множеств. Аналогично дизъюнкция соответствует объединению, а XOR — симметрической разности.

Какая связка отвечает разности множеств? В таблице такой связки нет. Чтобы выразить разность множеств, нам нужна ещё одна связка: отрицание (мы её будем обозначать $\neg$). Она применяется к одному утверждению и означает высказывание о ложности этого утверждения.

Таблица отрицания очень простая: $\neg 0 = 1$, $\neg 1 = 0$.

Задача 5.9. Проверьте, составив таблицы значений, что теоретико-множественной операции разности $A \setminus B$ отвечает связка, которая имеет вид $x \wedge \neg y$.

Что соответствует импликации на языке множеств? Посмотрев на таблицу значений, видим, что импликация истинна тогда и только тогда, когда элемент принадлежит второму множеству ИЛИ не принадлежит первому. Чтобы задать условие «не принадлежит», нужна операция дополнения (см. замечание 5.2). С её помощью теоретико-множественная операция, отвечающая импликации, записывается как $\bar{A} \cup B$.

Контрольный вопрос 5.10. Нарисуйте диаграмму Венна для операции $\bar{A} \cup B$.

У импликации есть еще один важный смысл на языке множеств. Давайте разберемся, что означает, что утверждение «если x принадлежит A, то x принадлежит B» выполняется для всех x. Нетрудно видеть, что импликация утверждений о принадлежности элемента множествам выполняется тогда и только тогда, когда всякий элемент первого множества лежит во втором. То есть, если это утверждение выполняется для всех x, то первое множество является подмножеством второго. Поэтому истинность импликации для всех x на языке множеств можно интерпретировать как утверждение о включении множеств $A \subseteq B$. Аналогично проверяется, что равносильность в этом же смысле интерпретируется как равенство множеств $A = B$.

Для связок выполняется довольно много тождеств, которые легко проверить по таблицам истинности (выполните эту проверку!).

Коммутативность конъюнкции, дизъюнкции и XOR:

$$x \wedge y = y \wedge x, \quad x \vee y = y \wedge x, \quad x \oplus y = y \oplus x. \tag{5.3}$$

Ассоциативность тех же связок:

$$(x \wedge y) \wedge z = x \wedge (y \wedge z), \quad (x \vee y) \vee z = x \vee (y \vee z), \quad (x \oplus y) \oplus z = x \oplus (y \oplus z). \tag{5.4}$$

Тождества дистрибутивности (пару из них вы уже видели в виде теоретико-множественных тождеств:

$$(x \wedge y) \vee z = (x \vee z) \wedge (y \vee z), \quad (x \vee y) \wedge z = (x \wedge z) \vee (y \wedge z), \quad (x \oplus y) \wedge z = (x \wedge z) \oplus (y \wedge z). \tag{5.5}$$

Формулы упрощения получаются подстановкой констант вместо одной из переменных

$$\begin{aligned} &0 \wedge y = 0, \quad 1 \wedge y = y, \quad 0 \vee y = y, \quad 1 \vee y = 1, \\ &0 \to y = 1, \quad 1 \to y = y, \quad x \to 0 = \neg x, \quad x \to 1 = 1, \\ &0 \oplus y = y, \quad 1 \oplus y = \neg y. \end{aligned} \tag{5.6}$$

Логические тождества выражают законы логики. Если истинно утверждение, выражающееся левой частью тождества, то истинно и утверждение, выражающееся правой частью тождества. И наоборот.

Тут уже уместно небольшое отступление о расстановке скобок. В логических тождествах возникают уже довольно сложные формулы, использующие разные логические связки. В таких формулах нам приходится расставлять скобки, чтобы было ясно, какая из связок применяется первой, какая второй, и так далее. Аналогично арифметическим операциям, чтобы сократить число скобок, можно договориться о том, какие связки применяются раньше, а какие позже. Принято считать, что связка отрицание $\neg$ имеет самый высокий приоритет и всегда применяется первой. Следующая по важности связка — это конъюнкция $\wedge$, она применяется второй. Следующей применяются связки дизъюнкция $\vee$ и XOR. Наконец, самый низкий приоритет у связки импликации $\to$. Между связками $\vee$ и XOR нет устоявшегося правила приоритетности. Обычно используют правило, что связки одного приоритета применяются, начиная с самой левой. Такая ситуация может возникнуть не только из-за применения $\vee$ и XOR, но и при применении нескольких импликаций подряд. Для надёжности в таких сомнительных случаях можно поставить лишнюю пару скобок. Впрочем, эти правила нам почти не понадобятся.

Поскольку логические переменные и связки выражают некоторые элементы нашего языка, то и некоторые логические тождества также выражают общеизвестные правила языка. Полезно разобрать несколько таких примеров логических тождеств, которые уже встречались в наших рассуждениях.

Логический закон двойного отрицания (отрицание отрицания утверждения равносильно самому утверждению) записывается тождеством

$$\neg\neg x = x, \tag{5.7}$$

которое совершенно очевидно из таблицы истинности отрицания.

Тождество

$$x \to y = \neg y \to \neg x \tag{5.8}$$

выражает принцип контрапозиции (теорема равносильна обратной к противоположной). Этот принцип часто используется в математических доказательствах: вместо доказательства утверждения «если А, то Б» зачастую удобнее изменить посылку и доказывать равносильное утверждение «если не Б, то не А». Проверка тождества (5.8) легко производится по таблице.

Законы де Моргана задают правило взятия отрицания от конъюнкции и дизъюнкции

$$\neg(x \vee y) = \neg x \wedge \neg y; \quad \neg(x \wedge y) = \neg x \vee \neg y. \tag{5.9}$$

Доказать эти тождества легко, разобрав случаи возможных значений переменных.

Равенства (5.9) обобщаются на случай нескольких переменных:

$$\neg(x_1 \vee x_2 \vee \cdots \vee x_n) = \neg x_1 \wedge \neg x_2 \wedge \cdots \wedge \neg x_n; \quad \neg(x_1 \wedge x_2 \wedge \cdots \wedge x_n) = \neg x_1 \vee \neg x_2 \vee \cdots \vee \neg x_n. \tag{5.10}$$

Справедливость этих тождеств проверяется индукцией по числу переменных (база — тождества (5.9)).

Задача 5.11. Докажите аккуратно тождества (5.10).

На языке множеств законы де Моргана формулируются так: (I) элемент x не принадлежит объединению семейства множеств тогда и только тогда, когда он не принадлежит ни одному из этих множеств; (II) элемент x не принадлежит пересечению семейства множеств тогда и только тогда, когда он не принадлежит хотя бы одному из этих множеств. В таком виде законы де Моргана применимы и к бесконечным семействам множеств.

В конце раздела вернёмся к теоретико-множественным тождествам и покажем пример использования логических тождеств для доказательства теоретико-множественных.

Пример 5.12. Докажем, что

$$(A_1 \cap A_2 \cap A_3 \cap \cdots \cap A_n) \setminus (B_1 \cup B_2 \cup \cdots \cup B_n) = (A_1 \setminus B_1) \cap (A_2 \setminus B_2) \cap \cdots \cap (A_n \setminus B_n). \tag{5.11}$$

Запишем логическую формулу для левой части

$$(a_1 \wedge a_2 \wedge \cdots \wedge a_n) \wedge \neg(b_1 \vee b_2 \vee \cdots \vee b_n).$$

Применяя закон де Моргана, получим

$$a_1 \wedge a_2 \wedge \cdots \wedge a_n \wedge \neg b_1 \wedge \neg b_2 \wedge \cdots \wedge \neg b_n.$$

Конъюнкция коммутативна, поэтому

$$a_1 \wedge a_2 \wedge \cdots \wedge a_n \wedge \neg b_1 \wedge \neg b_2 \wedge \cdots \wedge \neg b_n = (a_1 \wedge \neg b_1) \wedge (a_2 \wedge \neg b_2) \wedge \cdots \wedge (a_n \wedge \neg b_n).$$

Но это и есть логическая формула, отвечающая правой части (5.11).

5.4 Наблюдения

В этом разделе собрано несколько замечаний о логических связках и том, как они возникают в математике.

Во-первых, можно заметить, что постоянно используемые нами в рассуждениях *кванторы* «для любого» и «существует» можно рассматривать как конъюнкцию и дизъюнкцию, возможно с бесконечным числом переменных. Действительно, конструкция вида «для любого x выполняется свойство $A(x)$» соответствует конъюнкции

$$A(x_1) \wedge A(x_2) \wedge \ldots$$

по всем возможным значениям x и эта конъюнкция бесконечна, если x принимает бесконечное количество значений. А конструкция «существует x, для которого выполняется свойство $A(x)$» соответствует дизъюнкции

$$A(x_1) \vee A(x_2) \vee \dots,$$

которая также иногда оказывается бесконечной. Квантор «для любого» обычно называют *квантором всеобщности* и обозначают знаком $\forall$, а квантор «существует» называют *квантором существования* и обозначают знаком $\exists$.

Пример 5.13. Запишем формулу (5.1) с использованием квантора существования

$$\{x \in \mathbb{Z} \mid \exists y : (x = 2y \text{ и } y \in \mathbb{Z})\}.$$

Замечание 5.3. Обратите внимание на двоеточие после квантора существования. Это декоративный элемент, не несущий логического смысла, но часто используемый (так как многие воспринимают кванторы не как логические связки, а как стенографические сокращения русских слов; двоеточие таким людям заменяет слова «такой, что»). После квантора всеобщности двоеточие вроде бы никто не ставит.

Если теперь перевести на язык кванторов законы де Моргана, то получится, что при перестановке квантора и отрицания квантор меняется на противоположный.

Пример 5.14. Рассмотрим утверждение: «у любого целого числа n есть простой делитель k». Это утверждение построено с помощью квантора всеобщности по n и квантора существования по k. По законам де Моргана отрицание этого утверждения равносильно утверждению «существует целое число n, которое не делится ни на одно простое число k» (теперь вначале квантор существования, а потом — квантор всеобщности).

Поскольку число 1 не делится ни на одно простое число, второе утверждение истинно, а первое, соответственно, ложно.

Связь между логическими тождествами и законами логики помогает понять и другие свойства связок и правил логики, используемых в математических рассуждениях.

Во-первых, правила логики объясняют важность отсутствия противоречий в математических рассуждениях. Противоречие означает, что мы получили, что одновременно выполнено некоторое утверждение x и его отрицание $\neg x$, а такого не может быть. Логически это соответствует тому, что выражение

$$x \wedge \neg x$$

ложно независимо от того, чему равно x, а значит $x \wedge \neg x = 0$. Далее можно заметить, что выражение $0 \to y$ истинно независимо от того, чему равен y. Подставляя получаем, что $x \wedge \neg x \to y$. Таким образом из того, что доказано x и $\neg x$ сразу следует любое утверждение y. И если в математическое рассуждение закралось противоречие, то из него мгновенно можно вывести любое, даже самое нелепое, утверждение.

Так что наличие любого противоречия позволяет доказать что угодно и математика как наука теряет всякий смысл.

Возможно, стоит пояснить, почему в определении импликации мы считаем, что $0 \to 0 = 1$, $0 \to 1 = 1$, то есть, из лжи следует ложь и из лжи следует истина. На самом деле, это вполне согласуется с бытовым применением конструкции «если ..., то ...». Представим себе, что кто-то делает следующее заявление: “Если я завтра заболею, то не приду на занятия”. Если на следующий день этот человек не заболел и пришел на занятия, следует ли считать, что он соврал? А если он не заболел и не пришел? И то, и другое было бы странно, ведь человек и вовсе ничего не говорил про этот случай. Нечто содержательное было сказано только для той ситуации, когда он заболел. Так что, если посылка утверждения ложна, то и в обычной жизни утверждение считают истинным.

Пример на эту тему можно привести и в математике. Рассмотрим утверждение «если число n делится на 4, то n четно». Это утверждение верно (почему?). И оно не перестанет быть верным, если вместо произвольного числа подставить какое-нибудь конкретное, например 9. Посылка утверждения становится ложной, 9 не делится на 4, но само утверждение остается истинным.

Из правил логики можно также объяснить вырожденные случаи, которые встречаются, например, в теории графов.

Пример 5.15. Оказывается, что граф, состоящий из одной вершины, является одновременно и кликой, и независимым множеством. Развернув определения, видим, что про данный граф утверждается истинность двух высказываний: «любая пара вершин в этом графе связана ребром», «любая пара вершин в этом графе не связана ребром».

В графе на одной вершине вообще нет ни одной пары вершин, как нет и рёбер. Как применить квантор всеобщности, когда множество значений пусто (не содержит ни одного элемента)?

Это так сразу не ясно, зато ясно, как это делать в случае квантора существования. Ясно, что подобное высказывание будет ложно: оно утверждает существование элемента с некоторыми свойствами, выбранного из совокупности, в которой нет ни одного элемента. Если мы уже знаем, что эльфов не бывает, то утверждение «существует гладко выбритый эльф» следует признать ложным.

Таким образом, мы можем взять отрицания наших утверждений всеобщности. По законам де Моргана мы получаем утверждения «существует пара вершин в графе, которая не связана ребром» и «существует пара вершин в графе, которая связана ребром» соответственно. Как мы уже обсудили выше, это ложные утверждения. Значит, их отрицания истинны.

Тот же результат можно получить и другим способом. Утверждение «любая пара вершин в этом графе связана ребром» можно переформулировать в виде «если $\{x, y\}$ – пара вершин графа, то x и y соединена ребром». Если в графе только одна вершина, то для всех x и y посылка этой импликации ложна. А значит, как мы обсуждали выше, утверждение истинно.

Неформально можно заметить, что разобранный пример объясняет, почему конъюнкция пустого множества членов истинна, а дизъюнкция пустого множества членов ложна. Примеров такого рода есть очень много.

Задача 5.16. Проверьте, что граф на одной вершине связный и эйлеров.

Задача 5.17. Верно ли, что минимальное натуральное число, которое делится на все простые числа, больше 2015?

5.5 Какие связки необходимы?

В этой главе мы ввели разные логические связки, соответствующие разным конструкциям в русском языке. Но достаточно ли их чтобы выразить любое, сколь угодно сложное высказывание, основанное на данных простых? Все ли рассмотренные связки необходимы или без каких-то можно обойтись? Сейчас мы обсудим эти вопросы, но сначала уточним их.

Элементарные высказывания будем обозначать переменными $x_1, x_2, \ldots, x_n, \ldots$. Произвольное высказывание, зависящее от элементарных высказываний $x_1, \ldots, x_n$, мы будем обозначать через $f(x_1, \ldots, x_n)$. Все, что мы формально требуем от такого высказывания, — это чтобы его значение определялось значениями переменных $x_1, \ldots, x_n$. То есть при произвольных значениях переменных $x_1 = \alpha_1, \ldots, x_n = \alpha_n$ высказывание $f(\alpha_1, \ldots, \alpha_n)$ должно принимать однозначно определенное значение, либо истина, либо ложь. Другими словами, мы хотим, чтобы f была *функцией* от переменных $x_1, \ldots, x_n$. Подробнее функции мы обсудим в следующей главе.

Составные высказывания можно задавать таблицей – для всех значений переменных указывать значение высказывания. Такие таблицы мы рисовали в начале главы, когда определяли высказывание $f(x, y) = x \wedge y$ и другие. Также сложные высказывания можно задавать формулами с помощью связок. Например, можно определить $f(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee \neg x_3$.

Более экзотический пример получается если определить $f(x_1, x_2, x_3) = x_1$ или даже $f(x_1, x_2, x_3) = 1$. В первом случае мы говорим, что наше высказывание, зависящее от трех элементарных высказываний, есть просто первое из этих трех высказываний, а во втором, что оно просто всегда истинно. При этом в первом случае x_2 и x_3 никак не используются, а во втором и вовсе не используется ни одно из элементарных высказываний. Ничего страшного в этом нет, определением это не запрещается.

Заметим, что одно и то же высказывание можно задать разными формулами.

Пример 5.18. Рассмотрим высказывания (1) «из утверждений A_1, A_2, A_3 истинны более половины»; (2) «среди утверждений A_1, A_2, A_3 есть истинные и ложные»; (3) «не все утверждения A_1, A_2, A_3 истинны и не все утверждения A_1, A_2, A_3 ложны».

Высказывания (1) и (2) существенно различны: например, они различаются на наборе логических переменных $x_1 = 1$; $x_2 = 0$; $x_3 = 0$ (то есть A_1 истинно, а два других утверждения ложны).

Высказывания (2) и (3) по существу одинаковы. Понять это можно, например, из таблицы значений (приведём в ней и значения первого высказывания):

x_1	x_2	x_3	(1)	(2)	(3)
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	1	1
1	0	0	0	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	0	0

Какие связки можно использовать при построении таких формул? Здесь возможны разные варианты. Можно выбрать какие-то из связок, которые мы уже определили, можно добавить каких-то еще. Скажем, может оказаться полезным использовать связку «из утверждений A_1, A_2, A_3 истинны более половины».

Если некоторого набора связок достаточно, чтобы выразить любое составное высказывание, такой набор называется *полным*.

Теперь мы можем точнее сформулировать вопросы из начала этого раздела. Можно ли с помощью формул из уже введенных связок выразить любое высказывание? Можно ли обойтись меньшим набором связок?

Оказывается, верна такая теорема, которую мы постепенно докажем в этом разделе.

Теорема 5.1. *Отрицание и конъюнкция образуют полный набор.*

Остановимся на еще одном тонком моменте. Что все-таки разрешается использовать в формулах? Ясно, что можно использовать переменные, более того, разрешается даже использовать лишние переменные, которые в составном высказывании не участвуют (подумайте, зачем это может понадобиться). Ясно также, что разрешается использовать связки из заданного набора. Можно ли что-то еще? Например, можно ли использовать 0 и 1? В принципе, на последний вопрос можно было бы ответить и да, и нет — в обоих вариантах основные вопросы этого раздела остались бы осмысленными. Чтобы сохранить возможность ставить вопрос для всех вариантов, мы запрещаем использовать 0 и 1 по умолчанию, но разрешаем добавлять их в систему связок. Можно считать, что 0 и 1 – это связки с нулевым числом переменных.

Прежде чем переходить к доказательству полноты разных систем связок, полезно получше разобраться в объекте, который мы изучаем. Мы начнём с того, что определим, сколько вообще есть высказываний $f(x_1, \ldots, x_n)$ от n переменных $x_1, \ldots, x_n$.

Рассмотрим таблицу, задающую высказывание f. В таблице будет 2^n строк — это количество способов присвоить каждой из n переменных одно из двух логических значений.

Высказывание определяется последним столбцом, в котором также стоят 0 и 1. Для разных высказываний столбцы должны различаться.

Итак, осталось подсчитать количество столбцов из 2^n ячеек, каждая из которых содержит 0 или 1. Таких столбцов 2^{2^n} штук (то же самое рассуждение).

5.5.1 Полнота дизъюнкции, конъюнкции и отрицания

Таблицы значений помогают также понять, почему любое высказывание выражается через дизъюнкции, конъюнкции и отрицания. Пусть есть два составных высказывания f_1 и f_2. Как выглядит таблица для дизъюнкции $f_1 \vee f_2$, т.е. высказывания «f_1 или f_2»? В тех строках таблицы, в которых есть 1 для f_1 или для f_2, будет стоять 1 (см. таблицу дизъюнкции). В тех строках, где у f_1 и f_2 стоят нули, будет стоять 0.

Это рассуждение показывает, что любое высказывание может быть выражено как дизъюнкция таких высказываний, у которых ровно в одной строке стоит 1, а в остальных стоят нули. Действительно, выберем все строки таблицы высказывания, в которых стоят единицы; для каждой такой строки образуем высказывание, которое истинно только в данной строке, а в остальных ложно; дизъюнкция всех этих высказываний и будет выражать искомое высказывание.

Осталось научиться выражать через дизъюнкции, конъюнкции и отрицания высказывания того специального вида, который мы использовали в предыдущей конструкции. Одно высказывание такого вида построить легко, это конъюнкция всех переменных $x_1 \wedge \ldots \wedge x_n$. В её таблице значений есть ровно одна строка, в которой конъюнкция равна 1, в остальных она равна 0.

Чтобы получить высказывание, которое истинно ровно для одного (но произвольного) набора логических значений элементарных высказываний, нужно конъюнкцию подправить. Для этого используем наряду с элементарными высказываниями x_i их отрицания $\neg x_i$.

Обозначим значения элементарных высказываний через α_1, α_2, …, α_n. Если $\alpha_i = 1$, то включаем в конъюнкцию переменную x_i. Если $\alpha_i = 0$, то включаем в конъюнкцию отрицание переменной $\neg x_i$.

Построенная конъюнкция принимает значение 1 тогда и только тогда, когда все её члены равны 1. Из правила построения следует, что это происходит ровно на одном наборе значений элементарных высказываний $\alpha_1, \ldots, \alpha_n$. Действительно, проверим, что на любом другом наборе $\beta_1, \ldots, \beta_n$ она ложна. Раз наборы отличаются, значит $\alpha_i \neq \beta_i$ для некоторого i. Есть две возможности. Первая: $\alpha_i = 1$, $\beta_i = 0$, и в конъюнкцию входит переменная x_i, которая обнуляется на втором наборе. Вторая возможность: $\alpha_i = 0$, $\beta_i = 1$, и в конъюнкцию входит $\neg x_i$, который обнуляется на втором наборе.

Пример 5.19. Выразим указанным способом XOR через дизъюнкцию, конъюнкцию и отрицание:

$$x \oplus y = (x \wedge \neg y) \vee (\neg x \wedge y).$$

Первый член дизъюнкции отвечает строке $1, 0$ в таблице XOR, второй член — строке $0, 1$.

Мы выразили произвольное составное высказывание в виде дизъюнкции конъюнкций переменных или их отрицаний. Такое выражение называется дизъюнктивная нормальная форма (ДНФ).

Обратите внимание, что представление в виде ДНФ не единственно. Например, дизъюнкция переменных $x_1 \vee x_2 \vee \cdots \vee x_n$ является ДНФ. Но если применить описанную выше конструкцию, получится другая ДНФ для того же высказывания (как она выглядит?). Эта ДНФ намного длиннее. ДНФ, которые получаются из строк таблицы описанным выше способом, называют совершенными.

Задача нахождения самой короткой ДНФ, представляющей данное высказывание, в общем случае очень трудна.

Задача 5.20. КНФ (конъюнктивная нормальная форма) — это конъюнкция дизъюнкций переменных или их отрицаний. Докажите, что любое высказывание выражается в виде КНФ.

5.5.2 Полнота конъюнкции и отрицания

Теперь уже нетрудно доказать теорему – для этого осталось избавиться от дизъюнкции.

С помощью законов де Моргана дизъюнкция выражается через конъюнкцию и отрицание (и наоборот):

$$x \vee y = \neg(\neg x \wedge \neg y); \qquad x \wedge y = \neg(\neg x \vee \neg y)$$

(мы также использовали очевидное тождество $\neg\neg x = x$).

Возьмём высказывание, выраженное через дизъюнкции, конъюнкции и отрицания. Каждую дизъюнкцию заменим по закону де Моргана на выражение, содержащее лишь конъюнкции и отрицания. Выполнив все такие замены, получим высказывание, выраженное лишь через конъюнкции и отрицания.

Пример 5.21. Выразим указанным способом XOR через конъюнкцию и отрицание:

$$x \oplus y = \neg(\neg(x \wedge \neg y) \wedge \neg(\neg x \wedge y)).$$

5.5.3 Алгебраическое доказательство теоремы 5.1

Мы приведем еще одно доказательство теоремы. Для этого мы докажем полноту системы связок $\wedge, \oplus, 1$.

Почему из этого будет следовать теорема? Мы уже выразили $\oplus$ через конъюнкцию и отрицание, см. пример 5.21. Кроме того, через конъюнкцию и отрицание можно выразить и тождественно истинное утверждение:

$$1 = \neg(x \wedge \neg x).$$

Поэтому достаточно выразить любое высказывание через $\wedge, \oplus, 1$ и применить тот же приём, что в предыдущем разделе: последовательно убрать из выражения $\oplus$ и 1.

Чем хороши выражения, использующие только $\wedge, \oplus, 1$? Для связок $\wedge, \oplus, 1$ выполняются обычные свойства арифметических операций. Трюк состоит в том, чтобы посмотреть на 0 и 1 как на вычеты по модулю 2. Тогда $\wedge$ соответствует произведению вычетов, а $\oplus$ – сумме вычетов.

Поэтому неудивительно, что эти связки можно использовать для представления составных высказываний в виде многочленов. Многочлен по определению — сумма произведений. В данном случае сумма это XOR, а произведение — конъюнкция. Такие многочлены называются *многочленами Жегалкина*. Свободный член у этих многочленов равен 0 или 1.

Докажем индукцией по n, что произвольное высказывание $f(x_1, \ldots, x_n)$ можно выразить формулой со связками $\wedge, \oplus, 1$. База индукции — 0 высказываний. Высказываний f от нуля переменных всего два – 0 и 1. Константа 1 уже есть, осталось выразить константу 0: $0 = 1 \oplus 1$.

Пусть утверждение доказано для всех составных высказываний от n элементарных высказываний. Докажем выразимость для составных высказываний от $n+1$ элементарного высказывания. Для этого по высказыванию $f(x_1, \ldots, x_{n+1})$ определим два высказывания от n элементарных высказываний, а именно, $f_0(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, 0)$ и $f_1(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, 1)$.

По предположению индукции и f_0, и f_1 выражаются через связки $\wedge, \oplus, 1$. Выразим теперь f:

$$f = ((1 \oplus x_{n+1}) \wedge f_0) \oplus (x_{n+1} \wedge f_1)\,.$$

Действительно, при $x_{n+1} = 0$ обращается в 0 второе слагаемое, при $x_{n+1} = 1$ — первое. В любом случае получаем совпадение левой и правой частей равенства.

5.6 Формула включений-исключений

Теперь вернемся к перечислительной комбинаторике и приведем еще два доказательства формулы включений-исключений. Напомним, что формула включений-исключений обобщает правило суммы и даёт выражение для объединения нескольких, возможно пересекающихся, множеств. Для двух и трех множеств мы получили такие выражения

$$|A \cup B| = |A| + |B| - |A \cap B|,$$
$$|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |A \cap B| - |B \cap C| + |A \cap B \cap C|.$$

При доказательстве второй из этих формул уже использовались круги Венна: мы проверяли, что каждый элемент объединения посчитан ровно один раз, по отдельности для каждой части, на которые круги Венна разбивают объединение.

Формулы для двух и трёх множеств подсказывают общий вид формулы включений-исключений. Нужно взять сумму мощностей всех множеств. Некоторые элементы при этом посчитаны более одного раза. Поэтому нужно вычесть мощности попарных пересечений множеств, после чего некоторые элементы объединения вообще не будут посчитаны. Далее нужно последовательно прибавлять и вычитать

мощности тройных, четверных и т.д. пересечений, включая их в итоговую сумму с чередующимися знаками.

Способ построения итоговой формулы из этого описания понятен, она была написана в разделе 2.8.4 на с. 66. Но удобно представить итоговую формулу в более компактном виде. Для этого введём обозначения. Множества, для которых ищем объединение, обозначим A_1, A_2, …, A_n. Через S будем обозначать подмножество множества $\{1,\dots,n\}$, каждое такое подмножество выделяет некоторое семейство подмножеств $\{A_i : i \in S\}$ Через A_S обозначим пересечение всех множеств, входящих в семейство S, т.е.

$$A_S = \bigcap_{i\in S} A_i.$$

В таких обозначениях формула включений-исключений записывается достаточно компактно:

$$|A_1 \cup A_2 \cup \dots \cup A_n| = \sum_{S\neq\varnothing} (-1)^{|S|+1} |A_S|. \tag{5.12}$$

Знаки в сумме (5.12) определяются количеством множеств в семействе. Из примеров для $n = 2$ и $n = 3$ мы видим, что сами множества (т.е. пересечения по семействам множеств, содержащих одно множество) должны входить со знаком «+». Для остальных семейств знак определяется из правила чередования знаков.

Эта формула была доказана в разделе 2.8.4 вычислением с знакопеременными суммами биномиальных коэффициенов. Приведем ещё два доказательства этой формулы.

5.6.1 Первое доказательство

Индукция по числу множеств. База индукции — одно множество, формула очевидна: $|A| = (-1)^{1+1}|A|$.

Индуктивный переход использует формулу для количества элементов в объединении двух множеств:

$$\begin{aligned}|(A_1 \cup \dots \cup A_{n-1}) \cup A_n| &= |A_1 \cup \dots \cup A_{n-1}| + |A_n| - |A_n \cap (A_1 \cup \dots \cup A_n)| = \\ &= |A_1 \cup \dots \cup A_{n-1}| + |A_n| - |(A_n \cap A_1) \cup \dots \cup (A_n \cap A_{n-1})|.\end{aligned}$$

Для первого и третьего слагаемых по предположению индукции справедлива формула (5.12) для $n-1$ множества. Поэтому первое слагаемое дает вклад в сумму (5.12) для n множеств вида

$$\sum_{\substack{S\neq\varnothing \\ S\subseteq\{1,\dots,n-1\}}} (-1)^{|S|+1} |A_S|.$$

Второе слагаемое — это в точности $(-1)^{1+1} A_{\{n\}}$.

Рассмотрим теперь последнее слагаемое. По индуктивному предположению оно равно

$$\sum_{\substack{S\neq\varnothing \\ S\subseteq\{1,\dots,n-1\}}} (-1)^{|S|+1} |B_S|,$$

где вместо множеств A_i в формулу включений-исключений для $n-1$ множества подставлены множества $B_i = A_n \cap A_i$.

Для любого $S \subseteq \{1, \ldots, n-1\}$ выполняется равенство

$$B_S = \bigcap_{i \in S} (A_n \cap A_i) = A_n \cap A_S = A_{S \cup \{n\}}.$$

Получается, что мощность объединения $|(A_1 \cup \cdots \cup A_{n-1}) \cup A_n|$ представлена в виде суммы таких же слагаемых, что и в сумме (5.12): первое слагаемое отвечает семействам, не содержащим A_n, второе и третье — семействам, содержащим A_n. Нужно ещё проверить, что эти слагаемые входят с правильными знаками. Для первых двух слагаемых это ясно из самих формул. Для третьего заметим, что мощность S отличается от мощности $S \cup \{n\}$ на 1, так как $S \subseteq \{1, \ldots, n-1\}$. Это даёт лишний знак «$-$». Но в формулу для объединения двух множеств это слагаемое также входит со знаком «$-$». Поэтому окончательный знак будет правильным:

$$-(-1)^{|S|+1} = (-1)^{|S \cup \{i\}|+1}.$$

5.6.2 Второе доказательство

Формула (5.12) очень похожа на обычное алгебраическое тождество

$$1 - (1-x_1)(1-x_2)\ldots(1-x_n) = x_1 + x_2 + \cdots + x_n - x_1x_2 - \cdots + x_1x_2x_3 + \ldots \quad (5.13)$$

И это неслучайно. Введем индикаторную (или характеристическую) функцию множества. Для множества A по определению $\chi_A(x) = 1$, если $x \in A$, и $\chi_A(x) = 0$, если $x \notin A$.

Будем считать, что все рассматриваемые множества лежат в каком-то объемлющем множестве (*универсуме*), например, в объединении всех множеств, для которых доказывается формула. Мощность множества легко выражается как сумма индикатора по всему универсуму:

$$|A| = \sum_u \chi_A(u).$$

Теперь заметим, что индикаторная функция для дополнения множества (т.е. разности универсума и множества) равна $1 - \chi_A$, для пересечения множеств это просто произведение индикаторных функций этих множеств. А индикаторная функция для объединения $A = \cup_i A_i$ выражается как

$$\chi_A = 1 - (1 - \chi_{A_1})(1 - \chi_{A_2})\ldots(1 - \chi_{A_n}) \quad (5.14)$$

(используем закон де Моргана и выражаем дополнение к объединению как пересечение дополнений). Теперь заменим правую часть (5.14) на правую часть (5.13), произведения индикаторных функций — на индикаторные функции пересечений и просуммируем по универсуму.

5.6.3 Формула для симметрической разности

В симметрическую разность $A_1 \triangle A_2 \triangle \cdots \triangle A_n$ входят те элементы, которые принадлежат нечётному числу множеств из семейства A_i. Для мощности симметрической разности также есть формула через пересечения:

$$|A_1 \triangle A_2 \triangle \cdots \triangle A_n| = \sum_i |A_i| - 2\sum_{i<j} |A_i \cap A_j| + 4 \sum_{i<j<k} |A_i \cap A_j \cap A_k| - \dots \quad (5.15)$$

Эту формулу легко получить, несколько изменив рассуждение с индикаторными функциями. Пусть $A = A_1 \triangle A_2 \triangle \cdots \triangle A_n$. Тогда

$$2\chi_A = 1 - (1 - 2\chi_{A_1})(1 - 2\chi_{A_2}) \dots (1 - 2\chi_{A_n}).$$

Действительно, $1 - 2\chi_A$ принимает значения ± 1, причем -1 означает вхождение элемента в множество. Если элемент входит в четное число множеств, на таком элементе произведение будет равно $+1$ (вклад в правую часть равен 0), а если в нечетное — то -1 (вклад в правую часть равен 2). Теперь осталось раскрыть скобки и заменить произведения индикаторов на индикаторы пересечений.

Лекция 6

Функции

Понятие функции не менее важно для математики, чем понятие множества. Так же как и с множествами, функции уже встречались нам раньше в разных конкретных случаях. Теперь же мы хотим обсудить абстрактные понятия, связанные с функциями.

6.1 Пример

Мы начнем обсуждение функций с примера. Для этого вспомним один из вопросов, который мы обсуждали в главе о перечислительной комбинаторике.

Задача 6.1. Сколько есть k-элементных подмножеств в n-элементном множестве?

Мы обсуждали эту задачу в разделе 2.7 и даже привели два её решения. Сейчас мы повторим одно из них и выделим в нём использование понятия функции.

Мы начинаем с того, что подсчитываем количество *списков* подмножеств. Список отличается от подмножества тем, что его элементы упорядочены: какой-то элемент подмножества считается первым, какой-то вторым и т.д. Количество списков находится по правилу произведения и равно

$$(n)_k = n(n-1)(n-2)\cdot\ldots\cdot(n-k+1).$$

Далее мы находим количество списков, которые можно составить для одного подмножества. Оно равно $k!$, откуда получаем, что количество k-элементных подмножеств в n-элементном множестве равно

$$\binom{n}{k} = \frac{(n)_k}{k!} = \frac{n!}{k!(n-k)!}\ .$$

Мы уже решили задачу, не использовав слово «функция». И тем не менее, функции в этом решении присутствуют.

Как связаны списки и подмножества? Каждому списку соответствует ровно одно подмножество, это проиллюстрировано на рис. 6.1.

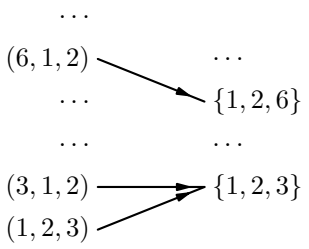


Рис. 6.1:

Вот это соответствие и есть функция. В решении мы использовали свойство этой функции: каждому подмножеству соответствует одно и то же количество списков (а именно, $k!$). Поэтому и получается, что количество подмножеств ровно в $k!$ раз меньше, чем количество списков.

6.2 Функции и связанные с ними понятия

Из разобранного примера видно, что под функцией мы понимаем соответствие: каждому элементу одного множества поставлен в соответствие какой-то один элемент другого множества.

Такое понимание функции не очень формально, но мы пока ограничимся им, а формальное определение дадим в следующем разделе.

6.2.1 Терминология и обозначения

Заметим, что в примере выше, каждому элементу множества ставится в соответствие ровно один элемент другого. Но для некоторых функций то множество, элементам которого мы ставим что-то в соответствие, устроено достаточно сложно. Например, если мы хотим рассмотреть функцию $f(x) = 1/x$ на действительных числах, то написанная формула не определяет функцию при $x = 0$. Но гораздо удобнее говорить об этой функции как о функции на действительных числах и не уточнять каждый раз, что в нуле она не определена. Поэтому нам будет удобно понимать функции немного шире.

Функцией из множества A в множество B мы назовём такое соответствие, которое сопоставляет некоторым элементам множества A какой-то элемент множества B. *Область определения* функции f из A в B состоит в точности из тех элементов x множества, которым сопоставлен элемент $f(x)$ множества B.

Наглядное представление функции[1] приведено на рис. 6.2.

Данному $x \in A$ (его называют *аргументом* функции) функция f из A в B либо не сопоставляет никакого элемента в B, либо сопоставляет ровно один такой элемент y. Во втором случае говорят, что x входит в область определения функции f, а

[1] В лекции 7 мы сопоставим таким картинкам (двудольные) графы.

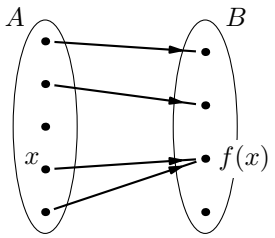


Рис. 6.2: функция

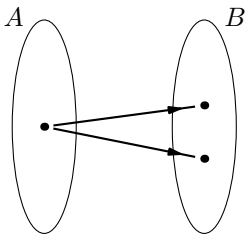


Рис. 6.3: не функция

этот самый единственный элемент y, сопоставленный элементу x, называют *значением функции f на x* и обозначают $f(x)$, как это сделано на рисунке 6.2. Элементы $f(x)$ для всех x из области определения функции f образуют *множество значений* функции f.

С помощью введенных обозначений мы уже можем записать область определения и область значения функции f на языке множеств. Для области определения получаем такое выражение:

$$\mathrm{Dom}(f) = \{x \in A \mid \exists y \in B\colon y = f(x)\},$$

а для области значений – такое:

$$\mathrm{Range}(f) = \{y \in B \mid \exists x \in A\colon y = f(x)\}$$

(см. замечание 5.3).

Область определения и область значения функции можно наглядно увидеть на рисунке, изображающем функцию из A в B. Из каждого элемента множества A ведёт не более одной стрелки. Это соответствует тому, что всякому элементу A соответствует не более одного элемента B. Те точки слева, из которых выходит стрелка, образуют область определения. Те точки справа, куда входят стрелки, образуют область значений функции. На рисунке 6.3 показано ключевая ситуация, которую мы в определении функции запрещаем: никакому элементу A не должно соответствовать несколько элементов B.

Пример 6.2. Функция «возведение в квадрат» определена для каждого целого x и ставит в соответствие этому x его квадрат (ещё говорят: отображает x в x^2). Множество значений этой функции имеет несколько названий, приведём два самых распространённых: «полные квадраты» и «квадратные числа».

Задача 6.3. Функция f из множества целых чисел в множество целых чисел сопоставляет числу x наименьшее простое число, которое больше x^2. Какова область определения f? Принадлежит ли число 19 множеству значений f?

Решение. Простых чисел бесконечно много. Поэтому для каждого числа найдутся простые числа, которые больше этого числа. Причём в этом множестве простых чи-

сел, как и в любом подмножестве множества натуральных чисел (т.е. целых неотрицательных), найдётся наименьшее число. Значит, областью определения f является всё множество целых чисел.

Функция «возведение в квадрат» обладает свойством строгой монотонности для неотрицательных чисел: если $0 \leqslant x < y$, то $x^2 < y^2$. Поэтому f нестрого монотонная на натуральных числах: если $x < y$, то $f(x) \leqslant f(y)$.

Так как $f(4) = 17$, а $f(5) = 29$, то 19 не является значением функции f: если $f(x) = 19$, то $4 < x < 5$, а такого целого числа нет. □

Как и для множеств, для функций возникает вопрос о способах их задания. Если функция имеет конечную область определения, её можно задать таблицей: для каждого элемента области определения достаточно указать значение в нем.

Если же область определения бесконечна или просто слишком большая для полного перечисления, то функцию можно задать с помощью правила, по которому из аргумента функции получается ее значение. Такое правило можно задавать разными способами, например, формулой, словесным описанием, алгоритмом или компьютерной программой. Подчеркнём, однако, что в математике функцией называется именно соответствие. Одну и ту же функцию можно задавать многими разными способами (сравните с заданием чисел: 2, $1 + 1$ и $\sqrt{4}$ задают одно и то же число разными способами).

Если область определения функции совпадает с множеством A, то пишут $f\colon A \to B$. Такие функции ещё называют *всюду определёнными*, а иногда *тотальными* (по-английски total) в отличие от общего случая *частичных* (или частично определённых, англ. partial) функций. Например, на рисунке 6.2 изображена частичная функция: легко увидеть элемент в множестве A, который не принадлежит области определения.

Вместе с обозначением $f\colon A \to B$ используется также и обозначение с «ограниченной стрелкой» $f\colon x \mapsto y$, которое указывает, что значение функции f на x равно y, т.е. это обозначение заменяет более привычное равенство $y = f(x)$.

6.2.2 Образ множества, полный прообраз

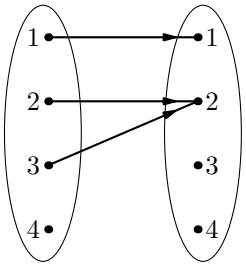
Функция f из множества A в множество B устанавливает соответствие между элементами множеств A и B. Это соответствие можно естественным образом продолжить до соответствия между подмножествами множеств A и B.

Пусть $X \subseteq A$ – подмножество множества A. Функция f сопоставляет ему *образ* $f(X) \subseteq B$ подмножества X. По определению $f(X)$ состоит в точности из тех элементов множества B, которые являются значениями элементов из X. Используя введённые нами для множеств обозначения, это можно записать как

$$f(X) = \{b \mid \exists x \in X : b = f(x)\}.$$

В частности, если в качестве X взять само множество A, то легко увидеть, что $f(A) = \mathrm{Range}(f)$.

Пример 6.4. Рассмотрим следующую функцию f из множества $A = \{1, 2, 3, 4\}$ в себя.



Если $X = \{1, 2\}$, то образом множества X будет $f(X) = \{1, 2\}$. Если же $X = \{3, 4\}$, то $f(X) = \{2\}$.

Чтобы немного привыкнуть к понятию образа, разберём несколько несложных задач. Прежде чем читать решения попробуйте решить задачи сами.

Задача 6.5. Может ли для некоторой функции f и множеств $X_1 \neq X_2$ быть так, что $f(X_1) = f(X_2)$?

Решение. Да так может случиться. Например, можно рассмотреть функцию f из примера 6.4 и множества $X_1 = \{2\}$ и $X_2 = \{2, 3\}$. Поскольку числа 2 и 3 под действием функции f обе переходят в число 2, то $f(X_1) = f(X_2) = \{2\}$.

Попробуйте придумать другую пару множеств с такими же свойствами для той же функции f. □

Попробуем немного усложнить задачу.

Задача 6.6. Может ли для некоторой функции f и множеств X_1, X_2, таких что $X_1 \cap X_2 = \varnothing$ быть так, что $f(X_1) = f(X_2)$?

Решение. Да, так тоже может случиться. Можно снова посмотреть на функцию f из примера 6.4 и взять $X_1 = \{2\}$ и $X_2 = \{3\}$. Аналогично прошлой задаче получим $f(X_1) = f(X_2) = \{2\}$, хотя теперь множества X_1 и X_2 не пересекаются. □

Задача 6.7. Верно ли для любых f, X_1, X_2 равенство $f(X_1 \cup X_2) = f(X_1) \cup f(X_2)$?

Решение. Ответ: да.

Условие $b \in f(X_1 \cup X_2)$ означает по определению, что $b = f(a)$, где $a \in X_1 \cup X_2$. Последнее означает, что $a \in X_1$ или $a \in X_2$. В первом случае получаем, что $b \in f(X_1)$, во втором — что $b \in f(X_2)$. Поскольку один из двух случаев имеет место, получаем, что $b \in f(X_1) \cup f(X_2)$. Значит всякий элемент множества $f(X_1 \cup X_2)$ лежит также в множестве $f(X_1) \cup f(X_2)$, то есть $f(X_1 \cup X_2) \subseteq f(X_1) \cup f(X_2)$.

С другой стороны, если $U \subseteq W$, то прямо из определения образа легко увидеть, что $f(U) \subseteq f(W)$. Значит, $f(X_1) \subseteq f(X_1 \cup X_2)$ и $f(X_2) \subseteq f(X_1 \cup X_2)$, то есть $f(X_1) \cup f(X_2) \subseteq f(X_1 \cup X_2)$.

Из включений в обе стороны следует равенство $f(X_1 \cup X_2) = f(X_1) \cup f(X_2)$. □

Функция f из множества A в множество B ставит в однозначное соответствие элементам A элементы B, но однозначного соответствия элементам B элементов A может и не задавать. Например, для функции из примера 6.4 в число 2 переходит сразу два числа 2 и 3. Если мы хотим отобразить всё в обратную сторону, то неясно, какое из этих чисел нужно поставить в соответствие числу 2.

Однако, если перейти к соответствию между подмножествами, то функция f устанавливает также однозначное соответствие и в обратную сторону — между подмножествами множества B и подмножествами множества A. Подмножеству $Y \subseteq B$ можно сопоставить *полный прообраз* $f^{-1}(Y) \subseteq A$ подмножества Y. По определению $f^{-1}(Y)$ состоит в точности из тех элементов A, значения которых лежат в Y. Или формально

$$f^{-1}(Y) = \{a : f(a) \in Y\}.$$

Аналогично образу, нетрудно заметить, что $f^{-1}(B) = \mathrm{Dom}(f)$, то есть прообраз всего множества B совпадает с областью определения функции.

Пример 6.8. Посмотрим еще раз на функцию f из примера 6.4. Для множества $Y = \{2\}$ прообразом являются все элементы множества A, которые переходят в Y под действием f. Нетрудно видеть, что $f^{-1}(Y) = \{2, 3\}$. Аналогично, если $Y = \{1\}$, то $f^{-1}(Y) = \{1\}$.

Заметим также, что число 4 не лежит ни в каком прообразе, то есть $4 \notin f^{-1}(Y)$ ни для какого Y, поскольку 4 никуда не переходит под действием f.

Следующие несколько задач полезно также попробовать решить самостоятельно, прежде чем читать решения.

Задача 6.9. Может ли для некоторой f и некоторых множеств $Y_1 \neq Y_2$ быть так, что $f^{-1}(Y_1) = f^{-1}(Y_2)$?

Решение. Да так может случиться и в качестве примера снова можно взять функцию f из примера 6.4. Рассмотрим множества $Y_1 = \{2\}$ и $Y_2 = \{2, 3\}$. Поскольку в число 2 под действием функции f переходят числа 2 и 3, а в число 3 ничего не переходит, то $f^{-1}(Y_1) = f^{-1}(Y_2) = \{2, 3\}$.

Вообще, если к множеству Y добавить элемент B, в который ничего не переходит, то прообраз не изменится (проверьте!). □

Задача 6.10. Верно ли, что если для некоторой функции f и множеств Y_1, Y_2 выполняется $Y_1 \cap Y_2 = \varnothing$, то выполняется и $f^{-1}(Y_1) \cap f^{-1}(Y_2) = \varnothing$?

Решение. Да, оказывается, что это верно. Действительно, предположим от противного, что существует $a \in f^{-1}(Y_1) \cap f^{-1}(Y_2)$. Тогда $a \in f^{-1}(Y_1)$ и $a \in f^{-1}(Y_2)$. Тогда по определению прообраза получаем, что $f(a) \in Y_1$ и $f(a) \in Y_2$. То есть у множеств Y_1 и Y_2 есть общий элемент, что противоречит пустоте их пересечения. □

Задача 6.11. Функция f из $\mathbb{N}$ в $\mathbb{N}$ сопоставляет числу n наибольший простой делитель числа n. Найдите полный прообраз множества чётных чисел.

Решение. Среди простых чисел есть всего одно чётное число 2. Поэтому в множество значений f входит лишь одно чётное число 2. Значит, $f^{-1}(2\mathbb{N}) = f^{-1}(\{2\})$.

Если $f(n) = 2$, то по определению функции f у числа n все простые делители не превосходят наименьшего простого числа 2. Значит, число n является степенью двойки.

Ответ: $f^{-1}(2\mathbb{N}) = \{2^k : k > 0\ k \in \mathbb{N}\}$. □

Обратите внимание, что функция f из предыдущей задачи частичная. Она не определена для $x = 1$ и $x = 0$, так как у 1 нет ни одного простого делителя, а 0 делится на все простые числа вообще и поэтому нет наибольшего простого делителя 0.

Задача 6.12. Верно ли для любых f, Y_1, Y_2 равенство $f^{-1}(Y_1 \cap Y_2) = f^{-1}(Y_1) \cap f^{-1}(Y_2)$?

Решение. Ответ: да.

Условие $a \in f^{-1}(Y_1 \cap Y_2)$ означает по определению прообраза, что $f(a) \in Y_1 \cap Y_2$, то есть $f(a) \in Y_1$ и $f(a) \in Y_2$. Поэтому $a \in f^{-1}(Y_1) \cap f^{-1}(Y_2)$. Значит, $f^{-1}(Y_1 \cap Y_2) \subseteq f^{-1}(Y_1) \cap f^{-1}(Y_2)$.

В другую сторону, если $a \in f^{-1}(Y_1) \cap f^{-1}(Y_2)$, то по определению $f(a) \in Y_1$ и $f(a) \in Y_2$. Значит, $f(a) \in Y_1 \cap Y_2$, то есть $a \in f^{-1}(Y_1 \cap Y_2)$. □

Задача 6.13. Функция f определена на множестве A и принимает значения в множестве B, при этом $Y \subseteq B$. Можно ли утверждать, что какое-нибудь включение между множествами $f(f^{-1}(Y))$ и Y выполняется всегда?

Решение. Рассмотрим произвольный $b \in f(f^{-1}(Y))$. По определению образа это означает, что есть некоторый $a \in f^{-1}(Y)$, такой что $f(a) = b$. По определению прообраза включение $a \in f^{-1}(Y)$, в свою очередь, означает, что $f(a) \in Y$, а значит $b \in Y$. Значит всякий элемент множества $f(f^{-1}(Y))$ является элементом Y, и мы доказали включение $f(f^{-1}(Y)) \subseteq Y$.

Обратное включение может не выполняться. Еще раз рассмотрим функцию f из примера 6.4 и рассмотрим множество $Y = \{2, 4\}$. Тогда $f^{-1}(Y) = \{2, 3\}$, а $f(\{2, 3\}) = \{2\}$. Так что множество $f(f^{-1}(Y)) = \{2\}$ не содержит множество $Y = \{2, 3\}$. На самом деле, мы сейчас применили следующее простое наблюдение: если взять какое-то множество и сначала перейти к прообразу, а затем к образу, то на стадии перехода к прообразу потеряются элементы множества, в которые ничего под действием функции не переходило.

Ответ: Включение $f(f^{-1}(Y)) \subseteq Y$ верно всегда, обратное включение может быть неверно. □

6.3 Декартово произведение множеств и графики функций

Для функции f, аргументами и значениями которой являются действительные числа, можно определить её *график* — множество всех точек координатной плоскости с

координатами $(x, f(x))$. Рисунок графика даёт наглядное и удобное представление о функции.

При этом график однозначно задаёт функцию: если множество Γ точек координатной плоскости является графиком некоторой функции f из $\mathbb{R}$ в $\mathbb{R}$, то значение $f(a)$ можно найти так: нужно рассмотреть пересечение Γ с прямой $x = a$. Если это пересечение пусто, то a не принадлежит области определения f. Если (a, b) — единственная точка, лежащая в этом пересечении, то $f(a) = b$. (См. рис. 6.4.)

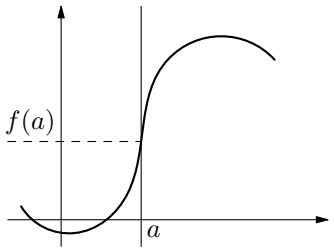


Рис. 6.4: Нахождение значения функции по её графику

Оказывается, график функции можно определить и в случае функций на произвольных множествах. Но для этого нам потребуется новая операция на множествах: декартово произведение.

Декартово произведение множеств A и B состоит из всех *упорядоченных пар* (a, b), в которых $a \in A$, $b \in B$.

Пример 6.14. Пусть $A = \{a, m, d\}$, а $B = \{1, 2\}$. Тогда всего есть 6 упорядоченных пар, в которых $a \in A$, $b \in B$, и по определению

$$A \times B = \{(a, 1),\ (a, 2),\ (d, 1),\ (d, 2),\ (m, 1),\ (m, 2)\}.$$

Упорядоченная пара (a, b) отличается от множества $\{a, b\}$, которое называется *неупорядоченной парой*, в двух отношениях. Во-первых, в упорядоченной паре различаются первый и второй элементы. Поэтому $(a, b) \neq (b, a)$ при $a \neq b$, а $\{a, b\} = \{b, a\}$. Во-вторых, в паре всегда два элемента — один на первом месте, а второй на втором. Поэтому хотя $\{a, a\} = \{a\}$, неверно, что $(a, a) = a$.

Другими словами, мы считаем, что

$$(a, b) = (a', b'),$$

если и только если $a = a'$ и $b = b'$.

Формально, упорядоченные пары также можно определить через множества. Это уже относится к довольно тонким вопросам теории множеств и обсуждается в следующей задаче. При первом чтении эту задачу можно пропустить.

Задача 6.15. Польский математик Куратовский в первой половине XX века придумал, как определить формально упорядоченные пары: он предложил считать, что

$$(a, b) = \{\{a\}, \{a, b\}\}.$$

Проверьте (аккуратно разобрав случаи), что при таком определении множества (a, b) и (a', b') равны тогда и только тогда, когда $a = a'$ и $b = b'$.

Задача может показаться совсем простой, но стоит вспомнить, что элементы множеств могут быть устроены довольно сложно, могут сами быть множествами и т.д. Так что формально для решения задачи требуется аккуратная проверка.

Решение. Пусть $\{\{a\}, \{a, b\}\} = \{\{a'\}, \{a', b'\}\}$. Равенство множеств означает, что они содержат одни и те же элементы. Поскольку в множествах не более двух элементов, есть ровно два способа отождествить их.

(I) $\{a\} = \{a'\}$, $\{a, b\} = \{a', b'\}$. Первое равенство означает, что $a = a'$. Тогда второе равенство приобретает вид $\{a, b\} = \{a, b'\}$. Поскольку b' обязан быть элементом левой части этого последнего равенства, возникают два подслучая:

(Ia) $a = b'$. Тогда $\{a, b\} = \{a\}$ и поэтому $b = a$. Значит, $a = a'$ и $b = a = b'$, что и требовалось доказать.

(Ib) $b = b'$. Что и требовалось доказать.

(II) $\{a\} = \{a', b'\}$, $\{a, b\} = \{a'\}$. Из первого равенства получаем $a' = a = b'$, из второго $a = a' = b$. Поэтому все четыре элемента совпадают, так что $a = a'$ и $b = a = b'$. Что и требовалось доказать. □

Теперь мы готовы дать определение графика функции.

Определение 6.1. *Графиком* функции f из A в B называется подмножество $\Gamma_f \subseteq A \times B$, состоящее из всех таких упорядоченных пар (a, b), что $f(a) = b$.

Пример 6.16. Графиком функции f из примера 6.4 является множество

$$\Gamma_f = \{(1, 1), (2, 2), (3, 2)\}.$$

Как нетрудно видеть, определение полностью аналогично определению графика числовой функции. Это объясняет одну часть термина «декартово произведение».

Слово «декартово» происходит от фамилии французского математика и философа Рене Декарта.[2] Он придумал, что на плоскости можно ввести систему «декартовых координат», проведя две перпендикулярные ориентированные прямые, оси координат. После этого каждая точка плоскости задаётся упорядоченной парой чисел (x, y), своих координат по оси абсцисс и оси ординат. Заметим, что пара именно упорядоченная: точки $(1, 2)$ и $(2, 1)$ — это разные точки. Можно сказать, что декартовы координаты устанавливают взаимно однозначное соответствие между плоскостью (ими снабжённой) и декартовым произведением $\mathbb{R} \times \mathbb{R}$ множества действительных чисел на себя.

[2]Как философ он попал в «Маскарад» к Лермонтову: один из игроков, предвосхищая современных экономистов, говорит Арбенину: «Что ни толкуй Волтер или Декарт — // Мир для меня — колода карт,// Жизнь — банк; рок мечет, я играю,// И правила игры я к людям применяю.» Декарт-философ также известен латинским изречением Cogito ergo sum (мыслю, следовательно, существую).

Можно ещё объяснить, откуда берётся слово «произведение». В комбинаторике произведение чисел возникает как раз при подсчёте пар. Если множество A содержит m элементов, а множество B содержит n элементов, то с каждым из m элементов множества A можно составить n пар, спаривая его с каждым из n элементов множества B. Всего получается $m \times n$ элементов.

Не всякое подмножество декартова произведения $A \times B$ является графиком какой-нибудь функции. Полезно понять, чем графики функций отличаются от произвольных подмножеств декартова произведения.

Как мы помним, функция сопоставляет аргументу из области определения единственное значение. Поэтому в графике функции нет двух пар с одинаковой первой компонентой и разными вторыми.

Верно и обратное: если $\Gamma \subseteq A \times B$ таково, что в пересечении Γ и любого множества вида $\{(a, y) : y \in B\}$ лежит не более одного элемента, то Γ является графиком некоторой функции из A в B.

Доказательство легко получить, следуя рис. 6.4. Определим функцию f_Γ из A в B следующим правилом:

$$f_\Gamma(a) = \begin{cases} b, & \text{если } \Gamma \cap \{(a, y) : y \in B\} = \{(a, b)\}, \\ \text{не определена}, & \text{если } \Gamma \cap \{(a, y) : y \in B\} = \varnothing. \end{cases}$$

Нетрудно проверить, что графиком функции f_Γ является множество Γ.

Таким образом мы доказали следующую лемму.

Лемма 6.2. *Множество $\Gamma \subseteq A \times B$ является графиком некоторой функции тогда и только тогда, когда для любых $x \in A$ и $y_1, y_2 \in B$ с $y_1 \neq y_2$ хотя бы одна из пар (x, y_1) и (x, y_2) не лежит в Γ.*

Теперь мы проделали достаточную работу, чтобы определить понятие функции формально. Оказывается, что функции можно определить через множества, а именно, формально мы будем отождествлять функции и их графики.

Определение 6.3. Функция из A в B — это такое подмножество F декартова произведения $A \times B$, что для любых $x \in A$ и $y_1, y_2 \in B$ с $y_1 \neq y_2$ хотя бы одна из пар (x, y_1) и (x, y_2) не лежит в F.

Этот формализм может показаться избыточным. Его основные достоинства лежат в области оснований математики и их обсуждение выходит за рамки данной книги.

Заметим лишь, что данное формальное определение фиксирует понятие равенства функций. Функции равны тогда и только тогда, когда их графики равны как множества.

Задача 6.17. Равны ли функции из $\mathbb{R}$ в $\mathbb{R}$, задаваемые формулами

$$\frac{x^2 - 1}{x + 1} \qquad \text{и} \qquad x - 1?$$

Решение. Ответ: нет.

Точка $(-1, 0)$ принадлежит графику функции, задаваемой второй формулой, и не принадлежит графику функции, задаваемой первой формулой (значение которой не определено при $x = -1$). Значит, эти функции как подмножества $\mathbb{R} \times \mathbb{R}$ различны. □

Мы пока говорили лишь о функциях от одного аргумента. Как определить функции от нескольких аргументов? Определение через график функции легко обобщается на функции многих аргументов: k-местной функцией из множеств A_1, A_2, …, A_k в множество B называется такое подмножество $F \subseteq A_1 \times A_2 \times \cdots \times A_k \times B$, что для любых $(x_1, x_2, \ldots, x_k) \in A_1 \times A_2 \times \cdots \times A_k$ и $y_1, y_2 \in B$ с $y_1 \neq y_2$ хотя бы один из списков $(x_1, x_2, \ldots, x_k, y_1)$ и $(x_1, x_2, \ldots, x_k, y_2)$ не лежит в F.

Мы использовали декартово произведение нескольких множеств, хотя и не определили его. Это определение аналогично определению декартова произведения двух множеств: *декартово произведение* $A_1 \times A_2 \times \cdots \times A_k$ — это множество, состоящее из всех списков[3] длины k, в которых на i-м месте стоит элемент из множества A_i.

6.4 Инъекции, сюръекции и биекции

Довольно часто приходится рассуждать только про всюду определённые функции. В таком случае хочется говорить короче. Всюду определённую функцию $f\colon A \to B$ мы будем называть *отображением* множества A в множество B, когда нужно подчеркнуть, что функция всюду определена. То есть, когда речь идёт о функции, мы допускаем, что она не всюду определена; а вот отображение — частный случай понятия функции — считаем всюду определённым.[4]

Введём три важных частных случая отображений.

6.4.1 Определения

Отображение $f\colon A \to B$ называется *инъекцией*, если его значения в различных точках различны, то есть разным элементам из множества A ставятся в соответствие разные элементы множества B. Иначе говоря, f — инъекция, если $f(x_1) = f(x_2)$ влечёт $x_1 = x_2$. (Переформулировка: f — инъекция, если $x_1 \neq x_2$ влечёт $f(x_1) \neq f(x_2)$ — тут мы ещё раз встречаемся с принципом контрапозиции.)

Пример инъекции изображён на рис. 6.5. Инъективность означает, что нет двух стрелок, ведущих из разных точек в одну, как на рис. 6.6.

Контрольный вопрос 6.18. Можно рассмотреть свойство, похожее на свойство инъекций, но с импликацией в другую сторону: «если $f(x_1) \neq f(x_2)$, то $x_1 \neq x_2$». Что за свойство мы таким образом определяем?

[3] Список, последовательность, упорядоченный набор — это синонимы, мы их используем взаимозаменяемо.

[4] Такое разделение терминов не является общепринятым. Советуем читателю быть бдительным и проверять, о чём идёт речь, когда в математической книге встречаются слова «функция» и «отображение».

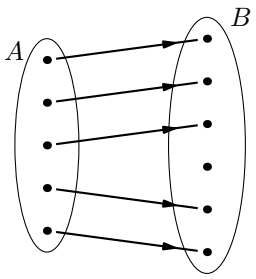


Рис. 6.5: инъекция

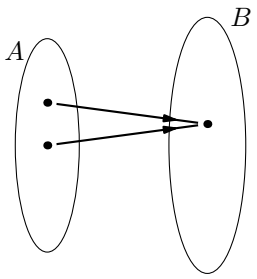


Рис. 6.6: не инъекция

Отображение $f\colon A \to B$ называется *сюръекцией*, если его область значений совпадает со всем множеством B, то есть если для всякого элемента $y \in B$ найдётся элемент $x \in A$, для которого $f(x) = y$.

Пример сюръекции изображён на рис. 6.7. Примеры не сюръекций изображены на рисунке 6.8 (а также на рисунке 6.5): для сюръективной функции не должно быть точек справа, в которые не ведёт ни одной стрелки.

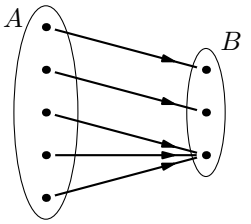


Рис. 6.7: сюръекция

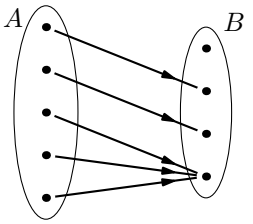


Рис. 6.8: не сюръекция

Пример 6.19. Отображение множества списков в множество подмножеств, которое мы рассмотрели в разделе 6.1, является сюръекцией (каждое конечное множество можно записать в виде списка его элементов).

Отображение $f\colon A \to B$ называется *биекцией*, если оно одновременно является и инъекцией, и сюръекцией. Другими словами, функция является биекцией, если всякому элементу из B соответствует ровно один элемент из A (здесь за "хотя бы один" отвечает свойство сюръективности, а за "не более одного" — свойство инъективности).

В изображении биекции из каждой точки слева выходит ровно одна стрелка, и в каждую точку справа ровно одна стрелка входит. То есть каждая точка с одной стороны соединена ровно с одной точкой с другой стороны. Поэтому биекции ещё называют *взаимно однозначными соответствиями*: соединённые вершины, то есть аргумент функции и её значение, *соответствуют* друг другу при этой биекции.

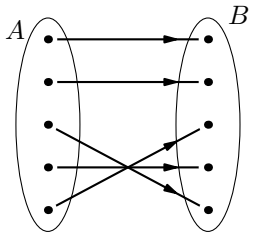


Рис. 6.9: биекция

Пример 6.20. Рассмотрим отображение $c\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$, заданное формулой

$$c(x,y)=\binom{x+y+1}{2}+y.$$

Докажем, что это биекция.

Для каждого n рассмотрим множество пар натуральных чисел

$$S_n=\{(x,y)\colon x+y=n\}.$$

Пары из множества S_n однозначно задаются ординатами, другими словами, есть биекция между S_n и целыми числами отрезка $[0,y]$.

На множестве S_n отображение c прибавляет к координате y число $\binom{n+1}{2}$. Поэтому оно задаёт биекцию между S_n и целыми точками отрезка

$$\left[\binom{n+1}{2};\binom{n+1}{2}+n\right].$$

Из определения ясно, что множества S_n и S_m не пересекаются при $n\ne m$, а всё множество пар натуральных чисел является объединением множеств S_n, $n\in\mathbb{N}$.

Если доказать, что отрезки указанного выше вида не пересекаются, а их объединение совпадает с $\mathbb{N}$, получим искомое: отображение c тогда является биекцией между $\mathbb{N}\times\mathbb{N}$ и $\mathbb{N}$.

По одному из основных свойств биномиальных коэффициентов имеем равенство

$$\binom{n+2}{2}=\binom{n+1}{2}+\binom{n+1}{1}=\binom{n+1}{2}+n+1.$$

Это и означает, что отрезки не пересекаются и в объединении дают всё $\mathbb{N}$: каждый следующий отрезок начинается непосредственно после предыдущего.

Замечание: понять, что отображение c является биекцией, намного проще, если начать выписывать его значения в точках множества $\mathbb{N}\times\mathbb{N}$ на координатной плоскости (попробуйте!).

6.4.2 Биекции и сравнение множеств

Задача 6.21. Пусть A и B — конечные множества, состоящие из a и b элементов. При каком условии на a и b существует инъекция $f\colon A \to B$? Тот же вопрос для сюръекций и биекций.

Пояснение. Если посмотреть на картинки, сопровождающие определения, ответ становится ясен: при $a \leqslant b$ существует инъекция $f\colon A \to B$, при $a \geqslant b$ существует сюръекция $f\colon A \to B$, при $a = b$ существует биекция $f\colon A \to B$.

Такие картинки кажутся понятными и не требующими каких-то дополнительных обоснований, и в данный момент мы оставим этот вопрос как раз на таком уровне строгости. Но мы вернёмся к нему в лекции 8 и обсудим, как такие вещи можно доказывать аккуратнее. □

Установление взаимно однозначного соответствия (биекции) часто является удобным способом показать, что в двух множествах одинаковое количество точек.

Задача 6.22. На окружности выбрано 10 точек. Чего больше: треугольников с вершинами в этих точках или выпуклых семиугольников с вершинами в этих точках?

Решение. Ответ: их поровну.

Есть взаимно однозначное соответствие между этими множествами: треугольнику соответствует выпуклый семиугольник, соединяющий остальные вершины На

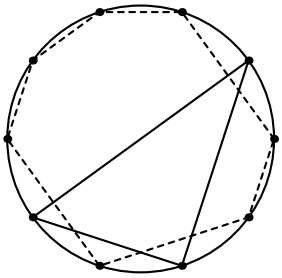


Рис. 6.10: Треугольник и соответствующий семиугольник

самом деле, конечно, треугольник и семиугольник тут только для рисунка, а речь идёт о трёх- и семиэлементных подмножествах десятиэлементного множества и равенстве $\binom{10}{3} = \binom{10}{7}$. □

Задача 6.23. На окружности выбрано несколько чёрных точек и одна белая. Чего больше: треугольников с чёрными вершинами или выпуклых четырёхугольников с тремя чёрными и одной белой вершиной? выпуклых многоугольников, у которых все вершины чёрные, или выпуклых многоугольников, у которых все вершины, кроме одной, чёрные, а одна белая?

Решение. Ответ на первый вопрос «поровну».

Важно заметить, что выпуклый многоугольник однозначно задаётся своими вершинами. Поэтому в первом случае нужно построить такую биекцию: каждым трём чёрным точкам (вершинам чёрного треугольника) сопоставим четвёрку точек, в которую входят эти же точки и ещё белая вершина. Это сюръекция: отбросив белую точку, получаем тройку чёрных точек. Эта тройка чёрных точек однозначно определяется выпуклым четырёхугольником с тремя чёрными и одной белой вершиной. Значит, это и инъекция. То есть мы установили биекцию между множествами, поэтому в них одинаковое количество элементов.

Во втором случае рассуждение аналогично, но ответ зависит от того, что понимать под выпуклым многоугольником. Мы используем стандартное школьное определение. Из этого определения, в частности, следует, что в многоугольнике не меньше 3 вершин. Соответствие, при котором многоугольнику с чёрными вершинами сопоставлен многоугольник с теми же вершинами и добавленной белой вершиной, является инъекцией аналогично предыдущему. Но теперь это соответствие не является сюръекцией: треугольник с двумя чёрными и одной белой вершинами не имеет прообраза.

Ответ на второй вопрос: выпуклых многоугольников, у которых все вершины чёрные, меньше, чем выпуклых многоугольников, у которых все вершины, кроме одной, чёрные, а одна белая. □

Приведём три важные и очень естественные биекции, которые уже встречались и будут дальше встречаться постоянно.

Для натурального n рассмотрим три множества.

1. Множество подмножеств n-элементного множества. Для определённости пусть это будет множество целых чисел от 1 до n, которое мы обозначим $\{1, 2, \ldots, n\}$;
2. Множество отображений множества $\{1, 2, \ldots, n\}$ в множество $\{0, 1\}$;
3. Множество двоичных слов длины n.

Между этими множествами есть очень простые и естественные биекции, которые позволяют переходить от высказываний об одном из этих множеств к высказываниям о другом.

Подмножеству $S \subseteq \{1, 2, \ldots, n\}$ соответствует индикаторная функция

$$\chi_S(x) = \begin{cases} 1, & \text{если } x \in S, \\ 0 & \text{в противном случае.} \end{cases}$$

Это соответствие взаимно однозначно: индикаторная функция однозначно определяется по множеству, но и для любого отображения $f\colon \{1, 2, \ldots, n\} \to \{0, 1\}$ множество находится однозначно. А именно, для $S = f^{-1}(\{1\})$ выполняется равенство

$$f(x) = \chi_S(x)$$

для всех x.

Если расположить числа от 1 до n в порядке возрастания и записать значения отображения $f\colon \{1, 2, \ldots, n\} \to \{0, 1\}$, то получим двоичное слово. И это соответствие также взаимно однозначно. По двоичному слову длины n соответствующая ему функция определяется так: $f(i)$ равно символу, который стоит в слове на i-м месте.

Аналогичная биекция есть и с подмножествами: слову соответствует подмножество номеров единиц в двоичном слове.

Заметим также, что первая биекция может быть определена и для бесконечных множеств.

Инъекции, сюръекции и биекции можно использовать для сравнения бесконечных множеств на предмет выяснения того, «где больше элементов». Но тут надо быть осторожным. Ещё Галилей (тот самый, которому приписывают фразу «а всё-таки она вертится» после допросов «компетентными органами»[5]) заметил, что точные квадраты натуральных чисел ($0^2 = 0$, $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $\ldots$) составляют лишь небольшую часть натуральных чисел, но находятся с ними во взаимно однозначном соответствии ($x \mapsto x^2$), и что маленький отрезок в этом смысле содержит столько же точек, сколько большой.

Задача 6.24. Как установить взаимно однозначное соответствие между отрезками разной длины?

Подсказка. Вспомните про подобия и пропорции. Или см. лекцию 8. □

Галилей заключил, что про бесконечные множества нет смысла спрашивать, какое из них больше, и единственное, что мы можем сказать, что они бесконечны. Лишь через три с лишним века Георг Кантор понял, что сравнение бесконечных множеств тоже имеет смысл, только нужно быть аккуратным. В частности, он понял, что на отрезке больше точек, чем натуральных чисел в натуральном ряду. Мы этим займёмся позже (см. лекцию 8).

6.5 Композиции функций

Функции можно применять одну за другой. В результате получается новая функция, которая называется *композицией*.

6.5.1 Определение

Определение 6.4. Пусть даны функции f из множества A в множество B и g из множества B в множество C. Их *композицией* называется функция $g \circ f$ из A в C, которая определена на тех x из области определения функции f, для которых $f(x)$ принадлежит области определения функции g, и равна $g(f(x))$.

[5] В истории науки Галилей более известен тем, что по праву считается основателем современной физики и целого ряда прикладных наук, включая метеорологию и сопротивление материалов.

С точки зрения программирования композицию можно описать так:

```
function GoF(a:A):C;
  var b:B;
begin b:=F(a); GoF:=G(b); end;
```

Здесь A, B, C задают типы соответствующих переменных; мы для наглядности упомянули переменную типа B, но можно было бы просто написать

```
function GoF(a:A):C;
begin GoF:=G(F(a)); end;
```

Композиция функций очень наглядно изображается графически, см. рис. 6.11. Стрелки, отвечающие композиции, получаются соединением стрелок, отвечающих функциям f и g.

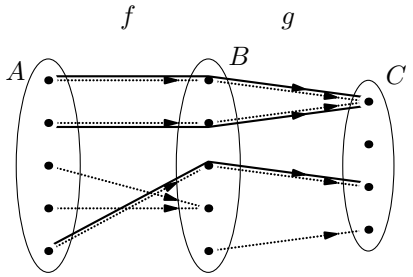


Рис. 6.11: композиция функций $g \circ f$

Обратите внимание, что в записи $g \circ f$ мы сначала пишем ту функцию, которая применяется второй, — просто потому, что такой же порядок в записи $g(f(x))$.

Задача 6.25. Пусть f всюду определена, а g не всюду определена. Верно ли, что композиция $g \circ f$ не всюду определена?

Решение. Ответ: нет.

Рассматриваем функции из $\mathbb{N}$ в $\mathbb{N}$. Пусть $f(x) = 0$ для всех x, $g(0) = 1$, а при $x > 0$ функция g не определена. Тогда $g \circ f(x) = 1$ для всех x. □

Задача 6.26. Какое преобразование плоскости является композицией двух осевых симметрий? (Тут есть два случая: когда оси симметрий пересекаются и когда они параллельны.)

Указание. Для знающих геометрию это несложное упражнение. Остальным читателям лучше пропустить эту задачу (и аналогичные задачи из раздела 6.7) и вернуться к ним после изучения геометрии. □

6.5.2 Ассоциативность

Композиция функций обладает свойством *ассоциативности*

$$(f \circ g) \circ h = f \circ (g \circ h).$$

Здесь мы предполагаем, что множества, для которых определены функции, выбраны согласованно: h — это функция из множества A в множество B; g — это функция из множества B в множество C; а f — это функция из множества C в множество D (иначе композиции нельзя определить).

Графически свойство ассоциативности выражается очень наглядно, см. рисунок 6.12.

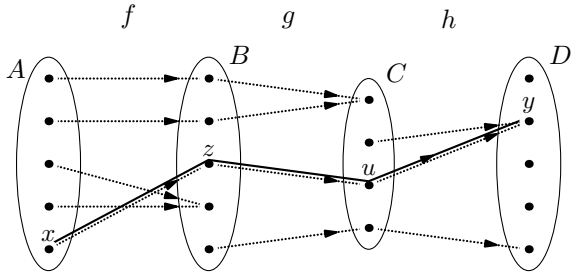


Рис. 6.12: ассоциативность композиции. Стрелка, отвечающая композиции трёх функций, заменяет путь по трём стрелкам, отвечающим функциям h, g и f соответственно

Формальное обоснование ассоциативности довольно техническое: обе части равенства задают функции r_1 и r_2 из множества A в множество D. Равенство $y = r_1(x)$ равносильно тому, что найдётся такое $z \in B$, что выполняются равенства $z = h(x)$, $y = (f \circ g)(z) = f(g(z))$.

Обозначим $u = g(z)$. Тогда $u = (g \circ h)(x)$ и $y = f(u)$, что по определению равносильно $y = r_2(x)$.

Контрольный вопрос 6.27. Проследите за предыдущим рассуждением по рисунку 6.12.

6.5.3 Обратная функция

Тождественной функцией на множестве A (или тождественным отображением множества A в себя) называется функция $\mathrm{id}_A \colon A \to A$, которая отображает всякий элемент $x \in A$ в себя: $\mathrm{id}_A(x) = x$. При композиции тождественные функции ведут себя, как единица при умножении: для любого отображения $f \colon A \to B$ выполнены равенства

$$\mathrm{id}_B \circ f = f \circ \mathrm{id}_A = f.$$

(Обратите внимание, что здесь две тождественные функции — одна на A, другая на B, иначе композицию нельзя определить.)

Если отображение $f\colon A \to B$ является биекцией (взаимно однозначным соответствием), то можно определить *обратную функцию* (или обратное отображение) f^{-1}: если f отображает x в y, то обратная функция f^{-1} отображает y в x. Инъективность f гарантирует, что это действительно функция, а сюръективность f гарантирует, что эта функция определена на всём B.

Заметим, что определение обратной функции симметрично: если g обратна к f, то и f обратна к g.

Пример 6.28 (Продолжение примера 6.20.)**.** Как задать отображение, обратное к отображению $c\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ из примера 6.20?

Давайте найдём $c^{-1}(11)$. Для этого полезно выписать ряд отрезков, соответствующих множествам S_n:

$$[0;0],\ [1;2],\ [3;5],\ [6;9],\ [10;14].$$

Число 11 попадает в отрезок $[10;14]$. Обозначим $c^{-1}(11) = (a, b)$. Тогда $a + b = 4$, причём из определения $c(x, y)$ получаем, что $11 = 10 + b$, то есть $b = 1$, $a = 3$.

Аналогично определяется обратная функция и в общем случае. Правило вычисления $c^{-1}(x)$ такое:

- найти наибольшее n такое, что $\binom{n}{2} \leqslant x$;
- присвоить $b := x - \binom{n}{2}$; $a := n - b - 1$;
- пара $(a; b)$ является значением $c^{-1}(x)$.

Обозначение f^{-1} для обратной функции не случайно: мы сравнили композицию с умножением, а тождественную функцию с единицей. Продолжая эту линию, можно заметить ещё одно свойство, аналогичное свойству умножения.

Утверждение 6.5. *Для биекции $f\colon A \to B$ выполнены равенства*

$$f^{-1} \circ f = \mathrm{id}_A \ \text{ и } \ f \circ f^{-1} = \mathrm{id}_B\,.$$

Доказательство. Возьмём $x \in A$. Пусть $f(x) = y$. Тогда по определению обратной функции $f^{-1}(y) = x$, а по определению композиции $(f^{-1} \circ f)(x) = f^{-1}(f(x)) = f^{-1}(y) = x$. Значит, $(f^{-1} \circ f) = \mathrm{id}_A$.

Возьмём $y \in B$. Пусть $f^{-1}(y) = x$. Тогда по определению обратной функции $f(x) = y$, а по определению композиции $(f \circ f^{-1})(y) = f(f^{-1}(y)) = f(x) = y$. Значит, $(f \circ f^{-1}) = \mathrm{id}_B$. □

Замечание 6.1. К сожалению, как это нередко бывает, обозначение f^{-1} двусмысленно: оно используется как в обозначении обратной функции, так и в обозначении полного прообраза множества. Различить эти случаи обычно легко из контекста. Важно помнить, что обратная функция определена только для биекций, а полный прообраз множества определён для любой функции.

Обращение утверждения 6.5 даёт простой критерий обратной функции.

Теорема 6.6. *Если для отображений $f\colon A \to B$ и $g\colon B \to A$ выполнены два равенства $g \circ f = \mathrm{id}_A$ и $f \circ g = \mathrm{id}_B$, то функция f является биекцией и g обратна к f.*

Доказательство. Пусть $f(x_1) = f(x_2)$. Тогда из первого условия на композиции получаем:

$$x_1 = (g \circ f)(x_1) = g(f(x_1)) = g(f(x_2)) = (g \circ f)(x_2) = x_2.$$

Значит, функция f инъективна.

Для любого $y \in B$ из второго условия на композиции следует, что $y = f(g(y))$, то есть y принадлежит множеству значений f. Значит, функция f сюръективна.

Итак, f биекция.

Если $y = f(x)$, то из первого условия на композиции получаем $g(y) = g(f(x)) = x$. Значит, g обратна к f. □

Также для биекций верен следующий важный факт.

Теорема 6.7. *Если отображения $f\colon A \to B$ и $g\colon B \to C$ — биекции, то и их композиция $g \circ f\colon A \to C$ является биекцией.*

Доказательство. Разные элементы A переходят в разные элементы B, потому что f инъекция, и эти разные элементы B переходят в разные элементы C, потому что g инъекция. Таким образом, при $a \neq a'$ получаем $f(a) \neq f(a')$ и затем $g(f(a)) \neq g(f(a'))$, так что $g \circ f$ — инъекция. Ещё надо проверить, что $g \circ f$ — сюръекция, то есть что в каждый элемент $c \in C$ что-то переходит. Но мы знаем, что g — сюръекция, так что $c = g(b)$ для некоторого $b \in B$; поскольку f — сюръекция, то $b = g(a)$ для некоторого $a \in A$, так что $c = g(b) = g(f(a)) = (g \circ f)(a)$. □

Утверждение теоремы очевидно в случае конечных множеств: если между конечными множествами A и B есть биекция, то в них одинаковое количество элементов, а если между множествами B и C есть биекция, то и в них одинаковое количество элементов, а значит в множествах A и C также одинаковое количество элементов. Поэтому между множествами A и C есть биекция. Для бесконечных множеств существование биекции между множествами означает, что множества имеют равные мощности, так что теорема 6.7 доказывает, что понятие мощности для бесконечных множеств определено осмысленно: из неё следует, что если множество A равномощно B, а B равномощно C, то A равномощно C.

6.5.4 Степени композиций

Если функция f отображает множество A в себя, то можно рассмотреть её композицию с собой. Получится функция $f \circ f$, которую обозначают ещё через f^2. Надо только не путать с обычным возведением в квадрат, поскольку $\sin^2$ может обозначать и функцию $x \mapsto \sin(\sin(x))$, и функцию $(\sin x)^2$. Аналогично определяется и $f^3(x) = f(f(f(x)))$, и вообще $f^k(x)$, называемая также *k-й итерацией* функции f.

Задача 6.29. Докажите, что для любой функции $f\colon A \to A$, отображающей конечное множество A в себя, найдутся различные целые положительные m, n, при которых f^m и f^n совпадают.

Решение. Всего функций из конечного множества в конечное имеется лишь конечное число (см. следующий раздел 6.6). Поэтому в последовательности функций $f^2, f^3, \ldots, f^n, \ldots$ обязательно есть одинаковые. □

Задача 6.30. Докажите, что для любой биекции $f\colon A \to A$ конечного множества A в себя существует целое положительное n, при котором $f^n = \mathrm{id}_A$.

Решение. Пусть $f^n = f^m = f^{m-n} \circ f^n$ (мы считаем, что $m > n$). Для f есть обратная g, для которой $f \circ g = \mathrm{id}_A$. Тогда

$$f^2 \circ g^2 = f \circ f \circ g \circ g = f \circ \mathrm{id}_A \circ g = f \circ g = \mathrm{id}_A$$

(используем ассоциативность композиции).

По индукции доказываем, что $f^k \circ g^k = \mathrm{id}_A$ (выше фактически выписан индуктивный переход от $k = 1$ к $k = 2$, общий случай записывается аналогично).

Поэтому

$$\mathrm{id}_A = f^n \circ g^n = f^{m-n} \circ f^n \circ g^n = f^{m-n}$$

(опять используем ассоциативность композиции). □

Биекции конечного множества в себя называют *перестановками* этого множества, а минимальное n с указанным в задаче свойством называют *порядком* перестановки.

Контрольный вопрос 6.31. Мы уже использовали слово «перестановка» в главе 2 про подсчёты немного в другом смысле (см. с. 57). Объясните связь между этими двумя значениями термина «перестановка».

Композицию перестановок называют также их *произведением* и говорят о *группе перестановок*. Группу перестановок n-элементного множества называют также *симметрической группой* и обозначают S_n.

Задача 6.32. Мы уже определили f^n при целом $n \geqslant 2$, а также f^{-1} для биекций. Как надо определить f^0, f^1 и отрицательные степени (для биекций), чтобы выполнялось равенство

$$f^{m+n} = f^m \circ f^n$$

(при любых целых m и n)?

Решение. Полагаем $f^0 = \mathrm{id}$, $f^1 = f$, а для биекции f, обратная к которой g, полагаем $f^{-n} = g^n$.

Выше мы уже проверили, что $f^n \circ f^{-n} = \mathrm{id} = f^0$. Для $m > 0 > -n$ проверка выполняется аналогично:

$$f^m \circ f^{-n} = f^m \circ g^n = f^{m-1} \circ g^{n-1} = \begin{cases} f^{m-n}, & \text{если } m > n, \\ g^{n-m}, & \text{в противном случае.} \end{cases}$$

В обоих случаях полученное выражение совпадает с f^{m-n}. При $n > 0 > -m$ рассуждения аналогичные. □

6.6 Подсчёты

В этом разделе мы рассмотрим функции и отображения на конечных множествах и посчитаем количество таких функций. Пусть есть два конечных множества A и B, причем в A содержится m элементов, а в B содержится n элементов.

Посчитаем, сколько есть разных отображений $f\colon A \to B$. Для каждой точки $x \in A$ (для каждого элемента $x \in A$) есть n возможных значений для $f(x)$ (это значение может быть любым элементом множества B). Они выбираются независимо, так что всего есть n^m функций. Упомянем кстати, что множество всех всюду определённых функций $A \to B$ часто обозначается как раз B^A.

Раньше мы считали слова длины m, составленные из букв n-элементного алфавита. Это в сущности тот же самый подсчёт, поскольку каждое такое слово есть отображение множества из m позиций в алфавит.

Можно подсчитать и не всюду определённые функции из A в B. Теперь для каждого $f(x)$ есть $n+1$ вариантов: это значение может быть не определено, а может быть любым элементом множества B. Соответственно получаем ответ $(n+1)^m$.

Таким образом мы установили следующую лемму.

Лемма 6.8. *Пусть множества A и B состоят из m и n элементов соответственно. Тогда число различных функций из A в B равно $(n+1)^m$, а число различных отображений из A в B равно n^m.*

Сколько есть биекций $A \to B$? Если $m \neq n$, то нет ни одной. А если $m = n$? Будем снова выбирать значения по очереди, расположив все n элементов A в каком-то порядке. Для первого элемента допустимы все n значений, для второго — все, кроме одного (уже использованного раньше), для третьего — $(n-2)$ и так далее, всего получается $n! = 1 \cdot 2 \cdot \ldots \cdot n$. Собственно говоря, мы повторили рассуждение о перестановках; биекции — это просто научное название для тех же перестановок.

Получаем следующую лемму.

Лемма 6.9. *Пусть каждое из множеств A и B состоит из n элементов. Тогда число различных биекций из A в B равно $n!$.*

Аналогично можно подсчитать инъекции. Если $m > n$, то их нет (большее множество не может быть инъективно отображено в меньшее). Если $m \leqslant n$, то аналогичное рассуждение даёт $n(n-1)(n-2) \cdot \ldots \cdot (n-m+1)$ (или $n!/(n-m)!$).

Получаем следующую лемму.

Лемма 6.10. *Пусть множества A и B состоят из m и n элементов соответственно, причем $m \leqslant n$. Тогда число различных инъекций из A в B равно $n!/(n-m)!$.*

А вот для сюръекций дела немного хуже, и формула содержит суммирование. (Впрочем, и факториал тоже скрывает в себе много операций, так что принципиально разница невелика.)

Теорема 6.11. *Количество сюръекций m-элементного множества в n-элементное при $m \geqslant n$ равно*

$$\sum_{k=0}^{n}(-1)^k\binom{n}{k}(n-k)^m.$$

(и равно нулю при $m < n$).

Напомним, что сюръекции должны быть определены на всех m элементах.

Доказательство. Надо воспользоваться формулой включений и исключений. Пусть B состоит из n элементов $b_1, \ldots, b_n$. Не-сюръекции $A \to B$ — это те функции, область значений которых не содержит одного из элементов $b_1, \ldots, b_n$, то есть объединение множеств

$$A(b_1) \cup A(b_2) \cup \ldots \cup A(b_n),$$

где через $A(b)$ обозначается множество тех функций, которые не принимают значения b. Легко понять, что все множества $A(b)$ имеют размер $(n-1)^m$ (мы просто выбросили одно значение). Более того, столь же легко посчитать размер их пересечений: если $b \neq b'$, то $A(b) \cap A(b')$ — это функции, не принимающие значений b и b', их будет $(n-2)^m$. Остаётся воспользоваться формулой включений и исключений: из всех функций (n^m) надо вычесть n множеств вида $A(b_i)$, то есть $n(n-1)^m$, затем посокрушаться, что мы вычли лишнее и вернуть обратно $\binom{n}{2}$ их попарных пересечений, всего $\binom{n}{2}(n-2)^m$, затем снова вычесть тройные пересечения, которых $\binom{n}{3}$ и каждое из которых имеет размер $(n-3)^m$, и так далее. □

С точки зрения программирования эта формула даёт быстрый — по сравнению с перебором всех сюръекций — алгоритм нахождения их количества.

Замечание 6.2. Для количества сюръекций выполняется рекуррентное соотношение с граничными условиями

$$\begin{aligned}
&\mathrm{Surj}(m,n) = n\mathrm{Surj}(m-1,n) + n\mathrm{Surj}(m-1,n-1),\\
&\mathrm{Surj}(0,0) = 1, \quad \mathrm{Surj}(m,0) = 0, \text{ при } m > 0,\\
&\mathrm{Surj}(m,n) = 0 \text{ при } m < n
\end{aligned}$$

(проверьте!).

Это соотношение также можно использовать для подсчёта сюръекций. Какой из способов лучше, сразу не очевидно. Но можно попробовать сравнить эти способы вычислений экспериментально. Попробуйте написать программы для обоих способов и сравнить время их работы при достаточно больших m и n (чтобы время счёта занимало десятки секунд — иначе разница во времени будет неубедительна).

Заметим, что подсчитанное нами число сюръекций делится на $n!$, поскольку сюръекцию можно строить в два шага: сначала разбить m элементов на n непустых групп, решив отнести в одну группу те элементы, у которых одинаковый образ, а затем решить, как соотносятся эти группы с элементами множества B (выбрав одну

из $n!$ биекций). Количество разбиений m элементов на n непустых групп (что то же самое, количество отношений эквивалентности на m-элементном множестве, имеющих n классов эквивалентности) называется иногда *числом Стирлинга второго рода* — в честь того же Стирлинга, что и приближённая формула $n! \approx \sqrt{2\pi n}(n/e)^n$. Оно обозначается $S(m,n)$ или $\left\{ {m \atop n} \right\}$ и вычисляется по той же формуле, только надо поделить на $n!$ в конце.

6.7 Задачи для самостоятельного решения

33. Функция f определена на множестве $A \cup B$ и принимает значения в множестве Y. Какой знак сравнения можно поставить вместо ?, чтобы утверждение

$$f(A \setminus B) \mathbin{?} f(A) \setminus f(B)$$

стало верным?

34. Функция f определена на множестве X и принимает значения в множестве Y, при этом $A \cup B \subseteq Y$. Какой знак сравнения можно поставить вместо ?, чтобы утверждение

$$f^{-1}(A \setminus B) \mathbin{?} f^{-1}(A) \setminus f^{-1}(B)$$

стало верным?

35. Верно ли для любых f, X, Y что $f(X) \cap Y = \varnothing$ равносильно $X \cap f^{-1}(Y) = \varnothing$?

36. При каких условиях на действительные числа a, b, c функция $f\colon \mathbb{R} \to \mathbb{R}$, задаваемая равенством $f(x) = x^3 + ax^2 + bx + c$, является инъекцией, при каких сюръекцией и при каких биекцией?

37. Докажите, что

a) композиция инъекций является инъекцией;

b) композиция сюръекций является сюръекцией.

38. Докажите, что $f^{-1}(g^{-1}(Z)) = (g \circ f)^{-1}(Z)$ для любого $Z \subseteq C$ и любых функций f из множества A в множество B и g из множества B в множество C. Здесь f^{-1} обозначает полный прообраз множества.

39. Какое преобразование является композицией двух центральных симметрий на плоскости?

40. Какое преобразование является композицией двух поворотов вокруг разных точек, если повороты на один и тот же угол, но в разные стороны (один по часовой стрелке, другой против)?

41. Докажите, что любое движение плоскости (преобразование, сохраняющее расстояния) является композицией двух или трёх осевых симметрий.

42. Дробно-линейной функцией называют функцию вида

$$f(x) = \frac{ax + b}{cx + d}\,,$$

где a, b, c, d — некоторые вещественные числа (константы). Покажите, что композиция двух дробно-линейных функций почти всегда дробно-линейна (и объясните, в каком смысле «почти»).

Замечание: Знающие линейную алгебру могут заметить, что коэффициенты композиции дробно-линейных функций вычисляются так же, как при умножении матриц 2×2 (и объяснить, почему).

43. Функция f на двоичных словах длины n задана следующим правилом: в слове x найдём самый правый 0 и заменим его на 1, а все 1, которые стоят правее этого нуля, заменим на нули. Получим слово $f(x)$. Это правило неприменимо к $x = 1^n$, в этом случае определим значение функции отдельно: $f(1^n) = 0^n$.

a) Проверьте, что f биекция. **b)** Найдите f^{-1} (задайте правило, по которому из слова x получается слово $f^{-1}(x)$).

44. Определение обратной функции включает два требования. Иногда их разделяют и говорят, что $g\colon B \to A$ является *левой обратной* (соответственно *правой обратной*) к f, если $g \circ f = \mathrm{id}_A$ (соответственно $f \circ g = \mathrm{id}_B$).

a) Приведите примеры, когда левая обратная не является правой обратной и когда правая обратная не является левой.

b) Может ли такое случиться для конечных множеств?

c) Может ли быть так, что у одной функции есть и левая, и правая обратные, но они различны?

d) Для каких функций существует левая обратная?

e) Для каких функций существует правая обратная?

45. Рассмотрим функцию $f\colon [0,1] \to [0,1]$, заданную формулой $f(x) = 4x(1-x)$. (Коэффициент подобран так, чтобы областью значений был как раз отрезок $[0,1]$.) Сколько решений имеют уравнения $f(x) = x$? $f(f(x)) = x$? $f^n(x) = x$? (В последнем случае мы имеем в виду композицию n копий функции f, а не возведение числа в степень n.)

46. Покажите, что порядок любой перестановки n-элементного множества делит нацело число $n!$.

47. Циклом $(a_1 a_2 \ldots a_n)$ называется перестановка (биекция на конечном множестве), при котором элемент a_1 переходит в a_2, в свою очередь a_2 переходит в a_3 и так далее до a_n, которое переходит в a_1. Остальные элементы множества переходят сами в себя (остаются на месте, как ещё говорят). (При $n = 2$ получается обмен двух элементов, или, как говорят, *транспозиция*; при $n = 1$ можно считать, что получится тождественная перестановка.) Покажите, что всякую перестановку можно представить как композицию нескольких непересекающихся циклов (более точно: не пересекаются множества тех элементов, которые циклы не оставляют на месте).

48. Покажите, что каждую перестановку можно представить в виде произведения некоторого числа транспозиций, причём чётность этого числа будет одинаковой во всех представлениях. (Скажем, любое разложение тождественной перестановки в произведение транспозиций содержит чётное число транспозиций.)

49. Найдите количество **a)** неубывающих функций $f\colon \{1,2,\ldots,n\} \to \{1,2,\ldots,m\}$; **b)** неубывающих инъекций $f\colon \{1,2,\ldots,n\} \to \{1,2,\ldots,m\}$; **c)** неубывающих сюръекций $f\colon \{1,2,\ldots,n\} \to \{1,2,\ldots,m\}$.

Функция неубывающая, если $x \leqslant y$ влечёт $f(x) \leqslant f(y)$.

Лекция 7

Отношения и их графы

Помимо множеств и функций математический формализм включает третий необходимый элемент: отношения. В этой лекции мы введём необходимые определения и обсудим основные свойства отношений.

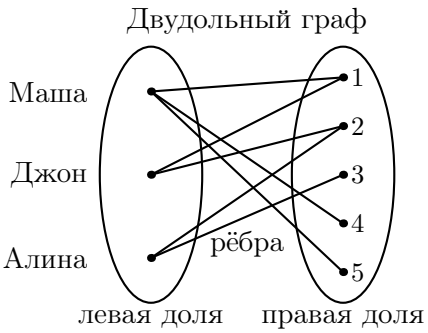
7.1 Отношения в естественном языке

В естественном языке множествам соответствуют свойства. Часто эти свойства выражаются прилагательными. Например, прилагательное «красный» соответствует множеству объектов, которые мы считаем красными. Лингвисты говорят о «предикатах» как свойствах, которыми может обладать (или не обладать) субъект — математики говорят о соответствующих множествах, которые могут содержать (или не содержать) этот самый субъект (он может быть их элементом или не быть). Как мы видели, этот перевод может быть продолжен — пересечению множеств соответствует союз «и», объединению — союз «или» (понимаемый в неисключительном смысле).

Но сейчас нас интересует другая конструкция языка и её формализация: когда после контрольной говорят, что «Маша решила задачу 4», это означает, что Маша и задача 4 находятся в некотором *отношении*, в котором они могут находиться или не находиться. Математики видят в этой фразе два множества G (все студенты группы, где была контрольная) и P (все задачи, предлагавшиеся на контрольной), и — как они говорят — «бинарное отношение» между множествами, которое указывает, кто чего решил. Это отношение можно представлять себе разными способами. Можно в списке группы у каждой фамилии написать номера решённых задач. Можно (хотя и менее привычно) у каждой задачи написать студентов, которые её решили. Можно составить таблицу наподобие такой:

	1	2	3	4	5
Маша	+		+	+	+
Джон	+	+			
Алина		+	+		

А можно нарисовать двудольный граф: в левой доле вершинами изобразить студентов, в правой доле вершинами считать задачи, и провести рёбра, соответствующие успеху при решении задач:



Контрольный вопрос 7.1. На этой картинке есть ошибка — точнее, есть несоответствие между картинкой и таблицей с плюсами выше. Найдите его. Как надо исправить таблицу или картинку, чтобы его устранить?

Во всех случаях мы представляем (разными способами) одну и ту же информацию (считаем, что несоответствие, о котором шла речь в задаче, исправлено). Различие в способах представления становится важным, если нас интересует удобство хранения и обработки этой информации — соответствующий раздел в программировании изучает различные «структуры данных» для представления множеств, отношений и других абстрактных объектов.

7.2 Отношения с точки зрения математики

С точки зрения математики, наше отношение задаётся (как мы уже говорили) множествами G, P и некоторым подмножеством декартова произведения $G \times P$ множеств G и P. В него входят следующие упорядоченные пары:

$$\begin{gathered}(\text{Маша}, 1), (\text{Маша}, 2), (\text{Маша}, 3), (\text{Маша}, 4), (\text{Маша}, 5),\\ (\text{Джон}, 1), (\text{Джон}, 2), (\text{Джон}, 3), (\text{Джон}, 4), (\text{Джон}, 5),\\ (\text{Алина}, 1), (\text{Алина}, 2), (\text{Алина}, 3), (\text{Алина}, 4), (\text{Алина}, 5).\end{gathered}$$

Мы перечислили их в этом порядке, но порядок не играет роли — множество $G \times P$ состоит из этих 15 элементов. Если вспомнить нашу таблицу, то элементы множества $G \times P$ соответствуют клеточкам таблицы: в каждой клеточке понятно, о каком студенте и о какой задаче идёт речь. Отношение, задаваемое нашей таблицей, состоит из 8 элементов, соответствующих плюсам в таблице:

$(\text{Маша}, 1), (\text{Маша}, 3), (\text{Маша}, 4), (\text{Маша}, 5), (\text{Джон}, 1), (\text{Джон}, 2), (\text{Алина}, 2), (\text{Алина}, 3).$

Определение 7.1. *Бинарным отношением на множествах A и B* называется любое подмножество R декартова произведения $A \times B$.

Мы говорим, что элементы $a \in A, b \in B$ находятся в отношении R, если $(x, y) \in R$. Есть несколько вариантов обозначения для этого: пишут, например aRb или $R(a, b)$. Скажем, в нашем примере можно ввести имя *Решил* для рассмотренного отношения, и сказать, что *Решил*(Маша, 3) истинно, а *Решил*(Джон, 4) ложно.

Слово «бинарное» подчёркивает, что речь идёт об отношении между элементами двух множеств. Наряду с бинарными отношениями мы можем рассмотреть отношения и любой другой «валентности», или «арности». Назовём *k-арным отношением на множествах* $A_1, \ldots, A_k$ любое подмножество множества $A_1 \times A_2 \times \ldots \times A_k$. При $k = 1$ говорят об *унарных* отношениях. В главе 5 мы обсуждали, что множество можно задать, задав свойство его элементов — это свойство и есть унарное отношение. Но чтобы описать свойство формально, нужно описать все элементы множества A_1, обладающие этим свойством. Таким образом, унарные отношения — это просто подмножества множества A_1.

При $k = 3$ получаются *тернарные* отношения. Одно и то же слово русского языка может использоваться для обозначения отношений разной валентности. Скажем, «Маша бьёт Джона» сообщает об элементе бинарного отношения, а в более подробном сообщении «Маша бьёт Джона веником» то же самое слово «бьёт» описывает тернарное отношении (как сказали бы лингвисты, «имеет три актанта»). Последнее сообщение можно было бы записать как *Бьёт*(Маша, Джон, веник).

Если множества A и B, на которых определено бинарное отношение, совпадают, то говорят просто о бинарном отношении на множестве A. Аналогично определяются k-арные отношения. Например, можно рассмотреть тернарное отношение «быть суммой» на множестве целых чисел: оно состоит из троек (a, b, c), для которых $a = b + c$.

В современных информационных системах используют так называемые «реляционные базы данных» — наверное, многие слышали о языке запросов SQL, который используют для обращения к таким базам. Слово «реляционные» (relational) происходит от английского слова “relation”, означающего «отношение». В реляционной базе данных информация (о мире) хранится в виде отношений: скажем, понятие «место жительства» рассматривается как отношение между множеством людей и множеством жилых помещений, и т. д. Это оказалось достаточно удобным и универсальным подходом к представлению различной информации.

Контрольный вопрос 7.2. Представим себе, что проверяющие контрольную ставят за каждую задачу один из знаков $0, -, \mp, \pm, +$. Как можно представить результаты контрольной в виде тернарного отношения?

Важнейший пример бинарных отношений, который мы подробно обсуждали в предыдущей главе — это функции. Вспомните, что функции мы отождествляем с их графиками, а график функции из множества A в множество B по определению является подмножеством декартова произведения $A \times B$.

Графики функций обладают особым свойством: при $y_1 \neq y_2$ графику функции может принадлежать лишь одна из пар (x, y_1), (x, y_2). Допуская вольность речи, можно сказать, что бинарные отношения — это «многозначные» функции.

7.3 Свойства бинарных отношений

Мы будем в первую очередь говорить о бинарных отношениях. Унарные отношения — это просто подмножества множеств, мы их уже обсуждали, а бинарные отношения сложнее, в них уже есть какая-то внутренняя структура, и тут есть о чём поговорить.

Бинарное отношение R на множестве X называют *симметричным*, если из $R(x, y)$ следует $R(y, x)$. Многие слова русского языка такую симметрию подразумевают: скажем, отношение «человек x — родственник человека y» на множестве всех людей обычно считают симметричным (хотя, конечно, может случиться и так, что А считает Б своим родственником, а Б и знать А не желает). Примерно то же самое можно сказать об отношении «x знаком с y», или отношении «x и y — соседи». Другие отношения, скажем «x посылал y письмо» или «x — мать y», не симметричны. В последнем примере $R(x, y)$ вообще несовместно с $R(y, x)$: из двух утверждений «x — мать y» и «y — мать x» хотя бы одно ложно.

Задача 7.3. Докажите, что если отношение знакомства между людьми симметрично, то количество людей, имеющих нечётное число знакомых, чётно.

Указание. Вспомните про соотношение между степенями вершин графа. □

Задача 7.4. Является ли отношение «x — брат y» (в самом простом, генеалогическом смысле) симметричным?

Указание. У Маши может быть брат Вася. □

Другое свойство бинарных отношений на некотором множестве: отношение R *транзитивно*, если из $R(x, y)$ и $R(y, z)$ следует $R(x, z)$. Например, отношение «быть предком» на множестве всех людей транзитивно: если А — предок Б, а Б — предок В, то А — предок В. А отношение «быть отцом» не транзитивно. Классическая фраза «вассал моего вассала не мой вассал» (не будем вдаваться в уточнение того, что это значит и в каких странах и когда так было) теперь может быть сформулирована научно: «отношение вассалитета не обязано быть транзитивным».

Контрольный вопрос 7.5. Будет ли отношение «x знаком с y» на множестве людей транзитивным?

Много транзитивных отношений возникает в процессе сравнений: естественно считать, что отношения «x выше y», «x сильнее y», «x красивее y» и т. п. подразумевают транзитивность — её отсутствие воспринимается как парадокс (как в игре камень–ножницы–бумага, когда камень сильнее ножниц, ножницы сильнее бумаги, а камень не сильнее бумаги).

Задача 7.6. Является ли отношение «быть делителем» на множестве положительных целых чисел симметричным? транзитивным? Тот же вопрос для отношения «быть взаимно простым» (не иметь общих делителей, кроме 1) на том же множестве.

Решение. Отношение «быть делителем» несимметрично, но транзитивно. Число 2 является делителем числа 4, а обратное неверно. С другой стороны, если a является делителем b, то $b = qa$ для некоторого целого числа q; если b является делителем c, то $c = rb$ для некоторого целого числа r. Значит, $c = rb = (rq)a$ и a является делителем c.

Отношение «быть взаимно простым» симметрично, но нетранзитивно. Нетранзитивность легко увидеть на примере: 2 взаимно просто с 3, 3 взаимно просто с 4, но 2 и 4 не взаимно просты: у них есть общий делитель 2.

Симметричность этого отношения вытекает прямо из определения: общий делитель есть у (неупорядоченной) пары чисел. □

Будет ли отношение «x — родственник y» транзитивным? Хочется сказать, что да и что родственник твоего родственника — твой родственник. С другой стороны, есть опасность таким образом распространить родство на всех ныне живущих людей (даже если и не считать, что все они потомки Адама — и без этого родственных связей довольно много). Это — типичная ситуация «парадокса кучи» — добавление одной песчинки не может превратить в кучу то, что раньше кучей не было, но тем не менее кучи бывают.

Третье свойство бинарного отношения, *рефлексивность*, означает, что $R(x, x)$ для любого x. (Слово «рефлексия» означает размышления о самом себе.) Интересно, что наша языковая интуиция не всегда отчётлива в этом месте: будет ли, например, отношение «x знаком с y» рефлексивным? Знаком ли человек с самим собой? Это уже вопрос с философским оттенком: совет «Познай самого себя» ничего не говорит о том, была ли эта попытка познания успешной. Скорее, видимо, нет — когда человек говорит, что у него есть трое знакомых, вряд ли он включает в них себя. Является ли человек своим собственным однофамильцем? своим собственным родственником? своим собственным соседом?

В математике, конечно, такая неопределённость недопустима, и говоря о бинарном отношении, мы должны заранее решить, что мы думаем по поводу отношения объекта с самим собой. Скажем, отношение $x \leqslant y$ будет рефлексивным (поскольку $x \leqslant x$ для любого x), а отношение $x < y$ не будет.

Более того, $x < x$ ложно для всех чисел x. Если $(x, x) \notin R$ для всех x, такое отношение называется *антирефлексивным*.

Заметим, что если мы хотим считать отношение «быть родственником» симметричным и транзитивным, то придётся считать его рефлексивным: если А родственник Б, то по симметрии Б будет родственником А, и по транзитивности (ведь мы же не говорили, что все три элемента разные!) получаем, что А будет родственником самому себе.

Повторим, что все эти свойства (транзитивность, рефлексивность, антирефлексивность, симметричность) не имеют смысла для бинарных отношений между элементами A и B при $A \neq B$.

7.4 Графы, матрицы и бинарные отношения

Мы уже вводили понятие простого неориентированного графа. Это множество вершин V и некоторое подмножество E неупорядоченных пар его вершин (рёбер). Петли не допускаются, поэтому пары состоят из разных вершин.

Если посмотреть на графы с точки зрения отношений, то простой неориентированный граф — это симметричное антирефлексивное отношение на множестве вершин. Словами это отношение можно обозначить как «вершина u связана ребром с вершиной v».

Ориентированные графы (без параллельных рёбер[1]) — это произвольные бинарные отношения на множестве вершин. (Мы рассматривали только конечные графы, но это необязательно.)

Мы уже видели на примере в разделе 7.1, что бинарному отношению можно сопоставить простой неориентированный граф, причём двудольный. Опишем эту конструкцию для любого бинарного отношения $R \subseteq A \times B$ на множествах A и B.

Вершинами графа, отвечающего отношению R, будут элементы множеств A и B. Рёбра соединяют те пары вершин, которые находятся в отношении R. Формально это можно записать так:

$$E = \{\{a, b\} : (a, b) \in R\}.$$

Второй способ записи бинарных отношений на конечных множествах, который использовался в примере раздела 7.1,— в виде таблиц — имеет в математике специальное название *матрицы*. Обычно строки и столбцы матриц нумеруются положительными целыми числами, но это необязательно. Вместо чисел можно использовать любые множества. Ведь эти числа нужны только для того, чтобы можно было указать на *элемент матрицы* M_{ij}.

Матричные элементы матриц, отвечающих отношениям, принимают булевы значения 1 или 0. Задание отношения в виде матрицы обобщает индикаторные функции. На индикаторную функцию можно смотреть как на матрицу с одной строкой.

7.5 Отношения эквивалентности

Отношение на некотором множестве, которое одновременно рефлексивно, симметрично и транзитивно, называют *отношением эквивалентности*. Например, отношение «быть однофамильцем» (иметь одинаковую фамилию; мы сейчас считаем каждого человека своим собственным однофамильцем) или «быть одноклассником» (с той же оговоркой).

Вообще, если множество X разбито на непересекающиеся подмножества, то отношение «попасть в одно подмножество» будет отношением эквивалентности. Немного другой пример: пусть для каждого элемента множества X определена какая-то характеристика, скажем, «цвет» из какого-то заранее выбранного множества цветов. Тогда отношение «быть одного цвета» является отношением эквивалентности.

[1] Напомним, что для таких графов есть не более одного ребра с данными началом и концом; мы только такие графы и рассматривали.

Оказывается, что это общая ситуация:

Теорема 7.2. *Любое отношение R, являющееся отношением эквивалентности на множестве A, делит A на классы эквивалентности — непересекающиеся подмножества множества X, при этом любые два элемента одного класса находятся в отношении R, а любые два элемента разных классов не находятся в отношении R.*

Доказательство. Пожалуй, тут сложнее понять, что тут вообще надо доказывать (и почему это не очевидно), чем доказать — но в качестве образца проведём подробно формальное рассуждение.

Для каждого $x \in A$ рассмотрим множество тех y, для которых верно $R(x, y)$. Обозначим его через $[x]$. Его можно было бы назвать «классом эквивалентности элемента x» — собственно говоря, так его и называют, но само по себе это название не гарантирует разбиения на классы, это ещё надо доказывать. А именно, надо доказать, что

- объединение всех множеств вида $[x]$ совпадает с множеством A;
- два множества $[x]$ и $[y]$ либо не пересекаются, либо совпадают;
- наконец, надо ещё доказать, что $[x] = [y]$ в том и только том случае, когда $R(x, y)$, то есть R совпадает с отношением «принадлежать одному классу».

Как это доказать?

(1) В силу рефлексивности множество $[x]$ содержит x в качестве своего элемента: $x \in [x]$, поскольку $R(x, x)$. Отсюда следует, что объединение всех этих множеств совпадает с A. (Выйти за пределы A они не могут, так как мы рассматриваем отношение на множестве A и элементы множества A.)

(2) Пусть для двух элементов $x, y \in A$ их классы $[x]$ и $[y]$ пересеклись. Это означает, что есть такой $z \in A$, что $R(x, z)$ и $R(y, z)$. Симметричность даёт $R(z, y)$, после чего мы применяем транзитивность к $R(x, z)$ и $R(z, y)$ и заключаем, что $R(x, y)$. Выведем отсюда, что $[x] = [y]$. В самом деле, если произвольный элемент t принадлежит $[y]$, то $R(y, t)$. Вспоминая, что $R(x, y)$и применяя транзитивность, получаем $R(x, t)$, то есть $t \in [x]$. Мы доказали, таким образом, что $[y] \subseteq [x]$. Аналогично доказывается, что $[x] \subseteq [y]$, так что $[x] = [y]$.

(3) Если для каких-то x, y верно $R(x, y)$, то x и y оба лежат в одном классе, а именно, в $[x]$. Обратно, если x и y лежат в каком-то $[z]$, то по определению имеем $R(z, x)$ и $R(z, y)$, симметричность даёт $R(x, z)$ и после этого транзитивность даёт $R(x, y)$. □

Замечание 7.1. Интересные вещи, связанные с этой теоремой, рассказывают биологи. Попытаемся разделить живые существа на виды, объединяя в один вид тех, которые скрещиваются. (Конечно, речь идёт о популяциях, а не об отдельных особях.) Возможность такого деления предполагает, что это отношение транзитивно. Однако так бывает не всегда. Рассказывают (см., например, статью про Ring species в википедии; русский вариант называется «Кольцевые виды»), что когда птицы живут

вокруг некоторого препятствия (скажем, пустыни), может оказаться так, что они образуют незамкнутое кольцо: всюду, кроме точки разрыва, соседи скрещиваются, но встречающиеся с разных сторон точки разрыва птицы уже не скрещиваются, хотя живут рядом. Русская статья пишет, что «такие примеры, обладающие промежуточными характеристиками между видом и более низкими таксономическими рангами, противоречат классическому представлению о дискретности видов», а английская статья не стесняется назвать вещи своими именами: «the issue is that interfertility (ability to interbreed) is not a transitive relation — if A can breed with B, and B can breed with C, it does not follow that A can breed with C — and thus does not define an equivalence relation. A ring species is a species that exhibits a counterexample to transitivity» (R. Brown, "Same species" vs. "Interfertile": concise wording can avoid confusion when discussing evolution).

В математике отношения эквивалентности используются очень часто и уже использовались нами выше. Так, при изучении графов мы исследовали отношение связанности на вершинах неориентированного графа. Оно является отношением эквивалентности, а классы эквивалентности этого отношения — это компоненты связности. Аналогично, для ориентированных графов отношение достижимости в обе стороны также является отношением эквивалентности, а его классы — компоненты сильной связности. Теоремы 3.2 и 3.2 являются следствием теоремы 7.2 — чтобы её применить, достаточно доказать, что упомянутые отношения действительно отношения эквивалентности. При изучении арифметики остатков мы имели дело с отношением эквивалентности «a имеет тот же остаток, что и b при делении на N». Классы эквивалентности этого отношения называются *вычетами по модулю* N. В лекции 4 мы фактически определили арифметические операции на множестве вычетов — построили на вычетах арифметику остатков.

Приведём и другие примеры. Формальное определение рациональных чисел схоже с построением арифметики остатков (на вычетах). Допустим, мы уже определили целые числа. Рассмотрим дроби, то есть упорядоченные пары (m, n), где m и n — целые числа и $n > 0$. Первый элемент пары будем называть числителем, второй — знаменателем, и записывать пару для наглядности как $\frac{m}{n}$ (не придавая пока разделяющей черте никакого смысла — просто решили так записывать пару, и всё). Определим отношение на множестве пар, считая, что $\frac{m}{n}$ находится в этом отношении с $\frac{u}{v}$, если произведения mv и nu равны. (Хочется сказать: если частные m/n и u/v равны, но мы пока не определили рациональных чисел и частного.) Обозначение для этого отношения: $\frac{m}{n} \sim \frac{u}{v}$.

Проверим, что это отношение эквивалентности. В самом деле, $\frac{m}{n} \sim \frac{m}{n}$ (рефлексивность), потому что $mn = nm$. Симметричность тоже сразу следует из определения. А вот транзитивность немного сложнее: если $\frac{m}{n} \sim \frac{u}{v}$ и $\frac{u}{v} \sim \frac{k}{l}$, то по определению $mv = nu$ и $ul = vk$. Домножая равенства на l и n и применяя транзитивность для обычного равенства чисел, получаем $mvl = nul = nvk$. Теперь надо воспользоваться таким свойством целых чисел: на ненулевое число (в нашем случае v) можно сокращать. Получится $ml = nk$, то есть $\frac{m}{n} \sim \frac{k}{l}$. Транзитивность проверена.

Имея отношение эквивалентности, можно рассмотреть соответствующие классы

эквивалентности и назвать их рациональными числами. Класс дроби $\frac{m}{n}$ после этого можно назвать рациональным числом, получающимся при делении m на n. Например при делении 2 на 3 и при делении 4 на 6 получается одно и то же рациональное число, множество всех пар $\frac{m}{n}$, где $2n = 3m$ (или $4n = 6m$, это то же самое).

После этого надо аккуратно определять арифметические операции над рациональными числами и доказывать их свойства. Скажем, чтобы сложить два рациональных числа, нужно выбрать их представители, скажем, $\frac{m}{n}$ и $\frac{u}{v}$, и рассмотреть рациональное число (класс эквивалентности), содержащий $\frac{mv+nu}{nv}$.

Задача 7.7. Проверьте, что это определение корректно, то есть что получившийся класс (сумма) не зависит от того, какие именно представители выбраны в двух классах-слагаемых.

Решение. Нужно проверить следующее утверждение:

$$\text{если } \frac{m}{n} \sim \frac{m'}{n'} \text{ и } \frac{u}{v} \sim \frac{u'}{v'}, \text{ то } \frac{mv+nu}{nv} \sim \frac{m'v'+n'u'}{n'v'} \ .$$

Если раскрыть определение эквивалентности в этом утверждении, получим такое (равносильное исходному) утверждение о целых числах:

$$\text{если } mn' = m'n \text{ и } uv' = u'v, \text{ то } (n'v')(mv+nu) = (nv)(m'v'+n'u') \ .$$

Выглядит сложно, а проверяется в одну строчку:

$$(n'v')(mv+nu) = n'mv'v + uv'n'n = m'nv'v + u'vn'n = (nv)(m'v'+n'u')$$

(в преобразованиях мы использовали равенства, которые входят в посылку утверждения). □

7.6 Композиция отношений

Все знают, что тёща — это мать жены, но не все знают, как это научно сформулировать. С точки зрения математика, тут есть два отношения

$$W(x,y) = \text{«}y \text{ — жена } x\text{»}$$

и

$$M(y,z) = \text{«}z \text{ — мать } y\text{»}$$

и мы применяем к ним операцию, называемую *композицией*.

Формальное определение композиции выглядит так. Пусть даны два отношения $R \subset A \times B$ и $S \subset B \times C$. Их *композицией* называется отношение $S \circ R \subseteq A \times C$, определяемое так:

$$(x,z) \in S \circ R \Leftrightarrow \text{существует такой } y \in B, \text{ что } (x,y) \in R \text{ и } (y,z) \in S.$$

В нашем примере A, B, C — это одно и то же множество, а именно, множество всех людей, $M \circ W(x, z)$ как раз и означает, что z — тёща x: определение композиции в данном случае читается как

$$\text{существует такой } y, \text{ что } W(x, y) \text{ и } M(y, z),$$

то есть что найдётся такой человек y, что y — жена человека x и z — мать человека y. Заметим ещё, что это определение вполне имеет смысл и в странах, где разрешено иметь много жён — тогда у одного человека может быть и много тёщ.

Еще один знакомый многим пример возникает в социальных сетях, где есть понятие «друзья друзей». Это ни что иное, как композиция с самим собой отношения «быть другом в данной социальной сети».

Обратите внимание, что композиция двух отношений может быть определена только в том случае, когда у них общее множество B. Если вас спросят на экзамене, чему равна композиция отношения $R(x, y)$, означающего, что человек y — отец человека x, и отношения $S(y, z)$, означающего, что число z на единицу больше числа y, то это скорее всего «проверка на вшивость» — если экзаменуемый начинает что-то вычислять, то на этом экзамен можно заканчивать. (Хороший студент в этот момент смотрит на экзаменатора как на идиота, и удовлетворённый экзаменатор переходит к следующему вопросу.)

Можно рассмотреть и более научный пример. Пусть A, B, C равны множеству действительных чисел, $R(x, y)$ означает, что $y = x^2$, а $S(y, z)$ означает, что $z = y + 1$. Тогда $S \circ R(x, z)$ означает, что найдётся y, для которого $y = x^2$ и $z = y + 1$, то есть что $z = x^2 + 1$.

Заметим, что для обозначения переменных можно использовать любые буквы: можно было бы сказать, например, что $R(y, z)$ означает, что $z = y^2$, а $S(x, y)$ означает, что $y = x + 1$. Это бы означало ровно то же самое: что R есть множество всех пар, в которых второе число равно квадрату первого, а S есть множество всех пар, в которых второе число на единицу больше первого. Но в таких обозначениях проще вычислить композицию в другом порядке $R \circ S$: она состоит из пар (x, z), для которых

$$\exists y (y = x + 1) \wedge (z = y^2).$$

Мы использовали здесь стандартное обозначение $\wedge$ для «и», а также «квантор существования» $\exists y$ вместо слов «существует y». Так или иначе, это означает, что $z = (x + 1)^2$. Видно, что композиция может зависеть от порядка, даже если и в том, и в другом порядке она имеет смысл. Что, впрочем, и по жизни ясно: отец брата — совсем не то же, что брат отца.

Контрольный вопрос 7.8. Проверьте, что композиция графиков функций как отношений совпадает с графиком композиции этих функций.

Задача 7.9. Закончить предложение: отношение R на множестве A транзитивно тогда и только тогда, когда композиция $R \circ R \ldots$

Решение. Ответ: ... является подмножеством отношения R.

Транзитивность R означает, что из $(x, y) \in R$ и $(y, z) \in R$ следует $(x, z) \in R$. Если пара (a, b) принадлежит $R \circ R$, это означает по определению композиции, что для некоторого $y \in A$ обе пары (a, y) и (y, b) принадлежат R. По транзитивности получаем, что $(a, b) \in R$. То есть, любой элемент $R \circ R$ является элементом R.

Пусть $R \circ R \subseteq R$. Возьмём две пары (x, y) и (y, z) из отношения R. По определению композиции $(x, z) \in R \circ R$, а поскольку квадрат отношения лежит в нём самом, то $(x, z) \in R$. Это и означает транзитивность. □

Задача 7.10. Рассмотрим на множестве действительных чисел $\mathbb{R}$ бинарное отношение $R(x, y)$, означающее, что $xy > 0$. Чему равно $R \circ R$?

Решение. Нужно описать все такие пары (x, y), что для некоторого числа z выполняются два строгих неравенства

$$xz > 0, \quad zy > 0.$$

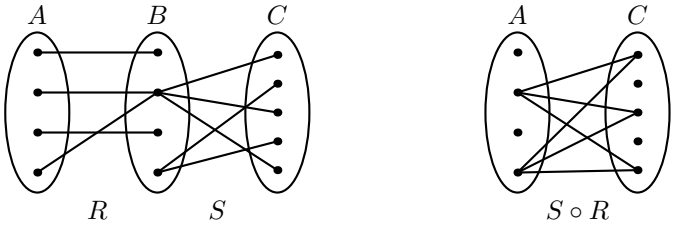
Ясно, что числа не равны 0. Поэтому из этих двух неравенств следует, что $x/y > 0$, откуда $xy = (x/y) \cdot y^2 > 0$.

С другой стороны, если $xy > 0$, то либо $x \cdot 1 > 0$ и $y \cdot 1 > 0$, либо $x \cdot (-1) > 0$ и $y \cdot (-1) > 0$. Значит, $(x, y) \in R \circ R$.

Ответ: $R \circ R = R$. □

По существу та же операция композиции встречается в реляционных базах данных под названием операции JOIN, применяемой к двум отношениям. Разница в обозначениях: там, говоря об отношениях, говорят не о первом или втором члене пары, а дают этим членам имена. Скажем, можно рассматривать отношение ПРОЖИВАТЬ между множеством людей и множеством помещений, и говорить, что оно имеет атрибуты ЖИЛЕЦ и ЖИЛЬЁ, а также отношение НАХОДИТЬСЯ между множеством помещений и множеством улиц, имеющее атрибуты ЖИЛЬЁ и УЛИЦА. Применяя операцию JOIN по атрибуту ЖИЛЬЁ, мы получаем отношение ПРОЖИВАТЬ-НА-УЛИЦЕ с атрибутами ЖИЛЕЦ и УЛИЦА (между множеством людей и множеством улиц). Ясно, что это та же самая композиция отношений, но в других обозначениях.

Можно описать операцию композиции и в терминах графов. Представим два отношения $R \subset A \times B$ и $S \subset B \times C$ в виде двудольных графов с общей долей B (правой долей для R и левой долей для S). Тогда их композиция будет двудольным графом с левой долей A и правой долей C: ребро a–c в таком графе означает, что найдётся b, для которого a соединено с b ребром, и b соединено с c ребром, то есть что из a в c можно пройти за два шага:



Для знакомых с линейной алгеброй композицию отношений можно описать ещё одним способом. Будем изображать отношение матрицей, как мы делали, только вместо плюсов будем писать положительные числа, а вместо пустых клеток — нули. Тогда композиция отношений будет изображаться произведением матриц. Понятно, почему?

Композиция отношений, как и функций, обладает свойством *ассоциативности*

$$(R \circ S) \circ T = R \circ (S \circ T).$$

Здесь мы предполагаем, что множества, для которых определены отношения, выбраны согласованно, то есть $R \subset A \times B$, $S \subset B \times C$ и $T \subset C \times D$ (иначе композиции нельзя определить).

Это проще понять, чем объяснить: обе части равенства задают отношение M, для которого $M(a,d)$ равносильно тому, что найдутся такие $b \in B$ и $c \in C$, что одновременно $R(a,b)$, $S(b,c)$ и $T(c,d)$:

$$M(a,d) \Leftrightarrow (\exists\, b \in B)(\exists\, c \in C)[R(a,b) \wedge S(b,c) \wedge T(c,d)].$$

(Композиция заменяет два шага по рёбрам графов на один; теперь у нас три графа, и всё равно, в каком порядке их склеивать.)

7.7 Отношения: что дальше?

В этой лекции, как и в нескольких предыдущих, почти не было интересных теорем, речь больше шла об определениях, соглашениях, обозначениях и т. д. Цель наша в том, чтобы привыкнуть к некоторому языку (множества, отношения, функции, композиция, ...) — и для этого поговорить на этом языке о чём-то сравнительно простом и знакомом. (Никто не ждёт откровений от диалогов в учебнике английского языка.)

Этот язык будет нам постоянно встречаться — будем ли мы говорить о логических операциях и кванторах, или о графах, или об упорядоченных множествах, или о вычислимых функциях и перечислимых свойствах. Да и не только нам — пытаясь уточнить смысл понятия падежа, Колмогоров и Зализняк предлагали считать падежом класс эквивалентности некоторого отношения.[2]. Умение увидеть какие-то из рассмотренных нами структур часто бывает полезно, даже если на первый взгляд ничего подобного не скажешь. Например, есть такая задача, которую может понять даже семиклассник: в десятизначном числе, составленном из разных цифр, можно вычеркнуть шесть цифр, чтобы оставшиеся четыре шли в возрастающем или в убывающем порядке. Однако (по-видимому, самое простое) её решение состоит в том, что на множестве цифр рассматривается подходящее бинарное отношение порядка («стоять левее и одновременно быть меньше» — мы разберём этот пример в лекции 9 об упорядоченных множествах).

[2]И при этом в русском языке обнаруживается больше падежей, чем традиционные шесть — именительный, родительный, дательный, винительный, творительный и предложный. Скажем, в обороте «в лесу» этот самый «лес» считают стоящим в «местном» падеже. См. подробности в http://www.kolmogorov.info/uspensky-k_opredeleniyu_padezha_po_kolmogorovu.html

7.8 Задачи для самостоятельного решения

11. Будем говорить, что одна прямоугольная коробка «меньше» другой, если одно из трёх измерений первой коробки меньше одного из трёх измерений второй (если их можно поставить на пол так, чтобы первая коробка была ниже второй). Будет ли это отношение транзитивным?

12. В ходе турнира каждая команда сыграла с каждой по одному разу, причём ничьих не было и каждая команда хоть кому-то да проиграла. Докажите, что найдутся три команды А, Б, В, нарушившие транзитивность: А выиграла у Б, Б выиграла у В, а В выиграла у А.

13. Педанты заметят, что в предыдущей задаче мы неявно предполагаем, что в турнире участвовало конечное число команд. Почему это предположение существенно?

14. На плоскости нарисовано некоторое количество равносторонних треугольников. Они не пересекаются, но могут иметь общие участки сторон. Мы хотим покрасить каждый треугольник в какой-нибудь цвет так, чтобы те из них, которые соприкасаются, были покрашены в разные цвета (треугольники, имеющие одну общую точку, могут быть покрашены в один цвет). Хватит ли для такой раскраски двух цветов?

15. a) Рассмотрим на множестве $\mathbb{R}$ бинарное отношение $R(x, y)$, означающее, что $y = x + 1$. Чему равно $R \circ R$?

b) Тот же вопрос, если $R(x, y)$ означает $x + y = 1$.

c) Тот же вопрос, если $R(x, y)$ означает $xy = 1$.

d) Тот же вопрос, если $R(x, y)$ означает $xy > 1$.

16. Является ли композиция отношений эквивалентности отношением эквивалентности?

17*. Докажите, что всякое отношение $R \subseteq A \times B$ на конечных множествах является композицией $U \circ f_R$ некоторой функции f_R, зависящей от R, и отношения U, которое зависит только от множеств A и B.

18. Пусть A состоит из m элементов, а B состоит из n элементов. Сколько всего бывает отношений $R \subseteq A \times B$?

Лекция 8

Мощность множеств

Мы уже говорили о том, что множества — и не только конечные, можно сравнивать по «размеру», «количеству элементов», или, как говорят, «мощности» (английский термин — cardinality), и при этом удивительным образом оказывается, что не все бесконечные множества «одинаково бесконечны», среди них тоже бывают бóльшие и меньшие. В этой лекции мы

- подробно изучим самые маленькие среди бесконечных множеств (они называются «счётными»;
- убедимся, что не все множества бывают счётными, и приведём несколько примеров несчётных множеств одной мощности («мощности континуум»);
- наконец, совсем кратко упомянем, что бывают и ещё бóльшие мощности.

8.1 Равномощные множества

8.1.1 Определение равномощности

Рассказывая про мощность множеств, обычно начинают с такой байки: как убедиться, кого больше в комнате: людей или стульев, не пересчитывая их? Понятно как: надо попросить всех сесть, и сразу будет видно, останутся ли свободные стулья или люди без мест (или как раз в точности всем хватит места). С математической точки зрения можно сказать: мы пытаемся установить взаимно однозначное соответствие между людьми и стульями, другими словами, объединить их в пары (человек, стул) — таким образом, чтобы каждый человек попал ровно в одну пару (никто не остался без места и не сидел на двух стульях) и чтобы каждый стул попал в одну пару (не было бы пустых и не сидели бы по двое). Если это удастся (такое соответствие существует), то мы заключаем, что людей и стульев одинаковое количество.

Кстати, интересный вопрос: может ли так случиться, что сразу это не удалось (скажем, остались лишние стулья), а потом всех попросили пересесть иначе, и лишних стульев не осталось? Мы все уверены, что нет (если, конечно, кто-то не пришёл

незаметно или стул не унесли тайно), но почему, и с какого возраста у людей появляется такая убеждённость? Были опыты знаменитого психолога Пиаже, которые вроде бы показывают, что она появляется скорее в некотором возрасте, чем с накоплением жизненного опыта.

Так или иначе, этот трюк с людьми и стульями можно произвести и для бесконечных множеств — он становится *определением*. А именно, мы говорим, что множество A называется *равномощным* множеству B, если существует биекция множества A в множество B. *Часто вместо «равномощным» говорят «одинаковой мощности», мы так тоже будем делать, но тут надо быть аккуратным, потому что возникает вопрос: что такое эта самая «мощность», какой природы этот объект, и не очень понятно, как на это отвечать.*

Мы уже обсуждали пример Галилея, который заметил, что множество натуральных чисел и множество точных квадратов равномощны, потому что есть биекция $n \mapsto n^2$. Аналогичным образом множество натуральных чисел и множество чётных натуральных чисел равномощны, потому что существует биекция $n \mapsto 2n$. А множество положительных действительных чисел и множество отрицательных действительных чисел равномощны, потому что существует биекция $x \mapsto -x$.

8.1.2 Свойства равномощности

Следующие свойства, которые для конечных множеств кажутся самоочевидными, в общем случае требуют доказательства.

Отношение равномощности симметрично: если A равномощно множеству B, то B равномощно A. В самом деле, как мы уже обсуждали, ко всякой биекции есть обратная функция — тоже биекция.

Отношение равномощности рефлексивно: каждое множество равномощно самому себе. В самом деле, тождественная функция, которая отображает каждый элемент какого-то множества A в себя, является биекцией между A и A.

Наконец, *отношение равномощности транзитивно*: если A равномощно B и B равномощно C, то A равномощно C. Этот факт доказан выше в теореме 6.7 о том, что композиция биекций — биекция.

Например, мы уже видели, что натуральные числа равномощны чётным натуральным числам, а также равномощны точным квадратам: применяя симметричность и транзитивность, заключаем, что множество чётных натуральных чисел равномощно множеству точных квадратов.

Задача 8.1. Как выглядит соответствующая биекция? (Надо описать, какой точный квадрат соответствует данному чётному натуральному числу, скажем, числу 46.)

Таким образом, равномощность множеств обладает всеми свойствами отношения эквивалентности.[1]

[1]Мы из осторожности не говорим «является отношением эквивалентности» и не говорим о соответствующих классах эквивалентности, которые можно было бы назвать «мощностями», потому что не очень понятно, на каком множестве определено отношение равномощности. Хочется сказать,

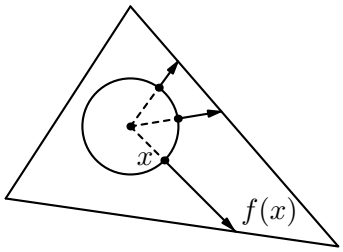
Вот ещё несколько примеров равномощных множеств. Чтобы увидеть, что отрезки $[0,1]$ и $[0,2]$ равномощны, достаточно рассмотреть биективную функцию f, заданную формулой $f(x) = 2x$. Эта функция умножает свой аргумент на 2; обратная биекция, напротив, делит свой аргумент на 2.

Чуть более сложный пример: интервал $(0,1)$ и полупрямая $(1,\infty)$ равномощны. Биекцию между ними устанавливает функция $x \mapsto 1/x$.

Задача 8.2. Равномощны ли интервал $(0,1)$ и полупрямая $(0,\infty)$?

8.1.3 Примеры равномощных множеств

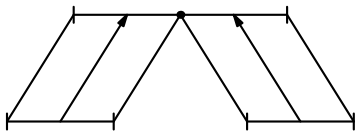
Приведём несколько геометрических примеров. Прежде всего заметим, что равные геометрические фигуры равномощны: они совмещаются движением плоскости, которое и устанавливает между ними взаимно однозначное соответствие. Равномощны и подобные фигуры, просто надо вместо движения рассматривать преобразование подобия (с растяжением в нужное число раз). Но даже и сохранение формы для равномощности не нужно. Если взять проволочное кольцо (окружность) и согнуть его так, чтобы оно приняло форму треугольника, то тем самым мы установим биекцию между окружностью и треугольником. При этой биекции начальному положению частицы окружности соответствует положение той же самой частицы в треугольнике после изгибания. Впрочем, построить биекцию между окружностью и треугольником можно и чисто геометрически:



Задача 8.3. Покажите, что квадрат (с внутренностью) и треугольник (с внутренностью) равномощны.

Может показаться, что разрезав проволочный отрезок на две части кусачками, мы установим биекцию и докажем, что отрезок равномощен множеству, состоящему из объединения двух отрезков половинной длины. Но тут физическая интуиция нас обманывает — ну, или математическая интуиция отрывается от реальности. Если мы захотим рассмотреть соответствие, возникающее при разрезании отрезка на два, то окажется, что оно не будет взаимно однозначным.

что на множестве всех множеств, потому что про любые два множества можно спросить, равномощны они или нет. Но это самое множество всех множеств так просто рассматривать нельзя, это быстро приведёт к противоречию, как мы увидим в конце этой главы.



В самом деле, точке разреза соответствуют два конца двух половинок, так что сверху вниз это не функция, а снизу вверх — не инъекция. Тут сказывается разница между отрезками и интервалами (отрезками без концов) — которой в проволочной модели не видно.

Эту разницу стоит обсудить подробнее. Если мы удалим из отрезка $[0,1]$ самую правую точку 1, то получится полуинтервал $[0,1)$. Какая в нём самая правая точка? Хочется сказать (и многие школьники так и говорят), что это точка $0.99999\ldots$. Тем не менее с точки зрения математики дробь $0.99999\ldots$ — это другая запись того же числа 1. А в полуинтервале $[0,1)$ самой правой точки, как это ни странно с наглядной точки зрения, нет: если какое-то x меньше числа 1 на какое-то ε, пусть очень маленькое, то между x и 1 есть ещё другие точки. Например, можно из 1 вычесть не само ε, а ещё меньшее число $\varepsilon/2$, получится точка посередине между x и 1.

Если мы хотим быть аккуратными, то надо сказать, что можно установить взаимно однозначное соответствие

$$f\colon [0,2] \to [0,1) \cup [2,3]$$

по формуле

$$f(x) = \begin{cases} x, \text{ если } 0 \leqslant x < 1; \\ x+1, \text{ если } 1 \leqslant x \leqslant 2. \end{cases}$$

Разумеется, можно было бы считать точку разреза попавшей в первую половину: множество $[0,2]$ равномощно и множеству $[0,1] \cup (2,3]$.

Возникает вопрос: а равномощны ли отрезок и полуинтервал (или интервал)? Можно ли построить, скажем, биекцию $f\colon [0,1] \to [0,1)$? Хочется сказать, что да: ясно, что в отрезке $[0,1]$ не меньше точек, чем в $[0,1)$, даже одна лишняя, и — с другой стороны — в $[0,1)$ не меньше точек, чем в отрезке половинной длины $[0,1/2]$, а мы знаем что отрезки разных длин равномощны (растяжение). Как мы увидим дальше, говоря о теореме Кантора–Бернштейна, это рассуждение можно узаконить, но просто так оно не имеет смысла, потому что понятия «меньше» и «больше» мы для множеств не определяли.

Тем не менее отрезок и полуинтервал (а также интервал) равномощны. Мы увидим это, используя свойства счётных множеств (следующий раздел), но уже сейчас можете попытаться придумать искомую биекцию, не заглядывая дальше.

Задача 8.4. Укажите биекцию $f\colon [0,1] \to [0,1)$.

Если это пока не получается, можно потренироваться на других примерах.

Задача 8.5. Укажите взаимно однозначное соответствие между прямой $\mathbb{R}$ и интервалом $(0,1)$. (Указание: у нас уже было соответствие между их половинами.)

Задача 8.6. Укажите взаимно однозначное соответствие между плоскостью и внутренностью круга (без границы).

А вот несколько негеометрических примеров.

Задача 8.7. Установите взаимно однозначное соответствие между бесконечными последовательностями нулей и единиц (бесконечными двоичными дробями без целой части) и подмножествами натурального ряда.

Задача 8.8. Установите взаимно однозначное соответствтствие между бесконечными последовательностями цифр $0, 1$ и бесконечными последовательностями цифр $0, 1, 2, 3$.

Задача 8.9. (Продолжение) Тот же вопрос для последовательностей цифр $0, 1, 2$.

8.2 Счётные множества

8.2.1 Определение и простейшие примеры

Среди бесконечных множеств выделяют так называемые *счётные множества*. Как мы увидим дальше, это «самые маленькие» бесконечные множества, а пока формальное определение: счётные множества — это множества равномощные множеству натуральных чисел $\mathbb{N}$. Другими словами, множество A счётно, если существует биекция между $\mathbb{N}$ и A. (Как мы знаем, отношение равномощности симметрично, так что биекцию можно представлять себе действующей в любую сторону.)

Наглядно это можно себе представлять так: каждый элемент множества пронумерован — ему присвоен номер (натуральное число), и каждое натуральное число использовано ровно один раз. Вообще, когда первоклассник считает камешки на столе, говоря «один, два, три, четыре. . . », или когда завхоз составляет список мебели в номере (типа «1. Кровать железная. 2. Шкаф деревянный.. . . »), то, научно говоря, они устанавливают взаимно однозначное соответствие между предметами и натуральными числами. Для счётных множеств происходит то же самое, только процесс этот бесконечный: мы пересчитываем элементы множества, присваивая им номера и располагая их в последовательность, и «в результате» (если представить себе, что этот бесконечный процесс завершился) каждый элемент приобретает свой номер (натуральное число). Биекция $f : \mathbb{N} \to A$ при этом сопоставляет с каждым натуральным числом предмет, пронумерованный этим числом. Сюръективность функции f означает, что каждый элемент A получит свой номер. А инъективность означает, что никакой элемент не будет пронумерован дважды.

Самый простой пример счётного множества — это само множество натуральных чисел $\mathbb{N}$. В качестве биекции можно взять тождественную функцию: само число и будет своим номером. Другой простой пример: множество положительных натуральных чисел. (Мы считаем нуль натуральным числом, так что это множество состоит из всех натуральных чисел, кроме нуля.) Здесь в качестве биекции можно взять функцию $f(x) = x + 1$ (отображающую множество натуральных чисел в множество

положительных натуральных чисел). Ещё два примера у нас уже были: множество чётных натуральных чисел с биекцией $x \mapsto 2x$ и множество точных квадратов с биекцией $x \mapsto x^2$.

Чуть более сложный пример — это множество целых чисел $\mathbb{Z}$. Оно тоже счетно. Чтобы это увидеть выпишем последовательно целые числа следующим образом:

$$0, 1, -1, 2, -2, 3, -3, \ldots.$$

Видно, что всякое целое число рано или поздно встретится в этой последовательности, и только один раз. По существу, мы только что перенумеровали множество $\mathbb{Z}$. Научно говоря, мы построили биекцию: в нуле она будет равна первому элементу последовательности, в единице — второму и так далее

$$\begin{array}{lcccccccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \ldots \\ & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ \mathbb{Z}: & 0 & 1 & -1 & 2 & -2 & 3 & -3 & 4 & -4 & 5 & \ldots \end{array}$$

Таким образом, чтобы показать счётность множества, надо выписать его элементы в последовательность, каждый по одному разу и ничего не пропустив.

8.2.2 Свойства счётных множеств

Лемма 8.1. *Объединение двух счётных множеств счётно.*

Доказательство. Рассмотрим два счетных множества A и B; каждое из них можно записать в последовательность:

$$\begin{array}{llllll} a_0, & a_1, & a_2, & a_3, & \ldots \\ b_0, & b_1, & b_2, & b_3, & \ldots \end{array}$$

Теперь нетрудно перечислить и элементы множества $A \cup B$, чередуя элементы из A с элементами из B:

$$a_0, b_0, a_1, b_1, a_2, b_2, \ldots.$$

Если A и B не пересекаются, то на этом рассуждение заканчивается — но если пересекаются, то в этой последовательности общие элементы встретятся по два раза. Как это исправить? Если очередной элемент уже встречался ранее (например, если элемент a_j совпадает с элементом b_i, где $i < j$), то мы его пропускаем и второй раз не выписываем. □

Лемма 8.2. *Всякое подмножество счетного множества конечно или счетно.*

Доказательство. Рассмотрим счётное множество A и его подмножество A'. Выпишем элементы A в последовательность

$$a_0, a_1, a_2, a_3, \ldots.$$

Вычеркнем из этой последовательности те элементы, которые не лежат в A'. В результате останется последовательность элементов A' — конечная или бесконечная. В первом случае множество будет конечным, во втором счётным. Формально говоря, для бесконечного подмножества $A' \subset A$ искомая биекция $f\colon \mathbb{N} \to A'$ ставит в соответствие числу n элемент множества A', который стоит n-м по счёту в последовательности (если считать только элементы A'). □

Лемма 8.3. *Всякое бесконечное множество содержит счётное подмножество.*

Доказательство. Рассмотрим прозвольное бесконечное множество A. Нам надо выписать последовательность из некоторых его элементов, не обязательно всех. Будем действовать самым простым образом. Первый элемент a_0 возьмем произвольно. Поскольку A бесконечно, в нем есть ещё элементы (кроме a_0). В качестве a_1 возьмем любой из них. И так далее. В общем случае, когда нам нужно выбрать очередной элемент a_n, мы рассматриваем подмножество $\{a_0, \ldots, a_{n-1}\}$. Оно конечно, а значит, не совпадает со всем множеством A (которое по предположению бесконечно). Значит, в A есть элементы, не лежащие в этом подмножестве — и мы можем взять любой из них в качестве a_n.

Получили бесконечную последовательность из элементов A, и множество элементов этой последовательности образует искомое счётное подмножество множества A. □

Две последние леммы объясняют, в каком смысле счётные множества — это «самые маленькие» бесконечные множества. Между ними и конечными нет ничего промежуточного (лемма 8.2), и всякое бесконечное множество «не меньше» счётного в том смысле, что в нём есть счётное подмножество (лемма 8.3).

Позже мы увидим, что бывают и бóльшие (несчётные) бесконечные множества. Например, таким оказывается множество действительных чисел (или точек на прямой), но и это не предел.

Пока же продолжим изучение счётных множеств.

Лемма 8.4. *Множество рациональных чисел $\mathbb{Q}$ счетно.*

Доказательство. Нам будет удобнее доказать отдельно, что множество неотрицательных рациональных чисел счётно и что множество отрицательных рациональных чисел счётно. Тогда счётность множества всех рациональных чисел сразу вытекает из леммы 8.1.

Проведем рассуждение для неотрицательных рациональных чисел. Для отрицательных чисел рассуждение аналогично (а можно заметить, что смена знака даёт биекцию между отрицательными и положительными числами, а к положительным рациональным числам применить лемму 8.2).

Неотрицательное рациональное число задается парой чисел — числителем и знаменателем. Числитель может быть произвольным натуральным числом, а знаменатель произвольным положительным натуральным числом. Выпишем все такие

числа в виде таблицы, бесконечной вниз и вправо:

$$\begin{array}{cccccc} 0/1 & 1/1 & 2/1 & 3/1 & \dots \\ 0/2 & 1/2 & 2/2 & 3/2 & \dots \\ 0/3 & 1/3 & 2/3 & 3/3 & \dots \\ 0/4 & 1/4 & 2/4 & 3/4 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{array}.$$

В строке с номером i этой таблицы стоят последовательно все числа со знаменателем i, а в столбце с номером j — все числа с числителем j. В этой таблице будут выписаны все рациональные числа, причем некоторые будут повторяться много раз (например, $0/1 = 0/2 = 0/3 = \dots$ и $1/2 = 2/4 = 3/6 = \dots$).

Числа из этой таблицы теперь уже легко выписать в последовательность. Например, можно идти по диагоналям (вниз-влево). Сначала выпишем единственное число на первой диагонали (0/1), потом два числа на второй (1/1, 0/2), потом три числа на третьей и так далее:

$$0/1, 1/1, 0/2, 2/1, 1/2, 0/3, 3/1, 2/2, 1/3, 0/4, \dots.$$

Другими словами, мы сначала выписываем все числа с суммой числителя и знаменателя 1, потом — с суммой 2, потом 3 и так далее. Конечно, нужно не забыть выбрасывать из последовательности повторяющиеся члены. То есть, когда мы доходим в таблице до очередного числа и видим что равное ему уже было выписано, мы пропускаем текущее число и переходим к следующему. Получится такая последовательность рациональных чисел:

$$0, 1, 2, 1/2, 3, 1/3, \dots.$$

□

В этом доказательстве на самом деле не имеет значения, что именно мы записываем в бесконечную вправо и вниз таблицу: верно такое общее утверждение.

Теорема 8.5. *Объединение конечного или счётного числа конечных или счётных множеств конечно или счётно.*

Доказательство. Пусть есть счётное количество счётных множеств $A_0, A_1, A_2, \dots$. Как и в прошлой лемме, расположим их элементы в виде таблицы:

$$\begin{array}{cccccc} A_0: & a_{00} & a_{01} & a_{02} & a_{03} & \dots \\ A_1: & a_{10} & a_{11} & a_{12} & a_{13} & \dots \\ A_2: & a_{20} & a_{21} & a_{22} & a_{23} & \dots \\ A_3: & a_{30} & a_{31} & a_{32} & a_{33} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}.$$

Здесь в первой строке мы последовательно выписали элементы A_0, во второй — элементы A_1 и так далее. Теперь снова соединяем эти последовательности в одну, идя по диагоналям:

$$a_{00}, a_{01}, a_{10}, a_{02}, a_{11}, a_{20}, a_{03}, a_{12}, a_{21}, a_{30}, \ldots.$$

При этом нужно следить, чтобы члены последовательности не повторялись: когда мы рассматриваем очередной элемент таблицы, нужно проверить, не встретился ли он раньше. Если он уже был, его нужно пропустить.

Мы предполагали, что все A_i счётны и что их счётное число. Если самих множеств лишь конечное число, или если какие-то из множеств конечны, то в таблице часть ячеек окажется пустой. Соответственно, мы будем их пропускать при составлении последовательности. В результате либо получится бесконечная последовательность, и тогда объединение счётно, либо получится только конечная последовательность — и тогда объединение конечно. □

По существу ту же теорему можно сформулировать иначе:

Теорема 8.6. *Декартово произведение двух счётных множеств $A \times B$ счётно.*

Доказательство. В самом деле, по определению декартово произведение есть множество всех упорядоченных пар вида (a, b), в которых $a \in A$ и $b \in B$. Разделим пары на группы, объединив пары с одинаковой первой компонентой (каждая группа имеет вид $\{a\} \times B$ для какого-то $a \in A$). Тогда каждая группа счётна, поскольку находится во взаимно однозначном соответствии с B (пара определяется своим вторым элементом), и групп столько же, сколько элементов в A, то есть счётное число. □

Например, множество пар натуральных чисел $\mathbb{N}\times\mathbb{N}$ счётно, поскольку его можно разбить в объединение счётного числа счётных множеств: $\{0\}\times\mathbb{N}, \{1\}\times\mathbb{N}, \{2\}\times\mathbb{N}, \ldots$.

А что можно сказать про $\mathbb{N}^3$, множество всех упорядоченных троек натуральных чисел? Можно было бы составить аналогичную трёхмерную таблицу и перечислять все тройки по «двумерным диагональным плоскостям», то есть в порядке увеличения суммы трёх членов тройки. (В самом деле, троек с данной суммой конечное число.) Тот же приём годится и для четвёрок и вообще для множества $\mathbb{N}^k$ при любом натуральном k.

Вместо этого можно воспользоваться индукцией по k. При $k = 1$ утверждение очевидно. Если мы уже доказали счётность $\mathbb{N}^k$, то можно представить $\mathbb{N}^{k+1}$ как произведение $\mathbb{N} \times \mathbb{N}^k$ (набор из $k + 1$ одного числа можно, как делают программисты на лиспе, считать парой (первое число, набор из k остальных чисел), и потому $\mathbb{N}^{k+1}$ счётно как произведение счётных множеств. Другими словами, можно разбить множество $\mathbb{N}^{k+1}$ в счетное объединение следующих множеств: $\{0\} \times \mathbb{N}^k, \{1\} \times \mathbb{N}^k, \{2\} \times \mathbb{N}^k, \ldots$. По предположению индукции каждое из этих множеств счетно, а значит счетно и их объединение.

Теперь можно доказать, что множество конечных последовательностей натуральных чисел счётно. Действительно, это множество можно разбить на части: последовательности длины 1, длины 2, длины 3 и так далее. Последовательностей

фиксированной длины k счётное число, как мы только что доказали, так что мы получили объединение счётного числа счётных множеств.

Вместо натуральных чисел можно рассматривать элементы произвольного конечного или счётного множества A. Такое множество называют *алфавитом*, элементы множества A называют *буквами*, или *символами* алфавита A, а конечные цепочки (последовательности) букв называют *словами*, или *строками* в алфавите A. Так что можно говорить, скажем, о словах в русском алфавите (и это не только слова русского языка, но и бессмысленные цепочки русских букв).

Задача 8.10. Докажите, что число слов в конечном или счётном алфавите счётно.

Частный случай этой задачи: слова в двоичном алфавите $\{0,1\}$, которые программисты называют битовыми строками; множество таких строк тоже счётно.

Задача 8.11. Доказывая счётность множества битовых строк, профессор говорит: «в самом деле, запись в двоичной системе даёт биекцию между множеством всех битовых строк и множеством всех натуральных чисел». Почему он неправ? Как можно исправить его ошибку и указать искомую биекцию?

8.3 Несчётные множества

8.3.1 Интервал и отрезок равномощны

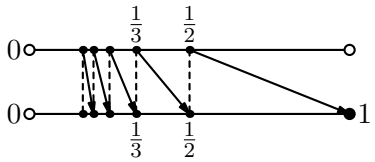
В этом разделе мы перейдём к несчётным множествам. Точнее, пока что мы должны говорить о множествах, про которые не ясно, счетны они или нет. К таким множествам, например, относятся интервал $(0,1)$, отрезок $[0,1]$, полупрямая $(0,\infty)$, прямая $(-\infty,\infty)$, а также соответствующие геометрические объекты. Мы уже видели, что интервал $(0,1)$ равномощен полупрямой $(0,\infty)$. Из этого легко получить, что интервал $(0,1)$ равномощен прямой $(-\infty,\infty)$. Действительно, интервал $(0,1)$ равномощен интервалу $(0,2)$ (биекция $f(x)=2x$), а он, в свою очередь, равномощен $(-1,1)$ (биекция $f(x)=x-1$). Теперь можно разбить интервал $(-1,1)$ на три части $(-1,0)$, $\{0\}$ и $(0,1)$. Первую из них можно биективно отобразить на полупрямую $(-\infty,0)$, точку 0 интервала можно отобразить в точку 0 прямой, а интервал $(0,1)$ можно отобразить в полупрямую $(0,\infty)$. Соединив эти три биекции в одну, мы получаем биекцию из $(-1,1)$ в $(-\infty,\infty)$. Теперь, пользуясь транзитивностью отношения равномощности, мы получаем, что интервал $(0,1)$ равномощен числовой прямой $(-\infty,\infty)$.

Это всё мы уже обсуждали — но теперь мы готовы пойти дальше и доказать, что интервал $(0,1)$ и полуинтервал $(0,1]$ тоже равномощны. Для этого мы воспользуемся известными нам сведениями про счётные множества. Выделим в интервале какое-нибудь счётное подмножество, например $A=\{1/2,1/3,1/4,1/5,\ldots\}$. Если мы добавим к нему точку 1, то оно останется счётным: $A\cup\{1\}=\{1,1/2,1/3,1/4,\ldots\}$. Таким образом, существует биекция f из множества A в множество $A\cup\{1\}$. Теперь нетрудно доопределить f до биекции всего интервала $(0,1)$ в полуинтервал $(0,1]$. Для этого скажем, что все точки, на которых f пока не определено, то есть все

точки из $(0,1) \setminus A$, переходят в себя: $f(x) = x$. Можно записать это формулой:

$$f(x) = \begin{cases} 1/(n-1), \text{если } x \in A \text{ и } x = 1/n \text{ при } n \geqslant 2; \\ x, \text{если } x \notin A. \end{cases}$$

и изобразить на картинке:



Полученная функция составлена из двух биекций и сама является биекцией из $(0,1)$ в $(0,1]$.

Задача 8.12. Докажите, что интервал $(0,1)$ равномощен полуинтервалу $[0,1)$.

Задача 8.13. Докажите, что интервал $(0,1)$ равномощен отрезку $[0,1]$.

8.3.2 Добавление счётного множества

На самом деле геометрический характер этого рассуждения — только видимость. По существу мы доказали такой общий результат:

Теорема 8.7. *Если множество A бесконечно, а множество B конечно или счётно, то множество $A \cup B$ равномощно A.*

Доказательство. Без ограничения общности можно считать, что множества A и B не имеют общих элементов: $A \cap B = \varnothing$. Действительно, если это не так, то можно просто не добавлять те элементы, которые уже есть в A, то есть вместо множества B рассмотреть множество $B \setminus A$. Оно тоже конечно или счётно (лемма 8.2), с A уже не пересекается, а объединение множеств от такой замены не изменится.

Как мы уже знаем (лемма 8.3), в множестве A есть счетное подмножество A_0. Объединение $A_0 \cup B$ тоже счётно (теорема 8.5). Значит, существует биекция $f\colon A_0 \to A_0 \cup B$. Остаётся продолжить её до биекции всего множества A в множество $A \cup B$, положив $f(x) = x$ для всех $x \in A \setminus A_0$. □

В сущности, мы повторили то же самое рассуждение, что и раньше, только без картинки (и для произвольного конечного или счётного множества вместо одной точки).

Задача 8.14. Докажите, что если множество A бесконечно, а множество B конечно, то разность $A \setminus B$ равномощна A. Почему в этом утверждении нельзя заменить «конечно» на «счётно»?

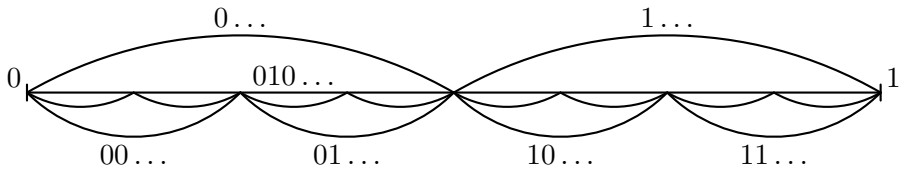
8.3.3 Числа и последовательности

Теперь мы можем установить соответствие между двумя множествами, которые мы рассматривали по отдельности:

Теорема 8.8. *Отрезок* $[0,1]$ *равномощен множеству бесконечных последовательностей из нулей и единиц.*

Отсюда следует, что и интервал, и прямая равномощны множеству бесконечных последовательностей нулей и единиц (поскольку они, как мы уже знаем, равномощны отрезку).

Доказательство. Доказательство этой теоремы требует каких-то знаний о действительных числах. Будем считать, что из курса анализа известно, что каждое число $x \in [0,1]$ можно записать в виде бесконечной двоичной дроби (аналогично тому, как его можно записать в виде бесконечной десятичной дроби). Напомним, как это делается. Первый знак (бит) после запятой равен 0, если x лежит в левой половине отрезка $[0,1]$, и равен 1, если в правой. Чтобы определить следующий бит, нужно поделить выбранную половину снова пополам. Если x лежит в левой половине, то следующая цифра 0, а если в правой, то 1. И так далее: чтобы определить очередной знак, нужно поделить текущий отрезок пополам и посмотреть, в какую половину попадает x.



Можно ли сказать, что мы тем самым построили взаимно однозначное соответствие между числами на отрезке $[0,1]$ и бесконечными двоичными последовательностями (дробями)? Не совсем. Проблема в том, что это соответствие не взаимно однозначно, некоторым числам соответствуют две последовательности. А именно, это происходит, когда точка попадает на границу очередного отрезка. Тогда мы можем относить её как к левой, так и к правой половине. В результате, например, последовательности $0,1001111\ldots$ и $0,101000\ldots$ соответствуют одному и тому же числу (какому?).[2]

Как же исправить положение? Мы получим взаимно однозначное соответствие, если исключим последовательности, в которых начиная с некоторого момента все цифры равны 1 (одну такую последовательность мы всё же должны оставить – это $0.111\ldots$). Но таких последовательностей счётное множество, так что их добавление не меняет мощности множества по теореме 8.7. □

[2]Тот же самый эффект хорошо известен для десятичной системы: дроби $0.19999\ldots$ и $0,20000\ldots$ представляют одно и то же число, $1/5$.

Естественно, не обязательно было пользоваться двоичной системой: можно было бы рассмотреть обычную десятичную систему, и показать, что отрезок $[0,1]$ равномощен множеству бесконечных десятичных дробей. Или троичную — и показать, что тот же самый отрезок $[0,1]$ равномощен множеству бесконечных последовательностей из цифр $0,1,2$. Кстати, отсюда следует, что все эти множества последовательностей равномощны — тем самым мы получаем решение задач 8.8 и 8.9. Правда, оно использует сведения о действительных числах, и получается довольно сложно описываемое соответствие (если всё развернуть), так что всё равно остаётся задача явно указать просто описываемое соответствие между этим множествами.

Задача 8.15. Покажите, что отрезок $[0,1]$ равномощен множеству бесконечных последовательностей натуральных чисел.

8.3.4 Отрезок и квадрат

Теперь мы можем доказать следующий довольно неожиданный результат. (Изобретатель теории множеств немецкий математик XIX века Георг Кантор даже надеялся доказать обратное и тем самым подкрепить интуитивное ощущение, что «на двумерной плоскости больше точек, чем на одномерной прямой» — и был очень удивлён, когда понял, что на самом деле это не так!)

Теорема 8.9. *Отрезок* $[0,1]$ *равномощен квадрату* $[0,1]\times[0,1]$.

Доказательство. Мы уже знаем, что отрезок $[0,1]$ равномощен множеству бесконечных последовательностей нулей и единиц. Тогда квадрат $[0,1]\times[0,1]$ равномощен множеству упорядоченных пар таких последовательностей. Действительно, чтобы получить пару, соответствующую точке (x,y), надо составить пару из последовательности, соответствующей x, и последовательности, соответствующей y.

Осталось установить взаимно однозначное соответствие между бесконечными последовательностями нулей и единиц и парами таких последовательностей (и воспользоваться транзитивностью). Это сделать уже не так сложно. Паре последовательностей

$$(a_0,a_1,a_2,a_3,\dots\ ,b_0,b_1,b_2,b_3,\dots)$$

можно поставить в соответствие последовательность

$$a_0,b_0,a_1,b_1,a_2,b_2,a_3,b_3,\dots.$$

Нетрудно увидеть, что это отображение взаимно однозначное (обратное к нему выделяет из последовательности отдельно чётные и отдельно нечётные члены). □

Хочется сказать по-простому: каждой паре чисел $(0.x_1x_2x_3\dots, 0.y_1y_2y_2\dots)$ мы ставим в соответствие число $0.x_1y_1x_2y_2x_3y_3\dots$. И всё было бы хорошо, если бы не эта проблема с двумя представлениями: числу $0.313939393939\dots$ соответствует пара $(0.333\dots, 0.1999\dots) = (1/3, 1/5)$, и та же самая пара соответствует другому числу $0.323030303030\dots$. Поэтому нам понадобилось предварительно использовать тот факт, что добавление счётного множества не меняет мощности.

Задача 8.16. Докажите, что множество точек пространства (которые можно записывать тремя координатами, то есть множество $\mathbb{R}^3$) равномощно $\mathbb{R}$. Докажите, что $\mathbb{R}^k$ равномощно $\mathbb{R}$ при любом k.

Задача 8.17. Докажите, что множество бесконечных последовательностей действительных чисел равномощно $\mathbb{R}$.

В этом разделе мы привели примеры бесконечных равномощных множеств. Скажем для определённости, что все они равномощны множеству действительных чисел $\mathbb{R}$. В следующем разделе мы докажем, что отрезок $[0, 1]$ — несчётное множество, а значит и $\mathbb{R}$, и все остальные множества из этого раздела несчётны. Их мощность называют словом «*континуум*» — устаревшим названием множества, которое обладает свойством непрерывности. Формально, множество A имеет мощность континуум, если A равномощно $\mathbb{R}$.

8.4 Диагональный аргумент Кантора и сравнение мощностей

8.4.1 Несчётность отрезка

Давайте на минутку остановимся и посмотрим, что мы уже доказали. У нас было много примеров счётных множеств, а также — отдельно от этого — много примеров множеств, равномощных между собой (отрезок, интервал, полуинтервал, множество последовательностей нулей и единиц, множество точек квадрата и пр.). Мы уже упоминали, что множества во втором списке несчётны, но пока что это не доказали. Давайте это сделаем. Поскольку они все равномощны, мы можем выбрать удобное нам.

Теорема 8.10 (Кантор)**.** *Множество бесконечных последовательностей нулей и единиц несчётно.*

Доказательство. Предположим, что это множество счётно, то есть что последовательности нулей и единиц можно пронумеровать. Обозначим последовательность с номером i через a_i, а её члены обозначим $a_{i0}, a_{i1}, \ldots$.

Удобно члены последовательностей писать слева направо, а саму последовательность последовательностей писать сверху вниз. Получится нечто вроде бесконечной таблицы:

$$\begin{array}{lcccccc} a_0 & = & a_{00} & a_{01} & a_{02} & \ldots \\ a_1 & = & a_{10} & a_{11} & a_{12} & \ldots \\ a_2 & = & a_{20} & a_{21} & a_{22} & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{array}$$

Теперь рассмотрим «диагональную» последовательность в этой таблице, то есть последовательность

$$a_{00}, a_{11}, a_{22}, \ldots$$

и заменим в ней все биты на противоположные. Другими словами, положим $b_i = 1 - a_{ii}$ и рассмотрим последовательность $b = (b_0, b_1, b_2, \ldots)$. Последовательность b отличается от любой последовательности a_i в i-й позиции, поскольку $b_i = 1 - a_{ii} \neq a_{ii}$, так что последовательности b нет в нашей нумерации всех последовательностей — а там по предположению были все. Мы пришли к противоречию, а значит, множество всех бесконечных последовательностей нулей и единиц несчетно. □

Поскольку это очень важная теорема, давайте изложим по существу то же доказательство немного по-другому. Докажем, что если дано счётное множество бесконечных двоичных дробей $a_0, a_1, a_2, \ldots$, то можно построить бесконечную дробь b, которая отлична от всех них. Нам надо удовлетворить счётное число требований: во-первых, b не должна совпасть с a_0, во-вторых, b не должна совпасть с a_1, и так далее. Заметим, что для совпадения двух дробей надо, чтобы совпадали все члены — а для *не*совпадения достаточно, чтобы не совпал какой-то один. Поэтому условия $b \neq a_0$ можно добиться, взяв первый бит в b не таким, какой он в a_0. Что бы мы ни делали дальше, b уже совпасть с a_0 не сможет. После этого мы гарантируем $b \neq a_1$, взяв второй бит b не таким, как у a_1, затем мы гарантируем $b \neq a_2$ с помощью третьего бита и так далее. В итоге полученная последовательность b будет отличаться от всех a_i.

Задача 8.18. Можно ли провести то же самое рассуждение для конечных последовательностей нулей и единиц? В каком месте перестаёт работать доказательство?

Мы уже знаем, что отрезок $[0, 1]$ равномощен множеству бесконечных последовательностей нулей и единиц, так что отрезок тоже будет несчётным множеством. Можно доказать несчётность сразу для отрезка. Пусть дана последовательность $a_0, a_1, a_2, \ldots$ точек отрезка (действительных чисел от 0 до 1). Как построить действительное число, которого в ней нет? Построим последовательность вложенных отрезков, общая точка которых (она существует, как учит математический анализ) будет отсутствовать среди a_i. Начнём с любого отрезка, не содержащего a_0. Уменьшим его так, чтобы в новый отрезок не попало и a_1 (скажем, поделим его на три части, a_1 в самом неудачном случае может попасть в две, а мы возьмём третью). Потом снова уменьшим, чтобы не попало и a_2. И так далее — мы постепенно отделимся (исключим возможность совпадения) от всех a_i.

Зная, что множество действительных чисел несчётно, а множество рациональных чисел (дробей с целыми числителями и знаменателями) счётно, мы получаем в качестве следствия, что существуют иррациональные числа (не представимые такими дробями). Впрочем, это знали ещё древние греки, доказавшие иррациональность $\sqrt{2}$. Но тот же метод можно использовать и для доказательства более сильного утверждения. *Алгебраическим числом* называется число, являющееся корнем ненулевого многочлена с целыми коэффициентами.

Задача 8.19. Докажите, что таких многочленов счётное число.

Задача 8.20. Докажите, что множество алгебраических чисел счётно. (Указание: из алгебры известно, что у многочлена степени n не больше n корней.)

Задача 8.21. Докажите, что существует действительное, но не алгебраическое число.

Такие числа называются трансцендентными. То, что они бывают, было доказано до Кантора. Первым это сделал французский математик Лиувилль: он показал, что бесконечная дробь

$$0.01001000000100000000000000000000010\ldots$$

в которой сначала один нуль, потом два, потом шесть (3!), потом $24 = 4!$ и так далее по факториалам, не будет алгебраической — основываясь на том, что она уж слишком хорошо приближается рациональными числами. Потом (гораздо более сложным образом) была доказана трансцендентность чисел e и π (второе показывает, в частности, что задача о квадратуре круга не решается циркулем и линейкой). Но все эти доказательства были про конкретные числа и требовали некоторой работы — а тут пришёл Кантор и вывел это из общих соображений о счётных и несчётных множествах.

Сначала это воспринималось с подозрением, но теперь эта конструкция стала стандартным средством и в анализе (теорема Бэра о том, что пересечение всюду плотных открытых множеств в полном пространстве непусто), и в computer science (диагональный метод — с его помощью, кстати, можно доказать, что в любом разумном языке программирования есть программа, печатающая свой текст),

Задача 8.22. Докажите, что множество всех подмножеств множества натуральных чисел несчетно.

Задача 8.23. Докажите, что множество всех функций $f\colon \mathbb{N} \to \{0,1\}$ несчётно.

Задача 8.24. Докажите, что существует функция $f\colon \mathbb{N} \to \{0,1\}$, для которой не существует программы на вашем любимом языке программирования, вычисляющей эту функцию.

8.4.2 Сравнение мощностей

Вернёмся к байке, с которой мы начинали — про людей и стулья. Мы говорили тогда, что если стульев не хватило, то бессмысленно вставать и пересаживаться, их всё равно не хватит. Говоря научно, если конечное множество A равномощно некоторой части конечного множества B, не совпадающей с B, то оно не равномощно самому B. Можно сказать, что A имеет «меньшую мощность». Для бесконечных множеств это не так: множество может быть равномощно своей части.

Тем не менее можно дать такое определение: множество A имеет мощность не большую, чем множество B, если A равномощно некоторому подмножеству множества B (возможно, совпадающему с B).

Задача 8.25. Докажите, что счётное множество имеет мощность не больше, чем любое бесконечное множество.

Задача 8.26. Докажите, что множество A имеет мощность не больше, чем множество B, в том и только том случае, когда существует инъекция $f\colon A \to B$.

Задача 8.27. Докажите, что множество A имеет мощность не больше, чем множество B, в том и только том случае, когда существует сюръекция $g\colon B \to A$.

Эта терминология («не больше») опасна: она подразумевает, что для такого сравнения множеств по мощности выполнены два свойства, которые мы знаем для обычного сравнения чисел — между тем мы их пока что не доказали. Вот эти свойства:

- Если мощность A не больше, чем у B, и одновременно мощность B не больше, чем у A, то A и B равномощны.
- Для любых двух множеств A и B мощность одного из них не больше, чем у другого.

Первое свойство называется (на языке упорядоченных множеств) антисимметричностью, а второе — линейностью порядка. Оба эти свойства верны. Второе — одно из главных достижений Кантора; он доказал его, придумав то, что теперь называется «трансфинитной индукцией», это обобщение метода математической индукции на множества произвольной мощности. Сам этот метод, увы, выходит за рамки этих лекций (про него можно прочесть, например, в книге «Начала теории множеств» Верещагина и Шеня [3]), так что это второе утверждение так и останется (увы) для нас недоказанным. А вот первое утверждение доказать легче, что мы сейчас и сделаем.

Обычно это утверждение называют *теоремой Кантора–Бернштейна*, хотя исторически это не вполне точно: первые её доказательства были опубликованы Шрёдером (1896) и Бернштейном (1897). Мы ее сформулировали выше в терминах мощности множеств, однако можно заметить, что ее несложно переформулировать в терминах функций.

Заметим, что взаимно однозначное соответствие между A и подмножеством B — это в точности инъекция A в B. Соответственно, взаимно однозначное соответствие между B и подмножеством A – это инъекция A в B. Таким образом, нам нужно доказать следующее утверждение.

Теорема 8.11. *Если для множеств A и B существует инъекция из A в B и инъекция из B в A, то существует и биекция между A и B.*

Доказательство. Пусть $f\colon A \to B$ и $g\colon B \to A$ – инъекции. Рассмотрим (возможно, бесконечный) ориентированный граф с вершинами $A \cup B$ (для простоты обозначений предположим, что A и B не пересекаются). Для точек $x \in A$ и $y \in B$ мы проводим ребро из x в y, если $f(x) = y$, и ребро из y в x, если $g(y) = x$. Если нарисовать множество A слева, а множество B справа, то можно сказать, что мы проводим рёбра слева направо согласно функции f и справа налево согласно функции g.

По построению из каждой точки выходит ровно одно ребро. А сколько рёбер входит? Поскольку функции инъективны, то не больше одного (но может не входить не одного).

Разобьем граф на компоненты связности (забыв для этого об ориентации рёбер) и рассмотрим каждую компоненту отдельно. Как устроены эти компоненты? Есть три возможности. Связная компонента может быть

- циклом из стрелок;
- бесконечной цепочкой стрелок, начинающейся в некоторой вершине (в которую ничего не входит);
- бесконечной в обе стороны цепочкой стрелок.

В самом деле, вперёд всегда можно идти по единственной стрелке, а назад либо можно пойти единственным образом, либо нельзя пойти вовсе. Если, идя вперёд, мы дважды попадём в одну вершину, то образуется цикл (и это возможно, лишь если мы вернёмся в начальную вершину). Если нет, то образуется бесконечная цепочка вперёд; её можно однозначно продолжать назад, при этом либо мы упрёмся в вершину, где назад не пройти, либо получим двустороннюю цепочку.

Задача 8.28. Проведите это рассуждение более подробно.

Это верно для любого ориентированного графа, в котором из каждой вершины выходит ровно одна стрелка и в каждую вершину входит не больше одной стрелки. В нашем конкретном случае есть дополнительная структура: вершины бывают левые и правые (из A и из B). Они чередуются, поэтому цикл может быть только чётной длины и содержит поровну вершин из A и из B. Любое из отображений f и g может быть использовано, чтобы построить биекцию между A- и B-вершинами цикла (так что есть минимум два варианта биекции). То же самое верно для бесконечной в обе стороны цепочки (два варианта). Если же цепочка бесконечна только в одну сторону, то для построения биекции годится только одно из отображений. Скажем, если она начинается с элемента $a \in A$, то годится только функция f (при которой a соответствует $f(a)$, затем $g(f(a))$ соответствует $f(g(f(a)))$ и так далее). Но в любом случае одна из функций f и g годится, так что внутри каждой связной компоненты у нас есть биекция, и остаётся их объединить для всех связных компонент. □

В доказательстве теоремы Кантора–Бернштейна искомая биекция h строится только из тех пар, которые отвечают двум инъекциям. В результате множество A разбивается на части A_1 и A_2, а множество B разбивается на части B_1 и B_2, при этом f осуществляет биекцию между A_1 и B_1, а g осуществляет биекцию между B_2 и A_2.

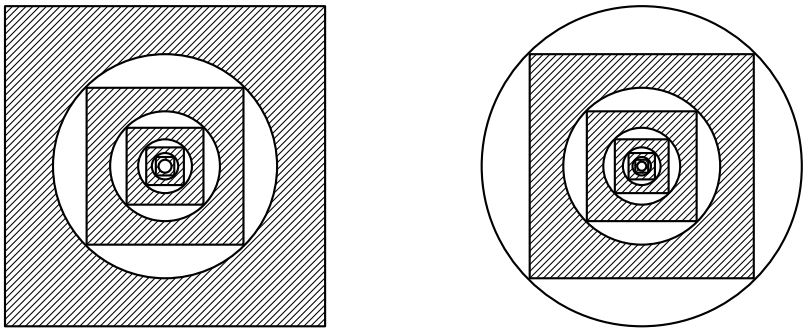
Это позволило предложить по существу теорему Кантора–Бернштейна в качестве задачи на одной из школьных олимпиад, сформулировав её так:

Теорема 8.12. *Квадрат и круг можно разбить на пары взаимно подобных частей.*

(Квадрат разбивается в объединение непересекающихся фигур A_1 и A_2, круг разбивается в объединение непересекающихся фигур B_1 и B_2, при этом A_1 подобно B_1, а A_2 подобно B_2.)

Доказательство. Возьмём преобразование подобия, которое отображает квадрат внутрь круга, оно будет инъекцией. (Бывает же внутри круга маленький квадрат!) Аналогично можно поместить круг в квадрат с помощью другого преобразования подобия. Остаётся применить доказательство теоремы Кантора–Бернштейна к этим двум инъекциям. □

Можно, конечно, сказать, что это нечестно — давать на олимпиаде задачу, использующую стандартный результат из «высшей математики». С другой стороны, можно решить эту задачу, не ссылаясь на Кантора и Бернштейна, а просто вдумчиво рассматривая картинку с чёрным квадратом — но не просто как у Малевича, а с кругом внутри.



(Две видеокамеры смотрят на круглый и квадратный экраны, показывая на них перекрёстно свои изображения).

Мы закончим наше знакомство с мощностями множеств ещё одним результатом Кантора. Теорему Кантора о несчётности отрезка можно интерпретировать как утверждение о том, что множество натуральных чисел по мощности меньше множества последовательностей нулей и единиц (в одну сторону биекция есть, поскольку в любом бесконечном множестве есть счётное подмножество, а в другую нет, потому что подмножество счётного множества конечно или счётно). Вместо последовательностей нулей и единиц можно рассматривать подмножества множества $\mathbb{N}$: каждая битовая последовательность $a_0a_1a_2\ldots$ соответствует множеству тех i, при которых $a_i = 1$. Таким образом, множество $\mathbb{N}$ имеет меньшую мощность, чем множество всех подмножеств $\mathbb{N}$.

Кантор заметил, что то же самое рассуждение проходит для любого множества (а не только для множества натуральных чисел).

Теорема 8.13 (Общая формулировка теоремы Кантора)**.** *Никакое множество X не равномощно множеству своих подмножеств.*

Доказательство. Предположим, что множество X и множество всех его подмножеств (оно обозначается 2^X) равномощны. Пусть f — биекция из X в 2^X. Рассмотрим те элементы x, которые не принадлежат соответствующим им подмножествам, то есть $x \notin f(x)$. Рассмотрим теперь множество всех таких элементов:

$$Y = \{x \in X \mid x \notin f(x)\}.$$

До противоречия осталось совсем немного: оказывается, что этому множеству Y не может соответствовать никакой элемент множества X. Действительно, пусть $Y = f(y)$ для некоторого $y \in X$. Тогда

$$y \in Y \Leftrightarrow y \notin f(y) \Leftrightarrow y \notin Y,$$

здесь первая равносильность верна по определению множества Y, а вторая – поскольку $f(y) = Y$. Мы получили противоречие, а значит такой биекции f не существует. □

Другими словами, мы строим множество Y, отличающееся от всех $f(y)$ — а именно, требуем, чтобы в точке y множество Y вело себя не так же, как $f(y)$ (ведь отличия в одной точке достаточно для различия множеств).

Заметим, что легко построить инъекцию из X в 2^X. Действительно, положим $f(x) = \{x\}$ для всякого $x \in X$. Таким образом, можно сказать, что мощность множества X всегда меньше мощности множества 2^X.

Задача 8.29. Докажите, что для всякого $n \in \mathbb{N}$ выполняется $n < 2^n$.

Задача 8.30. Покажите, что не существует сюръекции $X \to 2^X$ и не существует инъекции $2^X \to X$. (Это можно вывести из теоремы Кантора и теоремы Кантора–Бернштейна, а можно доказать, приспособив доказательство теоремы Кантора.)

Теорема 8.13 подводит нас к опасной черте, где в теории множеств начинаются парадоксы. Например, рассмотрим множество всех множеств U — его элементами являются произвольные множества. Оно выглядит как-то необозримо — мало ли какие бывают множества — и не зря: разрешив себе его рассматривать, мы сразу же придём к противоречию. Заметим, что все подмножества множества U (и вообще все подмножества любых множеств!) являются его элементами, то есть $2^U \subseteq U$. Но по теореме Кантора получается, что мощность 2^U больше U, то есть мы приходим к противоречию.

Если попытаться раскрутить это рассуждение, вспомнив доказательство теоремы Кантора, получится «парадокс Рассела»: если x — множество всех таких множеств, которые не являются своими элементами, то будет ли x своим элементом? Любой ответ приводит к противоречию: по построению

$$y \in x \Leftrightarrow y \notin y,$$

при любом y, в том числе при $y = x$, так что

$$x \in x \Leftrightarrow x \notin x.$$

Получается какая-то ерунда.

8.5 Что дальше?

Теория множеств, особенно множеств большой мощности — вещь довольно абстрактная, и изучает не пойми что. Если при некотором усилии можно убедить себя, что натуральные числа встречаются «в жизни» (хотя, конечно, очень большое натуральное число каких-нибудь фотонов и во вселенной-то может не поместиться), а с ещё большей натяжкой можно считать, что и действительные числа тоже бывают, то про какое-то нибудь множество всех подмножеств множества всех подмножеств действительных чисел это уже совсем малоубедительно.

Поэтому не так удивительно, что один из самых первый вопросов про множества, поставленный Кантором (есть ли множество промежуточной мощности между счётными множествами и континуумом, то есть бывает ли подмножество отрезка, которое не равномощно всему отрезку, но и не счётно), оказался неразрешимым. Сначала знаменитый логик Курт Гёдель в середине XX века доказал, что существование такого подмножества нельзя опровергнуть (пользуясь принятыми аксиомами теории множеств), а ещё через несколько десятилетий американский математик Пол Коэн доказал, что и доказать его (с помощью тех же аксиом) нельзя. Хочется спросить, «так что же на самом деле» — но вопрос этот сомнительный (примерно с тем же основанием можно было бы спрашивать, осталась ли на самом деле «век верна» пушкинская Татьяна своему нелюбимому мужу — из текста романа не следует ни того, ни другого, а никакого иного «самого дела» тут вроде как нет).

Информатика, конечно, интересуется в основном конечными объектами, и даже не очень большими (для неё 2^{1000} уже почти что бесконечность), так что теория множеств с её большими множествами вроде как тут не по делу. Тем не менее мы немного в ней поразбирались не зря — во-первых, жизнь не исчерпывается информатикой, и это просто забавно, во-вторых, тренировка в логических рассуждениях с использованием языка теории множеств в любом случае не помешает; наконец, многие идеи, возникшие в теории множества (например, диагональный аргумент) оказываются полезными и в других задачах, о чём мы уже упоминали.

Лекция 9

Упорядоченные множества

9.1 Отношения порядка

«Коля бутее, чем Вася, а Вася бутее, чем Таня. Кто бутее всех?» — такую задачу предлагал дошкольникам А. К. Звонкин на математическом кружке [9, с. 130]. И они, хотя и озадаченно, но отвечали, что самый бутой — это Коля, и только потом интересовались, что же значит «бутее».

Видимо, дело тут в том, что слово «бутее» воспринимается как прилагательное в сравнительной форме, и это подразумевает его свойства: если Коля бутее Васи, а Вася бутее Тани, то Коля бутее Тани (транзитивность). Эти подразумеваемые свойства можно формализовать.

9.1.1 Отношения строгого частичного порядка

Говорят, что бинарное отношение R является *строгим частичным порядком*, если выполнены такие свойства:

- если $R(a, b)$ и $R(b, c)$, то $R(a, c)$ (*транзитивность*);
- $R(a, a)$ всегда ложно (*антирефлексивность*).

Из этих свойств следует *антисимметричность*: $R(a, b)$ и $R(b, a)$ не могут выполняться одновременно. В самом деле, тогда по транзитивности (взяв $c = a$) получаем $R(a, a)$, что противоречит антирефлексивности.

Различные сравнительные прилагательные («меньше» для чисел, «ниже» для зданий, «дешевле» для товаров и пр.) дают много примеров таких отношений.

Перечисляя какие-то объекты списком, мы тем самым упорядочиваем их отношением «стоять раньше в списке»: скажем, для алфавита (множества букв языка) таким образом задаётся алфавитный порядок.[1]

[1] Таким же образом задаётся отношение протокольного старшинства — согласно www.protocolrf.ru/order.html, по состоянию на октябрь 2014 года имеются 44 категории, к которым, видимо, следует добавить подразумеваемую последнюю — «все остальные».

Для наглядности, говоря об отношениях строгого частичного порядка, часто используют обозначение $a < b$ вместо $R(a, b)$: тогда свойства транзитивности и антирефлексивности приобретают знакомый вид:

$$(a < b) \text{ и } (b < c) \text{ влечет } a < c;$$
$$a \not< a.$$

Но, конечно, от того, что мы обозначили какое-то отношение знаком $<$, оно не приобретает других свойств обычного сравнения чисел. Скажем, мы не требуем, чтобы любые два различных элемента a, b были сравнимы: может оказаться так, что не выполнено ни $a < b$, ни $b < a$. Скажем, если $a < b$ понимается так, что товар a одновременно легче и дешевле товара b, то может оказаться, что один товар легче, а другой дешевле, и тогда они не сравнимы друг с другом.

9.1.2 Строгие и нестрогие порядки

Прежде чем мы рассмотрим несколько примеров частично упорядоченных множеств, сделаем одно техническое замечание. Для чисел, помимо отношения «меньше», используется отношение «меньше или равно», определяемое (в соответствии с названием) как

$$a \leqslant b \Leftrightarrow (a < b) \text{ или } (a = b)$$

Аналогично можно поступить и с любом отношением строгого частичного порядка и получить другое отношение на том же множестве.

Лемма 9.1. *Полученное отношение обладает такими свойствами:*

- $a \leqslant a$ (*рефлексивность*)
- $(a \leqslant b)$ *и* $(b \leqslant a) \Rightarrow (a = b)$ (*антисимметричность*)
- $(a \leqslant b)$ *и* $(b \leqslant c) \Rightarrow (a \leqslant c)$ (*транзитивность*)

Доказательство. Рефлексивность прямо следует из определения. Чтобы доказать антисимметричность, предположим, что $a \leqslant b$, $b \leqslant a$, но $a \neq b$. Тогда по построению должно быть $a < b$ и $b < a$, что, как мы видели, невозможно. Транзитивность доказываем разбором случаев: если $a = b$ или $b = c$, то это очевидно, а если $a < b$ и $b < c$, применяем транзитивность для исходного отношения строгого частичного порядка. □

Возможен и обратный переход:

Лемма 9.2. *Если есть отношение* $\leqslant$*, удовлетворяющее трём указанным в лемме* 9.1 *требованиям, то полученное из него отношение*

$$a < b \Leftrightarrow (a \leqslant b) \text{ и } (a \neq b)$$

является отношением строгого частичного порядка.

Доказательство. Антирефлексивность ясна по построению. Надо проверить транзитивность: пусть $a < b$ и $b < c$, то есть (согласно определению строгого порядка) $a \leqslant b$, $a \neq b$, $b \leqslant c$, $b \neq c$. Надо получить $a \leqslant c$ и $a \neq c$. Первое сразу следует из транзитивности отношения $\leqslant$. Докажем второе: если $a = c$, то получаем $a \leqslant b$ и $b \leqslant a$, откуда по антисимметричности $a = b$ в противоречии с предположением. □

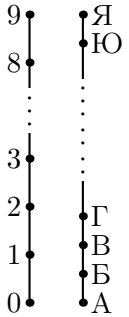
Эти две леммы показывают, что от строгого порядка можно перейти к нестрогому, добавив в отношение пары $\langle a, a\rangle$ (которых там нет по антирефлексивности). Наоборот: от нестрогого можно перейти к строгому, убрав такие пары (которые все там есть по рефлексивности). Так что по существу мы говорим об одном и том же, только немного по-разному. Обычно удобнее говорить о нестрогих порядках, и мы приходим к такому определению:

Бинарное отношение называется *отношением частичного порядка*, если оно обладает свойствами рефлексивности, антисимметричности и транзитивности. *Частично упорядоченным множеством* называется множество с отношением частичного порядка. Если пара $\langle a, b\rangle$ принадлежит этому отношению, то говорят, что *a меньше или равно b*, а также говорят, что *b больше или равно a*. Если при этом $a \neq b$, то говорят, что *a меньше b*, а также говорят, что *b больше a*. (Разумеется, смысл слов «больше» и «меньше» зависит от того, какой порядок выбран на множестве. На одном и том же множестве можно задать много частичных порядков, и для каждого из них эти слова имеют свой смысл.)

Замечание 9.1. Для обычного сравнения чисел вместо «меньше или равно» часто говорят «не больше» — это одно и то же. Но для произвольных частично упорядоченных множеств это не так. Если два различных элемента a и b не сравнимы друг с другом, то утверждение «a больше b» неверно, так что можно с полным правом сказать «a не больше b», понимая это буквально. Но сказать, что a меньше или равно b, при этом нельзя (a и не меньше b, и не равно b). Так что тут надо быть аккуратным.

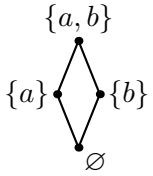
9.2 Примеры

Можно взять какое-то множество чисел, скажем, цифры $\{0, 1, 2, \ldots, 8, 9\}$ и рассмотреть на них обычное отношение порядка. Изобразим это упорядоченное множество на картинке, изображая меньшие элементы ниже бóльших:



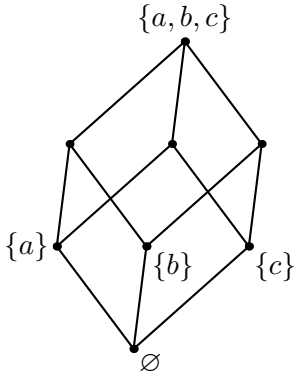
Аналогичным образом можно изобразить множество букв в алфавитном порядке.

Более сложный пример: рассмотрим множество $\{a, b\}$ и все его подмножества. Их четыре: $\varnothing$, $\{a\}$, $\{b\}$, $\{a, b\}$. Отношение порядка — это включение: $X \leqslant Y$, если X — подмножество множества Y. (Соответственно, $X < Y$, если X — собственное подмножество множества Y.) В этом порядке есть наименьший элемент $\varnothing$, наибольший элемент $\{a, b\}$ и два элемента $\{a\}$ и $\{b\}$ между ними. Эти два промежуточных элемента не сравнимы друг с другом (ни одно из множеств $\{a\}$ и $\{b\}$ не является подмножеством другого).



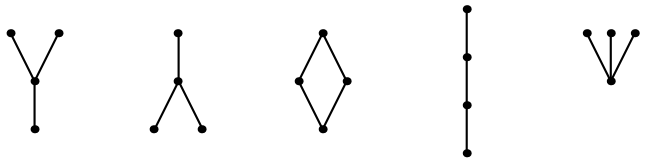
На картинке точки изображают элементы множеств, а линии отображают сравнимость: $x \leqslant y$, если из точки x можно пройти по линиям в y, идя всё время вверх.

Пример 9.1. Рассмотрим все подмножества трёхэлементного множества $\{a, b, c\}$ (их восемь) и тоже упорядочим их по включению ($X \leqslant Y$, если X — подмножество Y). На рисунке показаны некоторые из подмножеств.



Какие подмножества изображают остальные точки рисунка?

Другой пример: пусть множество состоит из каких-то целых положительных чисел, а $x \leqslant y$ понимается как «x является делителем y». Каждое число является своим делителем (с частным 1) — рефлексивность. Если a делится на b, и одновременно b делится на a, то эти числа равны — антисимметричность. Наконец, если a делит b и b делит c, то числа b/a и c/b — целые, и их произведение, равное c/a, тоже целое, то есть a делит c. Так что получаем частично упорядоченное множество. Его тоже можно изобразить на картинке.

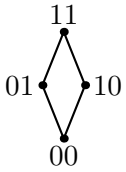


Задача 9.2. На рисунках показаны множества

$$\{1,2,4,8\},\quad \{1,2,3,6\},\quad \{1,2,3,5\},\quad \{2,3,6,12\}\quad \text{и}\quad \{1,2,4,6\}$$

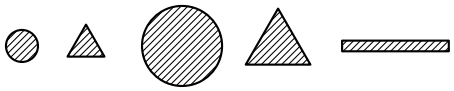
с отношением порядка «x является делителем y». Где какое множество?

Пример 9.3. Рассмотрим все двоичные слова длины n. Получится множество из 2^n элементов. Введём на нём отношение порядка, считая, что $x \leqslant y$, если y можно получить из x, заменив некоторые нули на единицы. (Другими словами, $x_1x_2\ldots x_n \leqslant y_1y_2\ldots y_n$, если $x_i \leqslant y_i$ для всех $i = 1, 2, \ldots, n$.) При $n = 2$ полученное множество изображено на рисунке.



Сделайте аналогичный рисунок для $n = 1$ и $n = 3$.

Задача 9.4. На рисунке показаны пять фигур.



Введём на этом множестве из 5 элементов отношение порядка, считая, что $X \leqslant Y$, если фигуру X помещается внутрь фигуры Y. Как изобразить это упорядоченное множество точками и линиями?

Как мы уже говорили, на данном множестве M можно определить много разных частичных порядков. Мы знаем, что любой частичный порядок должен включать в себя все пары $\langle m, m\rangle$ при всех $m \in M$ (рефлексивность). Можно ничего больше и не добавлять: получится порядок, в котором любые два разных элемента не сравнимы друг с другом. Ничего интересного — но формально это частичный порядок, в нём минимально возможное количество пар (k штук для множества M из k элементов).

Задача 9.5. А какое *максимальное* число пар может быть в отношении частичного порядка на множестве из k элементов?

9.3 Операции над частично упорядоченными множествами

Мы уже видели несколько примеров частично упорядоченных множеств. Сейчас мы опишем операции, которые позволяют строить новые частично упорядоченные множества из имеющихся.

Для краткости мы дальше называем порядком пару $(P, \leqslant)$ из множества и отношения частичного порядка на нем.

Если любые два элемента порядка сравнимы ($x \leqslant y$ или $y \leqslant x$), такой порядок называется *линейным* (хотя по аналогии с функциями его нужно бы называть тотальным или всюду определенным, но терминология уже устоялась).

Пример 9.3 демонстрирует операцию *покоординатного произведения порядков*. Пусть $(P, \leqslant_P)$ и $(Q, \leqslant_Q)$ — некоторые порядки. Покоординатный порядок на $P \times Q$ задаётся правилом:

$$(p_1, q_1) \leqslant (p_2, q_2) \text{ по определению означает } p_1 \leqslant_P p_2 \text{ и } \; q_1 \leqslant_Q q_2.$$

Пример 9.6 (покоординатный порядок на словах)**.** Напомним, что слова — это конечные последовательности элементов некоторого множества A (которое называется алфавитом, а его элементы — символами).

Во многих случаях алфавит упорядочен и даже линейно упорядочен (числа, буквы латиницы или кириллицы). В этом случае слова одинаковой длины сравниваются покоординатно.

Покоординатное произведение линейных порядков не обязательно линейно. Поэтому используется ещё одно определение произведения: лексикографическое. Лексикографический порядок на $P \times Q$ задаётся правилом:

$$(p_1, q_1) \leqslant (p_2, q_2) \text{ по определению означает, что } (p_1 <_P p_2) \text{ или } \; (p_1 = p_2) \text{ и } (q_1 \leqslant_Q q_2).$$

Лексикографическое произведение линейных порядков линейно. Дальше мы по умолчанию предполагаем лексикографический порядок на $P \times Q$.

Лексикографическое произведение определено не только для декартовых степеней P^n, но и для «бесконечной декартовой степени» $P^{\mathbb{N}}$, то есть на множестве бесконечных последовательностей элементов частичного порядка P.

Пример 9.7 (лексикографический порядок на всех словах)**.** Лексикографическое произведение задаёт линейный порядок на словах одинаковой длины.

А как сравнивать слова разной длины? Общепринятый способ, применяемый в словарях, таков: если слово u является началом слова w, то $u \leqslant w$. Если ни одно из слов u, v не является началом другого, то найдётся позиция, в которой эти слова различаются. Тогда меньше то слово, в котором на этой позиции стоит меньший символ.

То же самое правило получится, если добавить к алфавиту ещё один символ — пробел — и считать его наименьшим символом в алфавите. Слову в исходном алфавите сопоставим бесконечную последовательность в новом алфавите, которая начинается с символов слова и продолжается пробелами.

Легко видеть, что описанное выше правило задаётся лексикографическим сравнением полученных бесконечных последовательностей. Поэтому указанный порядок на всем множестве слов также называют лексикографическим.

Мы уже видели, что произведение порядков можно определять по-разному. То же самое и с суммой. Сумма порядков определяется для порядков, заданных на непересекающихся множествах. Если нужно сложить два порядка, множества которых пересекаются, то нужно изготовить копию одного из порядков на элементах, не входящих в множество второго порядка и лишь потом их складывать.

Самый осторожный способ определить сумму порядков — это больше ничего не делать. (Пары элементов из разных порядков несравнимы.) Недостаток этого способа в том, что он не сохраняет свойство линейности порядка.

Второй способ состоит в том, чтобы объявить все элементы первого порядка меньшими элементов второго порядка. При таком определении сумма линейных порядков является линейным порядком.

Пример 9.8. Рассмотрим сумму $\mathbb{N} + \mathbb{N}$. Её элементами будут обычные натуральные числа и их «копии». Копию числа n обозначим через n'. Тогда полученный линейный порядок выглядит так:

$$0 < 1 < 2 < \cdots < n < \cdots < 0' < 1' < \ldots\ .$$

Есть ещё один общий способ получить из одного порядка другой: ограничение на подмножество. Если множество X является подмножеством частичного порядка P, то на X определено *индуцированное* отношение частичного порядка: x меньше или равно y в индуцированном отношении, если x меньше или равно y в порядке P. Так мы получаем, например, порядок на каждом подмножестве действительных чисел, ограничивая общий порядок сравнения чисел.

9.4 Какие порядки считать «одинаковыми»?

Рассмотрим два порядка: порядок на подмножествах n-элементного множества по включению и покоординатный порядок на двоичных словах длины n. Это два разных отношения, заданные на разных множествах. Но по сути они одинаковы. Вспомним, что между подмножествами множества $\{1, \ldots, n\}$ и двоичными словами длины n есть биекция: подмножеству S сопоставляется слово x_S, в котором на i-й позиции стоит 1 тогда и только тогда, когда $i \in S$.

Эта биекция «уважает» порядок: если $A \subseteq B$, то $x_A \leqslant x_B$ в отношении покоординатного порядка. Справедливо и обратное: из $x_A \leqslant x_B$ следует $A \subseteq B$.

Поэтому естественно рассматривать два таких порядка как разные записи одного и того же. Для точных рассуждений используется специальный термин «изоморфизм».

Определение 9.3. Порядки P и Q называются *изоморфными*, если есть такая биекция $\varphi\colon P \to Q$, что $x \leqslant y$ равносильно $\varphi(x) \leqslant \varphi(y)$ для всех пар x, y.

Чтобы установить изоморфизм порядков достаточно указать такую биекцию. А как обосновать неизоморфность порядков? Разбирать все мыслимые биекции? Это затруднительно даже для конечных порядков.

Есть другой способ — указать различающее порядки *инвариантное* свойство. Если в формулировке свойства используются только сравнения пар элементов, то такое свойство должно сохраняться при изоморфизме.

Одним из простейших инвариантных свойств является наличие *минимального* элемента: такого элемента, что нет меньшего его. Аналогично определяется *максимальный* элемент.

В некоторых порядках минимальные элементы есть (скажем, порядок, индуцированный сравнением чисел на отрезке $[0,1]$). В других минимальных элементов нет (индуцированный на интервале $(0,1)$ порядок). Поэтому порядок на отрезке неизоморфен порядку на интервале.

Есть близкие понятия *наибольшего* и *наименьшего* элементов, которые нужно не путать с максимальными и минимальным элементами. Наименьший элемент меньше всех других элементов в порядке, наибольший — больше всех других.

Для линейных порядков эти понятия совпадают. В общем случае это не так.

Пример 9.9. Рассмотрим два единичных квадрата на координатной плоскости $\mathbb{R}^2$:

$$P = \{(x,y) : 0 \leqslant x \leqslant 1,\ 0 \leqslant y \leqslant 1\}, \qquad Q = \{(x,y) : |x| + |y| \leqslant \frac{1}{\sqrt{2}}\}$$

и порядки на них, индуцированные покоординатным порядком на $\mathbb{R}^2$.

В порядке P есть наименьший элемент $(0,0)$, он же единственный минимальный. В порядке Q наименьшего элемента нет, а минимальных — бесконечно много, весь отрезок между вершинами $(-1/\sqrt{2},0)$ и $(0,-1/\sqrt{2})$.

На порядки обобщается определение отрезка числовой прямой. Пусть $x \leqslant y$, тогда отрезком с концами x, y называется множество

$$[x,y] = \{z : x \leqslant z \leqslant y\}$$

и порядок, индуцированный на этом множестве.

Изоморфизм порядков должен порождать изоморфизм отрезков. В частности, отрезку из конечного числа элементов должен соответствовать отрезок из конечного числа элементов.

Поэтому $\mathbb{Z}$ неизоморфен $\mathbb{Q}$: в первом случае все отрезки конечны, а во втором — все бесконечны.

Порядок $\mathbb{Z}$ неизоморфен (линейному) порядку $\mathbb{Z}+\mathbb{Z}$: во втором случае есть бесконечные отрезки, скажем, $[0,0']$.

Два элемента x, y называют соседними, если между ними нет других элементов (то есть ни один элемент z не удовлетворяет сразу двум неравенствам $x < z < y$). Отрезок с концами в соседних элементах состоит в точности из своих концов.

Если соседних элементов нет, то порядок называется *плотным*. Примеры плотных порядков — $\mathbb{Q}$, $\mathbb{R}$.

Поскольку между любыми двумя элементами плотного порядка есть ещё хотя бы один, любой отрезок плотного порядка бесконечен.

9.5 Конечные линейные порядки

Проще всего разобраться с конечными линейными порядками.

Лемма 9.4. *В конечном линейном порядке есть наибольший и наименьший элементы.*

Доказательство. Рассмотрим *убывающие цепи*: последовательности элементов порядка $x_1 > x_2 > \ldots$, в которой каждый следующий элемент меньше предыдущего. Будем дополнять убывающую цепь новыми элементами, пока это возможно. Поскольку всего элементов конечное число, процесс рано или поздно остановится. Последний элемент такой убывающей цепи обязан быть наименьшим: в противном случае её можно было бы продолжить.

Аналогичное рассуждение с *возрастающими цепями* показывает существование наибольшего элемента. □

Теорема 9.5. *Все конечные линейные порядки с одинаковым числом элементов изоморфны.*

Доказательство. Индукция по числу элементов. База — один элемент в порядке — очевидна.

Индуктивный переход. Предположим, что все линейные порядки с n элементами изоморфны. Рассмотрим два линейных порядка P и Q с $n+1$ элементом. Выделим в них наименьшие элементы p_0, q_0. Порядки на оставшихся элементах изоморфны по предположению индукции. Продолжая этот изоморфизм соответствием $p_0 \mapsto q_0$, получаем искомый изоморфизм порядков P и Q. □

9.6 Порядки и индукция

Принцип математической индукции можно переформулировать таким образом:

> **Математическая индукция «без базы».** *Пусть для утверждения $A(n)$, зависящего от натурального параметра n, для любого n верно утверждение «если $A(m)$ верно при всех $m < n$, то и $A(n)$ верно». Тогда утверждение $A(n)$ верно при любом n.*

Отсутствие базы индукции тут мнимое. База скрыта в более сложном индуктивном предположении. Действительно, для $n = 0$ посылка условного индуктивного предположения всегда истинна (так как нет натуральных чисел, меньших n). Поэтому $A(0)$ обязано быть истинным.

В формулировке принципа математической индукции используется порядок на натуральных числах. Разберёмся, какие свойства этого порядка существенны для справедливости принципа математической индукции.

Теорема 9.6. *Следующие свойства порядка P равносильны:*

1. *каждое непустое подмножество имеет минимальный элемент;*
2. *любая убывающая цепь конечна;*
3. *для порядка P справедлив принцип индукции: если для утверждения $A(p)$, зависящего от элемента порядка, для любого p верно утверждение «если $A(q)$ верно при всех $q < p$, то и $A(p)$ верно». Тогда утверждение $A(p)$ верно при любом $p \in P$.*

Доказательство. Из 1 следует 2. Это равносильно тому, что из отрицания 2 следует отрицание 1. Возьмём бесконечную убывающую цепь $y_1 > y_2 > \ldots$. По определению в ней нет минимального элемента.

Из 2 следует 1 будем доказывать аналогично, проверив, что из отрицания 1 следует отрицание 2. Возьмём множество X без минимальных элементов и построим бесконечную убывающую цепь. Выберем какой-нибудь $x_0 \in X$. Он не минимален, поэтому есть $x_1 < x_0$. Аналогично рассуждаем с x_1 и так далее. Получаем бесконечную убывающую цепь.

Теперь выведем принцип индукции для P из существования минимальных элементов. Рассмотрим множество тех x, для которых не выполняется $A(x)$. Если оно непусто, в нем есть минимальный элемент m. Но тогда для всех $y < m$ утверждение $A(y)$ верно и в силу предположения индукции $A(m)$ тоже верно. Получили противоречие, так как по выбору m утверждение $A(m)$ ложно. Значит, имеет место единственная оставшаяся возможность: множество тех x, для которых не выполняется $A(x)$, пусто.

Осталось сделать ещё одну проверку: что из принципа индукции следует существование минимальных элементов в непустых множествах. Предположим, что множество X не имеет минимальных элементов. Возьмём в качестве утверждения $A(p)$ такое: $p \notin X$.

Индуктивное предположение выполняется: если для всех $q < p$ выполняется $q \notin X$, то и $p \notin X$ (иначе p — минимальный элемент). Поэтому $p \notin X$ для всех p, то есть $X = \varnothing$. □

Определение 9.7. Порядок, удовлетворяющий условиям теоремы 9.6, называется *фундированным*. Множество с фундированным порядком для краткости также называется фундированным.

Приведём примеры фундированных множеств, отличных от множества натуральных чисел.

Первый пример: лексикографическое произведение $\mathbb{N} \times \mathbb{N}$. В убывающей цепи пар натуральных чисел первые компоненты будут одинаковыми, начиная с некоторого места (так как $\mathbb{N}$ фундированное). Начиная с этого места, вторые компоненты образуют убывающую цепь в $\mathbb{N}$. Поэтому любая убывающая цепь в $\mathbb{N} \times \mathbb{N}$ конечна.

Точно такое же рассуждение показывает, что лексикографическое произведение любых фундированных множеств фундировано. Поэтому всякое $\mathbb{N}^d$ фундировано.

Покажем как использовать индукцию по фундированному множеству на примере известной олимпиадной задачи.

Задача 9.10. Купец заключил сделку с чёртом: каждый день он отдаёт черту одну монету и получает взамен любое количество монет меньшего достоинства. Брать монеты из других источников купцу запрещается. Когда монет не останется, купец проигрывает черту (душу).

Докажите, что черт выигрывает при любых действиях купца.

Состояние купца в любой момент времени выражается набором натуральных чисел

$$(n_1, n_2, \ldots, n_d),$$

где d — количество видов монет, n_1 — количество монет наибольшего номинала, n_2 — количество монет следующего номинала и т.д.

Тогда из условия задачи сразу следует, что состояние купца каждый день уменьшается в лексикографическом порядке. Поскольку бесконечных убывающих цепей нет, купец рано или поздно останется без монет.

9.7 Антицепи

Мы уже пользовались понятием цепи. Цепь — это линейный порядок. *Антицепь* — это множество попарно несравнимых элементов.

Неформально говоря, цепи и антицепи отвечают за «высоту» и «ширину» порядка. Более строго это соотношение демонстрирует следующее утверждение.

Задача 9.11. В конечном порядке на $mn + 1$ элементах есть либо конечная цепь размера $n + 1$, либо конечная антицепь размера $m + 1$.

Решение. Множество максимальных элементов порядка образует антицепь.

Построим такое разбиение элементов некоторого конечного порядка P_1. Первое множество разбиения M_1 — это максимальные элементы P_1. Отбросив эти элементы, получим на остальных элементах индуцированный порядок $P_2 = P_1 \setminus M_1$. Множество максимальных элементов этого порядка обозначим M_2. Продолжая этот процесс, получаем разбиение $P_1 = M_1 \cup M_2 \cup \cdots \cup M_h$, где

$$P_{i+1} = P_i \setminus M_i, \quad M_i \text{ — максимальные элементы порядка } P_i.$$

Заметим, что по определению для любого немаксимального элемента x в любом порядке найдётся максимальный элемент y, который больше x. Это позволяет построить возрастающую цепь $y_h < y_{h-1} < \cdots < y_1$, в которой $y_i \in M_i$, то есть из каждого множества разбиения взято по одному элементу. Для этого начнём с любого $y_h \in M_h$. Найдём для него максимальный в порядке P_{h-1} элемент y_{h-1}, который больше y_h, и так далее.

Осталось сделать простое арифметическое наблюдение: если в порядке на N элементах размер любой антицепи не превосходит s, то для построенного разбиения

должно выполняться неравенство $N \leqslant sh$, то есть в этом порядке есть цепь длины $\geqslant N/s$. □

Задача 9.12. Докажите, что в любом бесконечном порядке есть либо бесконечная цепь, либо бесконечная антицепь.

Есть более точная характеризация наибольшего размера антицепи в порядке через свойства цепей порядка.

Теорема 9.8 (Дилуорс)**.** *Наибольший размер антицепи в порядке равен наименьшему количеству цепей в разбиениях порядка на непересекающиеся цепи.*

Доказательство. В одну сторону очевидно: если порядок разбит на k непересекающихся цепей, то любая антицепь пересекается с каждой из цепей не более чем по одному элементу и в антицепи не больше k элементов.

В другую сторону сложнее. Индукция по числу элементов в порядке. База — порядок из одного элемента — очевидно выполнена.

Пусть утверждение теоремы справедливо для порядков с количеством элементов не больше n.

Рассмотрим порядок P, в котором $n+1$ элемент. В любом конечном порядке есть минимальные элементы. Пусть m — минимальный элемент в порядке P. Отбрасывая этот элемент, получаем порядок Q, для которого утверждение теоремы выполнено. Обозначим наибольший размер антицепи в Q через s. По предположению индукции порядок Q можно разбить на s непересекающихся цепей.

Наибольший размер антицепи в P либо равен s, либо равен $s+1$. В последнем случае m содержится в антицепи размера $s+1$ и порядок P легко разбивается на $s+1$ цепь: одна состоит только из элемента m, а остальные разбивают Q на s непересекающихся цепей.

Осталось рассмотреть случай, когда наибольший размер антицепи в P равен s. Элемент m тогда сравним с какими-то элементами порядка Q, а поскольку он минимальный, то он меньше каких-то элементов.

Рассмотрим в каждой цепи в Q минимальный элемент, входящий в антицепь максимального размера. Совокупность всех этих элементов образует антицепь M. Действительно, если для двух таких элементов $a < b$, то рассмотрим максимальную антицепь, в которой лежит a. Эта антицепь пересекается с цепью, в которой лежит b. В силу минимальности b по транзитивности получаем сравнимость двух элементов антицепи, что дает противоречие.

Один из элементов q построенной антицепи M сравним с m (иначе в P есть антицепь размера $s+1$). То есть, получается, что $m < q$, а все элементы, меньшие q в цепи C разбиения Q на s непересекающихся цепей, не входят в антицепи размера s.

Выделим цепь, состоящую из m и всех элементов цепи C, начиная с q и больше. Порядок на оставшихся элементах не содержит антицепей размера s (так как любая такая антицепь обязана пересекать остаток от цепи C) и в нем не больше n элементов. Значит, для этого порядка утверждение теоремы справедливо и его можно разбить на $s-1$ непересекающуюся цепь. Добавляя выделенную цепь, получаем разбиение исходного порядка P на s цепей.

Таким образом, утверждение теоремы справедливо и для порядка P. По принципу математической индукции теорема справедлива для всех конечных порядков. $\square$

С помощью теоремы Дилуорса легко решить задачу 9.11.

Лекция 10

Вероятность: первые шаги

Когда мы рассматриваем какой-то сложный процесс, то во многих ситуациях результат процесса не предопределён, но тем не менее возможны количественные утверждения о мере этой неопределённости. В этих ситуациях и возникает понятие «вероятности», точнее его математическая модель.

Простейшей моделью вероятности является модель с равновозможными исходами. В этой модели у процесса есть несколько возможных результатов процесса (*исходов*) и все они считаются одинаково возможными. Например, если мы подбрасываем монетку, то можно считать, что «орёл» и «решка» равновозможны. Аналогично, если мы бросаем шестигранный кубик, то любая из граней может оказаться верхней, как говорят, *с равной вероятностью*. *Событием* в этой модели называется некоторое подмножество множества возможных исходов. Мы считаем эти исходы благоприятными и интересуемся вероятностью этого события, то есть шансами того, что это событие произойдёт. Вероятностью события называется отношение числа благоприятных исходов к числу всех исходов или, другими словами, доля числа благоприятных среди всевозможных исходов.

Например, если мы подбрасываем монетку и интересуемся событием «выпадет орёл», то благоприятный исход один, а всего возможных исхода два. Таким образом вероятность выпадения орла равна $1/2$. Если же мы бросаем кубик, на гранях которого написаны числа от 1 до 6, и хотим посчитать вероятность того, что выпавшее число делится на 3, то благоприятными исходами будет выпадение 3 и 6, а всего исходов 6. Так что вероятность рассматриваемого события равна $1/3$.

Интуитивно равновозможность можно трактовать так: если повторять один и тот же эксперимент много раз, то все исходы будут встречаться примерно одинаковое число раз. И тогда доля благоприятных исходов среди всех будет приблизительно равна вероятности интересующего нас события.

Более общий вариант – неравновероятная модель, когда вероятности разных исходов различны.

Аккуратная формализация интуиции, стоящей за понятием вероятности, дело не простое. Теория вероятностей как математическая дисциплина и её продолжение — математическая статистика — предоставляют язык, на котором такая формализация

возможна. Мы здесь ограничимся *элементарной теорией вероятностей*, которую можно рассматривать как естественное обобщение перечислительной комбинаторики.

10.1 Элементарная теория вероятностей: определения

Перейдем к формальным определениям. *Вероятностным пространством* называется конечное множество U, его элементы называются *возможными исходами*. На вероятностном пространстве задана функция $\Pr\colon U \to [0,1]$, такая что $\sum_{x\in U} \Pr[x] = 1$. Функция Pr называется *вероятностным распределением*, а число $\Pr[x]$ называется *вероятностью исхода* $x \in U$. *Событием* называется произвольное подмножество $A \subseteq U$. Исходы, входящие в событие A, называются *благоприятными* (для события A). Вероятностью события A называется число $\Pr[A] = \sum_{x\in A} Pr[x]$.

В *модели с равновозможными исходами* функция Pr задается формулой $\Pr[x] = 1/|U|$ для всякого $x \in U$ (такое распределение называют также *равномерным*). Тогда вероятность события A равна $\Pr[A] = |A|/|U|$.

Приведём несколько примеров равномерных распределений на разных множествах и вычисления вероятностей событий. Фактически, это задачи перечислительной комбинаторики. Мы также указываем названия, которые обычно для них используются. Важно понимать, что все эти «монетки» и «кости» — условные названия, которые приняты для наглядности. Насколько подбрасывания реальной монеты соответствуют математической модели, — трудный вопрос и мы его здесь не обсуждаем.

Пример 10.1 («Подбрасывание монеты»)**.** В случае подбрасывания монетки исходом является выпадение одной из граней, орла или решки. Всего исходов, таким образом, два, и они считаются равновозможными. На самом деле, гораздо удобнее говорить не о сторонах монетки, а вместо этого ввести какое-нибудь более удобное обозначение для исходов. Чаще всего стороны монеты отождествляют с числами 0 и 1 и говорят об этих числах как об исходах. Мы будем пользоваться как раз такими обозначениями.

Итак, вероятностное пространство: числа 0 и 1. Все исходы равновозможны.

Пример 10.2 («Подбрасывание 6 монет»)**.** Вероятностное пространство: двоичные последовательности длины 6. Все исходы равновозможны.

Пример события, то есть множества в этом пространстве: ровно три элемента последовательности равны 1 (на неформальном жаргоне говорят «выпало три орла»). Чтобы найти вероятность этого события, нужно подсчитать количество исходов в нём и поделить на общее количество исходов.

Обе задачи легко решаются средствами перечислительной комбинаторики. Общее количество двоичных последовательностей длины 6 равно 2^6. Количество последовательностей, в которых ровно три единицы, равно $\binom{6}{3}$ (нужно выбрать из множества 6 позиций 3-элементное подмножество — те позиции, на которых стоят единицы).

Поэтому вероятность события равна

$$\frac{\binom{6}{3}}{2^6} = \frac{20}{64} = \frac{5}{16}.$$

Пример 10.3 («Подбрасывание n монет»)**.** Вероятностное пространство: двоичные слова длины n. Все слова равновозможны. Какова вероятность события «на i-й позиции в слове стоит 1»?

Всего исходов 2^n (количество двоичных слов длины n). Интересующее нас событие содержит 2^{n-1} исходов: каждый такой исход задаётся выбором 0 или 1 для всех позиций кроме i-й. Вероятность этого события равна по определению $2^{n-1}/2^n = 1/2$, как и вероятность дополнительного события «на i-й позиции в слове стоит 0».

Пример 10.4 («Подбрасывание двух игральных костей»)**.** Вероятностное пространство: последовательности (x_1, x_2) длины 2, состоящие из целых чисел в диапазоне от 1 до 6. Все исходы равновозможны. Нужно найти вероятность события «сумма выпавших чисел равна 7».

Общее количество исходов $6 \cdot 6 = 36$. Исходов, отвечающих указанному событию, ровно 6: если на первом месте в благоприятном исходе стоит число i, то на втором обязано стоять число $7 - i$. Поэтому вероятность равна $6/36 = 1/6$.

Пример 10.5 («Подбрасывание трёх игральных костей»)**.** Вероятностное пространство: последовательности (x_1, x_2, x_3) длины 3, состоящие из целых чисел в диапазоне от 1 до 6. Все исходы равновозможны.

Найдём вероятность события «сумма чисел в последовательности чётна».

Общее количество исходов $6 \cdot 6 \cdot 6 = 216$. Если на первых двух костях выпали числа i, j, то в благоприятном исходе есть три возможности для числа на третьей кости (три чётных числа, если сумма $i + j$ чётна, три нечётных числа, если $i + j$ нечётна).

Общее количество благоприятных исходов $6 \cdot 6 \cdot 3$, поэтому вероятность равна $6 \cdot 6 \cdot 3/6 \cdot 6 \cdot 6 = 1/2$.

Пример 10.6 («Случайная перестановка»)**.** Вероятностное пространство: перестановки $(a_1, a_2, \ldots, a_n)$ чисел от 1 до n. Все исходы равновозможны.

Проверим, что вероятность события «$a_2 > a_1 > a_3$» равна $1/6$.

Общее количество исходов (т.е. перестановок n элементов) равно $n!$. Разобьём их на непересекающиеся группы так, что в каждой группе на местах с 4-го по n-е стоят одни и те же числа.

В каждой из построенных групп $3! = 6$ перестановок (столькими способами можно разместить 3 оставшихся числа на первых трёх местах). Из этих 6 перестановок ровно одна является благоприятным исходом. Поэтому вероятность события равна $1/6$.

В общем случае (т.е., в не равновероятной модели) каждому исходу приписана некоторая вероятность, но теперь вероятности уже не равны: некоторые исходы вероятнее других. Можно понимать это так, что теперь у каждого исхода есть вес, и

у более вероятных исходов вес больше. В этом случае мы тоже могли бы сказать, что для подсчёта вероятностей событий нужно сложить веса благоприятных исходов и поделить на сумму весов всех исходов. Но чтобы не приходилось каждый раз делить, удобно считать, что сумма весов всех исходов равна 1. В частности, рассматривая одноэлементные события, мы получаем, что вес каждого исхода — это и есть его вероятность.

Пример 10.7. Пусть вероятностное пространство состоит из двоичных слов длины n и известно, что для каждого i вероятности событий «на i-й позиции в слове стоит 0», «на i-й позиции в слове стоит 1» равны $1/2$. Из этого не следует, что все слова равновозможны.

Действительно, рассмотрим такое вероятностное распределение на этом пространстве: вероятности слов из одних нулей и из одних единиц равны $1/2$, а вероятности остальных слов равны 0. Легко видеть, что вероятности указанных выше событий также равны $1/2$.

Пример 10.8 («Подбрасывание двух одинаковых монеток»)**.** Пусть мы подбрасываем две внешне очень похожие монетки и у нас нет цели различать, на какой что выпало. Все, что нас интересует, какая пара значений выпала на монетках, два орла, две решки или орёл и решка.

Тогда нам естественно рассматривать вероятностное пространство как раз с тремя такими исходами. Отличие от рассматривавшейся ранее ситуации с подбрасыванием нескольких монеток, состоит в том, что мы два исхода, когда орел выпал на одной из монет, а решка на другой, и когда наоборот, решка выпала на первой монете, а орёл на второй, объединили в один. Таким образом, вероятности исходов «два орла» и «две решки» получаются по $1/4$, а вероятность исхода «орёл и решка» в два раза больше и равна $1/2$.

Этот пример — простейший случай общей ситуации отображения вероятностных пространств. У нас есть вероятностное пространство (в предыдущем примере это подбрасывание двух монет, частный случай примера 10.3) и отображение множества его исходов X (в примере их четыре) в другое множество Y (в примере это 3-элементное множество «два орла», «две решки», «орёл и решка»). Вероятность исхода $y \in Y$ «наследуется» из X: она равна сумме вероятностей по исходам из прообраза y.

Пример 10.9 («Подбрасывание 5 одинаковых костей»)**.** Рассмотрим более сложный пример такого отображения. Его неформальный смысл состоит в том, что подбрасываются 5 одинаковых игральных костей. «Одинаковых» означает, что когда кости лежат на столе после броска, вы можете увидеть, какие очки выпали на каждой, но не можете их различить между собой.

Чтобы определить вероятностное пространство для этого примера, рассмотрим сначала вероятностное пространство «подбрасывание 5 (разных) костей»: последовательности $(x_1, x_2, x_3, x_4, x_5)$ длины 5, состоящие из целых чисел в диапазоне от 1 до 6. Все исходы равновозможны.

Для каждого исхода $(x_1, x_2, x_3, x_4, x_5)$ посчитаем количество единиц в этой последовательности, количество двоек и т.д. Получим тем самым всюду определённую функцию

$$f\colon \{1,2,3,4,5,6\} \to \mathbb{N},$$

которая возвращает по i количество вхождений числа i в последовательность. Сумма всех значений такой функции равна 5 (каждый элемент последовательности даёт вклад 1 в одно из значений суммирующей функции f).

Получили отображение исходов вероятностного пространства «подбрасывание 5 костей» в всюду определённые функции из $\{1,2,3,4,5,6\}$ в $\mathbb{N}$ с суммой значений 5. Как объяснялось выше, это отображение задаёт вероятностное пространство на множестве таких функций.

Чтобы найти вероятность функции f, нужно подсчитать количество последовательностей длины 5, члены которых — числа от 1 до 6, в которых $f(1)$ единиц, $f(2)$ двоек и т.д. Поскольку в изначальном вероятностном пространстве все исходы равновозможны, то это количество нужно умножить на 6^{-5} (вероятность исхода в изначальном пространстве).

Подсчёт делается аналогично формуле для биномиальных коэффициентов (аналогичные подсчёты уже проводились в главе 2): выберем множество $f(1)$ позиций, на которых стоят единицы, потом из оставшихся позиций выберем те, на которых стоят двойки и т.д. которых взяты $f(2)$ двоек и т.д. Получаем по правилу произведения

$$\binom{5}{f(1)} \cdot \binom{5-f(1)}{f(2)} \cdot \ldots = \frac{5!}{f(1)!(5-f(1))!} \cdot \frac{(5-f(1))!}{f(2)!(5-f(1)-f(2))!} \cdot \ldots$$

После сокращения в числителях и знаменателях останется

$$\frac{5!}{\prod_{i=1}^{6} f(i)!}.$$

Поэтому вероятность исхода f равна

$$\frac{5!}{6^5 \prod_{i=1}^{6} f(i)!}$$

(договоримся, что $0! = 1$).

Исходы в такой модели неравновозможны. Вероятность появления пяти единиц в 120 раз меньше вероятности появления 1, 2, 3, 4, 5 (сформулируйте это утверждение точно в терминах исходов указанного выше вероятностного пространства и проверьте результат по формуле для вероятностей).

Вероятностные пространство – это просто конечное множество с приписанными каждому элементу вероятностями. Но задавать такое множество можно по-разному. Некоторые способы задания могут оказаться существенно удобнее по сравнению с другими. Поскольку это оказывается важным при решении многих задач, мы поговорим об этом чуть подробнее.

Приведём один из основных примеров такого рода. Пусть вероятностное пространство состоит из перестановок чисел от 1 до n и все перестановки равновозможны. Оказывается, эту вероятностную модель можно задать другим способом. Выберем случайно и равновозможно первое число в перестановке. Затем второе число выберем случайно и равновозможно среди оставшихся и т.д.

Как задать исходы в модели последовательного случайного выбора? Каждый исход задаётся последовательностью $i_1, \ldots, i_n$, где i_k указывает на *порядковый номер* числа, которое мы выбираем в последовательности чисел, не использованных на предыдущих шагах. Все исходы равновозможны.

Такой процесс удобно изображать в виде дерева. На рисунке изображен пример для $n = 3$. Исходы в таком представлении — это листья дерева. На каждом ребре написан элемент соответствующей последовательности (каждое ребро входит ровно в один путь из корня в листья). Под каждым листом написана перестановка, которая отвечает этому исходу.

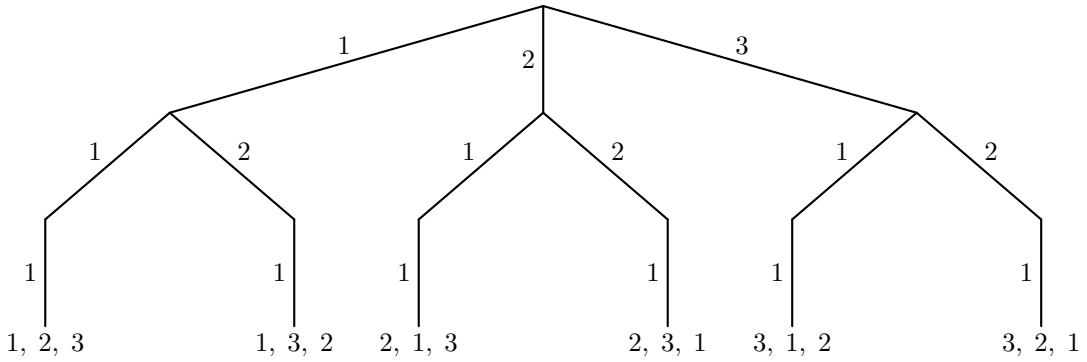


Рис. 10.1: Дерево порождения перестановок

Контрольный вопрос 10.10. Проверьте, что на рисунке 10.1 последовательность i_1, i_2, i_3 чисел на каждом пути из корня в лист соответствуют перестановке, написанной в листе.

Поскольку после $(k-1)$-го шага остаётся $n-k+1$ чисел, возможные последовательности номеров задаются условиями $1 \leqslant i_k \leqslant n-k+1$ для каждого k. Всего таких последовательностей $n \cdot (n-1) \cdot \ldots \cdot 1 = n!$, поэтому вероятность каждой перестановки в модели последовательного выбора равна $1/n!$, т.е. все перестановки равновозможны.

Аналогичным образом возможно задавать и другие вероятностные пространства. Например, пространство из примера 10.3 (двоичные строки длины n) задаётся полным бинарным деревом, вероятности на всех рёбрах равны $1/2$: на каждом шаге просто выбираем очередное число в последовательности равновероятно.

Заметим также, что вероятности выбора рёбер дерева, исходящих из данной вершины, вообще говоря, могут различаться. Чтобы получить распределение вероятностей на листьях дерева, достаточно удовлетворить двум условиям: (А) сумма ве-

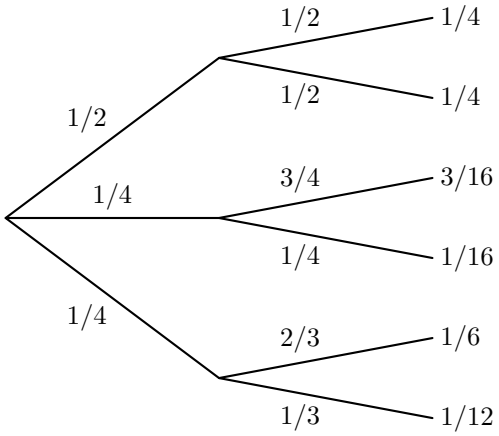


Рис. 10.2: Неравные вероятности выбора на каждом шаге

роятностей на всех рёбрах, исходящих из вершины, равна 1; (Б) вероятность листа равна произведению вероятностей на рёбрах пути, ведущего в этот лист.

На рис. 10.2 показан пример неравномерного распределения, удовлетворяющего этим условиям. Сумма вероятностей листьев (на рисунке написаны справа) равна 1, хотя это не так легко проверить сложением шести дробей. Удобнее провести проверку иначе, вынося общие множители:

$$\frac{1}{2}\cdot\frac{1}{2}+\frac{1}{2}\cdot\frac{1}{2}+\frac{1}{4}\cdot\frac{3}{4}+\frac{1}{4}\cdot\frac{1}{4}+\frac{1}{4}\cdot\frac{2}{3}+\frac{1}{4}\cdot\frac{1}{3}=$$
$$=\frac{1}{2}\cdot\left(\frac{1}{2}+\frac{1}{2}\right)+\frac{1}{4}\cdot\left(\frac{1}{4}+\frac{3}{4}\right)+\frac{1}{4}\cdot\left(\frac{2}{3}+\frac{1}{3}\right)=\frac{1}{2}\cdot 1+\frac{1}{4}\cdot 1+\frac{1}{4}\cdot 1=1$$

Задача 10.11. Докажите, что и в общем случае при соблюдении условий (А) и (Б) сумма вероятностей в листьях равна 1.

Очень часто задачи про вероятности формулируются как текстовые задачи: вместо явного указания вероятностного пространства и событий в нём, в задаче описывается некая условная ситуация, «как бы из жизни». В школьной математике такого рода задачи также активно используются: задачи про трубы и бассейны, про бригады полотёров или про велосипедистов, которые обгоняют идущие против течения катера, и т.п. Смысл в таких задачах простой: даже если не интересоваться приложениями (обычно такие задачи слишком условны, чтобы представлять интерес для приложений), решение таких задач помогает лучше представить стоящую за ними математику (в школьных примерах — уравнения и системы уравнений; в примерах про вероятности — понятия вероятностного пространства и события).

Решение таких текстовых задач про вероятности следует всегда начинать с явной формулировки вероятностной модели и событий, о которых идёт речь в задаче. Приведём пару примеров.

Задача 10.12. Десять учеников сдают экзамен по десяти билетам. Ученики по очереди заходят в кабинет и вытягивают случайный билет из оставшихся (в частности, последний берет единственный оставшийся билет). Вася выучил только один билет.

Какова вероятность, что Васе достанется билет, который он знает, если **a)** Вася тянет билет первым? **b)** Вася тянет билет последним?

Решение. В пункте (a) ясно, что Вася вытягивает случайно и равновозможно один из 10 билетов (вероятностное пространство: множество билетов, все исходы равновозможны). Поэтому вероятность вытянуть благоприятный билет 1/10.

Оказывается, в пункте (b) вероятность такая же. Чтобы ввести вероятностное пространство, занумеруем студентов числами от 1 до 10 в порядке очереди. Билеты также занумеруем числами от 1 до 10, номер 10 присвоим тому билету, который Вася выучил.

Тогда исходами будут перестановки чисел от 1 до 10. Все исходы равновозможны: процесс последовательного выбора билетов как раз и представляется в виде дерева последовательного случайного выбора.

Событие, вероятность которого нас интересует, состоит в том, что на 10-м месте стоит билет номер 10 (Вася идёт последним и знает только билет №10.)

Всего исходов 10!, а благоприятных — 9!. Искомая вероятность равна $9!/10! = 1/10$. □

Наметим другое решение задачи: все события «билет №10 стоит на месте i» имеют равные вероятности из симметрии. В пункте (a) мы уже нашли вероятность события «билет №10 стоит на месте 1». Поэтому в пункте (b) ответ такой же.

Конечно, наблюдение о симметрии задачи нужно как-то обосновать и сделать это зачастую не проще, чем решить задачу другим способом. Приведём поучительный пример.

Задача 10.13. В самолёт по очереди заходят 100 пассажиров. Первый садится на случайное место. Каждый следующий садится на своё место, если оно свободно, и на случайное свободное место, если его место занято. Какова вероятность того, что последний пассажир сядет на своё место?

Решение. Начнём с простого, короткого, но неформального решения задачи.

Какие места могут быть свободными перед посадкой последнего пассажира? Это либо его место, либо место первого пассажира. Действительно, если кто-то уже занял место первого пассажира раньше, то оставшиеся после этого пассажиры смогут сесть согласно купленным билетам, свободными останутся ровно их места. А значит, если место первого пассажира перед заходом последнего уже занято, то место последнего свободно и он на него и сядет.

Получается, что все исходы принадлежат ровно одному из двух событий: «последний пассажир сел на своё место» и «последний пассажир сел на место первого пассажира». Значит, в сумме вероятности этих событий дают 1.

Кроме того, вероятности этих событий одинаковы: если первый и последний пассажиры обменяются билетами, это не изменит рассадку, так как действия первого

и последнего пассажиров не зависят от номера билета. Но события при этом переставляются: исходы, которые были в первом событии, попадают теперь во второе, и наоборот.

Итак, вероятности двух указанных событий равны и в сумме дают 1. Поэтому каждая из них равна $1/2$.

Обратите внимание, что это неформальное решение нарушает наше правило: начинать решение текстовой задачи с явной формулировки вероятностной модели и событий, о которых идёт речь в задаче. К тому же, это решение подозрительно напоминает фольклорное рассуждение: «Какова вероятность встретить на улице динозавра? Она равна $1/2$ — либо встретишь, либо нет». Ошибки в неформальных решениях вероятностных задач — обычное дело, в социальных сетях можно легко найти примеры бурных обсуждений таких неправильных решений.

Тем не менее, в отличие от динозавров, решение задачи про посадку в самолёт можно сделать корректным. Ниже приводится такое корректное решение. Заметим, что задачу можно решить и многими другими способами (придумайте парочку), аккуратная запись некоторых существенно короче. Цель этой задачи состоит в том, чтобы показать, как можно превращать неформальные рассуждения в строгие.

Начнём с определения исходов. Занумеруем пассажиров в порядке очереди числами от 1 до 100. Исходом будет рассадка пассажиров по местам, то есть некоторая перестановка чисел от 1 до 100. Вероятности исходов неодинаковы. Их можно определить процессом последовательного случайного выбора: первый пассажир выбирает позицию в перестановке согласно равномерному распределению; i-й по очереди пассажир выбирает либо своё место, если оно свободно, с вероятностью 1 (вероятности остальных вариантов при этом равны 0), в противном случае он выбирает одно из свободных мест согласно равномерному распределению на них.

Таким образом, мы снова получаем представление нашего вероятностного пространства в виде дерева. Чтобы определить вероятность исхода, нужно перемножить вероятности для всех выборов на пути из корня в соответствующий лист.

Полученное распределение не равномерное. Например, любая перестановка, в которой первый пассажир сидит на своём месте, а какой-то пассажир — не на своём, имеет вероятность 0.

Заметим также, что вероятности перестановок-исходов зависят от раздачи билетов, то есть соответствия между пассажирами и местами. Обозначим через π это соответствие: у первого пассажира билет на место $\pi(1)$, у второго — на место $\pi(2)$ и т.д.

Мы фактически определили не одно распределение, а $100!$ распределений на одном и том же вероятностном пространстве. Одно такое распределение получается из другого перенумерацией мест, то есть перестановкой исходов. Кажется расточительным использовать $100!$ в сущности одинаковых распределений, но это помогает в объяснении трюка с симметрией.

Обозначим через G_π событие «последний пассажир сел на своё место», а через B_π — его дополнение, то есть «последний пассажир сел на место первого». Как мы

уже говорили, каждый исход попадает в одно из этих событий, поэтому

$$\Pr_\pi[G_\pi] + \Pr_\pi[B_\pi] = 1.$$

Индекс π указывает на распределение, задаваемое раздачей билетов π.

Нетрудно видеть, что вероятности интересующих нас событий не зависят от раздачи билетов, то есть $\Pr_\pi[G_\pi] = \Pr_\sigma[G_\sigma]$ для любых π, σ: при перенумерации мест перестановки, в которых последний сидит на своём месте, переходят в точности в перестановки, в которых последний сидит на своём месте.

Рассмотрим две раздачи билетов

$$\pi = (1,\ 2\ , \ldots, 99,\ 100), \quad \sigma = (100,\ 2\ , \ldots, 99,\ 1)$$

(первый и последний обменялись билетами).

Для любой рассадки α выполняется равенство

$$\Pr_\pi[\alpha] = \Pr_\sigma[\alpha].$$

Действительно, при вычислении вероятности исхода α все числа на пути из корня дерева случайного выбора в лист α одинаковы в обоих случаях. Первое равно $1/100$ (так как первый пассажир выбирает одно из 100 мест согласно равномерному распределению). Последнее равно 1 (так как у последнего нет выбора). Все промежуточные вероятности выборов равны, так как π и σ различаются только местами первого и последнего и это различие не меняет количества возможных выборов для i-го пассажира.

Однако интересующие нас события переставляются: $\alpha \in G_\pi$ (то есть $a_{100} = 100$) тогда и только тогда, когда $\alpha \in B_\sigma$ (ещё раз напомним, что последний пассажир садится либо на своё место, либо на место первого). Поэтому

$$\Pr_\pi[G_\pi] = \Pr_\sigma[B_\sigma].$$

Поскольку $\Pr_\sigma[G_\sigma] = \Pr_\pi[G_\pi]$, то получаем $\Pr_\sigma[G_\sigma] = \Pr_\sigma[B_\sigma]$, то есть $\Pr_\sigma[G_\sigma] = 1/2$. □

10.2 Вероятность объединения событий

События по определению являются множествами, поэтому для них определены все теоретико-множественные операции. В этом разделе мы рассмотрим объединения событий.

Лемма 10.1. *Если $A, B \subseteq U$, то $\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B]$. В частности, $\Pr[A \cup B] \leqslant \Pr[A] + \Pr[B]$ и, если $\Pr[A \cap B] = 0$, то $Pr[A \cup B] = \Pr[A] + \Pr[B]$.*

Доказательство. Действительно,

$$\Pr[A \cup B] = \sum_{x \in A \cup B} \Pr[x] = \sum_{x \in A} \Pr[x] + \sum_{x \in B} \Pr[x] - \sum_{x \in A \cap B} \Pr[x] = \Pr[A] + \Pr[B] - \Pr[A \cap B].$$

□

Следствие 10.2. *Для любых $A_1, \ldots, A_n \subseteq U$ верно*

$$\Pr[\bigcup_{i=1}^{n} A_i] \leqslant \sum_{i=1}^{n} \Pr[A_i],$$

а если множества A_i попарно не пересекаются, то неравенство обращается в равенство (это называется аддитивностью вероятностей для попарно несовместных событий).

Это следствие непосредственно вытекает из предыдущей леммы. Несмотря на простоту, это следствие оказывается весьма полезным и используется часто (в англоязычной литературе его обычно называют Union bound).

События A и B, которые не могут произойти одновременно, то есть для которых $\Pr[A \cap B] = 0$, называются *несовместными*. Таким образом вероятность объединения несовместных событий равна сумме их вероятностей.

Заметим, что если речь идёт о модели с равновозможными исходами, то вычисления и преобразования вероятности по существу мало отличаются от перечислительной комбинаторики. Нужно точно так же подсчитать нужное количество исходов, только в конце ещё разделить его на количество всех исходов.

В частности, как следствие принципа включений и исключений для множеств мы можем сразу получить принцип включений и исключений для вероятностей в равновозможной модели.

Следствие 10.3. *В равновозможной модели для произвольных множеств $A_1, \ldots, A_n \subseteq U$ верно*

$$\Pr[A_1 \cup A_2 \cup \cdots \cup A_n] = \sum_i \Pr[A_i] - \sum_{i<j} \Pr[A_i \cap A_j] + \cdots = \sum_{\varnothing \neq I \subset \{1,2,\ldots,n\}} (-1)^{|I|+1} \Pr\left[\bigcap_{i \in I} A_i\right]. \tag{10.1}$$

Доказательство. Действительно, по принципу включений и исключений для мощностей, аналогичное равенство верно для мощностей множеств. Чтобы получить равенство (10.1) для вероятностей, достаточно поделить обе части равенства на $|U|$. □

Оказывается, что на самом деле принцип включений и исключений верен не только для равновозможной модели, но и для любой другой.

Лемма 10.4. *Для всякой вероятностной модели и для произвольных множеств $A_1, \ldots, A_n \subseteq U$ верно*

$$\Pr[A_1 \cup A_2 \cup \cdots \cup A_n] = \sum_i \Pr[A_i] - \sum_{i<j} \Pr[A_i \cap A_j] + \cdots = \sum_{\varnothing \neq I \subset \{1,2,\ldots,n\}} (-1)^{|I|+1} \Pr\left[\bigcap_{i \in I} A_i\right]. \tag{10.2}$$

Доказательство. Здесь мы уже не можем просто сослаться на принцип включений и исключений для множеств. Однако, мы можем по существу повторить старое индуктивное доказательство принципа включений и исключений для множеств. База нами уже доказана выше.

Для доказательства шага индукции заметим, что

$$\begin{aligned}\Pr\left[(A_1 \cup \cdots \cup A_{n-1}) \cup A_n\right] &= \Pr\left[A_1 \cup \cdots \cup A_{n-1}\right] + \Pr[A_n] - \Pr\left[A_n \cap (A_1 \cup \cdots \cup A_n)\right] = \\ &= \Pr\left[A_1 \cup \cdots \cup A_{n-1}\right] + \Pr[A_n] - \Pr\left[(A_n \cap A_1) \cup \cdots \cup (A_n \cap A_{n-1})\right].\end{aligned}$$

Здесь мы воспользовались принципом включений и исключений для двух множеств. Осталось для каждого из объединений $A_1 \cup \cdots \cup A_{n-1}$, $(A_n \cap A_1) \cup \cdots \cup (A_n \cap A_{n-1})$ воспользоваться предположением индукции. □

В комбинаторике одним из стандартных приложений формулы включений и исключений является задача о количестве перестановок n различных объектов так, чтобы ни один не остался на своём месте. В терминах теории вероятности мы можем оценить долю таких перестановок.

Лемма 10.5 (Задача о беспорядках)**.** *Рассмотрим случайные перестановки n различных объектов, то есть рассмотрим вероятностное пространство всех перестановок n заданных объектов, причём все перестановки равновозможны. Пусть A_n — событие, означающее, что все объекты после перестановки оказались не на своих изначальных местах. Тогда $\lim_{n\to\infty} \Pr[A_n] = 1/e$.*

Доказательство. Зафиксируем n и обозначим через B_i, где $i = 1, \ldots, n$, событие «объект с номером i остался на месте». Тогда $B_1 \cup \ldots \cup B_n$ означает, что хотя бы один из элементов остался на месте. Дополнительное событие к этому – как раз событие A_n.

Применим формулу включений и исключений к событию $B_1 \cup \ldots \cup B_n$. Для этого нам нужно посчитать вероятности событий $\bigcap_{i\in I} B_i$ для всевозможных $I \subseteq [n]$. Но это сделать несложно. Подходящие перестановки – это в точности перестановки, оставляющие на месте элементы из I, и переставляющие остальные элементы произвольным образом. Таких перестановок $(n-|I|)!$. Таким образом, для всякого I

$$\Pr\left[\bigcap_{i\in I} B_i\right] = \frac{(n-|I|)!}{n!}.$$

Множеств I размера k всего $\binom{n}{k}$, так что по формуле включений и исключений мы получаем

$$\begin{aligned}\Pr\left[B_1 \cup \ldots \cup B_n\right] &= \binom{n}{1}\frac{(n-1)!}{n!} - \binom{n}{2}\frac{(n-2)!}{n!} + \binom{n}{3}\frac{(n-3)!}{n!} + \ldots + (-1)^{n+1}\binom{n}{n}\frac{1}{n!} \\ &= \frac{1}{1!} - \frac{1}{2!} + \frac{1}{3!} - \ldots + (-1)^{n+1}\frac{1}{n!}.\end{aligned}$$

Тогда для вероятности события A_n мы получаем формулу

$$\Pr[A_n] = 1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \ldots + (-1)^n \frac{1}{n!}.$$

Теперь мы сошлёмся на факт из математического анализа, а именно, что эта сумма совпадает с началом ряда Тейлора для функции e^x в точке $x = -1$. Более точно, воспользуемся тем, что $e^{-1} = \sum_{k=0}^{\infty}(-1)^k/k!$. Отсюда

$$\lim_{n\to\infty} \Pr[A_n] = 1/e.$$

□

10.3 Вероятностный метод

Мы уже готовы показать один из мощных методов, часто применяемых в комбинаторике и теоретической Computer Science, а именно, вероятностный метод. Этот метод позволяет доказывать существование объектов с заданными свойствами. Для этого выбирается случайный объект из некоторого семейства и показывается, что он удовлетворяет нужным свойствам с положительной вероятностью. Мы воспользуемся этим методом для нижней оценки чисел Рамсея. Напомним, что числом Рамсея $R(n,k)$ называется минимальное число вершин, что всякий граф на этих вершинах содержит клику размера n или независимое множество размера k. Мы видели, что числа Рамсея не слишком большие, а именно $R(n,k) \leqslant \binom{n+k-2}{k-1}$. Теперь мы покажем, что эти числа и не слишком маленькие.

Теорема 10.6. *Для всякого $k \geqslant 3$ верно $R(k,k) > 2^{(k-1)/2}$.*

Доказательство. Рассмотрим $n = 2^{(k-1)/2}$ вершин и рассмотрим на них случайный граф. То есть в качестве вероятностного пространства U мы рассматриваем все графы на этих вершинах, и мы приписываем каждому из них одинаковую вероятность. Можно сразу заметить, что всего таких графов $2^{\binom{n}{2}}$. Действительно, для каждой пары вершин есть два выбора: либо добавить эту пару в множество рёбер графа, либо нет. Всего пар вершин $\binom{n}{2}$, так что графов получается $2^{\binom{n}{2}}$.

Оценим вероятность того, что случайный граф содержит клику или независимое множество размера k. Обозначим это событие через A. Наша цель – показать, что эта вероятность меньше 1. Тогда мы получим, что существует граф без клики и независимого множества размера k. Чтобы оценить эту вероятность, разобьем событие A в объединение нескольких событий. Для этого для всякого подмножества $W \subseteq V$ множества вершин, такого что $|W| = k$ рассмотрим событие A_W, состоящее в том, что в случайном графе множество W образует клику или независимое множество. Нетрудно видеть, что

$$A = \bigcup_{W\subseteq V, |W|=k} A_W,$$

а значит

$$\Pr[A] \leqslant \sum_{W \subseteq V, |W|=k} \Pr[A_W].$$

Теперь оценим вероятность отдельного события A_W. Посчитаем количество графов, попадающих в это событие. Ребра между вершинами в W в таком графе должны либо все присутствовать, либо все отсутствовать. Ребра, хотя бы один конец которых лежит вне W, могут быть произвольными. Количество ребер, у которых хотя бы один конец лежит вне W, есть $\binom{n}{2} - \binom{k}{2}$ (все ребра минус ребра в W). Таким образом, количество таких графов есть $2 \cdot 2^{\binom{n}{2}-\binom{k}{2}}$, где первая двойка отвечает за выбор ребер внутри W, а второй множитель – за выбор остальных ребер. Тогда получается, что

$$\Pr[A_W] = \frac{2^{\binom{n}{2}-\binom{k}{2}+1}}{2^{\binom{n}{2}}} = 2^{-\binom{k}{2}+1}.$$

Таким образом, при $k \geqslant 3$

$$\Pr[A] \leqslant \sum_{W \subseteq V, |W|=k} 2^{-\binom{k}{2}+1} = \binom{n}{k} 2^{-\binom{k}{2}+1} \leqslant \frac{n^k}{2 \times 3} 2^{-\binom{k}{2}+1} =$$
$$= 2^{k(k-1)/2-\binom{k}{2}+1}/6 = 1/3.$$

Следовательно, вероятность дополнения события A положительна, а значит существует граф на N вершинах без клик и независимых множеств размера k. □

Заметим, что наше доказательство неконструктивно: мы не строим граф, в котором нет клики и независимого множества, мы только доказываем, что он существует.

Как мы уже говорили, наше доказательство использует вероятностный метод. Опишем его в общем виде более подробно.

Пусть есть некоторое конечное множество объектов U и нужно доказать, что среди них есть хотя бы один объект, обладающим некоторым свойством. Мы уже обсуждали, что с формальной точки зрения, свойство — это множества объектов, обладающих этим свойством. Итак, нас интересует, не пусто ли множество $A \subseteq U$, но по самому описанию множества, проверить это может оказаться сложно. Переформулируем задачу на языке теории вероятности, введя вероятностное распределение на U. Множество A не пусто тогда и только тогда, когда вероятность события A больше нуля! Казалось бы от этой переформулировки мы ничего не выигрываем, но теперь можно использовать методы теории вероятности, чтобы доказать, что вероятность события A положительна. Это и было сделано при доказательстве теоремы.

10.4 Условные вероятности

Помимо вероятностей тех или иных событий бывает нужным говорить и о вероятностях одних событий при условии других. Неформально говоря, мы хотим определить вероятность выполнения события A в том случае, когда событие B выполняется. В терминах вероятностного пространства определение этого понятия довольно

естественное: нужно сузить вероятностное пространство на множество B. Так, для равновозможной модели мы получаем, что вероятность A при условии B есть просто $|A \cap B|/|B|$, то есть число благоприятных исходов поделенное на число всех исходов (после сужения всего вероятностного пространства до B). В случае произвольного вероятностного пространства нужно учесть веса исходов, то есть нужно сложить вероятности исходов в $A \cap B$ и поделить на сумму вероятностей исходов в B.

Таким образом, мы приходим к формальному определению. *Условной вероятностью события A при условии B* называется число

$$\Pr[A|B] = \frac{\Pr[A \cap B]}{\Pr[B]}.$$

Заметим, что условная вероятность имеет смысл, только если $\Pr[B] > 0$. Иначе знаменатель обращается в ноль.

Определение условной вероятности можно переписать следующим образом:

$$\Pr[A \cap B] = \Pr[B] \cdot \Pr[A|B].$$

Другими словами, чтобы найти вероятность пересечения событий A и B достаточно найти вероятность события B и условную вероятность события A при условии события B.

Задача 10.14. В классе 50% мальчиков; среди мальчиков 60% любят мороженое. Какова доля мальчиков, любящих мороженое, среди учеников класса? Как переформулировать этот вопрос в терминах вероятностей?

Задача 10.15. Приведите примеры, в которых условная вероятность $\Pr[A \mid B]$ больше вероятности $\Pr[A]$, меньше её, а также равна ей.

Понятие условной вероятности не всегда легко соотнести с интуицией, основанной на обыденной жизни. Тут особенно важно следовать сформулированному нами правилу, явно обозначать вероятностное пространство и распределение на нём, после чего пользоваться формальным определением условной вероятности. Приведем несколько примеров.

Задача 10.16. Есть три внешне одинаковых мешочка. В одном лежит две золотых монеты, во втором — одна золотая и одна серебряная монета, в третьем — две серебряные. Случайно и равновозможно выбирается один из мешочков, затем из него случайно и равновозможно достают монету.

Какова вероятность того, что выбран мешок с золотыми монетами при условии, что выбранная монета золотая?

Поначалу может показаться, что вероятность очевидно равна $1/2$: золотая монета есть в двух мешках, так что мы выбрали один из них, и при этом мешки равноправны. Однако, это не так. Давайте аккуратно разберемся почему.

Решение. Исходами в данном случае являются монеты и все шесть исходов равновозможны. (В данном случае опять имеет место ситуация последовательного случайного выбора и на каждом шаге выбора вероятности одинаковые. Нарисуйте дерево для этого выбора.)

Событие A =«выбранная монета золотая» состоит из 3 исходов. Событие B = «выбран мешок с двумя золотыми монетами» состоит из двух исходов, причём оба они содержатся в A (напомним ещё раз, что события — это множества, в данном случае — подмножества монет). Значит, $A \cap B = B$ и поэтому

$$\Pr[B \mid A] = \frac{\Pr[A \cap B]}{\Pr[A]} = \frac{2}{3}\,.$$

□

Задача 10.17. Есть девять коробок и один шарик. Шарик помещается в одну из коробок по следующему правилу. Сначала случайно и равновозможно выбирается коробка. Затем с вероятностью $1/2$ в неё кладётся шарик, а с вероятностью $1/2$ — нет. Найдите вероятность того, что в последней коробке шарик есть при условии, что в остальных коробках его нет.

Решение. Перенумеруем коробки числами от 1 до 9 и будем считать, что последняя коробка имеет номер 9. Какое в этой задаче вероятностное пространство? Исход — это пара (i, j), где i — номер выбранной коробки, а j равно 1 или 0 (пусть 1 означает, что шарик положили в коробку). Все исходы равновозможны, так как опять имеем дело с ситуацией последовательного случайного выбора и на каждом шаге выбора вероятности одинаковые.

Событие-условие «в коробках с номерами от 1 до 8 шарика нет» состоит из 10 исходов:

$$(i, 0),\ 1 \leqslant i \leqslant 8, \quad (9, 0),\ (9, 1)$$

(первые 8 из этих исходов означают, что выбрана коробка i и в неё не положили шарик; последние два означают, что выбрана коробка 9: тогда шарика в остальных коробках заведомо нет). Интересующее нас событие состоит из одного исхода $(9, 1)$ (выбрана коробка 9 и в неё положили шарик).

Это событие содержится в событии-условии, поэтому искомая условная вероятность равна $1/10$. □

Из формального определения условной вероятности можно получить на первый взгляд неожиданные утверждения.

Лемма 10.7 (формула Байеса)**.** *Если вероятность событий A и B положительна, то*

$$\Pr[A|B] = \Pr[A] \cdot \frac{\Pr[B|A]}{\Pr[B]}.$$

У этой леммы есть вполне конкретный практический смысл, который мы проиллюстрируем на условном примере. Рассмотрим некоторую болезнь, и предположим,

что для её обнаружения мы можем делать недорогой анализ, который при этом с заметной вероятностью выдаёт неправильный результат, а также есть дорогостоящее исследование, которое уже наверняка сообщает, болен ли человек. Мы хотим обнаруживать болезнь следующим образом: сначала человек сдаёт недорогой анализ, а если он дал положительный результат, то проверяем человека с помощью дорогостоящего исследования. При таком подходе возможно, что больной человек не будет обнаружен нашим анализом.Можем ли мы на основании статистических данных понять, как часто это происходит? На первый взгляд кажется, что этого сделать нельзя, ведь мы же не обнаруживаем этих больных. Но оказывается, что на самом деле это можно сделать при помощи формулы Байеса. Пусть B — событие «быть больным», а A — событие «получить положительный результат на анализе». Собрав статистику, мы можем узнать $\Pr[A]$ — вероятность для человека получить положительный анализ.[1] Можно также оценить $\Pr[B]$ — долю больных рассматриваемой болезнью. Наконец, можно оценить и $\Pr[B|A]$ — вероятность того, что человек болеет при условии, что он получил положительный результат анализа. (Опять же, мы можем собрать статистику.) Теперь по формуле Байеса мы можем посчитать $\Pr[A|B]$ — вероятность того, что больной человек будет обнаружен нашим анализом. А это как раз то, что мы хотели найти.

Доказательство леммы 10.7. Доказательство формулы Байеса почти очевидно. Достаточно просто записать вероятность события $A \cap B$ через условные вероятности двумя способами:

$$\Pr[A \cap B] = \Pr[B] \cdot \Pr[A|B] = \Pr[A] \cdot \Pr[B|A].$$

Теперь второе равенство сразу даёт формулу Байеса. □

Понятие условной вероятности позволяет нам также говорить о независимых событиях. Неформально, событие A не зависит от события B, если вероятность события A при условии события B такая же, как и вероятность A при условии не выполнения события B. Формально удобнее выбрать другое определение: событие A *не зависит от события* B, если

$$\Pr[A] = \Pr[A|B].$$

Чтобы не возникало никаких тонкостей с нулевыми вероятностями мы всюду, когда говорим о независимых событиях предполагаем, что вероятности событий A и B ненулевые.

Из определения условной вероятности мы сразу получаем эквивалентное определение независимости событий. Событие A не зависит от события B, если

$$\Pr[A \cap B] = \Pr[A] \cdot \Pr[B].$$

[1] Строго говоря, из статистических данных можно лишь получать более-менее достоверные оценки вероятностей. В нашем условном примере мы пренебрегаем этой трудной проблемой.

Из этой формы определения видно замечательное свойство независимости событий: она симметрична. То есть, событие A не зависит от события B тогда и только тогда, когда событие B не зависит от события A.

Несложными алгебраическими преобразованиями проверяются и другие свойства независимых событий. Приведём два наиболее важных, оставляя доказательства в качестве упражнения для читателя.

Задача 10.18. Пусть события A и B независимы, и вероятность события $\bar{B}$ положительна. Докажите, что события A и $\bar{B}$ независимы. Здесь и далее $\bar{B} = U \setminus B$ обозначает событие, дополнительное к B.

Задача 10.19. Докажите, что события A и B, для которых $0 < \Pr[B] < 1$, независимы тогда и только тогда, когда $\Pr[A \mid B] = \Pr[A \mid \bar{B}]$.

Приведём несколько примеров проверки независимости событий. Первый очень простой, но важный.

Пример 10.20 (продолжение примера 10.3)**.** В примере 10.3 мы проверили, что для равномерного распределения на двоичных словах длины n вероятности событий «на i-й позиции стоит α» равны $1/2$ (здесь $1 \leqslant i \leqslant n$ и $\alpha \in \{0, 1\}$).

Теперь проверим, что эти события при $i \neq j$ независимы. Действительно, есть 2^{n-2} слов, принимающих заданные значения α и β в позициях i и j. Поэтому вероятность пересечения любых двух таких событий равна $2^{n-2}/2^n = 1/4 = 1/2 \cdot 1/2$.

Точно так же проверяется и независимость более сложных событий вида «на позициях $i_1, \ldots, i_s$ стоят символы $\alpha_1, \ldots, \alpha_s$», если множества позиций не пересекаются.

Менее очевидные примеры связаны с событиями на случайных перестановках.

Пример 10.21. Выбирается случайно и равновозможно перестановка $x_1, x_2, \ldots, x_{49}$ чисел от 1 до 49.

(а) Независимы ли события «x_{24} больше всех последующих» и «x_{25} больше всех последующих»?

Как мы уже видели выше в примере 10.12, другой способ получить случайно и равновозможно перестановку состоит в том, чтобы выбирать каждое число x_i в перестановке случайно и равновозможно из оставшихся к этому моменту вариантов (то есть среди чисел, которые ещё не включены в последовательность). Поэтому вероятности событий «x_{24} больше всех последующих» и «x_{25} больше всех последующих» равны $1/26$ и $1/25$ соответственно (в первом случае нужно выбрать на 24-м шаге максимальное из оставшихся 26 чисел, во втором аналогично).

Вероятность пересечения этих событий совпадает с вероятностью выбрать на 24-м шаге максимальное из 26 оставшихся чисел, а на 25-м — максимальное из 25 чисел, оставшихся к этому шагу. Вероятность такого события равна

$$\frac{1}{26 \cdot 25}$$

(исходы — упорядоченные пары из 26 чисел, благоприятный исход один).

Поскольку вероятность пересечения событий равна произведению вероятностей событий, эти события независимы.

(б) Независимы ли события «$x_{24} > x_{25}$» и «$x_{25} > x_{26}$»?

Вероятности этих событий равны 1/2. Перестановок, для которых $x_{24} > x_{25}$, ровно столько, сколько перестановок, для которых $x_{24} < x_{25}$: меняя местами 24-е и 25-е числа, мы из перестановки, для которой $x_{24} > x_{25}$, получаем перестановку, для которой $x_{24} < x_{25}$.

Аналогично рассуждаем и для второго события.

Теперь найдём вероятность пересечения рассматриваемых событий. Это событие «$x_{24} > x_{25} > x_{26}$». Рассуждаем аналогично примеру 10.6. Разобьём перестановки на группы из шести перестановок, в каждой группе одна и та же тройка чисел занимает места с 24-го по 26-е в каком-то порядке, а на остальных местах числа расставлены одинаково.

В каждой шестёрке есть ровно одна перестановка из события «$x_{24} > x_{25} > x_{26}$». Поэтому вероятность этого события 1/6.

Так как

$$\frac{1}{2} \cdot \frac{1}{2} \neq \frac{1}{6}\,,$$

рассматриваемые события не являются независимыми.

Из этих вычислений заключаем, что вероятность события «$x_{25} > x_{26}$» при условии «$x_{24} > x_{25}$» равна $1/3 < 1/2$.

Это довольно удивительно с обыденной точки зрения. Представьте, что в лототроне изначально находятся 49 шаров, пронумерованных числами от 1 до 49. Шары выкатываются из лототрона по очереди. Предположим, что нам известно, что номер 24-го шара оказался больше номера 25-го. Каковы шансы, что номер 25-го шара будет больше номера 26-го? Мало кто из людей способен догадаться до точного значения 1/3, большинство предположит, что эта вероятность больше.

Другим полезным утверждением об условных вероятностях является формула полной вероятности.

Лемма 10.8 (Формула полной вероятности)**.** *Пусть $B_1, \ldots, B_n$ — разбиение вероятностного пространства U, то есть $U = B_1 \cup \ldots \cup B_n$, где $B_i \cap B_j = \varnothing$ при $i \neq j$. Пусть также $\Pr[B_i] > 0$ для всякого i. Тогда для всякого события A*

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A|B_i] \cdot \Pr[B_i].$$

Доказательство. Согласно следствию 10.2 (свойству аддитивности вероятности),

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A \cap B_i] = \sum_{i=1}^{n} \Pr[A|B_i] \cdot \Pr[B_i],$$

где первое равенство получается по формуле сложения вероятностей непересекающихся событий, а второе равенство – по определению условной вероятности. □

Приведём примеры использования формулы полной вероятности.

Пример 10.22 (Продолжение примера 10.21)**.** Выбирается случайно и равновозможно перестановка $x_1, x_2, \ldots, x_{49}$ чисел от 1 до 49.

Какова вероятность события «$x_{25} > x_{26}$» при условии «$x_{24} < x_{25}$»?

Обозначим эту условную вероятность p. По формуле полной вероятности

$$\begin{aligned}\Pr[\text{«}x_{25} > x_{26}\text{»}] = {} & \Pr[\text{«}x_{25} > x_{26}\text{»} \mid \text{«}x_{24} < x_{25}\text{»}] \cdot \Pr[\text{«}x_{24} < x_{25}\text{»}] + \\ & + \Pr[\text{«}x_{25} > x_{26}\text{»} \mid \text{«}x_{24} > x_{25}\text{»}] \cdot \Pr[\text{«}x_{24} > x_{25}\text{»}].\end{aligned}$$

В примере 10.21 мы уже посчитали все вероятности, кроме искомой. Из формулы полной вероятности получаем

$$\frac{1}{2} = p \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{2},$$

т.е. $p = 2/3$. Конечно, посчитать искомую вероятность можно и другим способом, аналогично примеру 10.21.

В приложениях очень часто формула условной вероятности применяется вместе с формулой Байеса. Приведём условный пример. Для краткости мы пропускаем точные определения вероятностного пространства и вероятностного распределения для этого примера. Читателю рекомендуется восстановить их самостоятельно в качестве упражнения.

Пример 10.23. Редакционную колонку в некотором издании каждый раз пишет один из журналистов X и Y. Журналист X пишет колонку в два раза чаще журналиста Y. Известно, что X допускает фактические ошибки в 25% статей, а Y — в 50% статей.

(а) С какой вероятностью в случайной редакционной колонке обнаружится фактическая ошибка? Предполагаем, что выбор колумниста и наличие фактических ошибок в статье происходят случайно.

По условию задачи события «колонку писал журналист X» и «колонку писал журналист Y» образуют разбиение всего множества исходов, причём вероятность первого события 2/3, а второго — 1/3. Вероятность фактической ошибки первого журналиста 1/4, а второго — 1/2. Получаем по формуле полной вероятности

$$\begin{aligned}&\Pr[\text{ошибка}] = \\ &= \Pr[\text{ошибка} \mid \text{писал } X] \cdot \Pr[\text{писал } X] + \Pr[\text{ошибка} \mid \text{писал } Y] \cdot \Pr[\text{писал } Y] = \\ &= \frac{1}{4} \cdot \frac{2}{3} + \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{3}\,.\end{aligned}$$

(б) В редакционной колонке обнаружена фактическая ошибка. С какой вероятностью её написал журналист X?

Формула Байеса даёт

$$\Pr[\text{писал } X \mid \text{ошибка}] = \Pr[\text{писал } X] \cdot \frac{\Pr[\text{ошибка} \mid \text{писал } X]}{\Pr[\text{ошибка}]} = \frac{2}{3} \cdot \frac{1/4}{1/3} = \frac{1}{2}\,.$$

Приведём ещё пример использования формулы полной вероятности в теории графов.

Пример 10.24. Пусть G — простой неориентированный граф с n вершинами $v_1, \ldots, v_n$, степени которых равны некоторому числу d (такой граф называется регулярным). Как мы уже доказывали, у него $nd/2$ рёбер. Рассмотрим два вероятностных распределения на его рёбрах. Первое — равномерное, то есть каждое из $nd/2$ рёбер выбирается с одинаковой вероятностью. А второе такое: сначала случайным образом выбирается вершина (каждая с одинаковой вероятностью), а затем случайно с равными вероятностями выбирается ребро, соседнее с этой вершиной. Если немного подумать, то интуитивно понятно, что эти два распределения одинаковые — каждое ребро выбирается в обоих распределениях с одной и той же вероятностью. Но как это аккуратно доказать?

Рассмотрим произвольное ребро e и событие A, означающее, что выбрано это ребро. Подсчитаем $\Pr[A]$ в каждом из распределений. В первом случае это несложно — у нас задано равновероятное распределение на $nd/2$ рёбрах, так что вероятность равна $2/nd$. Чтобы посчитать вероятность во втором случае для всякого $i \in \{1, \ldots, n\}$ рассмотрим событие B_i, состоящее в том, что была выбрана вершина v_i. Эти события образуют разбиение вероятностного пространства. Так что по формуле полной вероятности получаем

$$\Pr[A] = \sum_{i=1}^{n} \Pr[A|B_i] \cdot \Pr[B_i].$$

На вершинах у нас задано равновероятное распределение, так что для каждого i $\Pr[B_i] = 1/n$. Теперь подсчитаем условную вероятность $\Pr[A|B_i]$. Если вершина v_i не является концом ребра e, то ребро никак не может быть выбрано, так что условная вероятность в этом случае равна нулю. Если же v_i является концом ребра e, то вероятность, что мы выберем его равна $1/d$: мы равновероятно выбираем одно из d рёбер с концом в v_i. У ребра два конца, так что все слагаемые кроме двух равны 0, а каждое из двух оставшихся равны $(1/d)\cdot(1/n)$. Таким образом, вероятность события A в случае второго распределения также равна $2/dn$, а значит оба распределения совпадают.

Следующий пример использования формулы полной вероятности показывает как упрощать вероятностное пространство, используя соображения симметрии.

Пример 10.25. Из n-элементного множества выбираются случайно, равновозможно и независимо два k-элементных множества X и Y. Какова вероятность события «$X \cap Y = \varnothing$»?

Условие означает, что вероятностное пространство — пары (X, Y) k-элементных подмножеств n-элементного множества, вероятности всех исходов одинаковы.

Событие «$X \cap Y = \varnothing$» не изменяется при переобозначении элементов множества. Поэтому вероятности условных событий «$Y \cap X = \varnothing \mid X = A$» и «$Y \cap X = \varnothing \mid X = B$» одинаковы для любых A и B. Из формулы полной вероятности получаем, что для

любого множества A

$$\Pr[X \cap Y = \varnothing] = \Pr[Y \cap X = \varnothing \mid X = A].$$

Пусть $A = \{1, \ldots, k\}$. Вероятность выбрать k-элементное множество Y, которое не содержит ни одного элемента из A, равна

$$\frac{\binom{n-k}{k}}{\binom{n}{k}}$$

(в числителе стоит количество k-элементных подмножеств в дополнении к A, а в знаменателе — количество k-элементных подмножеств в n-элементном множестве).

Преобразуем это выражение:

$$\frac{\binom{n-k}{k}}{\binom{n}{k}} = \left(1 - \frac{k}{n}\right) \cdot \left(1 - \frac{k}{n-1}\right) \cdot \ldots \cdot \left(1 - \frac{k}{n-k+1}\right) \leqslant \left(1 - \frac{k}{n}\right)^k.$$

При больших n и $k \approx c\sqrt{n}$ последнее выражение оценивается как e^{-c^2} (здесь мы опускаем подробности).

Получаем неожиданный с точки зрения интуиции факт: весьма малые случайные подмножества большого конечного множества почти заведомо пересекаются. Более того, мы получили количественную оценку этой «малости»: достаточно того, чтобы $k/\sqrt{n} \to \infty$ при $n \to \infty$. Аналогичными рассуждениями можно также проверить, что случайные множества размера $o(\sqrt{n})$ почти заведомо не пересекаются.

10.5 Случайная величина, математическое ожидание

Случайная величина — это числовая функция на вероятностном пространстве, то есть функция вида $f\colon U \to \mathbb{R}$. То есть, по сути, случайная величина — это обычная числовая функция, но теперь на её аргументах задано вероятностное распределение. Таким образом, например, мы можем говорить о вероятности того, что случайная величина f равна какому-то конкретному значению a: это есть просто вероятность события $\{u \in U \mid f(u) = a\}$. Случайные величины представляют собой числовые характеристики вероятностных экспериментов и, на самом деле, мы с ними уже неоднократно сталкивались, просто не говорили об этом. Например, если мы бросаем кубик, то исходом эксперимента является выпадение той или иной грани, а случайной величиной — число написанное на грани (каждой грани соответствует своё число — это функция).

Важным параметром случайной величины является её математическое ожидание. Неформально, это число, которое мы будем получать в среднем, если повторять эксперимент много раз и каждый раз смотреть на значение случайной величины.

Более конкретно, пусть вероятностное пространство состоит из k исходов, случайная величина $f\colon U \to \mathbb{R}$ принимает на них значения $a_1, \ldots, a_k$ соответственно и вероятности исходов равны $p_1, \ldots, p_k$ соответственно. В частности, $\sum_{i=1}^{k} p_i = 1$.

Предположим, что выбор случайного элемента из U повторяется n раз. Если n достаточно большое, то случайная величина f примет значение a_1 примерно $p_1 n$ раз, значение a_2 — примерно $p_2 n$ раз, и так далее, значение a_k — примерно $p_k n$ раз. (Этому утверждению можно придать строгий смысл. Простейший случай разобран в разделе 10.7.) Подсчитаем теперь примерное среднее арифметическое значений случайной величины f в этих экспериментах:

$$\frac{a_1 p_1 n + a_2 p_2 n + \ldots + a_k p_k n}{n} = \sum_{i=1}^{k} a_i p_i.$$

Этот неформальный рассказ приводит нас к следующему строгому (и очень важному) определению.

Математическим ожиданием случайной величины $f\colon U \to \mathbb{R}$ называется число

$$E[f] = \sum_{u \in U} f(u) \Pr[u] = \sum_{i=1}^{k} a_i p_i.$$

Например, случайная величина, равная числу, выпадающему на грани кубика, принимает значения $1, 2, 3, 4, 5, 6$ с вероятностями $1/6$. Её математическое ожидание равно

$$1 \cdot (1/6) + 2 \cdot (1/6) + 3 \cdot (1/6) + 4 \cdot (1/6) + 5 \cdot (1/6) + 6 \cdot (1/6) = 21/6 = 3.5.$$

То есть, при бросании кубика мы будем в среднем получать число 3.5 .

Математическое ожидание с одной стороны является осмысленной характеристикой случайной величины, а с другой обладает свойствами, делающими работу с математическими ожиданиями удобной.

Лемма 10.9 (линейность математического ожидания)**.** *Пусть $f\colon U \to \mathbb{R}$ и $g\colon U \to \mathbb{R}$ — две случайные величины на одном и том же вероятностном пространстве. Тогда*

$$E[f + g] = E[f] + E[g].$$

Доказательство. Доказательство леммы несложно получить непосредственно из определения математического ожидания (собственно, из чего же ещё?).

Пусть вероятностное пространство U состоит из исходов $u_1, \ldots, u_k$ с вероятностями $p_1, \ldots, p_k$ соответственно. Тогда

$$E[f + g] = \sum_{i=1}^{k} (f(u_i) + g(u_i)) p_i = \sum_{i=1}^{k} (f(u_i)) p_i + \sum_{i=1}^{k} (g(u_i)) p_i = E[f] + E[g].$$

□

Линейность математического ожидания во многих случаях заметно упрощает вычисления.

Пример 10.26. Том Сойер красит забор, состоящий из 20 досок. Он красит первую доску, а затем, покрасив очередную доску, он с вероятностью 4/5 переходит к следующей доске, а с вероятностью 1/5 уходит купаться (и больше забор не красит). Нужно найти математическое ожидание количества покрашенных досок. (Обратите внимание, что по правилам хотя бы одну доску он покрасит.)

Приведём два решения этой задачи: первое опирается только на определение, а второе использует линейность математического ожидания.

Исходя из определения, мы получим, что математическое ожидание случайной величины f, определённой в задаче, равно

$$E[f] = \sum_{k=1}^{19} k \left(\frac{4}{5}\right)^{k-1} \left(\frac{1}{5}\right) + 20 \left(\frac{4}{5}\right)^{19}.$$

Представим процесс покраски в виде дерева (как на рисунке 10.2). Действительно, ровно $k \leqslant 19$ досок покрашены, если первые $k-1$ раз случался переход по рёбрам с вероятностью 4/5 и после этого был переход по ребру с вероятностью 1/5. Чтобы покрасить ровно 20 досок, нужно каждый раз идти по ребру с вероятностью 4/5.

Чтобы привести получившийся ответ к числу, воспользуемся формулой суммы геометрической прогрессии и математическим анализом:

$$\sum_{k=0}^{n-1} x^k = \frac{1-x^n}{1-x};$$

продифференцировав эту формулу, получаем нужную нам формулу

$$\sum_{k=1}^{n-1} k x^{k-1} = \frac{1-x^n}{(1-x)^2} - \frac{n x^{n-1}}{1-x}.$$

В нашем случае $x = 4/5$, $n = 20$. Заметим, что первое слагаемое математического ожидания отличается от полученной формулы коэффициентом $1/5 = 1 - x$. Домножив обе части формулы на $(1-x)$ и подставив числа, получим:

$$E[f] = \left(\frac{1-(4/5)^{20}}{1-(4/5)} - 20\left(\frac{4}{5}\right)^{19}\right) + 20\left(\frac{4}{5}\right)^{19} = 5 - 5\left(\frac{4}{5}\right)^{20}.$$

Приведём теперь решение, использующее линейность математического ожидания. Докажем, что случайная величина f представима в виде суммы случайных величин $f_1 + f_2 + \ldots + f_{20}$, в которой

$$f_k = \begin{cases} 1, & \text{если } k\text{-ая доска была покрашена;} \\ 0, & \text{иначе.} \end{cases}$$

Действительно, если в исходе u было покрашено ровно m досок, то $f_k(u) = 1$ при $k \leqslant m$ и $f_k(u) = 0$, при $k > m$, а потому $\sum f_k(u) = m = f(u)$: k-ая доска вносит единичный вклад в сумму, если и только если была покрашена.

Заметим, что математическое ожидание случайной величины f_k имеет вид

$$E[f_k] = 1 \times \Pr[A_k] + 0 \times \Pr[\bar{A}_k] = \Pr[A_k],$$

где A_k — событие, состоящее в покраске k-ой доски. Случайную величину χ, принимающую только значения 0 или 1, называют *индикаторной*, поскольку её математическое ожидание равно вероятности события, для которого χ является индикаторной (характеристической) функцией.

Итак, найдём математическое ожидание каждой случайной величины f_k. Заметим, что k-ая доска красится только вместе с первыми $k-1$ досками, что возможно, если и только если путь по дереву проходил по рёбрам с вероятностью $4/5$ первые $k-1$ раз. Отсюда $\Pr[A_k] = (4/5)^{k-1}$, а значит

$$E[f] = E[\sum_{k=1}^{20} f_k] = \sum_{k=1}^{20} E[f_k] = \sum_{k=1}^{20} (4/5)^{k-1} = \frac{1-(4/5)^{20}}{1-(4/5)} = 5 - 5\left(\frac{4}{5}\right)^{20}.$$

Здесь мы воспользовались линейностью математического ожидания и формулой суммы геометрической прогрессии и получили ответ, не прибегая к более сложному анализу.

Следующий пример показывает, как с помощью линейности математического ожидания решить довольно непростую, на первый взгляд, задачу.

Пример 10.27 (Задача о днях рождения)**.** Рассмотрим n случайных людей и посмотрим на количество совпадений дней рождения у них, то есть на количество пар людей, имеющих день рождения в один день. Каким в среднем будет это число?

Сформулируем вопрос точно. Вероятностное пространство: всюду определённая функция из n-элементного множества людей $\{x_1, \ldots, x_n\}$ в 365-элементное множество дней в году. Все исходы равновозможные.[2]

Обозначим случайную величину, равную количеству пар людей с совпадающими днями рождения, через f. Нам требуется посчитать математическое ожидание случайной величины f. Но при этом случайная величина довольно сложная, и подсчитывать математическое ожидание непосредственно из определения трудно.

Идея состоит в следующем: давайте разобьём сложную случайную величину f в сумму нескольких простых случайных величин. Тогда мы сможем подсчитать отдельно математические ожидания всех простых величин, а затем, пользуясь линейностью математического ожидания, просто сложить результаты.

Обозначим через g_{ij} случайную величину, равную 1, если у людей x_i и x_j дни рождения совпадают, и равную 0 в противном случае. Тогда можно заметить, что

$$f = \sum_{i<j} g_{ij}.$$

[2]Мы выбрали такой вариант для простоты — в реальной жизни нужно учесть существование високосных лет и неравномерное распределение дней рождения в году. Да и независимость дней рождения в реальной жизни неочевидна — вспомним о близнецах.

Подсчитаем математическое ожидание случайной величины g_{ij}. Нетрудно увидеть, что вероятность того, что у двух случайных людей дни рождения совпадают, равна $1/365$, так что с вероятностью $1/365$ случайная величина равна 1, и с вероятностью $1-1/365$ равна 0. Так что $E[g_{ij}] = 1/365$ (для всякой пары i, j). Для математического ожидания f из линейности получаем

$$E[f] = E[\sum_{i<j} g_{ij}] = \sum_{i<j} E[g_{ij}] = \sum_{i<j} 1/365 = \frac{n(n-1)}{2 \cdot 365}.$$

Например, если число людей n больше 27, то $E[f] > 1$, то есть естественно ожидать, что будет не меньше одного совпадения дней рождений, что может показаться противоречащим интуиции (поэтому эту задачу иногда называют «парадоксом дней рождения»).

На самом деле, зная немного анализа, можно убедиться, что если распределение дней рождения по дням года считать неравномерным (что ближе к реальности), то математическое ожидание только вырастет.

Пример 10.28. Выбирается случайное множество двоичных строк длины n. (Все подмножества множества $\{0,1\}^n$ равновероятны.)

(а) Чему равно математическое ожидание суммарного числа единиц в строках этого подмножества?

Обозначим суммарное количество единиц S. Для каждой двоичной строки w обозначим через S_w случайную величину, которая равна $|w|$, т.е. количеству 1 в строке w, если эта строка попала в случайное множество, и 0 в противном случае. Тогда

$$S = \sum_w S_w.$$

Поскольку все подмножества равновероятны, вероятность для каждого попасть в случайное множество равна $1/2$ (одинаково количество тех подмножеств, которые содержат w, и тех, которые не содержат).

Поэтому математическое ожидание S_w равно

$$E[S_w] = \frac{1}{2} \cdot |w| + \frac{1}{2} \cdot 0 = \frac{|w|}{2}.$$

Значит, $E[S]$ равно общему количеству единиц в двоичных строках длины n, делённому на 2. В каждом разряде во всех строках вместе ровно 2^{n-1} единиц (нулей и единиц в каждом разряде поровну). Поэтому общее количество единиц равно $n2^{n-1}$, откуда получаем $E[S] = n2^{n-2}$.

(б) Тот же вопрос, но выбирается случайное подмножество, в котором ровно k строк.

Рассматриваем аналогичные случайные величины S и S_w как в п. (а). Но теперь у нас пространство состоит из подмножеств мощности k.

Какова вероятность события «строка w попала в случайное k-элементное подмножество строк X»? Всего k-элементных подмножеств строк $\binom{2^n}{k}$. Из них $\binom{2^n-1}{k-1}$ содержат данную строку w. Вероятность равна отношению этих двух чисел.

Получаем

$$E[S_w] = |w| \frac{\binom{2^n-1}{k-1}}{\binom{2^n}{k}} = \frac{|w|k}{2^n} \ .$$

Отсюда, пользуясь вычислениями из п. (а), находим ответ

$$E[S] = \sum_w E[S_w] = \frac{k}{2^n} \sum_w |w| = \frac{k}{2^n} \cdot n2^{n-1} = \frac{kn}{2}.$$

С помощью математического ожидания можно обобщить вероятностный метод.

Лемма 10.10 («среднее не больше максимума и не меньше минимума»)**.** *Пусть для какой-то случайной величины $f\colon U \to \mathbb{R}$ верно $E[f] = C$. Тогда существует такой исход $u \in U$, что $f(u) \geqslant C$. Аналогично, существует и такой исход $u \in U$, что $f(u) \leqslant C$.*

Доказательство. Докажем первое утверждение леммы, второе доказывается аналогично.

На самом деле, доказательство довольно простое. Предположим, что утверждение неверно, а значит для всякого $u \in U$ верно $f(u) < C$.

Тогда

$$E[f] = \sum_{u \in U} \Pr[u] f(u) < \sum_{u \in U} \Pr[u] C = C,$$

противоречие. □

Как и в разделе 10.3, данная лемма позволяет доказать существование объекта с определённым свойством, но на этот раз, используя вычисление математического ожидания. Как и раньше, нужно доказать, что есть хотя бы один объект, обладающий свойством $A \subseteq U$. Но теперь описание множества A имеет специальный вид: $A = \{a \mid f(a) \geqslant C\}$ (или $A = \{a \mid f(a) \leqslant C\}$), где C — некоторое число. Подобрав подходящее вероятностное распределение на U и показав, что $E[f] \geqslant C$, мы получаем из леммы, что множество A не пусто.

Задача 10.29. Объясните, почему старая формулировка вероятностного метода является частным случаем новой формулировки.

Новая формулировка удобна в некоторых случаях. Разберём один такой пример.

Рассмотрим простой неориентированный граф $G = (V, E)$. *Разрезом* графа называется разбиение множества его вершин на два непересекающихся подмножества: $V = V_1 \cup V_2$, $V_1 \cap V_2 = \varnothing$. Мы говорим, что ребро попадает в разрез, если один его конец лежит в V_1, а другой в V_2. Размером разреза называется число рёбер, попадающих в разрез. Нас будут интересовать большие разрезы графа.

Теорема 10.11. *Всякий граф $G = (V, E)$ имеет разрез размера не меньше $|E|/2$.*

Доказательство. Рассмотрим случайный разрез графа G. Более точно, мы берём равномерное распределение на множестве всех разрезов. Разрез задается подмножеством $S \subseteq V$: такому подмножеству ставится в соответствие разрез $(S, V \setminus S)$. Всего подмножеств (а значит и разрезов) 2^n, так что вероятность каждого разреза есть $1/2^n$. Аналогично примеру 10.20 можно проверить, что для каждой пары вершин $x \neq y$ все четыре события «$x \in S, y \in S$», «$x \notin S, y \in S$», «$x \in S, y \notin S$», «$x \notin S, y \notin S$» имеют вероятность $1/4$.

Итак, рассмотрим случайный разрез и рассмотрим случайную величину f, равную размеру разреза. Посчитаем её математическое ожидание. Для этого, как и раньше, стоит разбить случайную величину в сумму более простых случайных величин. Для всякого $e \in E$ рассмотрим случайную величину f_e, равную 1, если ребро e входит в разрез, и равную 0 в противном случае. Тогда нетрудно видеть, что $f = \sum_{e \in E} f_e$, а значит

$$E[f] = \sum_{e \in E} E[f_e].$$

Однако, для случайной величины f_e математическое ожидание уже нетрудно посчитать. Действительно, для всякого фиксированного ребра e вероятность, что оно попадёт в разрез равна $1/2$. А значит, $E[f_e] = 1/2$ для всякого $e \in E$, откуда

$$E[f] = \sum_{e \in E} 1/2 = |E|/2.$$

Из этого следует, что есть конкретный разрез, содержащий не меньше $|E|/2$ рёбер. □

На самом деле, этот результат не сильно удивителен, его можно доказать и «руками».

Задача 10.30. Постройте алгоритм, работающий за полиномиальное время и строящий разрез размера не меньше $|E|/2$.

Оказывается, что если проводить вероятностное рассуждение аккуратнее, то можно получить чуть более сильную оценку на размер разреза.

Теорема 10.12. *Рассмотрим граф $G = (V, E)$, в котором количество вершин $|V| = 2n$ — чётно. Тогда в G существует разрез размера не меньше $\frac{|E|n}{2n-1}$.*

Доказательство. Как и в прошлый раз, всякий разрез можно задать множеством $S \subseteq V$. Рассмотрим равномерное распределение на множествах $S \subseteq V$, таких что $|S| = n$ (в этом отличие от прошлого рассуждения).

Случайные величины f и f_e определим так же, как и в прошлом доказательстве. Оценим вероятность того, что $f_e = 1$. Число благоприятных исходов равно $2\binom{2n-2}{n-1}$, где двойка отвечает за выбор конца ребра e, лежащего в S, а биномиальный коэффициент отвечает за выбор остальных элементов S. Число всех исходов равно $\binom{2n}{n}$, так что

$$E[f_e] = \Pr[f_e = 1] = \frac{2\binom{2n-2}{n-1}}{\binom{2n}{n}} = \frac{2 \cdot n \cdot n}{2n \cdot (2n-1)} = \frac{n}{(2n-1)}.$$

Тогда аналогично доказательству предыдущей теоремы получаем

$$E[f] = \sum_{e \in E} E[f_e] = \frac{|E|n}{2n-1},$$

а значит существует такой разрез, в котором не меньше $\frac{|E|n}{2n-1}$. □

Такой разрез тоже можно построить напрямую, но это уже заметно сложнее. А наше доказательство было в сущности не очень сложным (во всяком случае, технически).

Задача 10.31. Докажите, что если в графе $G = (V, E)$ число вершин $|V| = 2n + 1$ — нечётно, то в нём есть разрез размера не меньше $\frac{|E|(n+1)}{2n+1}$.

Математическое ожидание позволяет давать оценки вероятностей некоторых событий.

Лемма 10.13 (неравенство Маркова)**.** *Пусть f — случайная величина, принимающая только неотрицательные значения. Тогда для всякого $\alpha > 0$ верно*

$$\Pr[f \geqslant \alpha] \leqslant \frac{E[f]}{\alpha}.$$

То есть, вероятность того, что случайная величина f сильно больше своего математического ожидания, не слишком велика (заметим, что лемма становится содержательной, когда $\alpha > E[f]$.

Доказательство. Взглянем на нужное нам неравенство с другой стороны. Нам нужно доказать, что

$$E[f] \geqslant \alpha \cdot \Pr[f \geqslant \alpha].$$

Пусть случайная величина f принимает значения $a_1, \ldots, a_k$ с вероятностями $p_1, \ldots, p_k$. Запишем, чему равно её математическое ожидание по определению:

$$E[f] = a_1 p_1 + a_2 p_2 + \ldots + a_k p_k.$$

Посмотрим отдельно на те a_i, которые меньше α, и отдельно на те a_i, которые не меньше α. Если первые заменить на ноль, то сумма может только уменьшиться. Если вторые заменить на α, то сумма также может только уменьшиться. После таких замен, у нас остаётся сумма нескольких слагаемых, каждое из которых есть αp_i, где p_i — вероятность некоторого значения случайной величины, не меньшего α. Нетрудно видеть, что такая сумма как раз равна $\alpha \cdot \Pr[f \geqslant \alpha]$, и лемма доказана. □

Задача 10.32. Где в нашем доказательстве мы использовали неотрицательность случайной величины? Остается ли лемма верной, если убрать условие неотрицательности случайной величины?

Задача 10.33. В лотерее на выигрыши уходит 40% от стоимости проданных билетов. Каждый билет стоит 100 рублей. Докажите, что вероятность выиграть 5000 рублей (или больше) меньше 1%.

Пример 10.34. Приведём алгоритмический пример применения неравенства Маркова.

Некоторые алгоритмы, использующие случайные числа, работают так, что всегда выдают верный ответ, но время работы может зависеть от значения случайных чисел и при некотором невезении алгоритм может работать долго. В таких ситуациях, чтобы тем не менее сказать что-то о времени работы алгоритма, говорят о среднем времени работы алгоритма. Действительно, время работы в данном случае — случайная величина (зависящая от случайных чисел, используемых алгоритмом), и среднее время работы — это просто математическое ожидание этой случайной величины.

Предположим, что у нас есть такой алгоритм A, работающий, скажем, за среднее время $O(n^2)$, где n — размер входных данных. Для наших практических целей хотелось бы, чтобы алгоритм всегда (то есть независимо от случайных чисел) заканчивал свою работу за время $O(n^2)$. Чтобы добиться этого, мы готовы даже смириться с тем, что в $0,01\%$ случаев алгоритм будет выдавать неправильный ответ. Можем ли мы получить такой алгоритм?

Оказывается, можем. Обозначим среднее время работы алгоритма A через T, и рассмотрим следующий алгоритм: запускаем алгоритм A и ждём пока он сделает $10000 \cdot T$ шагов. Если алгоритм успел выдать ответ, прекрасно. Если нет, выдаём произвольный ответ. Идея в том, что алгоритм A с очень большой вероятностью закончит свою работу за $10000 \cdot T$ шагов. Действительно, обозначим через f (неотрицательную) случайную величину, равную времени работы алгоритма A. Тогда $E[f] = T$. По неравенству Маркова получаем

$$\Pr[f > 10000 \cdot T] \leqslant \frac{T}{10000 \cdot T} = 1/10000.$$

Заметим, что время работы нового алгоритма действительно есть $O(n^2)$ (по сравнению со старым алгоритмом оно просто умножилось на константу), а ошибка может произойти только если старый алгоритм работал дольше $10000 \cdot T$ шагов. По нашей оценке это происходит с вероятностью не больше 0.01%.

10.6 Частота орлов при подбрасывании монеты и биномиальные коэффициенты

Если подбрасывать «честную» монету много раз, то разумно ожидать, что количество выпавших орлов будет примерно равно половине от числа подбрасываний. Именно это интуитивное наблюдение было положено в основу понятия «вероятность». Но что значит «примерно равно»? Оказывается, этому интуитивно ожидаемому результату можно придать точные количественные характеристики.

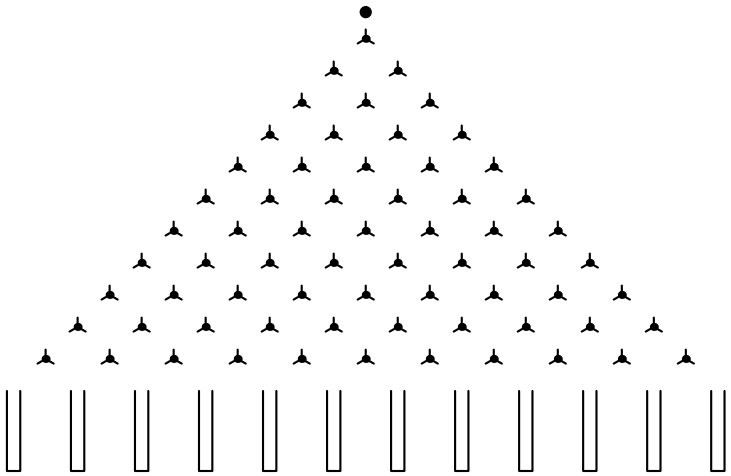


Рис. 10.3: Схема доски Гальтона

Во-первых, уточним, что разные подбрасывания «честной» монеты независимы. Если кто-то будет подбрасывать монету и после выпадения четвёртого орла подряд переворачивать монету, такие подбрасывания вряд ли стоит считать «честными».

Поэтому мы будем считать все возможные результаты n подбрасываний равновозможными. Будем записывать результаты, указывая 1, если выпал орёл, и 0, если выпала решка.

Как легко видеть, количество вариантов n подбрасываний, в которых выпало k орлов, равно количеству двоичных слов длины n, в которых ровно k единиц (и $n-k$ нулей), т.е. равно биномиальному коэффициенту

$$\binom{n}{k}.$$

Если же нас интересуют события вида «выпало не меньше k орлов» или «количество орлов не меньше k_1 и не больше k_2», то их вероятности — суммы биномиальных коэффициентов, делённые на 2^n.

Итак, нам нужно оценивать величины биномиальных коэффициентов. Это можно делать по-разному. Скажем, можно находить нужные величины экспериментально. Для этого можно использовать прибор, называемый доской Гальтона. Его схема изображена на рисунке 10.3.

Если набросать через такую решётку много шариков (в оригинальном исполнении это были бобы), то бункеры будут заполнены как раз пропорционально величине соответствующих биномиальных коэффициентов.

Конечно, в каждом конкретном эксперименте заполнение бункеров будет разным. Но форма этого распределения при достаточно большом количестве шариков всё больше будет напоминать кривую, изображённую на рисунке 10.4.

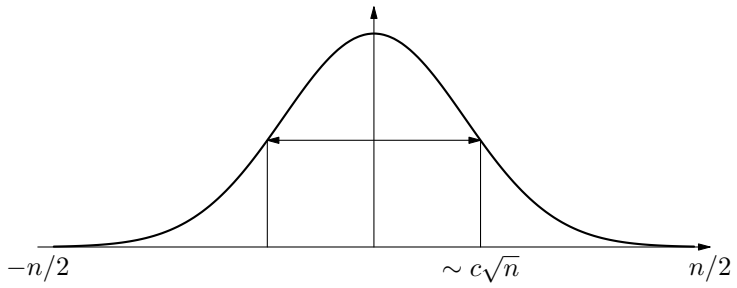


Рис. 10.4: Биномиальные коэффициенты: взгляд издалека

Задача 10.35. Рисунок 10.4 неточный: для наглядности масштаб по оси абсцисс выбран неравномерным. Попробуйте представить, как издалека выглядит график биномиальных коэффициентов при равномерном масштабе по оси абсцисс.

Другой подход к оценке сумм биномиальных коэффициентов состоит в использовании математики вместо бобов и гвоздиков.

Задача 10.36. Докажите, что биномиальные коэффициенты $\binom{n}{k}$ увеличиваются с ростом k вплоть до $n/2$, а затем убывают.

А насколько велик центральный коэффициент $\binom{2n}{n}$? Ясно, что совсем маленьким он быть не может: всего есть $2n+1$ коэффициент, их сумма равна 2^{2n}. Поэтому центральный коэффициент уж никак не меньше среднего значения:

$$\binom{2n}{n} \geqslant \frac{2^{2n}}{2n+1}\,.$$

Но интересно также получить верхнюю оценку для центрального биномиального коэффициента. В терминах вероятности это вероятность того, что в результате $2n$ подбрасываний монеты выпало ровно n орлов. Это то самое значение числа орлов, которое подсказывает нам интуиция. Мы пока лишь убедились, что эта вероятность не слишком мала, она не меньше $1/(2n+1)$.

Оказывается, она и не очень велика. Чтобы получить количественную оценку, можно воспользоваться асимптотической формулой Стирлинга для факториала:

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \tag{10.3}$$

(предел отношения этих выражений при $n \to \infty$ равен 1).

Подставляя в формулу для биномиального коэффициента, получаем

$$\binom{2n}{n} = \frac{(2n)!}{n!\cdot n!} \sim \frac{\sqrt{4\pi n}}{2\pi n}\left(\frac{2n}{e}\right)^{2n}\left(\frac{e}{n}\right)^{2n} = \frac{1}{\sqrt{\pi n}}2^{2n}. \tag{10.4}$$

Из этой оценки видим, что с ростом n вероятность получить ровно половину орлов стремится к нулю. Если вам показали результаты 10000 подбрасываний, в которых получилось ровно 5000 орлов, это повод задуматься о «честности» монеты.

Оценки биномиальных коэффициентов с помощью формулы Стирлинга довольно точные, но асимптотические и не очень простые из-за множителей вида $\sqrt{2\pi n}$. Очень часто оказываются удобными более грубые, но более простые оценки биномиальных коэффициентов. Приведём самую популярную пару.

Лемма 10.14. *Для всех целых положительных n и k, где $k \leqslant n$, выполняются неравенства*

$$\left(\frac{n}{k}\right)^k \leqslant \binom{n}{k} < \left(\frac{en}{k}\right)^k. \tag{10.5}$$

Доказательство левого неравенства в (10.5). Запишем выражение для биномиального коэффициента:

$$\binom{n}{k} = \frac{n\cdot(n-1)\cdot\ldots\cdot(n-k+1)}{k!} = \frac{n}{k}\cdot\frac{n-1}{k-1}\cdot\ldots\cdot\frac{n-(k-1)}{k-(k-1)}\,.$$

Дроби в это произведении увеличиваются слева направо, так как

$$\frac{a-1}{b-1} \geqslant \frac{a}{b} \quad \text{при } a \geqslant b > 1.$$

Заменяя каждую из этих k дробей на наименьшую среди них (это как раз n/k), получаем левую часть левого неравенства в (10.5). □

Доказательство правого неравенства в (10.5). Это неравенство уже требует применения математического анализа (откуда-то должно взяться число e).

Мы используем неравенство

$$e > \left(1+\frac{1}{k}\right)^k, \tag{10.6}$$

которое сразу следует из самого элементарного определения числа e. Перемножим неравенства (10.6) для $k = 1, 2, \ldots, n-1$, получим

$$e^{n-1} > \left(\frac{2}{1}\right)^1\cdot\left(\frac{3}{2}\right)^2\cdot\ldots\cdot\left(\frac{n}{n-1}\right)^{n-1} = \frac{n^{n-1}}{(n-1)!} = \frac{n^n}{n!}\,,$$

т.е. $n! > e(n/e)^n$. Отсюда

$$\binom{n}{k} < \frac{n^k}{k!} < \frac{1}{e}\cdot\left(\frac{en}{k}\right)^k,$$

что и требовалось, так как $e > 1$ (и даже $e > 2$, см. (10.6) при $k = 1$). □

Приведём ещё одно полезное неравенство для биномиальных коэффициентов.

Лемма 10.15. *Для любых целых чисел k, t, удовлетворяющих условиям $0 < t \leqslant k \leqslant n/2$, выполняется неравенство*

$$\binom{n}{k-t} < \frac{k}{t^2}\binom{n}{k}\,.$$

Доказательство. По формуле для биномиальных коэффициентов получаем

$$\binom{n}{k} \Big/ \binom{n}{k-t} = \frac{n\cdot(n-1)\cdot\ldots\cdot(n-k+1)}{k!} \cdot \frac{(k-t)!}{n\cdot(n-1)\cdot\ldots\cdot(n-k+t+1)} = \\ = \frac{(n-k+t)\cdot\ldots\cdot(n-k+1)}{k\cdot(k-1)\cdot\ldots\cdot(k-t+1)} = \frac{n-k+t}{k}\cdot\frac{n-k+t-1}{k-1}\cdot\ldots\cdot\frac{n-k+1}{k-t+1}$$

Поскольку $k \leqslant n/2$, числители дробей больше знаменателей. Как и раньше, самая маленькая дробь в этом произведении — первая. Поэтому получаем оценку

$$\binom{n}{k} \Big/ \binom{n}{k-t} > \left(\frac{n-k+t}{k}\right)^t = \left(1+\frac{n-2k+t}{k}\right)^t \geqslant 1+\frac{t(n-2k+t)}{k} > \frac{t^2}{k}\,,$$

что и даёт искомое неравенство. (Предпоследнее неравенство — это неравенство Бернулли.) □

Из этой леммы следует, что биномиальные коэффициенты довольно быстро убывают, начиная с расстояния $c\sqrt{n}$ от центрального. Более точные оценки приводятся в следующем разделе.

10.7 Большие уклонения: неравенство Чернова

Обозначим через X_n случайную величину, равную количеству выпавших орлов после n подбрасываний «честной» монеты, а через $\xi_n = X_n/n$ — частоту выпавших орлов. При больших n частота с очень большой вероятностью оказывается близкой к $1/2$. Имеет место следующая оценка, которая очень удобна в приложениях вероятностного метода в комбинаторике, а также в теоретической информатике.

Теорема 10.16 (неравенство Чернова)**.**

$$\Pr\left[|X_n - \tfrac{n}{2}| > \varepsilon n\right] = \Pr\left[|\xi_n - \tfrac{1}{2}| > \varepsilon\right] < 2e^{-2\varepsilon^2 n}.$$

Это и есть количественная формулировка того интуитивного представления, с которого мы начали обсуждение.

Заметим, что неравенство Чернова симметрично, оно оценивает вероятность уклонений частоты от $1/2$ в обе стороны. В силу симметрии биномиальных коэффициентов

$$\binom{n}{k} = \binom{n}{n-k}$$

достаточно оценивать вероятность превышения частоты над $1/2$ (это объясняет множитель 2 в правой части неравенства Чернова).

План доказательства неравенства Чернова. Изложим вначале общую схему доказательства, пропуская доказательства технических утверждений.

Оказывается, неравенство Чернова — это частный случай неравенства Маркова для подходящим образом подобранной функции от величины X_n.

Удобнее перейти к случайным величинам $Y_n = 2X_n - n$. Если X_n равна сумме n случайных величин, принимающих независимо и равновероятно значения 0 и 1, то Y_n равна сумме величин $y_{n,i}$, каждая из которых независимо принимает случайно и равновероятно значения -1 и $+1$.

Но это не всё: нужно взять экспоненту от Y_n с удачным основанием. Определим случайную величину

$$Z_n = e^{\lambda Y_n} = \prod_i e^{\lambda y_{n,i}}.$$

Оказывается, что в данном случае математическое ожидание произведения равно произведению математических ожиданий сомножителей (в отличие от линейности, мультипликативность не всегда выполняется для математических ожиданий):

$$E[Z_n] = \prod_i E[e^{\lambda y_{n,i}}] = \left(\frac{e^\lambda + e^{-\lambda}}{2}\right)^n = (\operatorname{ch}\lambda)^n. \tag{10.7}$$

В последнем равенстве использовано определение функции гиперболического косинуса $\operatorname{ch} x$. Здесь мы его используем лишь для краткости.

Интересующее нас событие $X_n - n/2 > \varepsilon n$ записывается через случайную величину Y_n как $Y_n > 2\varepsilon n$, а через величину Z_n как $Z_n > e^{2\lambda\varepsilon n}$.

Применим неравенство Маркова к величине Z_n:

$$\Pr[Z_n > e^{2\lambda\varepsilon n}] \leqslant \frac{E[Z_n]}{e^{2\lambda\varepsilon n}} = \left(\frac{\operatorname{ch}\lambda}{e^{2\lambda\varepsilon}}\right)^n$$

Осталось выбрать λ, чтобы сделать дробь в основании степени поменьше. Для этого нужна ещё одна порция анализа, а именно, неравенство

$$\operatorname{ch} x \leqslant e^{x^2/2}. \tag{10.8}$$

Подставляя это неравенство, получаем при $\lambda = 2\varepsilon$

$$\Pr[Z_n > e^{2\lambda\varepsilon n}] \leqslant e^{(2\varepsilon^2 - 4\varepsilon^2)n},$$

что и даёт неравенство Чернова. □

Для завершения доказательства нам нужно проверить два технических утверждения.

Доказательство формулы (10.7). Запишем выражение для математического ожидания Z_n:

$$E[Z_n] = \sum_{w\in\{0,1\}^n} 2^{-n} Z_n(w) = 2^{-n} \sum_{w\in\{0,1\}^n} \prod_{i=1}^n e^{\lambda(2w_i - 1)}.$$

Каждое слагаемое является произведением, в котором стоят e^λ (если $w_i = 1$) и $e^{-\lambda}$ (если $w_i = 0$). Значит, это те же самые слагаемые, которые получаются из бинома $(e^\lambda + e^{-\lambda})^n$ после раскрытия скобок (и до приведения подобных). Поэтому правая часть равенства равна $2^{-n}(e^\lambda + e^{-\lambda})^n$, что совпадает с $(\operatorname{ch}\lambda)^n$. □

Доказательство формулы (10.8). Тут нужно использовать разложение экспоненты в ряд Тейлора:

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \,.$$

Ряд для гиперболического косинуса получается отсюда почленным сложением рядов. Остаются только слагаемые в чётных степенях:

$$\operatorname{ch} x = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!} \,.$$

Второй ряд получается подстановкой $x^2/2$ в ряд для экспоненты. Опять есть только слагаемые для чётных степеней:

$$e^{-x^2/2} = \sum_{k=0}^{\infty} \frac{x^{2k}}{2^k k!} \,.$$

Осталось заметить, что при каждом k выполняется

$$\frac{1}{(2k)!} \leqslant \frac{1}{2^k k!} \,,$$

формула (10.8) получается почленным сравнением рядов. □

10.8 Подробности для любознательных

10.8.1 Ещё одна элементарная оценка отношения биномиальных коэффициентов

Лемма 10.15 даёт достаточно хорошее приближение к скорости убывания биномиальных коэффициентов, близких к среднему. Мы сейчас приведём доказательство оценки в другую сторону, которая не использует формулы Стирлинга. В некоторых случаях такие оценки предпочтительнее, так как не зависят от скорости сходимости в формуле Стирлинга.

Будем оценивать сверху величину

$$a_t = \frac{\binom{n}{n/2}}{\binom{n}{n/2-t}} \,,$$

предполагая n чётным (для нечётных всё аналогично). Как и раньше, запишем отношение биномиальных коэффициентов в виде произведения дробей (полагаем $k = n/2$):

$$a_t = \binom{n}{n/2} \Big/ \binom{n}{n/2-t} = \frac{n-n/2+t}{n/2} \cdot \frac{n-n/2+t-1}{n/2-1} \cdot \ldots \cdot \frac{n-n/2+1}{n/2-t+1} \,.$$

Теперь нас интересует верхняя оценка, поэтому заменим все дроби на наибольшую — последнюю. Получаем

$$a_t < \left(1 + \frac{t}{n/2 - t + 1}\right)^t . \tag{10.9}$$

Пусть $t < \sqrt{n/5}$. Тогда можно заметить, что слагаемые в разложении бинома

$$\left(1 + \frac{t}{n/2 - t + 1}\right)^t =$$
$$= 1 + \binom{t}{1}\frac{t}{n/2 - t + 1} + \binom{t}{2}\left(\frac{t}{n/2 - t + 1}\right)^2 + \ldots + \binom{t}{j}\left(\frac{t}{n/2 - t + 1}\right)^j + \ldots$$

убывают быстрее геометрической прогрессии со знаменателем $1/2$.

Задача 10.37. Докажите это утверждение.

Поэтому при $t < \sqrt{n/5}$ и $n > 2$ получаем неравенство

$$a_t < 1 + \frac{2t^2}{n/2 - t + 1} . \tag{10.10}$$

(Сравните с оценкой из леммы 10.15.)

10.8.2 Другое доказательство неравенства Чернова

Идея этого доказательства состоит в том, чтобы использовать другую монету, у которой вероятность выпадения орла $p = \frac{1}{2} + \varepsilon$, а вероятность выпадения решки $1 - p = \frac{1}{2} - \varepsilon$. Обозначим через $X_{n,\varepsilon}$ случайную величину, равную количеству выпавших орлов после n независимых подбрасываний этой «испорченной» монеты.

У испорченной монеты вероятности выпадения орлов больше. Оказывается, если сравнить вероятности событий $X_n = k$ (честная монета дала k орлов) и $X_{n,\varepsilon} = k$ (испорченная монета дала k орлов) при $k \geqslant pn$, то первая вероятность намного меньше второй при больших n. Но тогда сумма всех таких вероятностей для честной монеты намного меньше суммы тех же вероятностей для испорченной монеты. А эта вторая сумма уж точно не больше 1. Отсюда и получим верхнюю оценку на вероятность больших уклонений величины X_n.

Подбрасывания испорченной монеты независимые, поэтому по формуле произведения вероятностей независимых событий вероятность каждого результата, содержащего k единиц, равна $p^k(1-p)^{n-k}$. Суммируя по несовместным событиям (все результаты с k единицами), получаем

$$\Pr[X_{n,\varepsilon} = k] = \binom{n}{k} p^k (1-p)^{n-k}$$

и

$$\frac{\Pr[X_n = k]}{\Pr[X_{n,\varepsilon} = k]} = \frac{\binom{n}{k} 2^{-n}}{\binom{n}{k} p^k (1-p)^{n-k}} = \frac{1}{2^n (p^{k/n}(1-p)^{1-k/n})^n} .$$

Обозначим $q = k/n$ и перепишем это отношение вероятностей в виде

$$\frac{\Pr[X_n = qn]}{\Pr[X_{n,\varepsilon} = qn]} = (2p^q(1-p)^{1-q})^{-n}.$$

Мы хотим доказать, что при $p > 1/2$ основание степени в правой части равенства больше 1 и указать одну общую оценку для всех $p \leqslant q \leqslant 1$. Тогда получим желаемое: вероятности для честной монеты окажутся намного меньше, чем для испорченной (поскольку возводим число, бóльшее 1, в отрицательную степень).

Так как $p/(1-p) = (\frac{1}{2} + \varepsilon)/(\frac{1}{2} - \varepsilon) > 1$, функция

$$p^x(1-p)^{1-x} = (1-p) \cdot \left(\frac{p}{1-p}\right)^x$$

возрастающая по x. Поэтому при $x = p$ она принимает минимальное значение на луче $[p, +\infty)$. Поэтому для оценки отношения вероятностей достаточно сравнить с 1 функцию

$$2p^p(1-p)^{1-p}.$$

Удобнее взять логарифм, т.е. перенести функцию в показатель степени. Пусть это будет двоичный логарифм:

$$\log_2(p^p(1-p)^{1-p}) = p\log_2 p + (1-p)\log_2(1-p) \stackrel{\text{def}}{=} -h(p).$$

Заметим, что $1 = \log_2 2 = h(1/2)$.

Лемма 10.17. *Функция $h(x)$ на интервале $(0, 1/2)$ возрастает, а на интервале $(1/2, 1)$ убывает. Точка $1/2$ тем самым является точкой максимума.*

Доказательство. Нужно вычислить производную $h(x)$:

$$h'(x) = -\log_2 x - \frac{1}{\ln 2} + \log_2(1-x) + \frac{1}{\ln 2} = \log_2 \frac{1-x}{x}.$$

Так как $1 - x > x$ равносильно $x < 1/2$, получаем, что интервале $(0, 1/2)$ производная положительная, а на интервале $(1/2, 1)$ производная отрицательная. Отсюда и следует утверждение леммы. □

Теперь воспользуемся леммой и перепишем оценку на отношение вероятностей как

$$\frac{\Pr[X_n = qn]}{\Pr[X_{n,\varepsilon} = qn]} \leqslant 2^{-(h(1/2)-h(p))n} = 2^{-\eta^2 n}. \tag{10.11}$$

Число $\eta > 0$ зависит только от выбранного порога частоты ε.

Теорема 10.18. $\Pr\left[|\xi_n - \frac{1}{2}| > \varepsilon\right] < 2 \cdot 2^{-\eta^2 n}$.

Доказательство. Искомая вероятность в два раза больше, чем

$$\sum_{k > n/2 + \varepsilon n} \Pr[X_n > k] < \sum_{k > n/2 + \varepsilon n} \Pr[X_{n,\varepsilon} > k] 2^{-\eta^2 n} \leqslant 2^{-\eta^2 n}$$

(так как сумма вероятностей не превосходит 1). □

Мы получили, что вероятности больших уклонений убывают экспоненциально быстро.

Применив ещё чуть больше анализа, можно явно выразить η через ε. Вторая производная $h(x)$ на интервале $(1/2, 1)$ убывает, так как

$$h''(x) = -\frac{1}{\ln 2} \cdot \frac{1}{x(1-x)} \ .$$

Поэтому функция $h(1/2) - h(x) + \frac{h''(1/2)}{2}(x-1/2)^2$ выпуклая (вторая производная неотрицательна). Значение этой функции и её производной в точке $x = 1/2$ равны нулю. Поэтому касательная к графику этой функции в точке $x = 1/2$ горизонтальна. Так как функция выпуклая, её график лежит выше касательной, то есть функция неотрицательная.

Значит, выполняется неравенство

$$h(1/2) - h(1/2+\varepsilon) \geqslant -\frac{h''(1/2)}{2}\varepsilon^2 = \frac{2}{\ln 2}\varepsilon^2.$$

Подставляя в (10.11), получаем неравенство Чернова

$$\Pr\left[|\xi_n - \tfrac{1}{2}| > \varepsilon\right] < 2e^{-2\varepsilon^2 n}.$$

Лекция 11

Комбинаторные игры

Люди играют в разные игры, но нас будут интересовать не все. Мы предполагаем, что игра делится на ходы, игроки ходят по очереди, зная начальную позицию и ходы противника. Правила игры определяют, какие ходы возможны, когда игра кончается и кто выиграл. Как в крестиках-ноликах или шахматах (но не в преферансе, где расклад игрокам неизвестен).

Оказывается, что такие игры предопределены — в том смысле, что в каждой позиции теоретически известно, кто выигрывает «при правильной игре», независимо от игры противника. Теоретически — но не практически, иначе бы какой смысл был в турнирах по шахматам или го? На самом деле сказанное о «правильной игре» не так просто точно сформулировать — мы это сделаем, а также приведём примеры игр, которые удаётся проанализировать.

Вообще игры часто появляются в дискретной математике и теоретической информатике. В лекции 12 они встретятся при оценках сложности алгоритмов (так называемый «метод противника», adversary arguments).

11.1 Позиции

Вот совсем простая игра.

Пример 11.1 (игра в монетницу)**.** Игроки Первый и Второй по очереди кладут монеты в монетницу, которая вмещает самое большее 20 монет. Делая ход, можно положить 2 или 3 монеты. Кто не может сделать ход (не переполнив монетницу), проиграл. (Это бывает, если там 19 монет или больше). Кто выигрывает «при правильной игре» — Первый (начинающий игру) или Второй?

Наверно, вы уже сообразили: Второй с гарантией выиграет, если каждый раз будет доводить число монет для кратного пяти. Другими словами, если Первый положил 2 монеты, то Второй должен положить 3 и наоборот.

Почему Второй выиграет? Каждая пара ходов добавляет пять монет, за четыре раза их станет 20, Первый не сможет сделать ход и — согласно правилам — проиграет.

Задача 11.2. Проанализируйте вариант игры с монетницей, в котором игрокам разрешено класть 3 или 4 монеты, а всего помещается 21 монета. Тот же вопрос для 20 монет.

Пример 11.3 («штриховка»)**.** Есть полоска бумаги, расчерченная на квадратики. Первый и Второй ходят по очереди. Ход состоит в том, что игрок заштриховывает сколько-то (не меньше одного) квадратиков из оставшихся. Проигрывает тот игрок, который не может сделать ход (заштриховать хотя бы один квадратик).

Тут анализ, пожалуй, ещё проще. Если квадратик всего один, то Первый может заштриховать его и выиграть. Да и если не один, тоже может — надо заштриховать все. (Вот если разрешить полоску из нуля квадратиков, то на такой полоске выигрывает Второй, потому что Первый не может сделать ход.)

Объясним на этих примерах, что такое *позиция* в игре. Она определяет, какой игрок делает ход и как (точнее говоря, в какие позиции) он может пойти. В первом примере позиция включает в себя число монет и то, чья очередь хода. Получается всего $2 \cdot 21 = 42$ позиции (число монет от 0 до 20). Обратите внимание, что мы включаем и позиции, которые в реальной игре невозможны (скажем, с одной монетой).

Во втором примере, если исходных квадратиков было n, то в позицию можно включать число оставшихся квадратиков (от 0 до n) и очередь хода. Будет всего $2(n+1)$ позиций. В принципе мы могли бы включить в позицию и информацию о том, какие именно квадратики заштрихованы, но это только бы напрасно усложнило дело, поскольку для игры важно лишь *количество* оставшихся квадратиков.

Помимо описания всех позиций (мы будем всегда считать, что их конечное число), надо указать *начальную* позицию (с которой игра начинается). Надо также сказать, когда игра кончается, то есть какие позиции являются *заключительными* (больше ходы не делаются) и какой игрок выиграл (для каждой заключительной позиции). Чтобы предусмотреть и ничьи, будем считать, что в каждой заключительной позиции есть число, указывающее, сколько в ней выиграл, скажем, Первый (а Второй, соответственно, проиграл). Игра с ничейным исходом, вроде шахмат, теперь описывается так: в одних заключительных позициях это число равно $+1$ (в обычных терминах это выигрыш Первого), в других оно равно -1 (в обычных терминах — выигрыш Второго), в третьих оно равно нулю (в обычных терминах — ничья).

Замечание 11.1. Более общий случай — *неантагонистические игры*, когда выигрыш одного не равен проигрышу другого (скажем, могут быть позиции, выгодные для обоих — или наоборот, не выгодные ни для кого). Но такие игры в этой главе мы не рассматриваем.

Замечание 11.2. Что такое позиция в шахматах? Расположение фигур на доске и очерёдность хода? Почти так, но не совсем. Знатоки шахмат знают, что возможность рокировки зависит от истории игры, что при троекратном повторении любой игрок может потребовать зафиксировать ничью, и что при большом числе ходов без взятия фигур и перемещения пешек тоже фиксируется ничья. Значит, всю необходимую информацию (какие фигуры ходили, что уже встречалось и т.п.) нужно включить

в позицию — если действовать с запасом, то можно включить всю историю, то есть полный протокол игры.

Вот пример игры, где можно не просто выиграть или проиграть, а выиграть или проиграть разные суммы.

Пример 11.4 (игра в монетницу «на интерес»)**.** Играем как раньше, кладя по очереди в монетницу на 20 монет две или три монеты. Но теперь, если кладёшь две монеты, надо третью отдать противнику. А когда игра кончается и один из игроков не может сделать ход, то все монеты в монетнице (там может быть 19 или 20 монет) отдаются другому игроку.

По нашему соглашению результатом игры считается выигрыш первого игрока (отрицательное число, если на самом деле он проиграл). Например, если второй игрок придерживается описанной раньше стратегии (дополнять до кратного 5), то в конце он заберёт все 20 монет. Но часть из них ему самому придётся положить, при этом сколько именно, зависит от ходов первого игрока. Если первый игрок, скажем, всегда будет класть по три монеты, то второй будет добавлять две и одну отдавать первому. В итоге первый игрок положит 12 монет и получит 4, так что результат игры равен -8. Подумайте, стоит ли так действовать обоим игрокам или кто-то из них может гарантировать себе что-то лучше? Мы ещё вернёмся к этому вопросу.

Задача 11.5. Опишите множество позиций для игры в монетницу «на интерес». (Тут есть разные варианты — постарайтесь по возможности выбрать наиболее экономный, где хранится лишь то, что необходимо.)

После всех наших примеров, наверно, понятно, какого типа игры мы рассматриваем, но всё же нужно дать формальное определение. Будем называть игроков Макс (M) и Мин (m), подчёркивая, что Макс стремится увеличить результат игры, а Мин — уменьшить. В предыдущих примерах роль Макса играл Первый (так мы договорились), но теперь мы уже не будем настаивать, чтобы игроки ходили по очереди, поэтому лучше назвать игроков Макс и Мин.

Определение 11.1. Чтобы задать *конечную комбинаторную игру*, надо:

- Задать конечное множество S, элементы которого называют *позициями* игры.
- Присвоить каждой позиции один из трёх типов: в одних ходит Макс, в других ходит Мин, в третьих игра закончена. Позиции третьего типа называют *заключительными*. Будем обозначать множества позиций первого типа S_M, второго типа S_m (по именам игроков), а множество заключительных позиций S_f.
- Указать *начальную* позицию $s_0 \in S$;
- Для каждой незаключительной позиции указать, в какие вершины может из неё попасть игрок, которой в ней ходит.

- Задать функцию выигрыша $v\colon S_f \to \mathbb{R}$, которая в каждой заключительной позиции определяет результат игры (выигрыш Макса, как мы договорились).

Можно сказать, что игра задаётся ориентированным графом: его вершины являются позиции, а рёбра ведут из каждой позиции в те, куда может попасть делающий ход игрок. Тогда вершины, из которых нет рёбер, будут заключительными, и у них нужно написать результат, а у остальных нужно написать, чья эта вершина, то есть кто делает ход. Наконец, нужно указать начальную позицию. Собственно говоря, если нарисовать этот граф и положить рисунок на стол, то можно прямо на нём и играть. Фишка ставится в начальную позицию, и затем игроки двигают фишку по стрелкам (рёбрам), причём на каждой вершине написано, кто ходит. Когда фишка попадает в вершину, из которой стрелок не ведёт, игра кончается — а около вершины написано, сколько Мин платит Максу.

Задача 11.6. Составьте граф игры в монетницу из примера 11.1.

Замечание 11.3. Это определение не запрещает одному из игроков сделать несколько ходов подряд: если из Макс-вершины ход делается в другую Макс-вершину, то Макс снова делает ход. Но это не очень существенно: всегда можно объединить несколько ходов в один и перейти к играм, в которых игроки ходят строго по очереди.

Замечание 11.4. Наверно, у вас уже возник вопрос: а что делать, если игроки ходят (скажем) по циклу в графе игры и игра никогда не кончается? Что тогда? Давайте договоримся, что такие игры мы не рассматриваем: граф должен быть ациклическим. Но многое из сказанного дальше естественно переносится и на такие игры.

Замечание 11.5. В некоторых играх на самом деле не важно, кто ходил первым, а важно, кто ходит сейчас. Скажем, в игре в монетницу в её первоначальном варианте игроку, обдумывающему свой ход, важно лишь знать, сколько сейчас монет — а кто игру начал, уже без разницы. В варианте «на интерес» ещё надо знать, сколько монет ты потратил (чтобы определить окончательный результат — сколько всего выиграл или проиграл), но тоже не важно, кто начинал. Такие игры иногда называют *беспристрастными*.

Контрольный вопрос 11.7. Попробуйте дать формальное (наподобие определения 11.1) определение беспристрастной игры, в которой игроки ходят строго по очереди.

11.2 Стратегии

Часто в задаче про игру говорят что-то типа «Докажите, что при правильной игре Первый может выиграть». Но тут сразу же возникают вопросы и требуются уточнения. Что значит, что «Первый может выиграть»? а может и не выиграть, что ли? Имеется в виду, конечно, не это. Уточняя задачу, можно сказать, что «у Первого есть стратегия, гарантирующая выигрыш». Это уже лучше, но что такое «стратегия»? Как определить это понятие математически? И пусть мы даже определили,

что такое «стратегия» — что значит «гарантирует»? На все эти вопросы надо ответить, и мы попытаемся сейчас это сделать.

Неформально говоря, стратегия — это правило, которое предписывает игроку, как ему ходить во всех возможных ситуациях. Тут, однако, требуются два уточнения.

Во-первых, мы будем предполагать, что «ситуация» — это позиция в игре, а не весь ход игры (какие ходы уже сделали противники). Другими словами, давая игроку совет, стратегия не может ссылаться на историю игры; совет должен зависеть только от текущей позиции. Это отсекает некоторые естественные стратегии — собственно, даже стратегия игры в монетницу «дополнять ход противника до 5» тоже теперь запрещена, потому что совет игроку может зависеть только от числа монет. Но её можно переформулировать: скажем, что нужно доводить число монет до ближайшего кратного 5. Это уже вполне законная (и выигрышная для второго игрока) стратегия.

Почему мы ограничиваемся такими стратегиями (их называют *позиционными*)? В определении комбинаторной игры мы позаботились, чтобы все возможные продолжения партии определялись только текущей позицией, а не предшествующими ходами партии. По этой причине всегда можно ограничиться позиционными стратегиями: если есть непозиционная стратегия, гарантирующая выигрыш, то есть и позиционная.

Во-вторых, есть ещё такая тонкость: должна ли стратегия давать совет игроку в любой позиции или только в той, в которую игрок реально может попасть, следуя этой стратегии? Зачем предусматривать действия в ситуациях, которые не могут случиться, если игрок не отклоняется от стратегии? (Объясняя прохожему, что надо сначала повернуть направо, а потом идти прямо, вы не будете объяснять, куда он должен будет пойти, если сначала повернёт не направо, а налево.) Но, с другой стороны, можно считать, что стратегия рекомендует действия всегда — просто в ситуациях, куда нельзя попасть, ей следуя, эти действия можно выбрать произвольно. Так мы и будем делать.

Итак, мы можем дать формальное определение стратегии для одного из игроков.

Определение 11.2. Стратегией для одного из игроков (Макса или Мина) в данной игре называется функция, которая определена на всех позициях x, в которых этот игрок делает ход, и указывает один из возможных ходов (одну из позиций y, в которую по правилам можно попасть из x).

Теперь надо объяснить, что означает, что позиция гарантирует выигрыш какого-то размера. Сначала мы определим понятие партии, согласованной со стратегией.

Определение 11.3. Будем называть *партией* конечную последовательность позиций, в которой

- первая позиция — начальная;
- каждая следующая позиция получается из предыдущей по правилам игры (переход допустим);

- последняя позиция — заключительная.

Определение 11.4. Партия *согласована* со стратегий данного игрока (Макса или Мина), если все ходы этого игрока в партии совпадают с предписываемыми стратегией (в соответствующей позиции).

Определение 11.5. Стратегия для Макса *гарантирует результат не менее c*, если во всех партиях, согласованных с этой стратегией, результат игры (число в последней позиции этой партии — напомним, что она заключительная по определению партии) не меньше c. Симметричное определение для Мина: стратегия для Мина *гарантирует результат не более c*, если во всех партиях, согласованных с этой стратегией, результат игры не больше c.

Что даёт это определение для игр, в которых один из игроков выигрывает, а другой проигрывает? В заключительных позициях таких игр написаны числа 1 или -1, при этом 1 означает выигрыш Макса, а -1 означает выигрыш Мина. Стратегия, которая гарантирует Максу результат не менее 1, можно назвать *выигрышной стратегией* для Макса. Аналогично можно определить выигрышные стратегии для Мина — те, которые гарантируют ему результат не больше -1.

Если в игре возможен проигрыш, выигрыш и ничья, то их можно представить как $+1$ (выигрыш Макса), -1 (проигрыш Макса = выигрыш Мина) и 0 — ничья. Тогда, скажем, стратегия, гарантирующая Мину результат не больше 0 — это стратегия, позволяющая ему гарантированно не проиграть, и т. п.

Теперь вернёмся к вопросу «кто выигрывает при правильной игре?» и покажем, что он действительно имеет смысл.

Определение 11.6. Число C называется *ценой игры*, если у Макса и у Мина есть стратегии, гарантирующие выигрыш C.

Если у игры есть цена, то результат её в некотором смысле предопределён. Как бы ни играл Макс, больше C он не получит, если Мин придерживается той стратегии, которая гарантирует результат не больше C. Аналогично Мин не сможет получить меньше C, если Макс придерживается той стратегии, которая гарантирует результат не меньше C.

Контрольный вопрос 11.8. Что означает цена игры 1, 0 и -1 для игр, в которых возможен выигрыш, проигрыш и ничья?

Замечание 11.6. Тем не менее, в реальной жизни люди иногда играют в игры с известной ценой. Один из самых наглядных примеров — крестики-нолики. Маленькие дети во всем мире играют в эту игру, хотя хорошо известно, что её цена равна 0 (ничья).

В следующем разделе мы докажем, что всякая игра рассматриваемого нами класса имеет свою цену. Например, в шахматах возможно одно из трёх:

- у белых есть стратегия, гарантирующая им выигрыш;

- у чёрных есть стратегия, гарантирующая им выигрыш;
- и у белых, и у чёрных есть стратегия, гарантирующая как минимум ничью.

Контрольный вопрос 11.9. Являются ли эти три случая взаимно исключающими?

Из этого примера видно, что теорема существования цены игры мало что даёт на практике. Играя в шахматном турнире, вы вряд ли сможете использовать это своё теоретическое знание — для практической игры нужна не теорема существования стратегии, а сама стратегия.

Задача 11.10. Представьте себе, что у вас есть знакомый оракул, который про любую шахматную позицию сообщает, какова её цена (то есть какова цена игры, начинающейся с этой позиции). Его можно спрашивать во время игры — но стратегию (куда ходить) оракул не сообщает. Как с помощью оракула играть оптимальным образом?

11.3 Разбор с конца

В этом разделе доказывается, что для рассматриваемых нами игр (комбинаторных антагонистических игр с ациклическим графом) всегда существует цена игры. Идея доказательства — *разбор с конца*.

Пусть нам дана некоторая игра рассматриваемого класса, то есть дано множество позиций, в каждой известно, закончилась ли игра (и тогда есть число — результат игры) или ещё нет (и тогда известно, кто ходит и куда можно пойти), и указана начальная позиция. Как доказать, что у игры есть цена? Для начала мы немного обобщим задачу и будем рассматривать не одну эту игру, а все игры, которые получаются из исходной заменой начальной позиции, — и доказывать, что каждая из них имеет цену. Более того, гарантирующие цены этих игр стратегии, то есть функции из множества позиций в множество возможных ходов, для всех таких игр согласованы: все стратегии, определённые в данной позиции, указывают на один и тот же ход из этой позиции. Мы будем называть его *ходом стратегий* (в данной позиции).

Доказать существование ходов стратегий можно «индукцией с конца». Чтобы объяснить этот приём, введём дополнительную терминологию. Назовём позицию «хорошей», если игра, в которой она считается за начальную, имеет цену. Для каждой хорошей позиции мы выберем в качестве хода стратегий в этой позиции тот ход, который сопоставляет данной позиции какая-нибудь стратегия, гарантирующая цену игры, начинающейся в этой позиции.[1] Этот выбор важен для согласованности стратегий (которая, как будет видно ниже, важна для индуктивного рассуждения).

Теперь докажем такую лемму.

[1]Вообще говоря, может существовать много стратегий, гарантирующих один и тот же результат в игре, начинающейся из данной позиции. Мы выбираем какую-то одну из них и ход, который она сопоставляет данной позиции, объявляем ходом стратегий.

Лемма 11.7. *Если все позиции, в которые можно попасть из данной (быть может, за несколько ходов), хороши (следовательно, в каждой из них выбран ход стратегий), то и данная позиция хороша (следовательно, в ней также выбран ход стратегий).*

Что говорит эта лемма про заключительные позиции? По правилам логики хороши все позиции, в которые можно попасть из заключительной (таких позиций просто нет, так как из заключительной позиции попасть никуда нельзя). Но и утверждение леммы тривиально: в заключительных позициях результат уже известен, так что оба игрока его уже обеспечили.

Доказательство. Для доказательства леммы осталось рассмотреть случай незаключительной позиции. В ней ходит либо Макс, либо Мин. Рассуждения в обоих случаях аналогичны, разберём подробно только случай, когда в позиции x, принятой за начальную, ходит Макс.

Пусть $y_1, \ldots, y_k$ — все позиции, в которые Макс может попасть по правилам игры. По предположению все эти позиции хороши и потому имеют некоторые цены $c_1, \ldots, c_k$. Более того, мы предполагаем, что в каждой из этих позиций, как и во всех позициях, в которые можно попасть из них, выбран ход стартегий. Что должен делать Макс «при правильной игре»?

Понятное дело — он должен выбрать позицию с наибольшим c_i (одну из таких, если максимум достигается в нескольких) и пойти туда. И дальше следовать той стратегии для позиции y_i, которая гарантирует ему c_i и которая всегда выбирает ход стратегий (по условию такая существует). Так он обеспечит себе результат не меньше c_i. С другой стороны, в какую бы позицию y_j он ни пошёл, у Мина есть такая стратегия, которая обеспечивает ему результат не больше c_j и всегда выбирает ход стратегий. Заметим, что $c_j \leqslant c_i$, поскольку c_i было максимальным.

Повторим это рассуждение более формально. Не ограничивая общности, будем считать, что в начальной позиции x ходит Макс. Докажем, что цена игры с началом в x равна $c = \max(c_1, \ldots, c_k)$, где c_i — цена игры с началом в y_i, существующая по предположению индукции. Ходом стратегий в позиции x объявим ход в позицию y_i с наименьшим индексом i, для которой достигается максимум, то есть $c_i = c$.

По предположению позиция y_i хороша, то есть у Макса есть стратегия, которая выбирает ходы стратегий и гарантирует цену не ниже $c = c_i$. Значит, стратегия Макса, которая выбирает ход стратегий в каждой позиции, начиная с x, гарантирует ему результат не меньше c.

С другой стороны, стратегия Мина, которая выбирает ходы стратегий в позициях, которые возникают в партии, начинающейся с позиции x, гарантирует ему результат не больше c_i. Действительно, пусть Макс пошёл в y_j. Тогда по предположению у Мина есть стратегия, которая гарантирует Мину результат не больше $c_j \leqslant c$ (c — максимум) в игре с началом в y_j и всегда выбирает ходы стратегий.

Здесь существенен выбор ходов стратегий в каждой из позиций, благодаря чему стратегия Мина при игре из начальной позиции x корректно определена. □

Контрольный вопрос 11.11. Повторите это рассуждение для случая, когда ходит Мин.

Теперь легко доказать обещанное:

Теорема 11.8. *Для любой антагонистической игры на ациклическом графе существует цена.*

Доказательство. Как мы уже говорили, надо доказать, что все позиции в этой игре хорошие. Это можно изложить разными (но по существу эквивалентными) способами.

Первый способ. Пусть есть плохая позиция. Тогда по лемме есть позиция, в которую можно из неё перейти, которая тоже будет плохой (в частности, наша позиция не заключительная). Из этой новой плохой позиции по тем же причинам можно перейти в какую-то плохую позицию, и так далее. Рано или поздно (граф ведь конечный) получится цикл, которого быть не может — противоречие.

Второй способ. Для каждой позиции можно рассмотреть максимальное число ходов, которое из неё можно сделать. Это число — назовём его, скажем, «высотой» позиции, — определено корректно (конечно), поскольку граф ациклический. Для заключительных позиций (и только для них) высота равна нулю, и все они хорошие. Шаг индукции по высоте: если позиция x имеет высоту n, то все позиции, куда из неё можно попасть, имеют меньшую высоту, поэтому по предположению индукции они хорошие, и тем самым позиция x тоже хорошая.

Третий способ. Наконец, можно заметить, что в ациклическом графе отношение достижимости (x меньше y, если из y можно попасть в x) фундировано (нет убывающих бесконечных цепей), и можно сослаться на принцип индукции по фундированным множествам (теорема 9.6). □

Приведённое доказательство не просто доказывает теорему, но и позволяет найти цену игры и выигрышные стратегии — если только граф игры не слишком велик и помещается в память компьютера. Именно так рассчитывают шахматные окончания с небольшим количеством фигур, но мы приведём пример, который можно разобрать и вручную.

Пример 11.12 (Пристрастная игра в монетницу)**.** Рассмотрим вариант игры в монетницу на 20 монет, в котором Максу разрешается класть 2 или 3 монеты, а Мину — 1 или 4 монеты, проигрывает тот, кто не может сделать ход. Первый ход за Максом. У кого есть выигрышная стратегия?

Всего есть 42 позиции (21 вариант для числа монет надо умножить на 2, чтобы учесть, чей ход). Будем указывать не количество монет в монетнице, а количество монет, которые в монетницу ещё можно положить. Так позиция (Макс, 0) означает, что ход за Максом и в монетницу уже нельзя положить ни одной монеты.

Найдём цену игры для каждой позиции. Правила игры задают цену игры в за-

ключительных позициях. Запишем эти цены в таблицу:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Макс	-1	-1																			
Мин	$+1$																				

Из позиции (Мин, 1) Мин может сделать ровно один ход в позицию (Макс, 0). Поэтому цена игры в позиции (Мин, 1) равна -1 (Мин выигрывает). Из позиции (Мин, 2) тоже можно сделать только один ход; делая его, Мин переводит игру в позицию (Макс, 1) и тоже выигрывает.

Из позиции (Макс, 2) Макс также может сделать единственный ход в позицию (Мин, 0) и цена этой позиции $+1$. Получаем уточнённую таблицу:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Макс	-1	-1	$+1$																		
Мин	$+1$	-1	-1																		

Для позиции (Макс, 3) есть два хода. Один ведёт в позицию (Мин, 1) с ценой -1, а второй — в позицию (Мин, 0) с ценой $+1$. Для определения цены игры нужно взять максимум (ход за Максом). Поэтому цена позиции (Макс, 3) равна $+1$. Продолжая этот процесс, получим следующую таблицу:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Макс	-1	-1	$+1$	$+1$	-1	$+1$	$+1$	-1	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$
Мин	$+1$	-1	-1	$+1$	-1	-1	$+1$	$+1$	-1	$+1$	$+1$	-1	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$	$+1$

Из этой таблицы мы видим, что у Макса есть выигрышная стратегия для всех позиций, в которых больше 11 монет в монетнице независимо от очерёдности хода. Поэтому есть выигрышная стратегия, которая в позиции (Макс, 20) выбирает ход 2, равно как и есть выигрышная стратегия, которая в этой позиции выбирает ход 3. С другой стороны, в позиции (Макс, 14) ход 3 приводит к позиции, в которой выигрывает Мин. Поэтому любая выигрышная стратегия должна в позиции (Макс, 14) выбирать ход 2.

Для беспристрастных игр описание цены игры можно упростить и указывать в каждой позиции того игрока, для которого существует выигрышная стратегия в данной сокращённой позиции (без очерёдности хода). Приняты такие обозначения: позиция обозначается буквой N, если выигрышная стратегия существует для игрока, который делает ход в данной позиции, и буквой P, если выигрышная стратегия есть у игрока, который сделал ход в данную позицию.[2]

Индуктивное правило определения цены игры в данном случае также упрощается. Поскольку игроки ходят по очереди, то Первый игрок выигрывает, если у него

[2]Такое обозначение выглядит не вполне понятным для русскоязычного читателя, оно пришло из англоязычных книг по играм. “N” означает “next” (игрок, делающий следующий ход), а “P” означает “previous” (игрок, который сделал предыдущий ход). В начальной позиции “P” относится к игроку, который ходит вторым.

есть ход в позицию с ценой P (после сделанного хода Первый будет ходить вторым!). В противном случае выигрывает Второй игрок.

Пример 11.13 (продолжение примера 11.1)**.** Найдём цены позиций в первоначальном варианте игры в монетницу.

Разбором с конца нетрудно составить таблицу цены игры в монетницу. Во введённых выше N-P обозначениях получаем такую таблицу

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
P	P	N	N	N	P	P	N	N	N	P	P	N	N	N	P	P	N	N	N	P

Из этой таблицы видно, что в начальной позиции 20 выигрывает игрок, который делает ход вторым. А в позиции 19 выигрывает игрок, который делает ход первым (выигрышная стратегия сопоставляет этой позиции ход 3, так как для позиции 16 цена P).

Пример 11.14 (продолжение примера 11.4)**.** Вернёмся к игре в монетницу «на интерес». Мы уже видели, что у второго (Мина) есть стратегия, гарантирующая ему -8 (то есть выигрыш 8 монет).

Даёт ли эта стратегия цену игры? Для ответа на этот вопрос нужно понять, есть ли у Макса, который ходит первым, стратегия, гарантирующая -8.

Такая стратегия есть. Для её описания удобно использовать разметку позиций, сделанную в предыдущем примере. В позициях, которые кратны 5, Макс должен делать ход 3. В позициях, помеченных N, он должен придерживаться стратегии, гарантирующей выигрыш в обычной игре в монетницу. Стратегия Макса в остальных позициях несущественна.

Давайте оценим, какой выигрыш гарантирует данная стратегия для Макса. Если Макс оказался в N-позиции, то он заберёт в итоге монетницу. Его выигрыш при этом будет неотрицательным (дополнительно он отдаст не больше 4 монет, а получит не меньше 4: хотя бы два хода Мин сделает).

Осталось рассмотреть только тот случай, когда Макс делает ходы только в P-позициях. Глядя на таблицу, легко проверить, что партия тогда однозначно определена стратегией Макса: Макс кладёт 3 монеты, Мин кладёт 2 (и одну отдаёт Максу по правилам игры «на интерес») и т.д.

В этой партии Макс кладёт 12 монет в монетницу, но получает от Мина 4 монеты. Значит, его выигрыш равен -8.

Итак, если Макс придерживается описанной выше стратегии, то его выигрыш либо неотрицательный, либо равен -8. По определению это означает, что данная стратегия гарантирует ему выигрыш -8.

11.4 Симметричные стратегии

Разбор с конца требует анализа всех позиций игры. Для большинства игр это очень трудоёмкая, иногда непосильная, задача. Скажем, теорема 11.8 гарантирует, что в

шахматах есть цена игры.[3] То есть, либо у белых есть стратегия, гарантирующая выигрыш, либо у чёрных есть стратегия, гарантирующая выигрыш, либо у обоих игроков есть стратегии, гарантирующие ничью. Однако перебрать все шахматные позиции нереально на существующих компьютерах. Поэтому до сих пор неизвестно, какой из трёх вариантов выполняется на самом деле.

Однако доказательство существования выигрышной стратегии иногда возможно и тогда, когда разбор всех позиций игры неосуществим из-за их гигантского количества.

Одним из важных приёмов в доказательствах существования стратегий являются соображения симметрии. Приведём несколько примеров.

Пример 11.15 (Баловство с ладьёй)**.** Игровое поле: шахматная доска $n \times n$. (Настоящая шахматная доска получается при $n = 8$, но в данную игру можно играть и при других n.) На доске есть ровно одна фигура: ладья. Вначале ладья стоит в правом верхнем углу (рис. 11.1 слева). Далее игроки делают по очереди ходы. На каждом ходе игрок может сдвинуть ладью либо по горизонтали влево, либо по вертикали вниз (хотя бы на одну клетку), см. пример на рис. 11.1 в центре). Проигрывает тот, кто не может сделать ход.

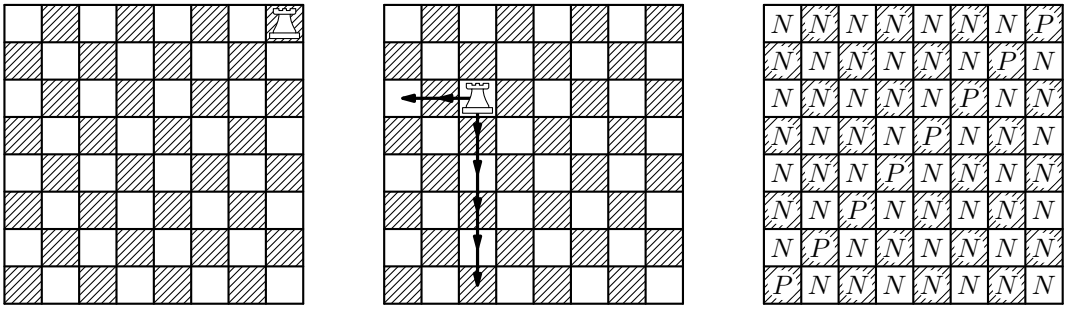


Рис. 11.1: Начальная позиция, возможные ходы и оценка позиций для баловства с ладьёй

Докажем, что позиции, в которых Первый проигрывает (P-позиции), — это диагональ из левого нижнего угла в правый верхний. Про левый нижний угол это очевидно: в этой позиции невозможно сделать ход. Стратегия Второго в остальных случаях — восстанавливать симметрию. Если Первый сделал ход, уводящий с диагонали, Второй на своём ходе может вернуть ладью на диагональ.

Если же исходное положение ладьи не на диагонали, то выигрывает Первый (N-позиция). Стратегия состоит в том, чтобы первым ходом перевести ладью на диагональ, а дальше следовать описанной выше симметричной стратегии Второго.

Пример 11.16 (монеты на стол)**.** В этой игре есть круглый стол и два игрока с неограниченным запасом одинаковых круглых монет. Игроки по очереди кладут

[3] Возможность применения к шахматам теоремы 11.8 неочевидна. Если вы знаете шахматные правила, попробуйте точно сформулировать, что считать шахматными позициями, чтобы получить конечный ациклический граф позиций.

монеты на стол так, чтобы монета полностью помещалась на столе и не накрывала уже положенные монеты. Выигрывает тот, кто положил последнюю монету.[4]

Это пример беспристрастной игры и выигрывает в ней тот, кто ходит первым. Выигрышная стратегия первым ходом помещает монету в центр стола (рис. 11.2 слева). На каждом следующем ходе Первый кладёт монету центрально-симметрично последнему ходу противника (пример на рис. 11.2 в центре).

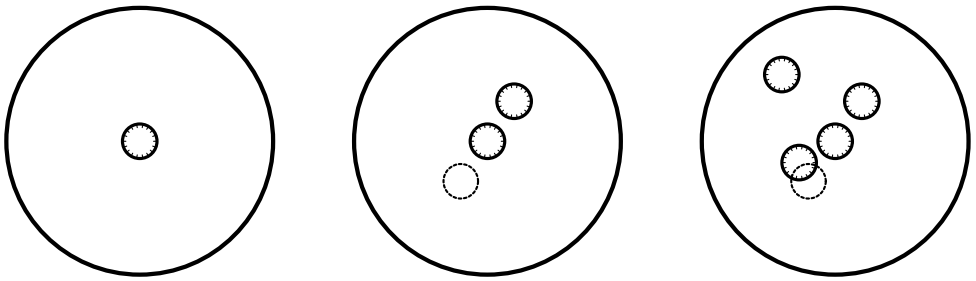


Рис. 11.2:

Обратите внимание, что эта стратегия не всюду определена: возможны позиции, в которых центрально-симметричный ход сделать невозможно, см. рис. 11.2 справа. Однако она всегда применима. После первого хода для свободной области доски выполняется такое свойство: если в какое-то место можно положить монету, то её можно положить в центрально-симметричное место. Если Первый следует описанной выше стратегии, то это свойство сохраняется после каждого хода.

Симметричная стратегия Первого гарантирует, что он всегда может сделать ход и потому не может проиграть. Значит, проигрывает Второй.

Пример 11.17 (гекс)**.** Игра «гекс» происходит на доске 11×11, составленной из правильных 6-угольников, см. рис. 11.3. Играют два игрока: Синий и Красный. Левая и правая стороны доски принадлежат Синему, а верхняя и нижняя — Красному (на рисунках синий цвет сторон обозначен горизонтальной штриховкой, а красный — вертикальной).

Игроки ходят по очереди, начинает Красный.

На каждом ходе игрок ставит на одно из свободных 6-угольных полей фишку своего цвета. Игрок выигрывает, если на доске возникает путь цвета этого игрока, соединяющий его стороны (пример см. на рис. 11.4, синий цвет обозначен заливкой, а красный — штриховкой).

В гексе не бывает ничьей[5]: если есть синий путь, соединяющий синие стороны как на рисунке, то заведомо нет красного пути, соединяющего красные стороны, аналогично для красного пути (докажите это утверждение!).

[4] В этой игре множество позиций бесконечно, так как для положений монет есть бесконечно много вариантов. Однако любая партия в этой игре конечна. Докажите, что в таком случае теорема 11.8 также справедлива.

[5] В соответствии с приведёнными правилами точнее будет сказать, что не бывает одновременного выигрыша обоих игроков.

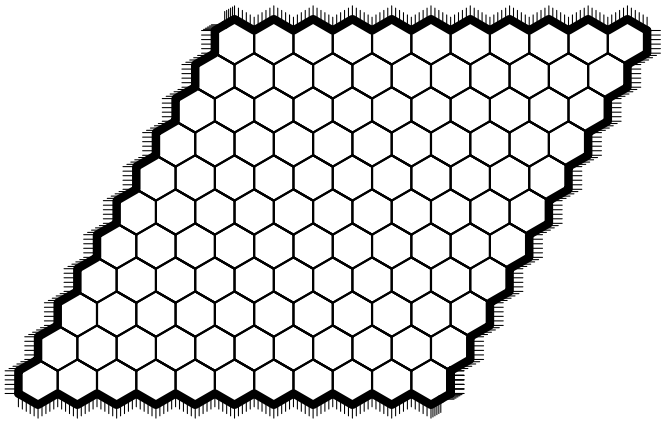


Рис. 11.3: Поле для игры в гекс

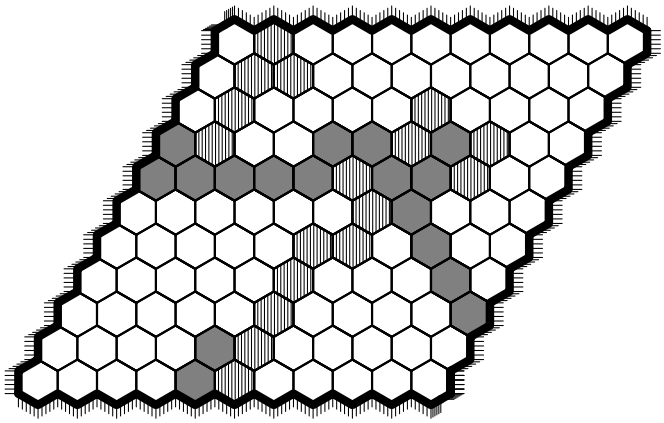


Рис. 11.4: Выигрыш Синего

Поэтому из теоремы 11.8 заключаем, что у одного из игроков есть выигрышная стратегия. Более того, этот игрок— Красный.

Это следует из симметрии доски и монотонности условия выигрыша. При отражении относительно диагонали синие и красные стороны меняются местами, а произвольное добавление на доску фишек своего цвета не ухудшает результат игрока (путь между красными сторонами не пропадёт, если добавить на доску красных фишек).

Докажем от противного существование выигрышной стратегии для Красного. Предположим, что выигрышная стратегия α есть у Синего. Рассмотрим такую стратегию для Красного: сделать первый ход произвольно, а на каждом следующем ходе играть по стратегии α, гарантирующей выигрыш Синего с точностью до симметрии (то есть отражать позицию на доске, меняя красные фишки на синие и наоборот, выбирать ход по стратегии α за Синего и делать ход в симметричную позицию). Может оказаться так, что поле, на которое нужно сделать ход по данному правилу,

уже занято. Тогда Красный делает ход на любое свободное поле.

В этом описании есть неточность. Позиция при ходе Красного содержит одинаковое количество красных и синих фишек, а при ходе Синего красных фишек на одну больше. Поэтому Красный мысленно удаляет с поля ту фишку, которую он поставил первой, и только после этого выбирает ход, симметричный ходу по стратегии α, как описано выше.

Почему это выигрышная стратегия? Предположим, что партия, в которой Красный придерживался симметричной α стратегии, закончилась проигрышем Красного. Последний ход Синего создал синий путь между синими сторонами. Но это означает, что в симметричной позиции у Красного был ход, создающий красный путь между красными сторонами, что невозможно по предположению о том, что α — выигрышная стратегия для Синего.

Гекс интересен тем, что хотя цена игры известна, играть в неё достаточно интересно. Доказательство от противного не даёт никакого намёка на то, как играть Красному в реальной партии.

11.5 Ним

Игра «ним» — ещё один пример беспристрастной игры. Имеется два игрока и три кучки камней. За один ход игрок может взять из какой-то кучки любое количество камней (хотя бы один нужно взять). Проигрывает тот игрок, который не может сделать ход.

Из такого описания видно, что (сокращённая) позиция в игре ним — это три натуральных числа. Причём позиция $(0, 0, 0)$ по определению проигрышная для первого игрока и выигрышная для второго (мы, как и выше, нумеруем игроков в том порядке, в каком они вступают в игру). В принятых нами обозначениях это P-позиция. Про остальные позиции нужно выяснить, кто в них выигрывает (напомним, что более точно говорить «имеет выигрышную стратегию»).

Часть случаев мы уже фактически разобрали раньше.

Позиции $(n, 0, 0)$ — это по сути позиции игры в штриховку (пример 11.3). Число n означает количество незаштрихованных квадратиков. Поэтому при $n > 0$ такие позиции выигрышны для Первого (N-позиции).

Позиции $(n, m, 0)$ — это позиции игры «баловство с ладьёй» (пример 11.15). Числа (n, m) — это попросту координаты ладьи при стандартном понимании координат (начало — левый нижний угол, координаты возрастают слева направо и снизу вверх).

Поэтому при $n \neq m$ позиции $(n, m, 0)$ являются N-позициями, а при $n = m$ являются P-позициями.

Отсюда также следует, что позиции (n, m, m) также выигрышны для Первого: своим первым ходом он забирает n камней из первой кучки и получается P-позиция $(0, m, m)$.[6]

[6]Заметим, что порядок кучек неважен и оценка игры в позиции не изменяется при перестановке кучек.

Разбором с конца легко проверить, что позиция $(1, 2, 3)$ является P-позицией (выигрышна для Второго).

Задача 11.18. Проведите полностью разбор с конца для позиции $(1, 2, 3)$.

Но тогда позиции $(k, 2, 3)$, где $k \neq 1$; $(k, 1, 3)$, где $k \neq 2$; $(k, 1, 2)$, где $k \neq 3$, являются N-позициями.

Если $k > 3$, то из каждой такой позиции Первый может перейти в позицию $(1, 2, 3)$ и далее следовать выигрышной стратегии Второго для этой позиции.

Задача 11.19. Докажите это утверждение при остальных значениях k.

Разбор с конца можно продолжать и расширять список позиций, для которых известна оценка игры. Прежде чем читать далее, попробуйте сделать такой разбор ещё для нескольких типов позиций и угадать общий ответ. Это не так легко сделать — ответ весьма неожиданный!

Пусть мы анализируем позицию (x, y, z). Запишем числа x, y, z в двоичной системе, начиная с младших битов:

$$\begin{aligned} x &= x_0 x_1 x_2 \ldots, \\ y &= y_0 y_1 y_2 \ldots, \\ z &= z_0 z_1 z_2 \ldots. \end{aligned}$$

Сложим биты в каждом столбце по модулю 2, получим двоичную последовательность

$$a_0 a_1 a_2 \ldots$$

Оказывается, у Первого игрока есть выигрышная стратегия тогда и только тогда, когда последовательность a_i содержит хотя бы одну 1.

Пример 11.20. Оценим ним в позиции $(1, 2, 3)$. Двоичная запись даёт

$$\begin{aligned} 1 &= 10, \\ 2 &= 01, \\ 3 &= 11. \end{aligned}$$

Поразрядная сумма по модулю 2 этих последовательностей равна 00. Значит, это P-позиция, как и утверждалось выше.

Пример 11.21. Оценим ним в позиции $(5, 15, 26)$. Двоичная запись даёт

$$\begin{aligned} 5 &= 10100, \\ 15 &= 11110, \\ 26 &= 01011. \end{aligned}$$

Поразрядная сумма по модулю 2 этих последовательностей равна 00001. Сформулированное правило утверждает, что Первый выигрывает. Пока неясно, в чём состоит его стратегия.

На самом деле, если поверить в справедливость сформулированного выше правила, то стратегия Первого на первом ходе ясна: нужно перейти в позицию, где среди поразрядных сумм нет единиц, — в такой позиции делающий первый ход проигрывает. Такой ход есть: нужно из 26 камней забрать 16. Получится позиция $(5, 15, 10)$.

Но почему позиции, в которых все порязрядные суммы по столбцам чётные, проигрышны для Первого? Опять-таки, если верить правилу, то на каждый ход Первого Второй должен ответить так, чтобы вернуться в позицию, в которой все порязрядные суммы чётные. Для каждой конкретной позиции это можно проверить перебором вариантов. Но, видимо, пора переходить к общему доказательству.

Теорема 11.9. *В игре ним с кучками из x, y, z камней выигрышная стратегия у Второго существует тогда и только тогда, когда поразрядная сумма по модулю* 2 *двоичных разрядов чисел x, y, z равна* 0*.*

Доказательство. Индукция по общему числу камней в кучках $x + y + z$.

База индукции: 0 камней, поразрядная сумма нулевая, выигрывает Второй, так как Первый не может сделать ход.

Предположим, что теорема доказана для всех позиций с общим количеством камней $< N$. Рассмотрим позицию с N камнями. Удобно её представлять как таблицу из трёх строк и какого-то количества столбцов. Таблица заполнена нулями и единицами.

Ход состоит в том, что в одной из строк часть нулей заменяется на единицы и наоборот. Конечно, нужно позаботиться, чтобы число, представленное новым набором нулей и единиц, было меньше числа, представленного исходным набором нулей и единиц. Для определённости мы считаем, как и в примерах выше, что старшинство разрядов идёт по столбцам слева направо.

Доказательство шага индукции разбивается на доказательство двух утверждений:

(I) если в таблице есть столбец с нечётным числом единиц, то можно сделать ход, который приведёт в таблицу, каждый столбец которой содержит чётное число единиц (в силу предположения индукции это означает, что в исходной позиции есть выигрышная стратегия у Первого);

(II) если в таблице все столбцы содержат чётное количество единиц, то любой допустимый ход приводит в таблицу, в которой есть столбец с нечётным числом единиц (в силу предположения индукции это означает, что в исходной позиции есть выигрышная стратегия у Второго).

Доказательство (I). Возьмём самый правый столбец, в котором нечётное количество единиц (а значит, хотя бы одна единица есть). Выберем строку, в которой в выбранном столбце стоит 1. Ход делаем в этой строке. Заменим единицу в выбранном столбце на 0. Во всех предыдущих столбцах действуем по следующему правилу: если в столбце чётное количество единиц, то ничего не меняем, а если нечётное — инвертируем бит в данном столбце. После этих действий в каждом столбце будет

чётное число единиц. Но нужно проверить, что такие изменения отвечают допустимому ходу. Для этого вспомним, что числа сравниваются в двоичной записи от старших разрядов к младшим. Мы изменили в каком-то разряде (столбце) 1 на 0, а далее все изменения происходили в разрядах, которые младше (левее). Поэтому полученная двоичная запись будет давать число, меньшее исходного.

Пример 11.22. Пусть дана таблица

0	1	1	0	1	1	1	0	1	число 374
1	0	1	1	0	0	0	1	0	число 141
1	0	1	0	1	1	0	1	0	число 181

Сделав ход по указанному правилу, получим таблицу

0	0	0	1	1	1	0	0	0	число 56
1	0	1	1	0	0	0	1	0	число 141
1	0	1	0	1	1	0	1	0	число 181

Задача 11.23. Придумайте таблицу, в которой ход по указанному правилу изменяет количество камней на 1.

Доказательство (II). Сейчас мы предполагаем, что в каждом столбце таблицы чётное количество единиц. Допустимый ход уменьшает количество камней. Поэтому в каком-то разряде (столбце) 1 заменилась на 0. После такого хода в этом столбце чётность числа единиц изменится. □

11.6 Сумма игр и функция Шпрага–Гранди

Правило оценки позиции в игре ним согласуется с играми, в которых одна или две кучки камней. Оказывается, можно утверждать и больше: это правило допускает обобщение на произвольное количество кучек и даже на более общую ситуацию.

Будем рассматривать беспристрастные игры, в которых выигрывает сделавший последний ход, а граф ходов ациклический (то есть невозможны бесконечные партии). Примером является обобщение игры ним на произвольное количество кучек камней (будем называть такую игру k-ним, где k — исходное количество кучек). А все игры такого вида будем называть играми обобщённого нима.

Анализ игр обобщённого нима состоит в том, что позициям игры сопоставляются позиции игры 1-ним (одна кучка, см. пример 11.3), то есть натуральные числа. Число, сопоставленное позиции, будем называть *оценкой позиции*.

Основное требование к оценке: она должна согласовывать правила данной игры и игры 1-ним.

Сформулируем это требование более точно. Пусть оценка позиции s в некоторой игре обобщённого нима равна v. В игре 1-ним из позиции с v камнями в кучке есть ход в позицию с любым меньшим количеством камней. Для согласования игр потребуем, чтобы для любого $v' < v$ существовал ход из s в позицию s' с оценкой v'. Назовём это условие *условием гомоморфизма игр*.

В 1-ниме все позиции с положительным количеством камней в кучке являются N-позициями (выигрышная стратегия у того, кто делает первый ход). А позиция с 0 камней, как уже отмечалось выше, является P-позицией (Первый не может сделать ход и потому проигрывает).

Мы хотим, чтобы эта классификация позиций сохранялась и в обобщённом ниме с данной оценкой. Вспоминая правила вычисления цены позиции в беспристрастной игре на выигрыш, получаем два условия, необходимые и достаточные для того, чтобы P-позициями были в точности позиции с нулевой оценкой:

(1) из каждой позиции с положительной оценкой существует ход в позицию с нулевой оценкой;

(2) из каждой позиции с нулевой оценкой все ходы ведут в позиции с положительной оценкой.

Условие (1) обеспечивается сформулированным выше условием гомоморфизма (это его частный случай). А условие (2) нужно обеспечивать отдельно, поскольку в 1-ниме позиция с 0 камней заключительная, из неё нет ходов.

Элегантный способ обеспечить оба условия предоставляет *функция Шпрага–Гранди*. Эта функция задаётся системой уравнений на значение оценки в позициях. В этой системе уравнений используется не вполне привычная функция mex, которая равна наименьшему натуральному числу, не встречающемуся среди её аргументов.

Пример 11.24. $\operatorname{mex}(0,2,3) = 1$; $\operatorname{mex}(0,1,2,10) = 3$, $\operatorname{mex}(2,3) = 0$, $\operatorname{mex}() = 0$. Как видно из этих примеров, область определения функции mex — последовательности натуральных чисел. Последний пример показывает значение этой функции на пустой последовательности.

У функции mex есть два свойства, непосредственно вытекающих из определения:

(1) если среди аргументов есть 0, то $\operatorname{mex}(\cdot) > 0$;

(2) если все аргументы положительные, то $\operatorname{mex}(\cdot) = 0$.

Эти свойства напоминают условия на оценку, которые мы хотим выполнить. Так что не очень удивительно, что функция mex возникает в уравнениях на оценку.

Определение 11.10. Функцией Шпрага–Гранди обобщённой игры ним называется функция $v\colon S \mapsto \mathbb{N}$ из множества позиций в натуральные числа, удовлетворяющая системе уравнений

$$v(s) = \operatorname{mex}(v(s_1), v(s_2), \dots), \tag{11.1}$$

в которой левые части пробегают всё множество позиций, а для каждой позиции s функция mex в правой части применяется к оценкам тех позиций s_1, s_2, …, в которые возможен ход из s.

Определение замысловатое, поэтому полезно посмотреть на примеры функций Шпрага–Гранди для очень простых игр.

Контрольный вопрос 11.25. Проверьте, что в заключительных позициях функция Шпрага–Гранди равна 0.

Пример 11.26. Вычислим функцию Шпрага–Гранди для 1-нима. Позиции нумеруем количеством камней в кучке.

Тогда $v(0) = 0$, поэтому

$$\begin{aligned} v(1) &= \operatorname{mex}(0) = 1, \\ v(2) &= \operatorname{mex}(0,1) = 2, \\ v(3) &= \operatorname{mex}(0,1,2) = 3 \end{aligned}$$

и т.д. По индукции получаем $v(n) = n$.

Пример 11.27. Вычислим функцию Шпрага–Гранди игры в монетницу. Позиции нумеруем количеством монет в монетнице.

Тогда $v(0) = v(1) = 0$ (это заключительные позиции), далее получаем

$$\begin{aligned} v(2) &= \operatorname{mex}(0) = 1, \\ v(3) &= \operatorname{mex}(0,0) = 1, \\ v(4) &= \operatorname{mex}(0,1) = 2, \\ v(5) &= \operatorname{mex}(1,1) = 0, \\ v(6) &= \operatorname{mex}(1,2) = 0, \\ v(7) &= \operatorname{mex}(0,2) = 1 \end{aligned}$$

и по индукции можно проверить, что далее функция Шпрага–Гранди периодична с периодом 5, т.е. $v(n+5) = v(n)$.

Хотя из соображений, предваряющих определение функции Шпрага–Гранди, уже ясно, что она позволяет находить цены позиций в обобщённом ниме, докажем этот факт строго.

Теорема 11.11. *Пусть $v\colon S \mapsto \mathbb{N}$ — функция Шпрага–Гранди. Тогда P-позиции — это в точности те позиции, для которых $v(s) = 0$. Остальные позиции являются N-позициями.*

Доказательство. Как и доказательстве теоремы 11.8, применяем индукцию по порядку достижимости на графе возможных ходов.

Теперь утверждение $A(s)$, которое доказывается индукцией по порядку достижимости, состоит в том, что $v(s) = 0$ равносильно для любой позиции s тому, что s является P-позицией.

Для индуктивного перехода нужно доказать такое утверждение: если для всех позиций s', достижимых из s, выполняется $A(s')$ (предположение индукции), то справедливо $A(s)$.

Если $v(s) = 0$, то все ходы из s ведут в позиции с положительными значениями функции Шпрага–Гранди и по предположению индукции все эти позиции являются N-позициями. Как уже обсуждалось выше, это означает, что s является P-позицией (выигрышная стратегия есть у Второго).

Контрольный вопрос 11.28. Убедитесь, что по правилам формальной логики рассуждение из предыдущего абзаца применимо к заключительным позициям.

Если $v(s) > 0$, то существует ход из позиции s в позицию s', для которой $v(s') = 0$. По предположению индукции позиция s' является P-позицией. Поэтому s является N-позицией (выигрышная стратегия есть у Первого — сделать ход в s' и следовать далее выигрышной стратегии Второго в игре, начинающейся из s').

Мы доказали индуктивный переход. По принципу индукции получаем отсюда утверждение теоремы. □

Из определения неясно, почему функция Шпрага–Гранди существует — ведь система уравнений может оказаться несовместной. Существование решения системы уравнений требует специального рассуждения.

Теорема 11.12. *Для любой игры обобщённого нима существует функция Шпрага–Гранди и она однозначно определена.*

Доказательство. Заметим, что система (11.1) такова, что в правой части каждого уравнения стоят лишь такие позиции, которые в порядке достижимости меньше позиции из левой части уравнения. Поэтому если уже удалось подобрать оценку, которая удовлетворяет уравнениям из системы (11.1) для всех позиций, которые меньше позиции s в порядке достижимости, то оценка в позиции s однозначно определяется правой частью уравнения

$$v(s) = \operatorname{mex}(v(s_1), v(s_2), \dots)$$

из системы (11.1).

Далее используем индукцию по порядку достижимости. Утверждение, которое будем доказывать по индукции, состоит в том, что для позиции s и всех меньших её в порядке достижимости можно найти решение уравнений из системы (11.1), в левых частях которых стоят эти позиции, причём это решение единственно.

Выше мы объяснили, почему выполняется шаг индукции для индукции по фундированному множеству позиций с порядком достижимости. Значит, это утверждение верно для всех позиций.

Выберем для каждой позиции s то единственное значение оценки, удовлетворяющей уравнениям из системы (11.1) для этой позиции и всех меньших. Это и есть функция Шпрага–Гранди: она удовлетворяет все уравнениям системы (11.1). Единственность уже обеспечена построением. □

В некоторых случаях игре можно сопоставлять не только 1-ним, но и k-ним. Это возможно для тех игр, которые являются *суммами игр*, и позволяет обобщить для них оценку позиций для игры ним.

Пусть есть две игры обобщённого нима с множествами позиций P_1 и P_2. Суммой $P_1 \oplus P_2$ называется игра, в которой позиции — это пары позиций в первой и второй игре, а ход состоит в переходе от позиции (p_1, p_2) либо к позиции (p_1', p_2), либо к

позиции (p_1, p_2'), где в первой игре возможен ход из позиции p_1 в позицию p_1', а во второй — из позиции p_2 в позицию p_2'.

Другими словами это можно объяснить так: игроки параллельно играют в игры P_1 и P_2, на каждом ходу игрок должен сделать ход в одной из игр.

Аналогично определяется и сумма бóльшего количества игр: игроки параллельно играют в несколько игр и на каждом ходу игрок должен сделать ход в одной из игр.

Контрольный вопрос 11.29. Проверьте, что k-ним — это сумма k игр 1-ним.

Теорема 11.13. *Пусть $v_1\colon P_1 \to \mathbb{N}$; $v_2\colon P_1 \to \mathbb{N}$ — функции Шпрага–Гранди для игр P_1, P_2. Тогда функция Шпрага–Гранди для игры $P_1 \oplus P_2$ задается как*

$$v(p_1, p_2) = v(p_1) \oplus v(p_2),$$

где $x \oplus y$ — число, двоичная запись которого является поразрядной суммой по модулю 2 двоичных записей чисел x и y.

Задача 11.30. Проверьте, что из теоремы 11.13 следует правило оценки позиций игры в ним с тремя кучками.

В доказательстве теоремы 11.13 нам потребуются свойства поразрядного сложения по модулю 2.

Лемма 11.14. *Для поразрядного сложения по модулю 2 выполняются следующие свойства.*

1. *$x \oplus y = y \oplus x$;*
2. *если $a \neq a'$, то $a \oplus x \neq a' \oplus x$ для любого x;*
3. *если $x < a \oplus b$, то либо $x = a' \oplus b$, $a' < a$; либо $x = a \oplus b'$, $b' < b$.*

Доказательство. Первое свойство очевидно из определения.

Если поразрядно прибавить к двум разным двоичным строкам одну и ту же строку, полученные суммы будут различаться в тех же самых позициях, что и исходные строки. Отсюда следует второе свойство.

Теперь докажем третье свойство. В самом старшем разряде, в котором x отличается от $a \oplus b$, в двоичной записи числа x стоит 0, а в двоичной записи $a \oplus b$ стоит 1. Поэтому в двоичных записях чисел a, b в этом разряде ровно одна единица. Считаем без ограничения общности, что в этом разряде запись a содержит 1, а запись b содержит 0.

Построим число a', поместив в этот разряд 0, а в более младших разрядах поставим такие значения, чтобы выполнялось равенство $x = a' \oplus b$ (число a' однозначно задаётся этими условиями и числами x, b). По правилу сравнения чисел $a' < a$, что и требовалось. □

Доказательство теоремы 11.13. Для позиций, из которых нельзя сделать ход, утверждение теоремы очевидно, так как $0 \oplus 0 = 0$.

Теперь проверим, что число $v(p_1) \oplus v(p_2)$ отличается от всех чисел $v(p_1') \oplus v(p_2)$; $v(p_1) \oplus v(p_2')$, где из p_1 есть ход в p_1' в игре P_1, а из p_2 есть ход в p_2' в игре P_2.

По построению функции Шпрага–Гранди $v(p_1') \neq v(p_1)$; $v(p_2') \neq v(p_2)$. Поэтому достаточно применить второе свойство из леммы 11.14.

Осталось доказать, что $v(p_1) \oplus v(p_2)$ — наименьшее из чисел, которые отличаются от всех чисел $v(p_1') \oplus v(p_2)$; $v(p_1) \oplus v(p_2')$. Для этого используем третье свойство из леммы 11.14.

Пусть $k < v(p_1) \oplus v(p_2)$. Тогда по указанному свойству либо $k = v_1 \oplus v(p_2)$, $v_1 < v(p_1)$; либо $k = v(p_1) \oplus v_2$, $v_2 < v(p_2)$. В обоих случаях из свойств функции Шпрага–Гранди соответствующей игры следует, что в первом случае $v_1 = v(p_1')$, а во втором $v_2 = v(p_2')$. Поэтому $k = v(p_1') \oplus v(p_2)$ или $k = v(p_1) \oplus v(p_2')$, причём в первом случае в первой игре есть ход из p_1 в p_1', а во втором случае во второй игре есть ход из p_2 в p_2'. □

Задача 11.31. Рассмотрим игру в три монетницы (ним с тремя кучками камней, разрешается брать 2 или 3 камня за ход). У кого из игроков есть выигрышная стратегия, если в монетницу помещается 12 монет?

Часть III

Вычислимость

Лекция 12

Разрешающие деревья

12.1 Задача об угадывании числа. Деление пополам. Мощностная нижняя оценка

Рассмотрим следующую игру. Алиса загадывает натуральное число от 1 до N, а Боб пытается это число отгадать. При этом Бобу разрешается задавать вопросы, на которые Алиса может ответить "да" или "нет", и Алиса должна на эти вопросы давать правильные ответы. Цель Боба состоит в том, чтобы задать как можно меньше вопросов. При этом мы не хотим полагаться на удачу, то есть нужно, чтобы число вопросов было гарантировано небольшим. Другими словами, мы хотим найти такое минимальное k, что у Боба есть алгоритм, позволяющий отгадать число за не более чем k вопросов, какое бы число ни загадала Алиса.

Оказывается, что Бобу всегда достаточно задать не более чем $\lceil \log_2 N \rceil$ вопросов. Чтобы это доказать мы воспользуемся *методом деления пополам*. Идея в том, что Боб каждым своим вопросом будет сокращать количество оставшихся возможных чисел примерно в два раза. Этого можно добиться, например, так. Обозначим число, загаданное Алисой, через x. На каждом шаге Боб будет знать, что x лежит в некотором "отрезке" $\{y \mid a \leqslant y \leqslant b\}$ для каких-то a и b. Изначально $a = 1$ и $b = N$. На очередном шаге Боб будет вычислять $c = \lfloor (a + b)/2 \rfloor$ и спрашивать, верно ли, что $x \leqslant c$. Если Алиса отвечает "да", то Боб переходит к отрезку $\{y \mid a \leqslant y \leqslant c\}$ и повторяет процедуру. Иначе, Боб переходит к отрезку $\{y \mid c + 1 \leqslant y \leqslant b\}$ и также повторяет процедуру. Нетрудно видеть, что каждый раз длина отрезка уменьшается почти в два раза (если в отрезке было нечетное число точек, то в следующем отрезке может оказаться чуть больше чем половина точек). Так что через приблизительно $\log_2 N$ шагов в отрезке останется одна точка и Боб узнает число Алисы. На самом деле, нетрудно доказать, что достаточно $\lceil \log_2 N \rceil$ вопросов, что мы сейчас и сделаем.

Чтобы максимально упростить оценку числа вопросов мы прибегнем к небольшому трюку и немного модифицируем алгоритм. Обозначим $k = \lceil \log_2 N \rceil$ и $N' = 2^k$. Видно, что $N' \geqslant N$. Пусть теперь Боб изначально считает, что Алиса может загадать число от 1 до N'. Поскольку на самом деле Алиса может загадывать только

числа от 1 до N, то Боб только рассматривает дополнительные возможности, которые никогда не будут реализовываться, и тем самым, Боб только усложняет свою задачу. Но при этом видно, что если Боб в предыдущем алгоритме начнет с отрезка $\{y \mid 1 \leqslant y \leqslant N'\}$, то на каждом шаге в каждом отрезке будет четное число точек и каждый отрезок будет делиться ровно пополам (N' – степень двойки). Так что, чтобы длина отрезка стала равной 1, потребуется ровно $\log_2 N' = k$ вопросов.

Задача 12.1. Мы доказали, что в модифицированном алгоритме потребуется не более $\lceil \log_2 N \rceil$ вопросов, но не доказали, что для изначального (не модифицированного) алгоритма верна такая же оценка (почему?). Докажите, что и для изначального алгоритма эта оценка верна.

Оказывается, что доказанная только что оценка точная: не существует алгоритма, который для всякого загаданного Алисой числа задавал бы меньше $\lceil \log_2 N \rceil$ вопросов. Сейчас мы докажем эту *нижнюю оценку* сложности нашей задачи.

Для доказательства нижней оценки мы применим так называемый *мощностной метод*. Пусть у Боба есть какой-то алгоритм сложности k (то есть, в нем всегда задается не более k вопросов), Алиса загадала какое-то число и Боб задал свои вопросы. Рассмотрим цепочку ответов Алисы. Для удобства будем обозначать ответ “да” цифрой 1, а ответ “нет” цифрой 0. Тогда последовательность ответов Алисы – это последовательность из 0 и 1 длины не больше k. Заметим, что для двух разных загаданных Алисой чисел последовательности не могут совпадать. Действительно, если для двух различных x и y Алиса дает Бобу на его вопросы полностью одинаковые ответы, то для Боба эти случаи неразличимы: его диалоги с Алисой для x и для y выглядят одинаково. При этом Боб после этого диалога выдает какой-то ответ, который определяется только состоявшимся диалогом. Значит в одном из случаев его ответ будет неправильным. Далее, заметим, что не может быть так, что для двух различных x и y, загаданных Алисой, цепочка ответов для x является началом цепочки ответов для y. Действительно, иначе диалог Боба с Алисой выглядит одинаково для x и y до того момента, когда будут заданы все вопросы из цепочки ответов для x. Значит к этому моменту Боб не может отличить x от y и должен делать для них одно и то же, тогда как он в одном случае задает следующий вопрос, а в другом нет.

Таким образом, мы получили, что каждому числу от 1 до N соответствует последовательность из не более чем k нулей и единиц, все эти последовательности различны, и ни одна не является началом другой. Заметим, что семейство этих последовательностей содержит не более 2^k элементов. Действительно, если какая-то из них имеет длину меньше k, то продолжим ее, например, нулями. Тогда для различных x и y полученные последовательности длины k различны: иначе они либо совпадают, либо одна (более короткая) является началом другой. Таким образом, каждому числу от 1 до N соответствует последовательность длины k из нулей и единиц, и все эти последовательности различны. Всего последовательностей длины k из нулей и единиц 2^k. По принципу Дирихле, чисел от 1 до N должно быть не больше 2^k (иначе двум разным числам соответствуют одинаковые последователь-

ности). Значит $N \leqslant 2^k$, то есть $k \geqslant \log_2 N$. Поскольку k –целое число, то отсюда следует, что $k \geqslant \lceil \log_2 N \rceil$.

Таким образом, мы получили следующий результат.

Лемма 12.1. *Для угадывания числа от* 1 *до* N *необходимо и достаточно* $\lceil \log_2 N \rceil$ *вопросов.*

12.2 Формализация модели

Рассуждения прошлого раздела не очень формальны. Например, мы не определяли, что мы подразумеваем под алгоритмом. В этом разделе мы формализуем задачу и заодно рассмотрим ее общую постановку.

Начнем с общей постановки задачи. Пусть фиксирована некоторая функция $f\colon A \to B$, где A, B – какие-то конечные множества. Требуется найти $f(x)$, но x явно не сообщается. Разрешается задавать вопросы вида $x \in S$ для подмножеств S множества A. Можно заметить, что в примере с угадыванием числа любой вопрос с ответом “да” или “нет” сводится к вопросу о том, принадлежит ли загаданное число некоторому подмножеству – подмножеству тех чисел, для которых ответ “да”.

Теперь формализуем нашу вычислительную модель. Протоколом вычисления (или разрешающим деревом) мы будем называть двоичное дерево, каждая промежуточная вершина которого (не лист) помечена некоторым подмножеством $S \subseteq A$. Каждый лист помечен элементом $b \in B$. Из каждой промежуточной вершины, выходит три ребра: одно к корню и два к листьям. Для каждой промежуточной вершины одно из ребер, ведущих к листьям, помечено единицей, а другое – нулем.

Вычисление согласно протоколу происходит следующим образом. Мы будем строить путь из корня дерева в какой-то из листьев. Элемент множества B, которым помечен лист, будет результатом вычисления. В начале пути мы находимся в корне дерева. Корень, как и всякая промежуточная вершина, помечен каким-то подмножеством множества A. Если вход x принадлежит этому подмножеству, то мы переходим по ребру, помеченному единицей, иначе – по ребру помеченному нулем. Если следующая вершина – лист, то путь построен. Если же она является промежуточной вершиной, то мы повторяем процедуру: спрашиваем, лежит ли x в подмножестве, которым помечена текущая вершина, и переходим по ребру, помеченному единицей, если x лежит в подмножестве, и по ребру, помеченному нулем, иначе. Мы говорим, что протокол вычисляет функцию f, если для всякого $x \in A$ протокол выдает $f(x)$. Сложностью протокола называется глубина дерева (нетрудно видеть, что она равна числу вопросов, которое потребуется задать в худшем случае).

12.3 Угадывание числа, неадаптивный вариант задачи

Когда мы рассматривали задачу об угадывании числа, то следующие вопросы могли зависеть от ответов на предыдущие. Такие вычислительные модели обычно называют *адаптивными*. Можно также рассмотреть и неадаптивную постановку той же

задачи: в ней вопросы не должны зависеть от ответов на предыдущие вопросы. Можно считать, что Боб должен составить на бумажке список вопросов и передать его Алисе. Алиса должна ответить на все эти вопросы и после этого Боб должен назвать загаданное число.

Во-первых, можно заметить, что неадаптивная модель слабее адаптивной: задача Боба стала только сложнее. Это значит, что в неадаптивной модели Бобу потребуется не меньше вопросов, чем в адаптивной. Действительно, если Боб может угадать число за k вопросов в неадаптивной модели, то он может сделать то же самое и в адаптивной модели – достаточно просто задать те же вопросы.

Таким образом, в новой модели Бобу также потребуется не меньше $\lceil \log_2 N \rceil$ вопросов, чтобы угадать число от 1 до N. Но хватит ли такого числа вопросов и в этой модели?

Оказывается, что ответ на этот вопрос положительный. Есть несколько способов объяснить, почему это так. Мы приведем два.

Первое рассуждение – прямое и универсальное. Можно заметить, что рассуждение для адаптивного алгоритма напрямую не проходит – вопросы зависят от ответов на предыдущие. Но это можно исправить за счет удлинения и усложнения вопросов. Для этого достаточно добавить перебор случаев в вопросы. Первый вопрос остается таким же, как и в адаптивном протоколе. Второй вопрос будет иметь следующий вид:

> “Если ответ на первый вопрос был ‘да’, то верно ли, что ..., если же ответ на первый вопрос был ‘нет’, то верно ли, что ... ?”,

где на места многоточий нужно подставить вторые вопросы из дерева адаптивного протокола. В общем виде, l-ый вопрос будет иметь следующий вид:

> “Верно ли следующее утверждение: (ответы на предыдущие вопросы были ‘нет’, ‘нет’, ..., ‘нет’ и ...) или (ответы на предыдущие вопросы были ‘нет’, ‘нет’, ..., ‘да’ и ...) или ... или (ответы на предыдущие вопросы были ‘да’, ‘да’, ..., ‘да’ и ...)”,

где в ‘или’ перебираются все варианты ответов на предыдущие вопросы, а вместо многоточий в скобках нужно подставить соответствующие вопросы из адаптивного алгоритма. Видно, что вопросы получаются очень длинными, но у нас нет ограничений на длину вопроса.

Предыдущее рассуждение универсально в том смысле, что оно работает для любой задачи такого типа и любого адаптивного алгоритма. В нашей частной задаче об угадывании числа это же рассуждение можно изложить очень коротко и просто: i-ым вопросом Боб спрашивает, верно ли, что i-й бит двоичной записи загаданного числа x – единица. После $k = \lceil \log_2 N \rceil$ вопросов Боб знает всю двоичную запись числа x, а значит знает само загаданное число.

Задача 12.2. Осознайте, что два приведенных выше алгоритма – по существу один и тот же.

12.4 Ограниченные модели разрешающих деревьев. Сортировка, взвешивания, булевы функции

Часто кроме описанной выше общей модели рассматриваются модели разрешающих деревьев с разными ограничениями. В большинстве из них накладываются ограничения на множества S, о которых можно задавать вопросы. Мы рассмотрим несколько таких примеров.

Сортировка. Первый пример – это задача о сортировке. Неформальная формулировка этой задачи такая. Дано n объектов, все разного веса. За один шаг разрешается сравнить веса двух объектов (мы узнаем, какой из этих объектов тяжелее). Требуется расположить эти объекты в порядке возрастания веса.

Опишем теперь задачу формально. Удобно считать, что объекты изначально расположены в виде последовательности. Обозначим в этой последовательности самый тяжелый объект единицей, второй по тяжести – двойкой, и так далее, самый легкий объект обозначим n. Таким образом, нам на вход по существу подается перестановка n-элементного множества. Чтобы упорядочить объекты по возрастанию нам нужно найти данную перестановку. Таким образом, в этом примере A – множество перестановок n-элементного множества и требуется вычислить тождественную функцию на A, то есть $f(x) = x$ для всякого $x \in A$.

При этом нам разрешается задавать не любые вопросы, а только вопросы о сравнении двух элементов перестановки. Формально это означает, что в вершинах разрешающего дерева могут стоять не любые подмножества множества перестановок, а только множества $S_{i,j}$ для $i, j = 1, \ldots, n$, состоящие из всех перестановок $(a_1, \ldots, a_n)$, в которых $a_i > a_j$.

Заметим, что если бы не было ограничения на вид множеств, то задача была бы полностью аналогична задаче об угадывании числа: на вход подается один из $n!$ объектов и требуется угадать, какой именно. Поскольку у нас добавляется ограничение на тип вопросов, то наша задача усложняется, а значит в задаче о сортировке требуется не меньше вопросов.

Следствие 12.2. *Сложность задачи о сортировке n объектов не меньше* $\lceil \log_2 n! \rceil$.

Что касается верхней оценки, то из оценки для задачи угадывания числа так сразу ничего не следует. Мы можем доказать следующую оценку.

Лемма 12.3. *Сложность задачи о сортировке n объектов не больше* $\sum_{k=1}^{n} \lceil \log_2 k \rceil$.

Доказательство. Доказательство будем вести индукцией по n. Для $n = 1$ оценка верна – никаких сравнений не требуется.

Пусть утверждение доказано для n, докажем его для $n + 1$. Сначала возьмем первые n объектов и упорядочим их, пользуясь предположением индукции. После этого у нас остается $\lceil \log_2(n + 1) \rceil$ сравнений, и нам нужно один оставшийся объект поместить в уже упорядоченный список из n объектов. То есть, для $(n + 1)$-го объекта есть $n + 1$ место среди упорядоченного списка из n объектов и нам нужно его

найти. Пользуясь тем же рассуждением, что и в задаче про угадывание числа, это можно сделать как раз за $\lceil \log_2(n+1) \rceil$ сравнение (сравниваем со средним объектом и сокращаем количество возможных позиций почти в два раза). □

Итак, мы получили верхнюю и нижнюю оценку числа сравнений, необходимого для сортировки n объектов. Можно заметить, что наша верхняя оценка есть $O(n \log n)$, а наша нижняя оценка есть $\Omega(n \log n)$.[1] Так что порядок роста сложности задачи о сортировке при росте n в нашей модели мы установили. Тем не менее, наши верхняя и нижняя оценка не совпадают. Более того, они обе не точны. Для $n = 5$ наша верхняя оценка дает 8 сравнений, тогда как можно убедиться, что сортировку можно сделать за 7 сравнений (попробуйте это сделать). Для $n = 12$ наша нижняя оценка дает 29 сравнений, тогда как на самом деле за 29 сравнений этого сделать также нельзя (в этом можно убедиться компьютерным перебором). Точное значение числа сравнений, необходимое и достаточное для сортировки для произвольного n неизвестно.

Взвешивания. Второй пример – это задачи на взвешивание. Для примера мы рассмотрим задачу поиска самого тяжелого из n объектов разного веса. Более точно, есть n объектов, за один ход разрешается сравнить по весу два из них. Требуется найти самый тяжелый объект. Видно, что формально модель точно такая же, как и в предыдущем примере. Меняется только функция, которую мы хотим вычислить: теперь по данной перестановке мы хотим найти номер позиции, в которой стоит число 1.

Лемма 12.4. *Для нахождения самого тяжелого из n объектов необходимо и достаточно $n - 1$ взвешивания.*

Доказательство. Сначала докажем, что $n-1$ взвешивания достаточно. Проще всего вести рассуждение по индукции. Если $n = 1$, то ничего взвешивать не нужно. Пусть мы доказали утверждение для $n - 1$. Рассмотрим n объектов. Возьмем любые два и сравним их. Заметим, что более легкий из них не может быть самым тяжелым, так что его можно выбросить из рассмотрения. Таким образом у нас остается $n - 1$ объект и по предположению индукции мы можем найти самый тяжелый из них за $n - 2$ оставшихся взвешивания.

Теперь докажем, что меньше чем за $n - 1$ взвешивание найти самый тяжелый объект нельзя. Пусть мы сделали $n - 2$ взвешивания. Рассмотрим следующий граф. Его вершинами будут наши объекты, и мы соединяем ребрами те из них, которые мы сравнили в одном из взвешиваний. Тогда в этом графе n вершин и $n - 2$ ребра. Значит этот граф не связен. Рассмотрим множество V_1 объектов в одной из его компонент связности и множество V_2 всех остальных объектов. Предположим, для определенности, что самый тяжелый объект находится в V_1. Увеличим вес всех объектов в V_2 на одно и то же очень большое число, такое чтобы все объекты в V_2 стали

[1] Напомним, что $\Omega(f(n))$ обозначает функции, для которых $f(n)$ является *нижней* асимптотической оценкой с точностью до константного множителя. (Отличие от $O(f(n))$ в знаке сравнения: «больше», а не «меньше».

тяжелее всех объектов в V_1. При этом результаты всех взвешиваний не изменятся, поскольку все сравнения были либо внутри V_1, либо внутри V_2, а самая тяжелая монета станет другой (теперь она будет в V_2). Таким образом, все взвешивания дадут один и тот же результат в обеих ситуациях, а самый тяжелый объект будет разным. Значит в одной из двух ситуаций наш протокол выдает неправильный ответ. Мы пришли к противоречию, а значит для нахождения самого тяжелого объекта требуется не меньше $n-1$ сравнения. □

В этом месте уместно вспомнить о неадаптивной модели. Мы видели, что в общей модели разрешающих деревьев разницы между адаптивной и неадаптивной моделью нет — сложность в обоих случаях получается одинаковая. Но это достигается за счет использования весьма нетривиальных запросов в неадаптивной модели. Так что, если мы добавляем ограничения на вид запросов, то естественно ожидать, что разница между адаптивной и неадаптивной моделью все же появится. Это хорошо видно на примере задачи поиска самого тяжелого объекта.

Лемма 12.5. *Для нахождения самого тяжелого из n объектов в неадаптивной модели необходимо и достаточно $\binom{n}{2} = n(n-1)/2$ взвешиваний.*

Доказательство. Верхняя оценка здесь получается совсем просто – достаточно сравнить попарно все объекты друг с другом. Ясно, что если мы сравнили все объекты, то мы можем сказать, какой из них самый тяжелый.

Для доказательства нижней оценки предположим, от противного, что у нас есть протокол, который делает меньше $\binom{n}{2}$ сравнений. Заметим, что сейчас у нас модель неадаптивная, так что протокол – это по существу просто список всех вопросов. Раз вопросов в нем меньше $\binom{n}{2}$, значит какие-то два объекта между собой не сравниваются. Рассмотрим два таких входа, при котором эти два объекта тяжелее всех остальных и в первом случае, первый объект тяжелее, а во втором – второй (а все остальные объекты сравниваются друг с другом одинаково). Тогда наш протокол на этих входах получит одни и те же ответы, а значит выдаст один и тот же результат. Поскольку самые тяжелые объекты в этих двух входах разные, наш протокол на одном из них ошибается, а значит мы пришли к противоречию. □

Булевы функции. Третий пример – это разрешающие деревья для булевых функций. В этой модели требуется вычислить булеву функцию $f\colon \{0,1\}^n \to \{0,1\}$, то есть на вход подается $\vec{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ и при этом за один ход разрешается спрашивать значение одной переменной из $x_1, \ldots, x_n$. Формально в рамках нашей общей модели это означает, что в промежуточных вершинах разрешающего дерева могут быть написаны только подмножества множества $\{0,1\}^n$, состоящие из всех векторов, в которых одна фиксированная координата одинакова. Но на самом деле удобно в случае булевых функцию в промежуточных вершинах дерева просто писать переменную, которую мы спрашиваем, и в пути, соответствующему вычислению, переходить по ребру помеченному нулем, если эта переменная равна нулю, и по ребру помеченному единицей, если переменная равна единице.

Сложностью разрешающих деревьев для булевой функции f называют минимальную сложность (то есть, глубину) дерева, вычисляющего f. Эту величину обычно обозначают через $D(f)$.

Нетрудно заметить, что для всякой $f\colon \{0,1\}^n \to \{0,1\}$ верно $D(f) \leqslant n$. Действительно, достаточно просто спросить последовательно все переменные. После этого нам известен вход функции, и мы можем выдать ответ (как выглядит дерево этого протокола?). Оказывается, что для большинства функций $D(f) = n$, но мы не будем останавливаться на этом подробно.

12.5 Рассуждение с противником

В последних двух леммах рассуждения в доказательствах нижних оценок во многом похожи. И там, и там мы предполагали, что протокол есть, рассматривали его и подбирали вход таким образом, чтобы протокол ошибался. Таким образом, мы пользовались тем, что протокол должен работать для всех входов, и по существу, рассматривали "худший случай" для протокола.

У этого рассуждения есть другая (игровая) форма, которая оказывается более удобной для доказательства нижних оценок: *рассуждение с противником*. Представим себе, что вход для алгоритма выбирается не произвольно, а есть некое лицо (противник), которое его выбирает. При этом противник может выбирать вход не заранее, а по ходу работы протокола, вычисляющего функцию. То есть, изначально противник вход не фиксирует, а по мере поступления запросов от протокола дает на них ответы так, чтобы ответы были с каким-то входом согласованы, и при этом противник стремится заставить протокол задать как можно больше вопросов. Таким образом, по существу противник старается реализовать"худший случай". Это довольно просто понять на примере самой первой задачи об угадывании числа: Алиса вместо того, чтобы загадывать число заранее, может выбирать его по ходу дела так, чтобы Боб из ответов на вопросы получал как можно меньше информации.

Чем хороша такая форма рассуждения? Для этого полезно подумать о том, что нужно сделать для доказательства верхних и нижних оценок. Чтобы доказать верхнюю оценку нужно *построить* протокол. А чтобы доказать нижнюю оценку нужно доказать, что протокола *нет*. Если в первом случае нужно сделать что-то вполне конструктивное, то во втором не сразу ясно, что вообще делать. Рассуждение с противником позволяет свести доказательство нижней оценки также к чему-то конструктивному: вместо того, чтобы доказывать, что протокола *нет* (то есть, нет стратегии для вычисляющего игрока), мы можем *строить* стратегию для противника, которая будет работать против всех протоколов.

По сути, мы рассматриваем нашу модель как игру. В игре есть два игрока: протокол и противник. Протокол задаёт вопросы, а противник даёт на них ответы. Цель протокола – найти значение функции за не более чем k вопросов, для какого-то фиксированного k, а цель противника – помешать протоколу это сделать. Теория игр учит, что один из игроков при правильной игре может гарантировать себе победу, как бы ни играл другой. Наше использование этой игровой интерпретации такое.

Мы хотим доказать, что протокол не может гарантировать себе победу. Для этого мы показываем, что это может сделать противник.

Чтобы лучше понять происходящее, полезно вспомнить уже рассмотренные задачи и понять, как получить в них нижние оценки с помощью рассуждений с противником. В задаче об угадывании числа противником является Алиса. Рассмотрим для неё такую стратегию. На каждом шаге Алиса помнит множество тех чисел, которые согласованы со всеми данными ранее ответами. Изначально это множество есть просто множество всех чисел от 1 до N. При каждом вопросе множество Алисы разбивается на два подмножества: те числа, для которых ответ на новый вопрос 'да', и те числа, для которых ответ на новый вопрос 'нет'. Алиса выбирает большее из двух подмножеств и дает ответ, согласованный с этим подмножеством. Таким образом, множество Алисы за один шаг уменьшается не более чем в два раза. И, пока множество не станет одноэлементным, протокол не может выдать правильный ответ. Отсюда получается та же самая нижняя оценка $\lceil \log_2 N \rceil$.

В случае задачи о нахождении самого тяжёлого объекта из n объектов в адаптивной модели противнику даже не нужна никакая специальная стратегия ответов на вопросы. Он просто выбирает изначально какой-то вход, даёт ответы в соответствии с ним и ждёт пока протокол (задающий не более $n-2$ вопросов) выдаст какой-то ответ, а затем изменяет свой вход, добавив большой вес ко всем объектам в одной из компонент связности (см. доказательство леммы 12.4). При этом все ответы согласованы с новым входом, а ответ протокола становится неправильным.

В случае задачи о нахождении самого тяжёлого объекта из n объектов в неадаптивной модели протокол должен сразу сообщить противнику список вопросов. В этом случае стратегия противника уже по существу была нами описана. Нужно просто выбрать два объекта, которые протокол не сравнивает, самыми тяжёлыми, а после выдачи протоколом ответа, выбрать какой из них сделать тяжелее.

Связность графа. Рассмотрим более содержательный пример рассуждения с противником.

А именно, рассмотрим следующую задачу. Дан неориентированный граф G на вершинах $\{1, \ldots, n\}$. За один ход разрешается спрашивать наличие или отсутствие конкретного ребра. Нужно проверить, является ли граф связным, то есть выдать 1, если является, и 0 иначе. Заметим, что эта задача – частный случай модели разрешающих деревьев для булевых функций. Действительно, можно считать, что рассматривается функция от множества переменных $\{x_{ij} \mid 1 \leqslant i < j \leqslant n\}$, где $x_{ij} = 1$ тогда и только тогда, когда между вершинами i и j есть ребро. Обозначим функцию через $CONN$. Всего переменных у функции $\binom{n}{2} = n(n-1)/2$, а значит, как мы упоминали выше, $D(CONN) \leqslant \binom{n}{2}$.

Для начала докажем, что сложность разрешающих деревьев для этой функции квадратична. Позже мы усилим эту оценку.

Лемма 12.6. *Пусть n – чётно. Тогда $D(CONN) \geqslant n^2/4$.*

Доказательство. Опишем стратегию противника, вынуждающую протокол задать не меньше $n^2/4$ запросов.

Поделим вершины графа на две равные доли A и B, по $n/2$ вершин в каждой. Противник отвечает на все вопросы о рёбрах между долями отрицательно, а на остальные вопросы – положительно. Тогда, пока протокол не задаст вопросы про все рёбра, между A и B он не может сказать, связен ли граф: если между A и B рёбер нет, то граф не связен, но если хотя бы одно ребро присутствует, то он будет связен. Всего рёбер между A и B как раз $n^2/4$. □

Таким образом, мы доказали верхнюю и нижнюю квадратичную оценку сложности разрешающих деревьев для проверки графа на связность. Оказывается мы можем установить точное значение сложности этой задачи.

Лемма 12.7. $D(CONN) \geqslant \binom{n}{2}$.

Доказательство. Опять же, опишем стратегию противника. Для этого в каждый момент времени будем рассматривать два графа MAX и MIN на том же самом множестве вершин $\{1, \ldots, n\}$. В MIN будут только те ребра, про которые противник сказал "да", а в MAX – те, про которые противник не сказал "нет". Таким образом, MIN – это минимальный возможный граф, согласованный с уже данными ответами, а MAX – максимальный. Стратегия противника такая. Если его спрашивают про ребро e, то он отвечает "нет", в том случае если MAX после этого остаётся связным, а иначе отвечает "да".

Нам нужно доказать, что при такой стратегии противника протоколу придётся задать вопросы про все ребра. Заметим, что по определению стратегии противника MAX всегда связен. Кроме того, если $MIN = MAX$, то про каждое ребро был задан вопрос (с ответом «да» для рёбер из MIN и с ответом «нет» для остальных рёбер). Наш план – доказать, что если $MIN \neq MAX$, то MIN не связен. Это означает, что мы не знаем значение функции: с текущими ответами согласован как некоторый связный граф, так и некоторый не связный.

Заметим, что если в MAX есть цикл, то ни одно его ребро не принадлежит MIN. Действительно, пусть это не так. Рассмотрим ребро цикла, которое попало в MIN первым. Это означает, что противник ответил на вопрос об этом ребре положительно, то есть граф MAX при удалении этого ребра переставал быть связным. Но этого не может быть, потому что в любом пути, проходящем через это ребро, его можно заменить обходом по циклу. Противоречие.

В частности, из этого получается, что в MIN нет циклов: иначе такой цикл лежал бы и в MAX и все его ребра были бы при этом в MIN.

Теперь, если бы MIN был связен, то он был бы остовным деревом для MAX. При этом $MAX \neq MIN$, то есть в MAX есть ребро, которого нет в MIN. Тогда это ребро вместе с графом MIN содержит цикл, а значит мы нашли цикл в MAX, часть ребер которого (все, кроме одного) лежат в MIN. Противоречие. Получается, что MIN не связен. □

Заметим, что в принципе рассуждение с противником не является обязательным: всякое рассуждение с противником можно перевести на язык "худшего случая". Однако это сделает, например, последнее доказательство заметно труднее. Рассужде-

ние с противником – это удобный способ излагать и придумывать доказательства нижних оценок.

Лекция 13

Булевы схемы и формулы

В этой главе мы рассматриваем частный случай алгоритмов. Они представляются программами, состоящими только из присваиваний. Например:

```
sec_in_hour := minutes_in_hour * sec_in_minute
sec_in_day := hour_in_day * sec_in_hour
sec_in_year:= sec_in_day * sec_in_day
```

После исполнения трёх таких присваиваний переменная `sec_in_year` будет равна количеству секунд в году, если, конечно, остальным переменным присвоены правильные значения.

В общем случае последовательность присваиваний вычисляет некоторую функцию. В каждом присваивании аргументы и значения промежуточных присваиваний входят как аргументы некоторой функции (в примере это функция умножения).

Программы такого вида и называются в теоретической информатике *схемами*.

В реальных программах встречаются более сложные присваивания, в которых правая часть является формулой:

```
hypotenuse := sqrt(cathetusA * cathetusA + cathetusB * cathetusB)
```

С математической точки зрения формулы — это тоже схемы специального вида. Мы обсудим этот вопрос позже, см. раздел 13.2.

Как указано в названии главы, мы ограничиваемся частным случаем, когда и аргументы, и значения функций булевы. Это не слишком ограничивает общность с практической точки зрения: ведь данные в программах представляются в двоичном виде и потому все преобразования данных являются булевыми отображениями (функциями вида $f\colon \{0,1\}^n \to \{0,1\}^m$).

Два основных вопроса про схемы: существует ли схема, вычисляющая данную функцию? насколько сложны схемы, вычисляющие данную функцию? Первый вопрос мы уже обсуждали раньше в главе 5, раздел 5.5, хотя и не вводили формального определения схемы. В этой главе нас главным образом интересует второй вопрос.

13.1 Булевы схемы

Дадим формальные определения.

Булевой схемой от переменных $x_1, \ldots, x_n$ мы будем называть последовательность булевых функций $g_1, \ldots, g_s$, в которой всякая g_i получается из предыдущих функций последовательности и переменных применением одной из логических операций отрицание, конъюнкция и дизъюнкция. Другими словами, для всякого i имеет место одно из равенств

$$\begin{array}{llll} g_i = g_j \wedge g_k & (j,k<i), & g_i = g_j \vee g_k & (j,k<i), \\ g_i = g_j \wedge x_k & (j<i), & g_i = g_j \vee x_k & (j<i), \\ g_i = x_j \wedge x_k, & & g_i = x_j \vee x_k, & \\ g_i = \neg g_j & (j<i), & g_i = \neg x_k. & \end{array}$$

Имея в виду эти связи между элементами последовательности (схемы), будем также называть элементы схемы *присваиваниями*.

В булевой схеме также задано некоторое число $m \geqslant 1$ и члены последовательности $g_{s-m+1}, \ldots, g_s$ называются *выходами схемы* (их как раз m). Число m называют числом выходов схемы. Мы говорим, что схема вычисляет булево отображение $f\colon \{0,1\}^n \to \{0,1\}^m$, если $f(x) = (g_{s-m+1}(x), \ldots, g_s(x))$ для всякого $x \in \{0,1\}^n$. *Размером схемы* называют число s, то есть количество присваиваний в схеме.

Замечание 13.1. Выбор конъюнкции, дизъюнкции и отрицания связан с тем, что они образуют полный набор: через эти связки выражается любая булева функция (теорема 5.1 говорит, что достаточно даже только конъюнкции и отрицания).

Ничто не мешает рассматривать более общие схемы, в которых используются другие функции вместо выбранного нами стандартного набора связок.

Хотя в основном мы рассматриваем схемы со стандартным набором связок, иногда нам потребуются и схемы более общего вида.

Приведём простые примеры схем. Для краткости мы указываем только правые части присваиваний.

Пример 13.1. Схема

$$x_1 \wedge x_2$$

с одним выходом вычисляет конъюнкцию переменных x_1 и x_2. Размер этой схемы равен 1.

Схема

$$\neg x_1,\ \neg x_2,\ x_1 \wedge \neg x_2,\ \neg x_1 \wedge x_2,\ (x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2) \tag{13.1}$$

с одним выходом вычисляет, как нетрудно проверить, функцию $x_1 \oplus x_2$. Размер этой схемы равен 5.

Схема

$$x_1 \vee x_2,\ (x_1 \vee x_2) \wedge x_3,\ (x_1 \wedge x_2),\ ((x_1 \vee x_2) \wedge x_3) \vee (x_1 \wedge x_2) \tag{13.2}$$

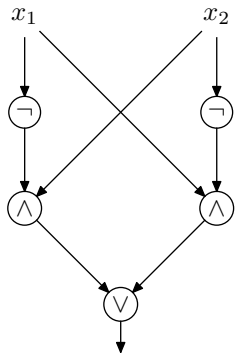


Рис. 13.1: Схема для функции $x_1 \oplus x_2$

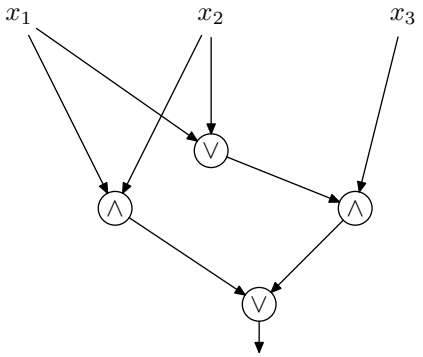


Рис. 13.2: Схема для функции MAJ_3

с одним выходом вычисляет функцию $MAJ_3(x_1, x_2, x_3)$. Эта функция равна 1 тогда и только тогда, когда хотя бы две из её переменных равны 1. Размер этой схемы равен 4.

Часто схемы представляют не в виде последовательности функций, а в виде ориентированного графа. Это можно делать по-разному. Нам будет удобно представление схемы ориентированным графом с $n + s$ вершинами $x_1, \ldots, x_n, v_1, \ldots, v_s$ (см. примеры на рис. 13.1, 13.2). Вершины этого графа соответствуют функциям и переменным схемы. Вершина v_i соответствует функции g_i. Вершина x_j соответствует переменной x_j. Всякая вершина v_i помечена логической связкой, с помощью которой функция g_i получена из предыдущих. При этом, если функция g_i была получена из функций g_j и g_k, мы проводим ребра из вершин v_j и v_k в вершину v_i. Если функция g_i получена как отрицание g_j, то мы проводим только ребро из v_j в v_i. Вершины $v_{s-m+1}, \ldots, v_s$ помечены как выходные вершины схемы (из них проведены выходящие стрелки).

Контрольный вопрос 13.2. Проверьте, что граф любой схемы ациклический.

Обратный переход — от графа к последовательности присваиваний — неоднозначен. Нужно выбрать одну из нумераций вершин v_i, в которой рёбра между вершинами идут от меньших номеров к бóльшим. Это всегда возможно, вспомните теорему 3.9 на с. 128. Но таких нумераций обычно много.

Контрольный вопрос 13.3. Напишите схему для графа на рис. 13.2, которая отличается от схемы (13.2) из примера 13.1.

Как говорилось во введении, нас интересует сложность схем, которые вычисляют функции. Основной мерой сложности является размер.

Определение 13.1. *Схемная сложность* булева отображения $f\colon \{0,1\}^n \to \{0,1\}^m$ (в частности, булевой функции) — это наименьший размер схемы, вычисляющей это отображение.

Рассмотрим примеры оценки схемной сложности сверху. Такая оценка получается, если предъявить схему, вычисляющую заданное булево отображение.

Пример 13.4. Пусть нам даны две n-битовых двоичных записи чисел x и y и мы хотим вычислить двоичную запись их суммы $z = x + y$. Для удобства обозначим $x = x_{n-1} \dots x_1 x_0$, где x_0 – младший разряд двоичной записи. Аналогично, $y = y_{n-1} \dots y_1 y_0$. Во-первых, заметим, что в двоичной записи z будет не более $n+1$ разрядов. Так что мы хотим построить схему с $2n$ входами и $n+1$ выходом.

Идея конструкции схемы будет та же, что и в обычном школьном сложении в столбик. Мы будем складывать числа x и y поразрядно, попутно вычисляя биты переноса в следующий разряд.

Для удобства будем обозначать через b_i бит, который переносится в i-ый разряд из предыдущих.

Заметим, что мы уже готовы вычислить первый разряд ответа $z_0 = x_0 \oplus y_0$. Конечно, мы не можем сразу применить операцию $\oplus$, но выше мы показали, как её можно вычислить небольшой схемой (13.1). Добавим эту маленькую схему в нашу как подсхему. Далее, заметим, что $b_1 = x_0 \wedge y_0$, добавим соответствующий элемент в схему. Перейдём к следующему разряду. Здесь $z_1 = x_1 \oplus y_1 \oplus b_1$ и $b_2 = MAJ_3(x_1, y_1, b_1)$. Для вычисления первого добавим сначала подсхему, вычисляющую промежуточную величину $c_1 = x_1 \oplus y_1$, а затем подсхему, вычисляющую $z_1 = c_1 \oplus b_1$. Для вычисления b_2 просто добавим подсхему, вычисляющую функцию MAJ_3. Такая схема также приведена выше. Дальше, случай произвольных z_i и b_i полностью аналогичен случаю z_1 и b_1 и мы можем последовательно вычислить все эти значения.

На рис. 13.3 изображена получившаяся схема, точнее, её начальная часть. Мы используем на этом рисунке дополнительную вольность, изображая подсхемы в виде блоков с входящими и выходящими стрелками. Другими словами, мы построили схему, используя более широкий набор присваиваний, а затем реализовали схемой со стандартными связками каждую из более сложных функций (в нашем случае это $x \oplus y$, $x \oplus y \oplus z$, $MAJ_3(x, y, z)$).

Оценим теперь размер описанной схемы. Для каждого разряда ответа нам нужно не больше двух раз применить подсхему для вычисления функции $\oplus$ и не более одного раза подсхему для вычисления MAJ_3. Все эти схемы имеют фиксированный размер, так что для вычисления каждого разряда z мы используем фиксированное число элементов, не зависящее от числа входных переменных. Поэтому всего в схеме $O(n)$ элементов.

Пример 13.5. Построим теперь схему для умножения n-битовых чисел. Пусть на вход снова подаются два числа $x = x_{n-1} \dots x_1 x_0$ и $y = y_{n-1} \dots y_1 y_0$. На этот раз мы хотим вычислить $z = x \cdot y$. Заметим, что z имеет не больше $2n$ разрядов. Действительно, $x, y < 2^n$, так что $z = x \cdot y < 2^{2n}$, а значит для его записи достаточно $2n$ разрядов.

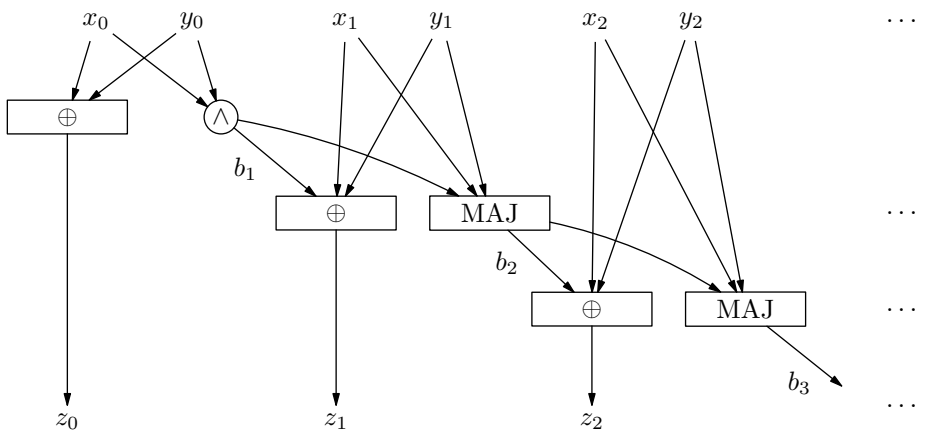


Рис. 13.3: Схема для сложения положительных целых чисел

Для вычисления z снова воспользуемся школьным методом. В нем умножение двух чисел сводится к сложению n чисел. Действительно, чтобы умножить x на y достаточно для всякого $i = 0, \ldots, n-1$ умножить x на y_i, приписать в конце числа i нулей и затем сложить все полученные числа.

Умножение x на y_i легко реализуется с помощью n конъюнкций. Чтобы приписывать нули, их нужно иметь. В нашем определении не разрешается использовать константы. Поэтому нуль нужно вычислить. Для этого годится, например, такая схема

$$\neg x_1,\ x_1 \wedge \neg x_1 = 0. \tag{13.3}$$

После этого остаётся сложить n чисел длины не более $2n$. Для этого мы можем $n-1$ раз применить схему для сложения, описанную выше. Размер каждой схемы для сложения линейный, так что суммарная сложность схемы для умножения получается $O(n^2)$.

Пример 13.6. Мы разобрали, как в модели булевых схем выполнять базовые арифметические операции. Теперь обсудим одну из простейших комбинаторных задач. Дан неориентированный граф G на n вершинах. Нужно проверить, является ли он связным.

Во-первых, стоит обсудить, как задавать граф в пригодном для схем виде. Для этого удобно воспользоваться так называемой *матрицей смежности* графа. Перенумеруем вершины графа $v_1, v_2, \ldots, v_n$. Матрицей смежности графа G называется матрица $A \in \{0,1\}^{n \times n}$, в которой на пересечении строки i со столбцом j стоит 1 тогда и только тогда, когда в графе есть ребро v_i–v_j. Это фактически та таблица $edge[i, j : 0..n-1] : \texttt{Boolean}$, которой задавался неориентированный граф в главе 3, с. 106.

Матрица смежности полностью описывает, какие пары вершин соединены рёбрами, так что булевой схеме достаточно подать на вход матрицу смежности графа. Более того, заметим, что матрица смежности симметрична и на диагонали у неё обязательно стоят нули (мы запрещали петли — рёбра, ведущие из вершины в неё же саму). Так что на вход схеме можно подать, скажем, только верхнюю половину матрицы смежности.

Оказывается, матрица смежности также удобна для проверки связности графа. Можно матрицу A интерпретировать следующим образом: на пересечении строки i и столбца j написано количество путей длины 1 из вершины v_i в вершину v_j. Теперь возведём матрицу A в квадрат (над действительными числами). Если посмотреть на формулу для произведения матриц можно заметить, что на пересечении строки i и столбца j матрицы A^2 записано количество путей длины 2 из вершины v_i в вершину v_j. По индукции можно доказать, что на пересечении строки i и столбца j матрицы A^k записано число путей длины k из вершины v_i в вершину v_j. Заметим теперь, что если между какими-то двумя вершинами в графе есть путь, то обязательно есть путь длины не больше $n-1$ (из пути всегда можно выкинуть циклы, если они там есть, после этого все вершины в пути разные). Так что для проверки связности у нас появляется такой план: вычислим все матрицы $A, A^2, \ldots, A^{n-1}$ и для каждой пары вершин v_i и v_j проверим, есть ли между ними путь длины не больше $n-1$. Если это так, то граф связен, иначе не связен.

Этот план можно несколько упростить. Во-первых, можно немного модифицировать матрицу смежности. Рассмотрим матрицу A', которая отличается от матрицы A тем, что у неё на главной диагонали стоят единицы, а не нули (в остальном матрицы совпадают). В терминах графов это означает, что к каждой вершине мы добавляем петлю. В модели простых неориентированных графов мы этого не допускали, но ничего не мешает нам рассмотреть графы с петлями. Идея состоит в том, что теперь, если между двумя вершинами есть путь длины меньше $n-1$, то есть и путь длины ровно $n-1$ (достаточно добавить к пути нужное количество петель). Так что теперь не обязательно смотреть на все степени матрицы смежности, достаточно взглянуть на $(A')^{n-1}$. Если в ячейках этой матрицы нет нулей, то граф связен, иначе не связен.

Второе упрощение связано со способом возведения матрицы в степень. Выше мы считали число путей и для этого нужно складывать и умножать целые числа. Чтобы делать это с помощью булевых схем, нам придётся использовать описанные выше в примерах 13.4 и 13.5 схемы для сложения и умножения. Чтобы оценить размер получившейся схемы, придётся оценивать величину возникающих в процессе вычислений целых чисел. От этих сложностей можно избавиться.

Решение состоит в том, чтобы вместо умножения матриц над целыми числами воспользоваться так называемым *булевым умножением матриц*. В нем формулы для умножения матриц такие же, как и в обычном умножении, только вместо операции умножения используется конъюнкция, а вместо сложения – дизъюнкция. Тогда по индукции можно доказать, что в (булевой) матрице $(A')^k$ на пересечении строки i и столбца j стоит 1 тогда и только тогда, когда в графе есть путь из v_i в v_j длины

не больше k.

Теперь мы готовы описать схему для проверки графа на связность. На вход схема (по существу) получает матрицу смежности A'. Схема последовательно вычисляет булевы степени этой матрицы $(A')^2, \ldots, (A')^{n-1}$. Затем схема вычисляет конъюнкцию всех ячеек матрицы $(A')^{n-1}$ и подаёт её на выход.

Оценим размер получившийся схемы. Для булева умножения двух булевых матриц $n \times n$ достаточно $n^2 \cdot O(n) = O(n^3)$ операций (каждая ячейка произведения матриц вычисляется за линейное число операций, всего ячеек n^2). Всего нам нужно $(n-1)$ умножение матриц, так что для вычисления матрицы $(A')^{n-1}$ достаточно $O(n^4)$ операций. На последний этап (конъюнкция ячеек $(A')^{n-1}$) нужно $O(n^2)$ операций, итого получается $O(n^4) + O(n^2) = O(n^4)$ операций.

Задача 13.7. При построении схемы, реализующей описанный нами алгоритм, и при подсчёте числа её элементов мы действовали довольно грубо. То же самое можно сделать за $O(n^3 \log n)$ операций. Как это сделать?

Мы увидели, что конкретные функции можно вычислять схемами, причём размер схем получается не слишком большим (полиномиальным). Далее заметим, что всякую булеву функцию можно вычислить булевой схемой. По существу мы уже обсуждали этот вопрос, когда обсуждали дизъюнктивную нормальную форму функции. Для полноты изложения повторим кратко это рассуждение.

Лемма 13.2. *Всякую функцию $f\colon \{0,1\}^n \to \{0,1\}$ можно вычислить схемой размера не больше $O(n2^n)$.*

Доказательство. Для всякого $a \in \{0,1\}^n$ рассмотрим такую функцию $f_a\colon \{0,1\}^n \to \{0,1\}$, что $f(x) = 1$ тогда и только тогда, когда $x = a$. Будет удобно ввести обозначения $x^1 = x$ и $x^0 = \neg x$. Тогда функцию f_a можно записать формулой

$$f_a(x) = \bigwedge_{i=1}^{n} x_i^{a_i},$$

где $x = (x_1, \ldots, x_n)$ и $a = (a_1, \ldots, a_n)$.

Произвольная функция f выражается через функции f_a с помощью дизъюнкции:

$$f(x) = \bigvee_{a \in f^{-1}(1)} f_a(x).$$

Эти формулы без труда переделываются в схему. Наша схема сначала будет вычислять отрицания всех переменных, на это нужно n элементов. После этого можно вычислить все функции f_a. Для вычисления каждой такой функции нужно $n-1$ раз применить конъюнкцию. Всего получается $2^n(n-1)$ элемент. Наконец, для вычисления f нужно взять дизъюнкцию нужных функций f_a, на это уйдёт не более 2^n элементов. Суммарно в нашей схеме получается $O(n2^n)$ элементов. □

Из этого несложно получить схему для произвольной функцию с многими выходами $f\colon \{0,1\}^n \to \{0,1\}^m$.

Следствие 13.3. *Всякое булево отображение* $f\colon \{0,1\}^n \to \{0,1\}^m$ *можно вычислить схемой размера не больше* $O(nm2^n)$.

Доказательство. Для данного отображения f зададим функции $f_i\colon \{0,1\}^n \to \{0,1\}$ для $i = 1, \ldots, m$ так, что $f(x) = (f_1(x), \ldots, f_m(x))$ для всякого $x \in \{0,1\}^n$. Иными словами f_i выдаёт i-ю координату f. Мы можем вычислить каждую функцию f_i схемой размера $O(n2^n)$. Объединив эти схемы в одну, мы получим схему для f размера $O(mn2^n)$. □

Мы показали, что всякую функцию можно вычислить схемой, но размер схемы при этом получился большим: экспоненциальным по числу переменных. Оказывается, это неизбежно — существуют функции экспоненциальной схемной сложности.

Теорема 13.4. *Для всякого* $n \geqslant 10$ *существует функция* $f\colon \{0,1\}^n \to \{0,1\}$, *которую нельзя вычислить схемой размера меньше* $2^n/10n$.

Доказательство. Для доказательства применим мощностной метод: докажем, что функций больше, чем маленьких схем. Тогда маленьких схем не хватит, чтобы вычислить все функции.

Всего булевых функций от n переменных 2^{2^n}.

Заметим, что если схемная сложность функции не больше S, то существует схема размера ровно S, вычисляющая эту функцию (добавим в схему столько присваиваний $x_1 \wedge x_1$, сколько нужно для выравнивания размера).

Оценим (весьма грубо, но для наших целей такой оценки будет достаточно) количество схем размера S от n переменных. Для этого заметим, что всякую схему размера S с n переменными можно описать с помощью не больше чем $S \cdot 2(1+\log(n+S))$ битов. Для описания схемы удобно её расширить, добавив в начало все переменные.

Теперь для каждого из S элементов схемы нужно указать его тип (конъюнкция, дизъюнкция, отрицание), на что достаточно потратить два бита. Кроме того, нужно указать, к каким из предыдущих элементов применяется операция. Достаточно указать номера элементов в расширенной последовательности, начинающейся со всех переменных схемы. На это требуется не более $2(1 + \log(n + S))$ битов.

Мы применим эту оценку на длину описания схемы при $S = \lfloor 2^n/10n \rfloor$. В этом случае, как легко проверить, $S > n$ при $n \geqslant 10$. Оценка на длину описания упрощается:

$$S \cdot 2(1 + \log(n + S)) \leqslant 2S \cdot (2 + \log_2 S) \leqslant 4S \log S.$$

Таким образом, при $n \geqslant 10$ и $S = \lfloor 2^n/10n \rfloor$ всякую схему размера S можно описать строкой из не более $4\frac{2^n}{10n}(n - \log_2(10n)) \leqslant 2 \cdot 2^n/5$ битов. Поэтому количество схем размера S не больше, чем количество таких строк, то есть не больше $2^{2 \cdot 2^n/5}$. Видно, что это меньше 2^{2^n}, а значит не всякую функцию можно вычислить схемой размера S (или меньшего, как мы заметили с самого начала). □

Мы доказали верхнюю и нижнюю оценку сложности функций n переменных, причём наши оценки достаточно близки: и та, и та оценка экспоненциальная. Если интересоваться более точным значением максимальной сложности функции, то

оказывается, что наша нижняя оценка точнее, то есть можно доказать, что всякую функцию от n переменных можно вычислить схемой размера не больше $O(2^n/n)$. Однако доказать этот факт не так просто, и мы не будем приводить здесь его доказательство.

Заметим, что наше доказательство существования трудной функции неконструктивное, «явной» функции мы не предъявляем. И неудивительно: предъявить достаточно простую функцию хотя бы со сверхлинейной схемной сложностью — это трудная нерешённая проблема теории вычислительной сложности.

Покажем, как доказывать хотя бы линейные нижние оценки сложности конкретных булевых функций.

Для этого рассмотрим функцию XOR_n от n переменных. Эта функция задается формулой

$$XOR_n(x_1, \ldots, x_n) = x_1 \oplus x_2 \oplus \ldots \oplus x_n.$$

Ясно, что её можно вычислить схемой линейного размера. Действительно, можно сначала вычислить $x_1 \oplus x_2$, пользуясь описанной выше схемой (13.1), затем вычислить $(x_1 \oplus x_2) \oplus x_3$, пользуясь той же схемой, и так далее. На добавление каждой переменной требуется 5 элементов, так что общий размер схемы получается $5(n-1) = 5n - 5$.

Теперь мы докажем нижнюю линейную оценку.

Теорема 13.5. *Схемная сложность функции XOR_n не меньше $2n - 1$.*

Доказательство. Будет удобнее доказать более общий факт. Давайте рассматривать схемы, в которых используются не только операции конъюнкции, дизъюнкции и отрицания, но и произвольные операции вида $(x^a \wedge x^b)^c$. Иначе говоря, мы можем использовать конъюнкции и дизъюнкции и брать при этом отрицания бесплатно, то есть не учитывать их при подсчёте размера схемы. Ясно, что теперь мы рассматриваем более общие схемы, а значит доказывать нижние оценки будет только труднее.

Мы будем вести доказательство индукцией по n. Причём удобно будет усилить доказываемое утверждение: мы будем доказывать нижнюю оценку $2n - 1$ одновременно для двух функций XOR_n и $\neg XOR_n$.

Итак, для $n = 1$ нам нужно доказать, что размер схемы не меньше 1, что очевидно. Пусть мы доказали утверждение для n, докажем его для $n+1$. Индуктивный переход доказывается от противного.

Пусть есть схема размера $S < 2(n+1) - 1 = 2n + 1$ для функции XOR_{n+1} или для функции $\neg XOR_{n+1}$. Не ограничивая общности, рассматриваем минимальную схему, то есть предполагаем, что схем размера меньше S для этих функций нет.

Рассмотрим первый элемент g_1 этой схемы. Ясно, что ему на вход подаются две переменные. Для определённости, пусть это переменные x_1 и x_2. Заметим, что можно так зафиксировать переменную x_1, что элемент g_1 равен константе при любых значениях остальных переменных. Действительно, пусть $g_1 = (x_1^a \wedge x_2^b)^c$. Тогда выберем такое значение x_1, что $x_1^a = 0$ (то есть $x_1 = 1$, если $a = 0$, и $x_1 = 0$, если $a = 1$). Далее заметим, что g_1 подаётся на вход какому-то ещё элементу (иначе g_1 можно

было бы просто удалить из схемы, что противоречит минимальности S). Пусть это элемент g_k и второй его вход — элемент g_j. Заметим, что после того, как g_1 обратился в константу, элемент g_k равен либо g_j, либо его отрицанию.

Зафиксируем переменную x_1 так, чтобы g_1 стал константой. Удалим из схемы элементы g_1, g_k. Все элементы, в которые подставлялась переменная x_1 или элемент g_1, по-прежнему можно вычислить — они вычисляют либо какие-то предыдущие элементы, либо их отрицания. Если в какой-то элемент подставлялся g_k, то вместо него можно подставить g_j, либо его отрицание. Так что после удаления этих двух элементов мы снова получили некоторую булеву схему. Эта схема вычисляет функцию XOR_n или функцию $\neg XOR_n$ от оставшихся n переменных. При этом размер схемы равен $S - 2 < 2n - 1$. Мы получили противоречие с предположением индукции. $\square$

До сих пор мы обсуждали размер схем. Вторым важным параметром схемы является её глубина. Рассмотрим схему как ориентированный граф. Глубиной вершины в схеме называется длина наибольшего пути из какой-нибудь переменной в эту вершину. Глубиной схемы называется максимальная глубина вершины в этой схеме. Например, глубина схемы для функции $x_1 \oplus x_2$ на рисунке 13.1 равна 3. Глубина схемы для MAJ_3 на рисунке 13.2 также равна 3.

Неформально, глубина схемы характеризует время, необходимое для вычисления значения схемы на данном входе, если вычисление элементов схемы можно производить параллельно: на первом шаге можно параллельно вычислить все элементы глубины 1, на втором — все элементы глубины 2, и так далее, на некотором k-м шаге можно параллельно вычислить все элементы глубины k. Это можно сделать, так как на вход элементам глубины k подаются только элементы меньшей глубины, а их значения уже вычислены.

Пример 13.8. Рассмотрим функцию $x_1 \wedge x_2 \wedge \ldots \wedge x_n$. Её можно вычислить схемой размера $n - 1$, просто вычислив последовательно $x_1 \wedge x_2$, $x_1 \wedge x_2 \wedge x_3$ и так далее. Однако глубина такой схемы также равна $n - 1$, поскольку каждый следующий элемент получает на вход предыдущий. Ту же функцию можно вычислить схемой глубины $\lceil \log_2 n \rceil$, если организовать схему как двоичное дерево. Разобьём переменные на пары x_1 и x_2, x_3 и x_4 и так далее. Вычислим конъюнкцию каждой пары. Все эти элементы имеют глубину 1. Повторим рассуждение: разобьём полученные элементы на пары и вычислим конъюнкцию парных элементов. Если количество переменных n есть степень двойки, то мы получим полное двоичное дерево глубины $\log_2 n$. Если количество элементов не равно степени двойки, то на некоторых шагах у некоторых элементов не будет парных и мы будем брать их конъюнкцию с самими собой (по существу, мы будем переносить их на следующий шаг). В результате получится поддерево полного двоичного дерева глубины $\lceil \log_2 n \rceil$.

То же самое рассуждение применимо к функции $x_1 \vee x_2 \vee \ldots \vee x_n$ и к функции XOR_n. Их обе можно вычислить схемой глубины $O(\log n)$.

Из приведённого выше примера и конструкции схемы для произвольной функции из леммы 13.2 видно, что всякую функцию можно вычислить схемой глубины $O(n)$. Действительно, в конструкции с дизъюнктивной нормальной формой возникают

конъюнкции n элементов и дизъюнкции 2^n элементов. Если вычислять их так, как описано выше, то глубина схемы будет $O(n)$.

Обсудим подробнее глубину схем для функции «связность графа».

Лемма 13.6. *Проверку связности графа на n вершинах можно выполнить схемой глубины не больше $O(\log^2 n)$ и полиномиального размера.*

Доказательство. Идея состоит в том, чтобы модифицировать построенную в примере 13.6 схему полиномиального размера для задачи проверки связности в духе примера 13.8.

Мы будем вычислять степени матрицы смежности A' с единицами на диагонали. Во-первых, оценим за какую глубину можно по данной матрице $B \in \{0,1\}^{n\times n}$ вычислить матрицу B^2, где умножение матриц — булево. Ячейка (i,j) матрицы B^2 вычисляется по формуле

$$\bigvee_{k=1}^{n} (b_{ik} \wedge b_{kj}).$$

На вычисление всех конъюнкций нам потребуется глубина 1, а на вычисление дизъюнкции, как мы обсудили выше, достаточно глубины порядка $\log_2 n$. Поскольку все ячейки матрицы B^2 можно вычислять параллельно, мы получаем, что для возведения матрицы в квадрат достаточно глубины $O(\log n)$, если быть точными, то $\lceil \log_2 n \rceil + 1$.

Далее, как мы уже обсуждали, чтобы проверить, есть ли между двумя данными вершинами путь в графе на n вершинах, достаточно проверить, есть ли путь длины не больше n. Вместо этого нам будет удобнее проверить, есть ли в графе пути длины не больше 2^k, где $k = \lceil \log_2 n \rceil$. Ясно, что $2^k \geqslant n$, так что этого для нас будет достаточно. С другой стороны, для такой проверки нам достаточно вычислить матрицу $(A')^{2^k}$. Для этого мы можем просто последовательно возвести матрицу A' в квадрат k раз. Матрица A' нам по существу подаётся на вход, на каждое возведение в квадрат мы тратим глубину $O(\log n)$, так что, поскольку мы возводим в квадрат k раз, суммарная глубина есть $kO(\log n) = O(\log^2 n)$.

После этого нужно лишь взять конъюнкцию всех ячеек матрицы $(A')^{2^k}$, что можно сделать за глубину $O(\log n^2) = O(\log n)$. Так что общая глубина всей схемы есть $O(\log^2 n) + O(\log n) = O(\log^2 n)$. □

Задача 13.9. В доказательстве, чтобы вписаться в глубину $O(\log^2 n)$, мы вычислили матрицу $(A')^{2^k}$ вместо матрицы $(A')^n$. Но на самом деле, можно было обойтись и без этого трюка. Докажите, что матрицу $(A')^n$ также можно вычислить схемой глубины $O(\log^2 n)$.

Задача 13.10. В предыдущей лемме мы рассматривали задачу проверки на связность неориентированного графа. Можно рассмотреть аналогичную задачу для ориентированных графов (проверка сильной связности). Докажите, что и для этой задачи есть схема глубины $O(\log^2 n)$.

Отметим, что вопрос о том, можно ли проверить граф (ориентированный или неориентированный) на связность схемой глубины $O(\log n)$, является важным открытым вопросом теории вычислительной сложности. Можно, однако, сказать, что ориентированный случай не проще неориентированного.

Задача 13.11. Докажите, что если для задачи проверки ориентированного графа на связность есть схема глубины $O(\log n)$, то такая схема есть и для проверки неориентированного графа на связность.

В оставшейся части лекции мы сосредоточимся на схемах с одним выходом. Такие схемы от n переменных вычисляют функции вида $f\colon \{0,1\}^n \to \{0,1\}$.

13.2 Формулы

Во введении к этой главе уже упоминалось, что формулы можно рассматривать как частный случай схем. В результате получаются булевы схемы специального вида, которые также называются *формулами*.

Формула задаёт способ вычисления некоторой функции, как и схема. Входящие в формулы операции нужно применять в определённом порядке, который указывается либо скобками, либо правилами старшинства операций. Например, посмотрим на последние формулы в схемах (13.1) и (13.2) и пронумеруем операции в них в порядке выполнения:

$$(x_1 \overset{2}{\wedge} \overset{1}{\neg} x_2) \overset{5}{\vee} (\overset{3}{\neg} x_1 \overset{4}{\wedge} x_2), \tag{13.4}$$

$$\Big((x_1 \overset{1}{\vee} x_2) \overset{2}{\wedge} x_3\Big) \overset{4}{\vee} (x_1 \overset{3}{\wedge} x_2) \tag{13.5}$$

Записав полученные последовательности присваиваний, мы получим булевы схемы для XOR_2 и MAJ_3 соответственно.

Контрольный вопрос 13.12. Совпадает ли схема, построенная по разметке формулы (13.4), со схемой (13.1)?

Теперь опишем переход от формул к схемам в общем случае. Рассмотрим формулу булевой логики ϕ, то есть формулу от переменных $x_1, \ldots, x_n$ со связками конъюнкция, дизъюнкция и отрицание. Построим по этой формуле схему, каждый элемент которой отвечает нетривиальной подформуле формулы ϕ (то есть подформуле, содержащей хотя бы одну связку).

Найдём связку, которая применяется в формуле первой, и добавим соответствующий элемент в схему. Найдём связку, применяющуюся следующей и добавим в схему соответствующий элемент и т.д. Каждый раз связка в схеме применяется к тем элементам схемы, которые соответствуют подформулам, к которым соответствующая связка применяется в формуле (если подформула — переменная, то применяем связку к той же переменной).

В итоге получаем схему, размер которой равен количеству связок в исходной формуле.

Нетрудно заметить, чем полученные схемы отличаются от общих схем. В них каждый элемент входит в правую часть присваиваний не более одного раза, так как он однозначно задаётся одной из связок формулы. Переменные могут входить много раз, так как одна и та же переменная может входить в формулу неоднократно.

Определение 13.7. Схема, в которой каждый элемент встречается в правых частях присваиваний не более одного раза, называется *формулой*.

Рассмотрим теперь обратный переход: от схем-формул к обычным формулам. Для этого последовательно выписываем обычные формулы для каждого элемента схемы, как мы уже делали в примере 13.1, соединяя предыдущие формулы (или переменные) связкой, соответствующей данному элементу схемы-формулы.

Нетрудно видеть, что эти два сведения обратны друг к другу, то есть, построив по формуле булевой логики схему, а затем по ней обратно формулу булевой логики, мы придём к той же формуле, с которой начали.

Если представлять схемы ориентированными графами, то схемы-формулы выделяются таким условием: из каждой вершины за исключением переменной, выходит ровно одно ребро. (Напомним, что выходы схемы мы отмечаем выходящим ребром со свободным концом.)

Для единообразия удобнее считать, что в графе допускается несколько вершин, помеченных одной и той же переменной, и из каждой такой вершины также выходит ровно одно ребро. На рисунке 13.4 приведён пример графа для формулы (13.4), выражающей функцию $x_1 \oplus x_2$.

Задача 13.13. Пусть в ациклическом графе из каждой вершины выходит не более одного ребра, а неориентированный граф, полученный заменой ориентированных рёбер на неориентированные, связен. Тогда этот граф — дерево.

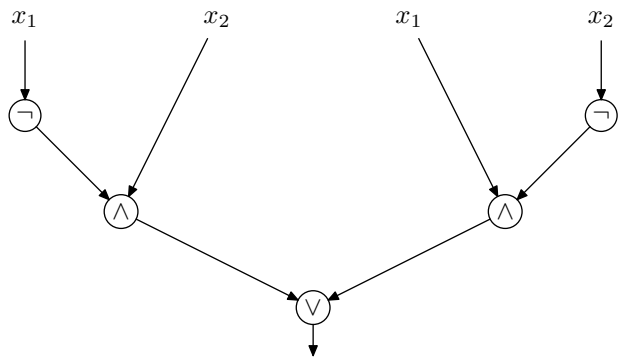


Рис. 13.4: Формула для функции $x_1 \oplus x_2$

Из задачи 13.13 следует, что формулы — это такие схемы, графы которых являются деревьями (после размножения вершин, отвечающих переменным).

Формулы — это частный случай схем. При этом всякую схему можно переделать в формулу. Глубина при этом не увеличится, однако размер может вырасти экспоненциально.

Лемма 13.8. *По всякой схеме C от переменных $x_1, \ldots, x_n$ глубины d и размера s можно построить формулу, вычисляющую ту же функцию, глубины не больше d и размера не больше $2^s - 1$.*

Доказательство. Доказательство можно вести индукцией по s и d. По существу мы доказываем утверждения для глубины и размера отдельно, но рассуждение одинаково, так что мы сделаем это параллельно. В качестве базы рассмотрим формулу, вычисляющую одну из переменных. Её размер n и глубина равна нулю. Собственно, эта схема уже является формулой и утверждение леммы очевидно ($2^n - 1 \geqslant n$ для всякого n).

Предположим, что для схем размера меньше s (глубины меньше d) утверждение уже доказано. Рассмотрим схему размера s (глубины d). Посмотрим на её последний элемент g_s. Он получается применением какой-то логической связки к предыдущим элементам g_i и g_j. Рассмотрим подсхемы нашей схемы, вычисляющие g_i и g_j. Их размер меньше s (глубина меньше d), так что по предположению индукции их можно вычислить формулами размера не больше $2^{s-1} - 1$ (глубины меньше d).

Добавим к этой паре формул новую вершину, соответствующую элементу g_s, и проведём в неё ребра из элементов, вычисляющих g_i и g_j. Полученная схема имеет размер не больше $2 \cdot (2^{s-1} - 1) + 1 = 2^s - 1$ (глубину не больше d).

Этот же процесс можно описать и напрямую. Для этого отсортируем вершины по слоям по их глубине, то есть рассмотрим отдельно вершины глубины 1, вершины глубины 2, и так далее. Будем для всех вершин последовательно добиваться того, чтобы из них выходило не более одного ребра, при этом будем рассматривать вершины по убыванию их глубины. Из вершин глубины d рёбер не выходит (или выходит одно ребро, если учитывать те стрелки, которые мы рисуем для выделения выходов схемы), так что для них условие выполняется изначально. Пусть мы уже добились выполнения условия для вершин глубины больше k. Рассмотрим вершины глубины k. Если из какой-то из них выходит больше одного ребра, то сделаем несколько копий этой вершины, по числу выходящих рёбер, и из каждой копии вершины выпустим одно ребро.

Зачем мы делаем этот процесс по убыванию глубины: «размножение» вершин глубины больше k может увеличить выходную степень вершин глубины k, так что мы сначала размножаем все вершины глубины больше k, и только потом смотрим на вершины глубины k, их выходная степень уже не вырастет. □

Рассмотрим теперь вопрос о соотношении глубины и размера схем. Между этими величинами несложно доказать неравенство в одну сторону.

Лемма 13.9. *Всякая схема глубины d эквивалентна (то есть, вычисляет ту же функцию) некоторой схеме, и даже формуле, глубины d и размера не больше 2^{d+1}.*

Доказательство. По существу, строить эквивалентную схему не нужно. Достаточно просто удалить из текущей схемы все неиспользующиеся элементы. Более точно, если в схеме есть вершина, не являющаяся выходом, из которой не выходит рёбер, то такую вершину можно безболезненно удалить из схемы. Будем удалять подобные вершины до тех пор, пока их не останется.

После этого можно утверждать, что размер схемы не больше 2^{d+1}. Действительно, можно переделать эту схему в формулу по предыдущей лемме. При этом процессе размер схемы не уменьшается. Но получившаяся в конце формула является поддеревом двоичного дерева глубины не больше d, так что в ней не больше 2^{d+1} вершин. □

Таким образом, мы доказали, что размер схемы не больше экспоненты от глубины схемы. Неизвестно, выполняется ли обратное соотношение: это трудная открытая проблема.

Задача 13.14. Докажите, что если для всякой схемы от n переменных размера s существует эквивалентная схема глубины не больше $O(\log s)$, то достижимость в графе можно проверить схемой глубины $O(\log n)$.

Однако оказывается, что для частного случая формул обратное утверждение также верно.

Теорема 13.10. *Для всякой формулы размера s есть эквивалентная формула глубины не больше $1 + 4 \log s$.*

Вспомнив, что формула — это дерево, можно заметить, что результат этой теоремы означает, что для всякой формулы есть эквивалентная формула, которая не сильно больше изначальной, и в которой ветви имеют примерно одинаковую длину.

В доказательстве теоремы 13.10 нам уже придётся всерьёз перестраивать формулу. Для этого потребуется вспомогательное утверждение: нужно выделить «среднюю» вершину в дереве.

Утверждение 13.11. *Пусть формула ϕ имеет размер s. Тогда в ней есть такая подформула ψ размера не меньше $s/2$, что все подформулы формулы ψ имеют размер меньше $s/2$.*

Доказательство. Начнём из выходного элемента спускаться по рёбрам формулы в подформулы так, чтобы каждый раз оставаться в подформуле размера не меньше $s/2$. В выходном элементе размер подформулы равен s, а в переменных размер подформулы равен 1, так что в какой-то момент мы не сможем спуститься из очередной вершины в подформулу. Это как раз и будет означать, что у текущей подформулы размер не меньше $s/2$, а во всех её подформулах уже меньше. □

Доказательство теоремы 13.10. Мы применяем индукцию по s. База индукции — формулы размера 1 — очевидна, так как глубина таких формул уже равна 1.

Теперь предполагаем, что теорема верна для формул размера меньше s. Докажем, что тогда она верна и для формул размера s.

В формуле размера s найдём «среднюю» подформулу ψ, удовлетворяющую свойствам из утверждения 13.11. Удалим из дерева формулы поддерево, которое растёт из вершины, соответствующей подформуле ψ. (Более точно, те вершины дерева, путь из которых в корень проходит через ψ.)

Теперь у дерева образовался новый лист: та вершина, где раньше была подформула ψ. Пометим этот лист константой 0. Мы получили новую формулу ϕ_0. Аналогично сделаем для константы 1: пометим новый лист единицей и обозначим полученную подформулу через ϕ_1.

Описанная конструкция не соответствует нашим определениям: мы не разрешали использовать константы вместо переменных. Однако нетрудно видеть, что от констант можно избавиться, не увеличивая ни размера, ни глубины формулы. Действительно, отрицание константы — это опять константа. Конъюнкция с 0 — это 0, с 1 — это второй аргумент конъюнкции. Аналогично, дизъюнкция с 1 — это 1, с 0 — это второй аргумент дизъюнкции.

Пользуясь этими соотношениями, в схеме с константами можно удалять лишние элементы. Ни размер схемы, ни глубина при таком удалении не увеличиваются, а если исходная схема была формулой, то и преобразованная схема останется формулой. Завершается такое удаление эквивалентной схемой без констант (формулой, если исходная схема была формулой).

Далее под ϕ_0 и ϕ_1 понимаем формулы, полученные избавлением от констант.

Теперь рассмотрим формулу $\alpha = (\phi_1 \wedge \psi) \vee (\phi_0 \wedge \neg\psi)$. Нетрудно видеть, что она вычисляет ту же функцию, что и изначальная формула.

Поскольку размер формулы ψ не меньше $s/2$, то размеры формул ϕ_0 и ϕ_1 не больше $s/2$ и к ним применимо по предположение индукции. Для этих формул есть эквивалентные формулы глубины $\leqslant 1 + 4\log_2 s - 4 = 4\log_2 s - 3$.

Для формулы ψ есть эквивалентная формула глубины $4\log_2 s - 2$. Действительно, по предположению индукции для самых больших подформул ψ есть эквивалентные формулы глубины $\leqslant 4\log_2 s - 3$. Ещё единица глубины добавляется из-за последней связки в формуле ψ.

Окончательно получаем, что для формулы α, которая эквивалентна исходной, есть эквивалентная формула глубины $\leqslant 4\log_2 s - 2 + 3 = 1 + 4\log_2 s$ (последнее слагаемое в оценке глубины появляется из-за связок в формуле α, которая имеет глубину 3). Это доказывает шаг индукции. □

Таким образом, мы получаем, что для всякой функции $f\colon \{0,1\}^n \to \{0,1\}$ минимальная глубина схемы и логарифм минимального размера формулы линейно связаны. То есть, изучение размера формул — это по существу то же самое, что изучение глубины схем.

Лекция 14

Алгоритмическая неразрешимость

Последним обширным сюжетом нашего курса будет введение в теорию алгоритмов. Это самая сложная часть курса. Хотя какое-то представление об алгоритмах есть сейчас почти у всех, дать аккуратное математическое определение алгоритма не так просто.

Поэтому мы начнём рассказ о теории алгоритмов с не очень формального рассказа об основных её идеях. Дальше мы повторим примерно то же, с некоторыми уточнениями и расширениями, ещё дважды, каждый раз делая изложение более строгим.

Если какие-то конкретные утверждения в этой главе окажутся непонятными, то можно попробовать разобрать аналогичные утверждения в последующих главах. И наоборот, если в двух последующих главах вы потеряете общую нить рассказа, полезно вернуться к этой главе и разобраться вначале с затруднением на менее формальном уровне.

14.1 Игра FRACTRAN

Наверно, вы слышали про игру «Жизнь», которую придумал Джон Конвей (эволюция на клетчатой бумаге по определённым правилам). Тот же самый Конвей придумал и другую игру, которую он назвал FRACTRAN[1]. Она происходит так: есть список дробей (числитель и знаменатель — целые положительные числа)

$$\frac{p_1}{q_1}, \ldots, \frac{p_n}{q_n}$$

и целое положительное число a. Игра[2] состоит в том, что это целое число надо умножить на какую-то дробь из списка, причём оно должно остаться целым после умножения. Если такой дроби нет, игра заканчивается. Если подходящих дробей несколько, выбирается первая в списке. После умножения получается новое целое

[1] От слов “fraction” (дробь) и “FORTRAN” (старинный язык программирования).

[2] Точнее её можно было бы назвать пасьянсом, так как игрок только один и он следует заранее известным правилам, никакого выбора у него нет.

положительное число, и всё повторяется вновь (выбирается первая дробь, дающая целое произведение, и так далее).

Для любой начальной позиции есть две возможности: либо (1) этот процесс будет продолжаться бесконечно, либо (2) в какой-то момент он остановится, потому что ход по правилам будет сделать нельзя.

Теорема Конвея: *нельзя написать программу, которая, получив на вход начальную позицию* (список и число), *говорила бы, какая из этих двух возможностей реализуется для этой начальной позиции.*

Можно сказать, что «проблема остановки для игры Конвея» алгоритмически неразрешима.

Одна из наших целей — доказать этот факт. Но сначала надо обсудить, что конкретно тут утверждается.

14.2 Что утверждается?

Мы говорим «нельзя написать программу» — но что это означает? Программа на каком языке программирования имеется в виду? Может быть, на каком-то одном языке написать нельзя, а на другом — можно? Или на существующих сейчас языках нельзя, а завтра какой-то новый Кнут придумает совершенно новый язык программирования, и на нём будет можно?

Удивительным образом оказывается, что выбор языка не имеет значения: все известные языки эквивалентны в том смысле, что если решение какой-то задачи можно запрограммировать на одном, то можно и на другом. Тут, конечно, нужны некоторые оговорки:

- Бывают языки, которые с самого начала предназначены для какого-то узкого класса задач (скажем, регулярные выражения для лексического разбора). К ним сказанное не относится.

- Если говорить о реальных языках программирования, в них есть разные количественные ограничения: скажем, часто индекс в массиве может быть лишь целым числом ограниченного размера, и потому используемая память ограничена, и так далее. Чтобы утверждение было верным, нужно рассматривать идеализированные варианты реальных языков, в которых такие количественные ограничения сняты.

- Мы ничего не говорим ни о длине программы, ни тем более об удобстве её написания, ни о скорости её работы. С практической точки зрения это всё очень важно, соответственно выбор подходящего языка программирования — важная часть профессиональных умений программиста.

Конечно, тут можно спросить — что значит «оказывается»? Это что, можно как-то доказать как математическую теорему? Нет, конечно, поскольку слова «язык

программирования» не имеют математического определения (по крайней мере, общепринятого). Теоретически можно себе представить такое доказательство для двух конкретных языков программирования (в которых как-то сняты количественные ограничения, как мы обсуждали). Но тогда для начала нужно строго определить эти языки программирования, что вряд ли реально.[3]

Почему же мы так уверены, что все языки программирования позволяют запрограммировать одно и то же? Можно пытаться объяснить это так. Программы на любом языке программирования в конечном счёте должны исполняться процессором (с помощью интерпретатора, компилятора или чего-то промежуточного) — иначе это не язык программирования, а не пойми что. Значит, любой алгоритм, который можно запрограммировать, в конечном счёте выполняется процессором. А эмулятор процессора можно написать на совсем простом языке программирования. Поэтому даже очень продвинутый язык программирования не позволяет запрограммировать чего-то большего по сравнению с простыми. (Тут опять надо повторить сделанные выше практические оговорки, конечно.)

14.3 Отступление о процессорах

Мы говорили о языках программирования, но то же самое можно сказать и о языках низкого уровня — системах команд процессора. Честертон когда-то писал, что полезно посмотреть на привычные вещи со стороны и удивиться чудесам, не замечаемым за их повседневностью. Он говорил про христианство, но это и вообще верно. Ведь удивительно же, если подумать, что одни и те же процессоры используются, условно говоря, в стиральных машинах и в шахматных автоматах[4]. Какой-нибудь писатель-фантаст в старину легко мог представить себе, что в будущем будут и шахматные автоматы, и машины, которые сами стирают бельё по команде человека. (Мало ли какие сложные машины может изобрести человек!) Но вряд ли бы ему могло прийти в голову, что в шахматном автомате и стиральной машине может использоваться одно и то же устройство (процессор) — как так, сказал бы он, это явно какая-то ерунда. А собственно, именно из-за универсальности процессоров и возможности определять их работу хранимой в памяти программой всё это так дёшево.

Так или иначе, любой не совсем узкоспециализированный процессор может худобедно (с замедлением, и если отвлечься от количественных ограничений памяти) эмулировать работу любого другого. Поэтому класс разрешимых программистских задач (в теории, со всеми сделанными нами оговорками) не зависит от выбора процессора.

[3] Описание языка программирования часто разделяют на *синтаксис* (какие программы считаются допустимыми) и *семантику* (как выполняется синтаксически допустимая программа). Если для описания синтаксиса есть разные средства, то семантику обычно описывают весьма неформально. Памятник попыткам «строгого» описания семантики — знаменитая когда-то книжка Revised Report on the Algorithmic Language ALGOL 68 (её даже перевели на русский язык).

[4] Современный шахматный автомат — это робот, состоящий из шахматного компьютера и управляемой им руки-манипулятора, которая выполняет ходы на шахматной доске.

14.4 Кодирование

Ещё одна важная деталь: формулируя теорему Конвея, мы не говорили, в каком виде алгоритм (программа) получает на вход набор дробей и число. Можно считать, например, что это последовательность байтов, в которой числители и знаменатели записаны в десятичной системе, разделены знаками деления и запятыми. А можно — если в языке есть соответствующие конструкции, как в питоне — считать, что рациональное число это пара целых чисел неограниченной длины, и из таких пар сформирован список. Но ясно, что от этого ничего не зависит. В самом деле, мы можем предварительно перекодировать данные из одного формата в другой (в нашем примере это совсем просто). По аналогичным причинам и представление ответа не играет роли: если наша программа даёт ответ «остановится» и «не остановится», а нам хочется получать ответ в виде значения типа Boolean, легко переделать её и под этот формат, и т. п.

Поэтому, когда мы говорим о программах, работающих с конечными объектами (будь то графы, рациональные числа или многочлены с целыми коэффициентами), мы обычно не уточняем, как именно они представлены — считая, что если разные разумные люди имеют в виду разные разумные представления, то это ни на что не повлияет и всегда можно алгоритмически перекодировать данные из одного формата в другой. Поэтому, развивая теорию вычислимости, можно считать, что алгоритмы работают только со словами (строками, последовательностями битов или символов какого-то алфавита), а все остальные конечные объекты как-то кодируются строками. Или можно считать, что алгоритмы работают с натуральными числами — это по существу то же самое, поскольку есть простое взаимно однозначное соответствие между числами и битовыми строками. Например, можно отождествить число n с двоичной записью $n + 1$ без первого бита. (Понятно, кстати, почему надо отбрасывать первый бит и почему мы берём двоичную запись $n + 1$, а не n?)

Проблемы начинаются, если мы хотим иметь дело, скажем, с действительными числами (или другими бесконечными объектами). Если мы говорим, скажем, что написали алгоритм вычисления синуса действительного числа, что мы имеем в виду? Можно действовать цинично и считать, что действительное число — это значение типа float (или real в других языках), то есть по существу элемент некоторого конечного множества значений, выбранных для приближённого представления действительных чисел в этом языке. Тогда наш алгоритм должен выдавать другое значение типа float, желательно, близкое к синусу своего аргумента. Но возможны и более изощрённые понимания: можно считать, что аргумент синуса задан в виде доступа к внешней функции, выдающей приближения любой запрошенной точности, и наша программа аналогичным образом выдаёт приближения к результату. Мы в это дело вдаваться не будем, и будем рассматривать только алгоритмы, работающие с конечными объектами, когда проблем такого рода не возникает.

14.5 Класс вычислимых функций

Теперь теорему Конвея можно сформулировать как *утверждение о невычислимости* некоторой функции. Как это понимать? Будем рассматривать функции, аргументами и значениями которых являются двоичные слова (=битовые строки). Пусть f — такая функция; область определения f — это некоторое множество слов (мы разрешаем ей не быть всюду определённой, на некоторых двоичных словах она может быть не определения), значения f тоже двоичные слова.

Определение. Функция f называется *вычислимой*, если существует программа p, её вычисляющая. Это значит, что:

- если функция f определена на слове x, то программа p на входе x останавливается и даёт ответ $f(x)$;
- если функция f не определена на слове x, то программа p на входе x не останавливается или останавливается, не дав никакого ответа.

Замечания

- Мы говорим о «программе», не уточняя языка программирования (мы обсуждали, почему это не играет роли). Иногда, чтобы подчеркнуть это, говорят про «алгоритм, вычисляющий функцию f».
- Мы предусматриваем возможность «безрезультатной остановки» (которая в разных языках может быть разной — дойти до конца процедуры без return, или какая-то ошибка при выполнении, или ещё что-то). Можно было бы это запретить, это не изменило бы понятия вычислимости: программа, вместо того чтобы останавливаться без ответа, могла бы войти в искусственный бесконечный цикл.
- В этом определении говорится только о вычислении значения функции по аргументу, поданному на вход программы. Интерактивные программ, которые ведут диалог с клиентом, или рисуют что-то на экране по запросу пользователя, в эту схему не укладываются: результат работы такой программы зависит от предыстории, а не только от текущего входа.

Как теперь сформулировать теорему Конвея? Рассмотрим функцию Terminates(x), аргументом которой является позиция в игре Конвея, закодированная с помощью двоичного слова, а значение равно 0, если игра продолжается бесконечно, и 1, если игра заканчивается. Так вот, теорема Конвея утверждает, что *функция* Terminates *невычислима*.

Педантичный читатель спросит — а как определена функция Terminates на двоичных словах, которые не являются кодами позиций? Можно отвечать по-разному. Например, можно искусственно доопределить кодирование, сочтя все такие слова кодами какой-то фиксированной позиции в игре. Или договориться, что на всех них значение функции Terminates равно нулю. Или как-то ещё: это не влияет на вычислимость, потому что при разумном кодировании легко проверить алгоритмически, будет ли данное слово кодом позиции.

14.6 Определение вычислимости?

Ну хорошо, скажет дотошный читатель, допустим, мы поверили в то, что все языки программирования позволяют запрограммировать одно и то же, и достаточно доказать, что задачу Конвея нельзя решить на каком-то одном языке программирования. Но как вы собираетесь доказывать, если не определили *ни одного* языка программирования?

Это хороший вопрос (как отвечают докладчики, когда по существу ничего сказать не могут). Действительно, честный способ действий состоял бы в следующем. Надо определить конкретный язык программирования (будь то язык высокого уровня или систему машинных команд), в котором не было бы количественных ограничений. При этом желательно выбрать его попроще, чтобы определение не было очень длинным. Затем надо для этого языка строго доказать, что никакая программа не вычисляет (в смысле этого языка) функцию Конвея.

И действительно, так люди и делали. Первое определение вычислимости было предложено для функций на натуральных числах, был определён класс функций, названных «рекурсивными» (recursive functions). Если совсем грубо, то в соответствующем языке программирования были базовые функции прибавления единицы и константа нуль, а также конструкции подстановки одной функции в другую, рекурсивные определения и перебор (он назывался тогда μ-оператором). Это было сделано в начале 1930-х годов, окончательное определение сформулировал Клини. Были предложены и другие определения — Чёрч использовал так называемое «λ-исчисление» (которое лежит в основе функциональных языков программирования), Тьюринг придумал знаменитые «машины Тьюринга» — устройства с лентой, по которой движется головка, производя вычисления. Примерно тот же тип машин предложил Пост. Оказалось, что все эти варианты «языков программирования» (как мы сказали бы теперь — тогда до первых реально действующих машин оставалось ещё полтора десятилетия) эквивалентны — задают один и тот же класс вычислимых функций. В учебниках по теории вычислимости выбиралось какое-то из этих определений и более или менее аккуратно доказывались его свойства.[5]

Но дело это, увы, хлопотное, скучное, и не очень понятно, что оно даёт измученному читателю, так что в современных книжках так делают редко.

14.7 Компромисс

Мы тоже не будем давать строгих определений и точных доказательств. Вместо этого мы предложим вам представить себе какой-нибудь знакомый язык программирования (вряд ли кто-то, читающий это, не пробовал программировать на каком-то языке, правда?). Мы не будем допытываться, что это за язык, или навязывать вам свой вариант. Вместо этого мы будем убедительно просить согласиться, что этот не

[5] Можно отметить тут достижение Андрея Андреевича Маркова-младшего (сына того, в честь которого названы марковские цепи) — он впервые дал действительно точное доказательство корректности интерпретатора для предложенного им класса алгоритмов. Это было в 1950-е годы.

уточняемый язык имеет те или иные свойства и говорить «при необходимости вы легко докажете это, ведь правда?», а вы будете вежливо соглашаться. При этом и мы, и вы, по правде говоря, понимаем, что это лицемерие — реально это доказать не так просто, да и определение языка дать не просто, и что начав это делать, вы запутаетесь в деталях, ровно так же, как запутались бы мы.

Кто-то из великих математиков (Гильберт?) сказал, что аксиоматический метод имеет перед честным построением требуемых объектов те же преимущества, что и воровство перед честным трудом. Конечно, это лишь шутка, и да и слова «аксиоматический метод» можно понимать по-разному. Но продолжая эту аналогию, можно сказать, что мы будем заниматься скорее грабежом, чем воровством — в том смысле, что будем не скрывать, а подчёркивать те места в рассуждениях, где по-честному надо было бы сделать большую работу и доказать то или иное утверждение.

Что имеется в виду? Ну вот, например:

Теорема. Функция проверки простоты IsPrime(n), которая получает на вход натуральное число n и возвращает 1, если n простое, и 0, если n составное, вычислима.

«**Доказательство**». Достаточно проверить по очереди все числа от 2 до $n-1$, если среди них нет делителей числа n, то n простое, а если есть, то n составное. Ну вы же можете, разумеется, запрограммировать этот простой алгоритм на вашем любимом языке программирования? Можете, да? Ну вот и славно, как говорил профессор Стравинский (не композитор) у Булгакова.

Замечания.

- Иногда говорят ещё, что множество простых чисел *разрешимо*, это как раз и значит, что функция проверки принадлежности ему вычислима, то есть что существует алгоритм (программа на вашем любимом языке), которая даёт ответ 1 и 0, когда его вход принадлежит (соответственно не принадлежит) этому множеству.

- С практической точки зрения описанный алгоритм очень неэффективен — есть гораздо более быстрые (но и более сложные) алгоритмы. Некоторые практические алгоритмы вероятностные — они используют датчик случайных битов и с некоторой небольшой вероятностью могут ошибаться, но даже и детерминированные алгоритмы могут работать за полиномиальное от числа битов в записи числа время. (Долгое время это было открытой проблемой, но в 2002 году такой алгоритм придумали Агравал, Каял и Саксена, и теперь он известен под названием AKS primality testing. Но на практике всё равно используют вероятностные алгоритмы.)

14.8 Композиция вычислимых функций

Вот другой пример подобных «доказательств».

Теорема. Композиция двух вычислимых функций вычислима.

Напомним, что такое композиция. Пусть есть две функции f и g, аргументами и значениями которых являются двоичные слова. Тогда их композицией называ-

ется функция h, которая на аргументе x равна $g(f(x))$, если $f(x)$ определено (то есть x принадлежит области определения f) и $g(f(x))$ определено (то есть $f(x)$ принадлежит области определения функции g). В остальных случаях функция h не определена.

«**Доказательство**». По предположению существуют алгоритмы, вычисляющие функции f и g по отдельности. Напишем алгоритм, вычисляющий композицию этих функции. Получив вход x, этот алгоритм сначала вызывает функцию f на входе x. Затем, если этот вызов заканчивает работу и даёт какой-то результат y, наш алгоритм запускает функцию g на входе y и её результат (если он получится) выдаёт на выход.

Если алгоритм для f зациклится, или не выдаст результата, то и наш алгоритм зациклится (соответственно не выдаст результата), аналогично для g, то есть действительно алгоритм вычисляет композицию.

Бывают, конечно, языки программирования, где нет конструкций вызова функции (например, машины Тьюринга, которые мы обсудим позже). В таком случае надо написать программу, которая сначала действует как алгоритм для f (на входе x), а потом — если этот алгоритм завершит работу и даст какой-то результат y — как алгоритм g на входе y. Конкретный вид этой программы, конечно, зависит от выбранного языка программирования.

14.9 Не все функции вычислимы

Теорема. Существует функция с натуральными аргументами и значениями, которая не является вычислимой.

Это ещё не теорема Конвея — там говорится о невычислимости конкретной функции, а здесь пока лишь о том, что невычислимые функции вообще бывают. Но и доказать это сильно проще.

Если вы помните про счётные и несчётные множества, то доказательство сводится к одной фразе: для каждой вычислимой функции есть алгоритм, её (и только её) вычисляющий, поэтому множество вычислимых функций счётно, а множество всех функций несчётно.

(Когда говорят о несчётности множества функций с натуральными аргументами и значениями, обычно рассматривают только всюду определённые функции. Но если мы добавим и не всюду определённые, то функций станет только больше.)

Давайте развернём это доказательство — это поучительно и для тех, кто помнит про счётные и несчётные множества, а те, кто не помнят, убедятся, что ничего сложного в этом нет.

Программы в вашем любимом языке программирования можно записать в виде двоичных слов, правда? Будем перечислять все эти программы в порядке увеличения длины слов (для данной длины есть только конечное количество программ). Пусть это будут программы $p_0, p_1, \ldots, p_n \ldots$. Чего мы теперь хотим? придумать такую функцию $\delta(n)$, которая *не вычислялась бы никакой программой* p_i. Что это значит? что для любой программы p_i есть какой-то вход n, на котором она дей-

ствует не так, как требуется для вычисления функции $\delta(n)$. Можно сказать, что мы хотим построить функцию δ, которая отклоняется от всех программ p_i — хоть где-то отклоняется от каждой программы p_i. Хоть где-то — давайте договоримся, что она отклоняется от p_i на входе i. Можно, например, положить

$$\delta(i) = \begin{cases} k+1, \text{если программа } p_i \text{ на входе } i \text{ останавливается с результатом } k; \\ 0, \text{если программа } p_i \text{ на входе } i \text{ не останавливается или не даёт результата.} \end{cases}$$

Заметим, что мы построили всюду определённую функцию δ, чего даже и не обещали заранее.

Обычно это рассуждение называют *диагональной конструкцией* Кантора. Про Кантора понятно — он впервые такой трюк применил, но понятно ли, где здесь диагональ?

Задача. Как надо видоизменить это рассуждение, чтобы построить невычислимую функцию, которая определена на всех натуральных числах и принимает значения 0 и 1? (Множество аргументов, где она равна единице, будет тогда неразрешимым множеством, см. обсуждение выше.)

14.10 Неразрешимость проблемы остановки

Если вам кажется, что доказательство с помощью диагональной конструкции понятное и правильное, тогда вопрос. Как же так получается, что функция δ невычислима, когда прямо там и написан алгоритм её вычисления? Получив i на вход, надо выписать программу p_i, затем запустить её на входе i, посмотреть, остановится ли она и в зависимости от этого действовать. Чем это не алгоритм вычисления функции δ?

Тут есть, конечно, тонкие моменты. Как найти программу p_i алгоритмически? Мы должны предположить, что в нашем языке программирования все двоичные слова можно считать программами (а если получается ошибка при компиляции или интерпретации, значит, соответствующая программа не даёт результата, то есть вычисляет нигде не определённую функцию). Или, по крайней мере, что по двоичному слову можно определить, программа это или нет (тогда не-программы можно вычёркивать и найти p_i для данного i). Так что с этим проблем нет, правда?

Ещё нам нужен алгоритм, который можно запустить на паре (p_i, i), и он будет вести себя как программа p_i на входе i. Такой алгоритм программисты называют *интерпретатором* — и действительно, для всех нормальных языков программирования его можно написать[6]. Так что и тут нет подвоха. А где же он?

Если вы ещё не догадались, то вот где: в невинной фразе «посмотреть, остановится ли она». Что значит «посмотреть»? Ну вот мы запустили программу и смотрим, а она не останавливается и не останавливается. В какой момент мы должны сдаться и решить — всё, бросаем, она не остановится? Иногда об этом можно уверенно

[6]Если можно объяснить, как выполняется программа, то это и есть алгоритм интерпретатора, и что же это за язык, если на нём этот алгоритм нельзя запрограмиировать. А если нельзя объяснить, как выполняется программа, то что ж это за язык программирования? Правда ведь?

судить, посмотрев, что делает программа. Но если она, скажем, ищет контрпример к теореме Ферма перебором, то чтобы сказать с уверенностью, что она не остановится, нужно доказать теорему Ферма — как показал опыт, дело это совсем не такое простое. А мало ли какие ещё могут быть программы. Так что это «посмотреть» — чистой воды жульничество. Если бы можно было узнать по программе и входу, останавливается ли эта программа на этом входе, тогда другое дело — мы могли бы это узнать и потом (если останавливается) дождаться результата, тем самым вычислив функцию δ. И получить противоречие. Так что мы по ходу доказали теорему:

Теорема. Проблема остановки неразрешима: не существует алгоритма, который бы по программе p и входу i определял бы (через конечное число шагов, как и всякий алгоритм), остановится ли программа p на входе i или нет.

Другими словами, функция

$$\mathsf{Halt}(p,i) = \begin{cases} 1, & \text{если программа } p \text{ останавливается на входе } i; \\ 0, & \text{если не останавливается} \end{cases}$$

невычислима.

Ещё одна переформулировка: область определения интерпретатора неразрешима.

Это, пожалуй, самая главная теорема теории вычислимости (если надо выбрать одну самую главную) — но, как видите, доказательство её совсем не сложное.

14.11 Самоприменимость

В предыдущем доказательстве p было двоичным словом (программой), а i числом. Хотя мы уже объясняли, что в этом нет ничего страшного и при необходимости можно переходить от слов к числам и обратно, для тренировки можно изложить те же рассуждения в варианте, когда используются только двоичные слова. Будем рассматривать программы в нашем любимом языке программирования как двоичные слова, и пусть их входом будут тоже двоичные слова.

Теорема. Не существует алгоритма, который по паре (p,x) определяет (давая ответ 0 или 1), останавливается ли программа p на входе x.

Диагональная конструкция теперь превращается в применение программы к себе. Будем говорить, что программа (двоичное слово) *самоприменима*, если при подаче на вход её собственного текста[7] она останавливается с каким-то результатом. Теперь теорему можно вывести из боле сильного утверждения

Теорема. Не существует алгоритма, который по программе p определяет, является ли она самоприменимой.

[7] Что за глупость, скажете вы? Не так уж и глупо, скажем, применить компилятор языка программирования, написанный на том же языке программирования, к самому себе. (Эта технология называется bootstrapping.)

Доказательство. Пусть такой алгоритм S существует. Тогда с его помощью можно построить алгоритм, вычисляющий такую функцию

$$d(p) = \begin{cases} \text{пустое слово, если } p \text{ не самоприменима} \\ \text{результат применения } p \text{ к входу } p \text{ с приписанным нулём, если } p \text{ самоприменима.} \end{cases}$$

(Алгоритм S нужен для того, чтобы определить, какой из двух строк надо пользоваться.) Пусть q — программа, которая вычисляет функцию d. Будет ли она самоприменимой? если нет, то по определению $d(q)$ равно пустому слову, и, значит, q останавливается на q, то есть q самоприменима. Если же q самоприменима, то $d(q)$ получается дописыванием нуля к результату применения q к q, значит, q не вычисляет d. Получилось противоречие. А почему? потому что мы предположили, что алгоритм S для выяснения самоприменимости существует.[8]

14.12 Перечисление останавливающихся программ

Только что мы доказали, что область определения интерпретатора, то есть множество D пар слов (p, i), для которых программа p останавливается на входе i, неразрешима (нельзя алгоритмически проверить принадлежность к этому множеству). Но всё-таки что-то хорошее об этом множестве сказать можно. А именно, оно *перечислимо*. Это значит, что существует алгоритм (программа), который не имеет входа, и после запуска время от времени выдаёт на выход пары из множества D, причём любая пара из D там рано или поздно появится (а никаких пар не из D не появится). Если мы это докажем, то получим такое следствие.

Теорема. Существует перечислимое неразрешимое множество.

Как же доказать, что множество D перечислимо? Для этого вспомним, что на вашем любимом языке программирования можно написать не только интерпретатор, но и отладчик этого самого любимого языка, который исполняет указанную ему программу с указанным входом, делая указанное число шагов. В отличие от интерпретатора, который не определён на паре (p, x), если p не останавливается на x, отладчик определён на всех тройках (p, x, k) и говорит, что произойдёт за k шагов исполнения программы p на входе x — остановится ли она, и если остановится, то какой будет ответ.[9] Ведь правда, вы можете написать такой отладчик для своего любимого языка?[10]

Теперь — как перечислять множество D, имея отладчик. Будем по очереди перебирать все тройки (p, x, k) в каком-то порядке (например, в порядке увеличения

[8] В такой формулировке видна аналогия с рефлексивными парадоксами. Будем называть прилагательное «самоприменимым», если оно само обладает свойством, которое описывает. Скажем, прилагательные «русский» или «трёхсложный» самоприменимы, а «английский» или «пятисложный» — нет. Теперь скажите, самоприменимо ли прилагательное «несамоприменимый»? Вот то-то и оно...

[9] Таким образом, выходом отладчика можно считать пару из булева значения и слова.

[10] Ответ тут не так очевиден, если ваш любимый язык функциональный, вроде лиспа или хаскеля — где там шаги и как их считать? Но тем не менее и для таких языков есть их интерпретаторы, работающие на стандартных процессорах, и можно считать такты работы процессора.

суммарной длины их двоичной записи — ведь правда, вы можете написать алгоритм, который перечисляет их в таком порядке?), и для каждой из них с помощью отладчика проверять, останавливается ли p на x за k шагов. Если останавливается, то на выход мы подаём пару (p, x) и продолжаем работать. При таком способе мы перечислим все пары из D (каждую — бесконечное число раз, поскольку если программа останавливается за k шагов, то и за большее число шагов она тоже остановится). Поэтому D перечислимо.

Ну, а неразрешимость D мы уже доказали.

14.13 Как доказать неразрешимость?

Мы доказали неразрешимость проблемы остановки. Теперь можно доказывать неразрешимость других проблем, *сводя к ним проблему остановки*. Это значит, что мы показываем, что *если* интересующая нас проблема была бы разрешима, *то* и проблема остановки была бы разрешима. Вот простой пример, иллюстрирующий эту схему. Будем называть программу (только в этом разделе — это не стандартный термин, просто для краткости) «гдетоприменимой», если она останавливается хотя бы на одном входе (в противном случае можно было назвать её «нигденеприменимой»).

Теорема. Множество гдетоприменимых программ неразрешимо.

Другими словами, не существует алгоритма, который по программе определяет, есть ли вход, на котором она останавливается, или таких входов нет.

Доказательство. Пусть такой алгоритм S существует. Покажем, как тогда определить, останавливается ли данная программа p на данном входе x. Преобразуем программу p, заменив чтение со входа присваиванием константы («литерала», как говорят программисты) x. Получится программа, которая со входа не читает (то есть её действия не зависят от входа) и которая останавливается тогда и только тогда, когда программа p останавливалась на входе x. Теперь подсунем эту программу алгоритму S, он скажет, будет ли она гдетоприменимой. Но поскольку её действия от входа не зависят, гдетоприменимость означает, что программа p останавливается на x, и тем самым мы алгоритмически получим ответ на вопрос, для которого (как мы доказали) алгоритма не существует.

Тут мы использовали, что описанное преобразование программ (подстановка литерала вместо входа) может быть выполнено алгоритмически: по паре (p, x) можно алгоритмически построить программу q, которая действует на любом входе ровно так же, как p действует на входе x (точнее, нам важно, чтобы для любого y программа q останавливалась на y в том и только в том случае, когда программа p останавливается на x). Но ведь правда, что в вашем любимом языке программирования такое преобразование возможно? Вот и славно.

Отметим ещё одно важное обстоятельство. В «доказательстве» алгоритмической неразрешимости проблемы остановки мы нигде не фиксировали язык программирования. Поэтому это рассуждение пригодно для любого достаточно выразительного языка программирования.

Именно это позволяет доказывать алгоритмическую неразрешимость проблемы

остановки для FRACTRAN'а, как и любое другое утверждение об алгоритмической неразрешимости конкретной задачи. В отличие от рассуждения о неразрешимости проблемы остановки, любое такое доказательство опирается на какой-то язык программирования (ещё говорят, *модель вычислений*) и его свойства.

Требования к такому языку программирования радикально отличаются от обычных. Он нужен не для написания реальных программ, а для доказательства. Поэтому такие языки стараются делать как можно проще. Самым популярным вариантом являются *машины Тьюринга*, о которых речь пойдёт дальше в главе 16.

14.14 Язык программирования для доказательства теоремы Конвея

Однако для доказательства теоремы Конвея удобен другой вариант.[11]

Мы сейчас опишем некоторый примитивный язык программирования. Программы на этом языке используют лишь неотрицательные целые переменные, но это настоящие натуральные числа, а не до 2^{32} или 2^{64}, как обычно бывает — никаких ограничений на их величину нет.

Программа представляет собой последовательность команд, а команды бывают всего двух видов. Во-первых, команда может увеличивать значение какой-то из переменных на 1, то есть иметь вид

```
a <- a+1
```

Следуя примеру программистов, мы будем иногда сокращать эту запись до `a++`. Эта команда выполнима всегда, поскольку ограничений на величину чисел у нас нет. Вторая команда, как можно догадаться, уменьшает число на 1. Тут уже может выйти, что называется, облом — или, как говорят программисты, exception (исключение). Это бывает, когда переменная равна нулю и уменьшать её некуда. В этом случае значение переменной не меняется и происходит «обработка исключения» — мы переходим к исполнению строки программы с номером X, который указан в команде. Короче говоря, разрешены команды вида

```
a <- a-1; exception: go to X
```

Исполняя такую команду, мы смотрим, можно ли уменьшить переменную `a` на единицу, то есть положительно ли `a`. Если да, то уменьшаем, если нет, то переходим к строке с номером `X`. (Можно также считать, что `X` — это не номер строки, а поставленная нами около неё метка, так сказать, перейдя от программирования в машинных кодах к ассемблеру.)

В начале исполнения программы значения всех переменных равны нулю.

Несмотря на убогость такого языка программирования, мы сейчас покажем, как в этом языке можно реализовать все традиционные программистские конструкции (и тем самым транслировать любую программу на этот язык — правда, с сильным замедлением работы программы).

[11] Научное название: машины Минского или автоматы с несколькими счётчиками.

- **Безусловные переходы**. Допустим, мы хотим использовать в программе команды безусловного перехода типа

  ```
  go to X
  ```

  Дойдя до такой команды, исполнитель переходит к строке с номером (или меткой) `X`. Как это сделать? Давайте выделим специальную переменную `null`, которую никогда не будем менять, она как была вначале нулём (по нашему соглашению), так и будет оставаться. Тогда можно написать

  ```
  null <- null-1; exception: go to X
  ```

- **Условные переходы**. Пусть мы хотим реализовать условный переход вида

  ```
  if a>0 then go to X else go to Y
  ```

  (если `a` положительно, переходим к строке `X`, а если равно нулю, то к строке `Y`). Можно попробовать написать так:

  ```
  a <- a-1; exception: go to Y
  go to X
  ```

  (мы уже знаем, как реализовать безусловные переходы, и можем их использовать).

  Это почти правильно, плохо только, что при положительном `a` мы не только переходим к строке `Y`, но и уменьшаем `a` на единицу. Понятно, как это исправить:

  ```
  a <- a-1; exception: go to Y
  a <- a+1
  go to X
  ```

- **Циклы** типа «while» (пока $\langle\ldots\rangle$ повторять $\langle\ldots\rangle$) теперь легко реализовать стандартным образом: вместо

  ```
  while a>0 do
       <...>
  od
  ```

  можно написать

  ```
  A: if a>0 go to B else go to C
  B:     <...>
     go to A
  C:
  ```

(после метки `C` следует то, что шло за циклом). Собственно, примерно это и делает компилятор любого языка высокого уровня, содержащего циклы while.

- **Присваивания**. Мы говорили об управляющих конструкциях, забыв, что в языке не предусмотрены даже присваивания `a<-b` (после которого значение переменной `b` помещается в переменную `a`) или присваивания вида `a<-0`. Последнее, впрочем, легко реализовать в виде цикла

```
while a>0 do a--; od
```

(здесь `a--` сокращает `a<-a-1`; заботиться об исключении не надо, так как условие цикла гарантирует положительность `a`). Можно попробовать примерно так же реализовать и присваивание `a<-b`, сначала поместив в `a` нуль уже известным способом, а затем написав

```
while b>0 do a++; b--; od
```

После этого действительно значение переменной `b` помещается в `a`, но само `b` становится равным нулю. Получается такое «разрушающее» присваивание. Легко сообразить, как исправить положение: надо ввести вспомогательную переменную `tmp`, и написать

```
tmp<-0
while b>0 do a++; tmp++;  b--; od
b<=tmp
```

(последняя строка обозначает разрушающее присваивание, восстанавливающее `b` из `tmp`).

- **Арифметические операции и сравнения**. Теперь понятно, как реализовать операцию сложения `a<-b+c`:

```
a<-0
while b>0 do a++; b--; od
while c>0 do a++; c--; od
```

При этом, правда, разрушатся `b` и `c`, но мы уже умеем копировать без разрушения, так что их можно сохранить заранее. Аналогично можно запрограммировать «безопасное вычитание» , при котором $a-b$ считается равным нулю, когда $a < b$. А именно, безопасно вычесть единицу можно так:

```
if a>0 then a--;
```

а вычесть произвольное `b` можно с помощью цикла:

```
while b>0 do b--; [a--;] od
```

Здесь `[a--;]` обозначает безопасное вычитание единицы. Имея безопасное вычитание, мы очевидным образом программируем переходы вида `if a>=b`, и так далее. После этого можно реализовать умножение (как сложение в цикле), проверку делимости, проверку простоты. Видно, что все управляющие конструкции стандартного языка программирования теперь у нас есть, и дальше мы можем программировать как обычно.[12]

- **Массивы произвольного размера**. Единственное, чего нам ещё не хватает — это массивов произвольного размера (не фиксированного заранее в тексте программы), в которые можно записывать и из которых можно читать числа, указав номер ячейки. В самом деле, любая программа на нашем языке содержит конечное число переменных (их можно подсчитать, читая программу). Как же нам поступить, если надо хранить массив произвольного размера?

  Вспомним, что множество всех конечных последовательностей натуральных чисел счётно: скажем, последовательность $x_1, \ldots, x_n$ можно закодировать числом

  $$2^{x_1} 3^{x_2} 5^{x_3} \ldots,$$

  и разным последовательностям будут соответствовать разные числа в силу однозначности разложения на простые множители. Хотя нет, минуточку — тут мы погорячились: если к последовательности x_i в конце дописать несколько нулей, то ничего не изменится. Более точно надо сказать, что мы кодируем не конечные последовательности, а бесконечные последовательности натуральных чисел, в которых все члены, начиная с некоторого места равны нулю. (Такие бесконечные последовательности иногда называют *финитными*.) Используя это кодирование, можно считать, что одна переменная с натуральными значениями представляет собой бесконечный виртуальный массив натуральных чисел, в котором почти все члены равны нулю. (Нулевой массив, где все числа равны нулю, кодируется при этом как 1.) Остаётся реализовать операции чтения и записи в i-ю ячейку этого массива. Для начала мы пишем код, который по i находит i-е простое число p_i (обычным образом, при этом используется лишь конечное число переменных), затем можем увеличивать значение в i-ой ячейке виртуального массива, умножая на p_i, и уменьшать, деля на p_i, а после этого можем и копировать число из этой виртуальной ячейки с помощью уменьшения в цикле, как мы это делали выше. Мы не будем сейчас вдаваться в подробности — надеясь, что все, кто дочитали до этого места, легко могут это сделать сами (и эти подробности всё равно читать не будут).

[12] Более сложно реализовать рекурсию — но это и не удивительно, первые компиляторы тоже этого не умели. Как это делается, описывается в учебниках по реализации языков программирования — для этого нужен стек произвольного размера, и о массивах произвольного размера речь пойдёт дальше. Поскольку бывают языки программирования, в которых нет рекурсии и которые тем не менее универсальны (можно запрограммировать всё, что можно запрограммировать на других языках), то рекурсия для доказательства теоремы Конвея не нужна.

14.15 Сведение проблемы остановки: от программ к пасьянсам

Теперь вспомним, для чего нам нужен был такой простой язык программирования, а именно, покажем, что по всякой программе π на этом языке можно написать последовательность дробей и начальное целое число таким образом, что FRACTRAN-пасьянс остановится тогда и только тогда, когда останавливается программа π. На самом деле там не только вопросы об остановке равносильны, а просто каждый шаг исполнения программы соответствует одному действию по правилам FRACTRAN.

Вот как эта последовательность пишется. Пусть в программе π у нас используется n переменных $x_1, \ldots, x_n$. Текущие значения этих переменных будут закодированы в текущем целом числе FRACTRAN-пасьянса. А именно, в его разложении на множители i-е простое число p_i будет входить в степени x_i. Тогда операция увеличения x_i на единицу соответствует в пасьянсе умножению на p_i, а уменьшение x_i на единицу соответствует делению на p_i. Заметим, что логика перехода по исключению как раз соответствует правилам пасьянса: если мы пытаемся уменьшить x_i на 1, а оно уже равно нулю, то число не делится на p_i, и соответствующая операция пасьянса неприменима, и надо искать дальше (следующую применимую).

Как знают конструкторы микропроцессоров, помимо собственно переменных, нужно где-то хранить program counter — информацию о том, какая строка программы сейчас выполняется. Для этого мы тоже будем использовать простые числа, но не те, которые зарезервированы для переменных, а другие. Пронумеруем строки программы простыми числами (не занятыми на обслуживание переменных) $q_1, \ldots, q_m$. Договоримся, что если текущая строка программы помечена числом q, то текущее число пасьянса содержит простой множитель q (в первой степени). Таким образом, в каждый момент ровно одно из чисел q_i входит в разложение текущего числа пасьянса (в первой степени).

Всё это пока описания, как всё хорошо будет — но надо объяснить, как мы этого добьёмся. Допустим, в программе есть строка

$$\begin{array}{ll} q: & \texttt{x++} \\ q': & \ldots \end{array}$$

Здесь q — простое число, выбранное номером этой строки, а q' — другое простое число, выбранное номером следующей строки. Эта строка выполняется в тот момент, когда текущей является строка номер q (когда число в пасьянсе делится на q), в результате переменная x увеличивается на 1 (число умножается на p_i, если x кодируется как показатель степени при p_i), а текущей строкой становится строка с номером q'. Все эти действия будут обеспечены, если в список дробей в пасьянсе включить дробь

$$\frac{p_i q'}{q},$$

и позаботиться о том, чтобы не сработали более ранние дроби.

Вычитание единицы чуть сложнее, поскольку надо заботиться об исключениях. Для строки

q : `x--; if exception, go to` q''
q' : ...

надо добавить в список дробей группу из двух дробей

$$\frac{q'}{qp_i}, \frac{q''}{q}$$

Они могут сработать, если число в пасьянсе делится на q (то есть если текущая строка имеет метку q). Первая сработает, если к тому же число делится на p_i, то есть если значение переменной x, которая кодируется как показатель степени при p_i, больше нуля. Если же это значение равно нулю, то первая дробь не сработает, и исключение подхватит вторая: в ней никакие переменные не изменяются, а текущей становится строка с меткой q''.

Группы дробей, соответствующие разным строкам программы, могут идти в любом порядке, поскольку всё равно в каждый момент текущее число пасьянса делится только на один номер строки (мы об этом уже говорили, и это свойство сохраняется при умножении на построенные дроби). В качестве начального числа пасьянса надо взять простое число, являющееся меткой первой строки программы. После этого исполнение правил пасьянса будет в точности соответствовать исполнению программы, и остановка пасьянса равносильна остановке программы.

Следовательно, если бы у нас был алгоритм, проверяющий, остановится ли данный пасьянс, его можно было бы использовать для проверки остановки программ на нашем мини-языке программирования. А поскольку на этот язык можно алгоритмически транслировать программы с любого другого, то и вообще проблема остановки была бы разрешима.

Что и завершает доказательство теоремы Конвея.

Задача 14.1. Покажите, что существует последовательность дробей, для которой задача «определить по данному числу m, остановится ли игра FRACTRAN, начатая с числа m», алгоритмически неразрешима.

Разница с теоремой Конвея в том, что там требовался алгоритм, который по списку дробей и начальному числу даёт ответ (и его не было), а теперь мы спрашиваем: может быть, для каждого отдельного списка существует алгоритм, который по начальному числу даёт ответ. Это, вообще говоря, более слабое свойство — но в данном случае и такого алгоритма тоже нет.

Лекция 15

Вычислимые функции, разрешимые и перечислимые множества

Рассуждения предыдущей главы были не очень формальными и во многом опирались на наши интуитивные представления об алгоритмах. Чтобы превратить их в настоящие математические доказательства, нужно по крайней мере дать строгое определение алгоритма, то есть программы на некотором достаточно мощном языке программирования, средствами которого выразим любой другой язык программирования.

Все известные точные определения понятия «алгоритм» довольно сложные и громоздкие. В главе 16 мы дадим такое определение, одно из стандартных для математики и теоретической информатики — так называемые машины Тьюринга. Как уже говорилось в прошлой главе, целью определения алгоритма являются доказательства, а не написание реальных программ. Машины Тьюринга как язык программирования весьма просты, если не сказать убоги, что облегчает доказательства. Но даже такая простая вычислительная модель, как машина Тьюринга, порождает много технических деталей, за обсуждением которых не всегда легко разглядеть суть.

Между тем очень многие теоремы о вычислимости легко понять, не привлекая точного определения алгоритма. Вместо этого мы будем опираться на интуицию, возникающую из опыта использования реальных языков программирования. А именно, мы будем использовать в рассуждениях возможность алгоритмической реализации некоторых процедур. Обычно про каждую такую процедуру нетрудно сообразить, как она реализуется на каком-то языке программирования (а значит, и на любом другом достаточно мощном языке программирования).

Чтобы превратить рассуждения этой главы в строгие доказательства, придётся после выбора формального определения алгоритма доказать большое количество лемм вида «такое-то действие реализуется алгоритмом» (в том строгом смысле, который фиксируется определением). Отчасти мы проделаем такую работу в следующей главе 16, используя определение алгоритма через машины Тьюринга.

Пока мы будем указывать некоторые свойства алгоритмов и расширять спи-

сок таких свойств по мере необходимости. Фактически именно так мы поступали в предыдущей главе.

Свойство 0. *Алгоритм имеет конечное описание.*

Это довольно естественное свойство. Ведь мы договорились считать, что алгоритм — это программа на некотором языке программирования. Текст программы и есть описание алгоритма.[1] Заметим, что мы не накладываем никаких ограничений на размер описания. Важно лишь, что оно конечно. Так что любая сколь угодно сложная программа удовлетворяет этому свойству.

Свойство 1. *Алгоритм выполняется по шагам.*

Это важное для нас свойство, и оно действительно выполняется для алгоритмов в жизни. Когда мы выполняем цепочку инструкций, то мы последовательно переходим от одной инструкции к другой. Когда мы пишем программу, то у нас есть возможность запустить программу по шагам.

Свойство 2. *На каждом шаге алгоритма должно быть чётко и однозначно определено, что нужно делать на следующем шаге.*

Это принципиальное свойство. Все действия в алгоритме должны быть однозначно определены, не должно быть никакой неопределённости и недосказанности.

В точном определении алгоритма должно быть указано, в частности, в каком именно виде алгоритм получает входные данные и в каком виде он выдаёт результат работы. Как уже обсуждалось в прошлой главе, форматы входных данных и результата работы не очень важны. В этой главе мы будем предполагать, что все математические объекты, к которым применяются алгоритмы, кодируются натуральными числами (то есть целыми неотрицательными числами).

Для простоты рассуждений удобны взаимно однозначные кодировки (чтобы не обсуждать, что делать с теми числами, которые ничего не кодируют).

Приведём два примера таких биективных кодировок, которые используются ниже в рассуждениях. В этих примерах кодировки достаточно просты, чтобы их можно было реализовать программно.

Пример 15.1. Между двоичными словами и натуральными числами есть такая биекция: слову a, состоящему из нулей и единиц, сопоставим число n, которое на единицу меньше числа, заданного в двоичной записи словом $1a$.

Контрольный вопрос 15.2. Докажите, что отображение, заданное в предыдущем примере, и в самом деле биекция. (Основное, что нужно понять — зачем вычитается единица. Для этого вспомните определение биекции из главы 6.)

[1] Программа может дополнительно вызывать какие-то библиотеки, но и эти библиотеки записаны в файлах, содержащих конечное число байтов.

Пример 15.3. Между парами натуральных чисел и натуральными числами есть биекция, описанная в главе 6, пример 6.20: паре натуральных чисел (x, y) сопоставлено число

$$[x, y] = \mathrm{code}_2(x, y) = \binom{x + y + 1}{2} + y$$

(мы изменили название функции по сравнению с примером 6.20).

Обозначение этой биекции квадратными скобками удобно в длинных формулах, поскольку оно короче. Но оно выглядит непривычно (и многозначно — что только не обозначают математики квадратными скобками), так что мы в более простых ситуациях будем использовать мнемоническое обозначение code_2.

Этих двух несложных примеров нам будет достаточно для многих рассуждений в этой главе.

А теперь дадим основное для математической теории алгоритмов определение. Отметим, что оно ссылается на определение алгоритма, которое мы пока не фиксируем, используя вместо него некоторый открытый список свойств алгоритмов.[2]

Определение 15.1. Функция $f\colon \mathbb{N} \to \mathbb{N}$ называется вычислимой, если существует алгоритм P, который на любом входе $n \in \mathbb{N}$ выдаёт $f(n)$.

Чтобы говорить о вычислимости функции, заданной на каком-то другом множестве, нужно задать кодировку этого множества натуральными числами.

У алгоритмов есть такая особенность, что они в принципе не обязаны выдавать какой-то ответ. На некоторых входах вполне может получиться так, что цепочка инструкций или программа приводит либо к завершению работы без ответа, либо и вовсе к зацикливанию, то есть продолжению выполнения алгоритма до бесконечности без выдачи ответа. Поэтому вычислимые функции могут быть *частичными* (определёнными не для всех натуральных чисел).

Нам удобно в этой и следующей главе под записью $f\colon A \to B$ понимать частичные функции, то есть такие функции, что для некоторых аргументов $n \in A$ значение функции $f(n)$ может быть не определено. Напомним, что если функция f определена на всех элементах $n \in \mathbb{N}$, мы говорим, что функция f *всюду определённая* или *тотальная*.

Наше определение вычислимой функции относится и к не всюду определённым функциям. Для тех аргументов $n \in \mathbb{N}$, для которых $f(n)$ не определено, алгоритм P не выдаёт никакого результата.

15.1 Примеры вычислимых функций

Функции на натуральных числах, с которыми мы сталкиваемся в обычной математике, как правило оказываются вычислимыми. Как уже говорилось, мы будем аргументировать вычислимость конкретных функций ссылками на интуицию работы

[2] Если бы мы зафиксировали список свойств алгоритмов, то получили бы обычное для математики аксиоматическое определение алгоритма. Однако такой подход в случае алгоритмов не работает: алгоритмы приходится в конечном счёте определять «явно».

с реальными языками программирования. Поэтому чтение данного раздела необязательно для понимания остального материала главы. Однако полезно посмотреть на примеры и выделить те свойства алгоритмов, которые в них используются.

Функция $f(n) = n$ вычислима очень простым алгоритмом: получив n на вход, нужно подать его на выход.

Функция $\mathrm{next}(n) = n + 1$ также вычислима. Прибавление единицы реализовано в обычных языках программирования. Но и сам алгоритм прибавления единицы несложен и всем известен, если число задано в позиционной системе. (Для определённости мы далее в этом разделе предполагаем задание числа в позиционной системе.)

Функция $f(n) = n^2$ также вычислима: получив n на вход, нужно возвести его в квадрат и передать результат на выход. Операция возведения в квадрат (или умножения) также реализована в языках программирования. Человек же может реализовать эту функцию «руками» с помощью умножения в столбик.

Аналогично, функция $f(n) = 2^n$ также вычислима. Нужно умножить число 2 само на себя n раз.

Функция $f(n) = \lceil \log n \rceil$ (не определённая для $n = 0$) вычислима: взятие логарифма и целой части реализовано в обычных языках программирования. Если же мы хотим вычислить эту функцию «элементарными средствами», то можно перебирать все m от 1 до n и последовательно для каждого вычислять 2^m. На выход тогда нужно будет выдать последнее m, для которого $2^m \leqslant n$ (для $n = 0$ такого m не будет, так что и никакого результата мы не выдадим). Ничего, что предложенный нами алгоритм — не самый эффективный с точки зрения числа шагов работы. Мы не учитываем эффективность алгоритмов при определении вычислимости функции.

Какие свойства алгоритмов мы использовали в этих примерах? Разумеется, не только те три абстрактных свойства, которые названы выше. Скажем, для умножения в столбик нужны циклы и вызов подпрограммы (сложения). Аналогично для намеченного выше алгоритма вычисления $\lceil \log n \rceil$. В этом алгоритме ещё нужно сравнивать числа по величине, алгоритм такого сравнения общеизвестен. Кроме того, в этом алгоритме в цикле вызывается вычисление 2^m.

Сформулируем два способа комбинирования алгоритмов, которых достаточно для предыдущих примеров (впрочем, и далее других нам не понадобится).

Свойство 3 (комбинирование алгоритмов)**.** *Пусть имеются алгоритмы A, B, C, причём результат работы алгоритма C «логический» (равен 0 или 1). Тогда их можно использовать при составлении других алгоритмов следующими способами.*

Вызов подпрограммы: *допустима инструкция «исполнить алгоритм A и сохранить результат его работы для последующих вычислений».*

Условный переход: *допустима инструкция «исполнить алгоритм C; если результат его работы равен 1, то исполнить алгоритм A; в противном случае исполнить алгоритм B».*

Контрольный вопрос 15.4. Как реализовать с помощью конструкций из свойства 3 исполнение некоторого алгоритма A на всех входах от 0 до n? Какие алгоритмы, помимо A, потребуются в этих конструкциях?

Замечание 15.1. Поскольку мы не экономим ни время работы, ни память, используемую алгоритмом, без ограничения общности можно считать, что алгоритм хранит результаты всех проделанных вычислений и использует их по мере надобности.

Замечание 15.2. Чтобы комбинированный алгоритм выдал результат, каждый из исполняемых алгоритмов обязан закончить работу. Скажем, что случится, если исполнить конструкцию условного перехода

```
if a/b = 1 then A else B
```

при `b=0`? В реальных языках программирования в таком случае как правило выдаётся какое-нибудь сообщение об ошибке.

У нас никаких сообщений вообще не предполагается. Поэтому по нашим правилам нужно считать, что алгоритм, исполняющий такую инструкцию при таком значении переменной `b`, не даёт никакого результата.

Пример 15.5. Напомним, что через $\mathrm{code}_2(x, y)$ мы обозначили биекцию между парами натуральных чисел и натуральными числами. Обратная функция code_2^{-1} является отображением множества натуральных чисел на пары натуральных чисел. Из неё можно построить две функции из натуральных чисел в натуральные, выбирая первое или второе число в паре. Более точно, если $\mathrm{code}_2^{-1}(n) = (x, y)$, то по определению

$$\begin{aligned} l(n) &= x, \\ r(n) &= y. \end{aligned} \tag{15.1}$$

Функции $l(n)$ и $r(n)$ вычислимы. Вспомним, что в главе 6, в примере 6.28 мы обсуждали нахождение обратной функции $\mathrm{code}_2^{-1}(n)$ и нашли такое правило (формулируется с изменением обозначений):

- найти наибольшее k такое, что $\binom{k}{2} \leqslant n$;
- вычислить $y := n - \binom{k}{2}$ и $x := k - y - 1$;
- передать пару $(x; y)$ на выход.

Легко видеть, что это ещё один пример комбинирования алгоритмов с помощью свойства 3.

Чтобы получить значения функций $l(n)$ и $r(n)$, нужно передать на выход x и y соответственно.

Контрольный вопрос 15.6. Вычислимость каких функций предполагается в указанном выше алгоритме вычисления $l(n)$ и $r(n)$?

Считая выполненным свойство 3 уже несложно доказать «без кавычек» замкнутость вычислимых функций относительно композиции.

Лемма 15.2. *Если две функции $f, g\colon \mathbb{N} \to \mathbb{N}$ вычислимы, то и их композиция $g \circ f$ вычислима.*

Стоит уточнить, что если для какого-то $n \in \mathbb{N}$ функция f не определена, то не определена будет и композиция $g \circ f$. Если $f(n)$ определено, но функция g не определена на $f(n)$, то вновь композиция $g \circ f$ не определена на n. Во всех остальных случаях композиция определена (и равна $g(f(n))$).

Доказательство леммы 15.2. Для вычисления композиции мы можем применить следующий алгоритм. Получив на вход n, этот алгоритм исполняет алгоритм вычисления $f(n)$. Если он что-то выдал, то на результате его работы исполняется алгоритм вычисления g. Если этот алгоритм в свою очередь что-то выдаёт, то результат его работы подаётся на выход комбинированного алгоритма.

Свойство 3 говорит, что такой комбинированный алгоритм существует. Этот комбинированный алгоритм выдаёт результат в точности в тех случаях, когда композиция определена, и этот результат равен значению композиции функций. □

Пример 15.7. Из леммы 15.2 сразу следует, что, например, функции 2^{n^2}, 2^{2^n}, $\lceil \log n \rceil^n$ вычислимы как композиции вычислимых функций.

Заметим, что свойство 3 обеспечивает вычислимость не только композиций, но и более сложных функций, которые получаются подстановкой вычислимой функции в один из аргументов вычислимой функций от нескольких аргументов. По этой причине вычислимы функции $n + 2^n$, $\lceil \log n \rceil^n \cdot 2^{n^2}$, и т.п.

Пример 15.8. Для тренировки определим вычислимую биекцию между тройками натуральных чисел и натуральными числами правилом

$$\mathrm{code}_3(x, y, z) = \mathrm{code}_2(\mathrm{code}_2(x, y), z).$$

Вычислимость этой функции обеспечивается свойством 3. Биективность легко проверить: отображение $(x, y, z) \mapsto (\mathrm{code}_2(x, y), z)$ является биекцией между тройками натуральных чисел и парами натуральных чисел, а композиция биекций является биекцией.

Пример 15.9. Можно пойти и дальше, построив вычислимые биекции $\mathbb{N}^k \to \mathbb{N}$ при $k > 2$ по рекуррентному правилу

$$\mathrm{code}_{k+1}(x_1, \ldots, x_k, x_{k+1}) = \mathrm{code}_2(\mathrm{code}_k(x_1, \ldots, x_k, x_{k+1}), x_{k+1}). \tag{15.2}$$

Каждое из отображений code_k является композицией биекций, как и в примере с тройками натуральных чисел (только теперь это композиция $k - 1$ биекций). Вычислимость code_k вытекает из свойства 3 (многократный вызов подпрограммы).

Пример 15.10. Следующий шаг: задать вычислимую биекцию множества конечных последовательностей натуральных чисел $\mathbb{N}^*$ и натуральных чисел. Последовательности $x_1, \ldots, x_\ell$ длины ℓ сопоставим $f(x_1, \ldots, x_k) = \mathrm{code}_2(\ell, \mathrm{code}_\ell(x_1, \ldots, x_\ell))$.

Проверим, что это биекция, предъявив обратное отображение. Числу n взаимно однозначно соответствует пара натуральных чисел $(\ell, s) = \mathrm{code}_2^{-1}(n)$. Числу s взаимно однозначно соответствует последовательность $\mathrm{code}_\ell^{-1}(s)$ длины ℓ. Она и только она отображается отображением f в число n.

Алгоритм вычисления такой функции запускает вычисления по рекуррентному соотношению (15.2) в цикле (подумайте, как организовать такое вычисление).

Замечание 15.3. Мы настолько увлеклись примерами вычислимых функций, что почти добрались до одного из формальных определений алгоритма: с помощью так называемых *рекурсивных* функций. Это определение постулирует вычислимость очень простых функций (выбор одного из аргументов функции, прибавление единицы) и замкнутость вычислимых функций относительно трёх несложных операций: подстановки функций в функцию, рекурсивного определения функции (как в примере 15.10 и частичного обращения (как в алгоритме вычисления функции $\lceil \log n \rceil$). Эти операции выражаются через наши правила комбинирования алгоритмов. Кроме того, определение утверждает, что других вычислимых функций нет. И это самая важная его часть, совершенно удивительная на первый взгляд и не имеющая простого наглядного объяснения. К каким только ухищрениям ни прибегают программисты при написании программ. И все они в принципе[3] выражаются очень ограниченными инструментами. (Напомним, что в предыдущей главе мы проделали аналогичную работу и наметили построение очень бедного языка программирования, на котором тем не менее можно переписать любую программу любого другого языка программирования.)

Такое определение вычислимых функций наиболее удовлетворительно с точки зрения математики: оно сравнительно короткое и не использует ничего лишнего. Но для доказательства теорем о вычислимых функциях это становится серьёзным недостатком: приходится формулировать и доказывать много технических результатов.

Мы предпочитаем не связывать себе руки и будем более вольно пользоваться сформулированными выше свойствами, чтобы сделать рассуждения более понятными, пусть и в ущерб строгости изложения.

В этой книге мы не обсуждаем рекурсивные функции. Для первого знакомства с ними читателю рекомендуется книга Н. К. Верещагина и А. Шеня «Вычислимые функции».

Подведём итог этого обширного набора примеров. Обычно мы встречаемся с вычислимыми функциями на натуральных числах. Чтобы предъявить невычислимую функцию, нужно постараться. Этим мы и займёмся далее.

[3]Это очень важная оговорка: вспомните, что мы не заботимся об эффективности программ, в отличие от программистов.

15.2 Не все функции вычислимы (повторение)

Но начнём с другого вопроса, позволяющего лучше понять суть понятия вычислимости.

Обсудим **неправильное утверждение** и его «доказательство» (содержащее ошибку, разумеется). Чтобы сформулировать это утверждение, нам нужно понятие продолжения функции. Функция $g\colon \mathbb{N} \to \mathbb{N}$ называется *продолжением* функции $f\colon \mathbb{N} \to \mathbb{N}$, если для всякого $n \in \mathbb{N}$, при котором определено $f(n)$, определено и $g(n)$, и при этом $f(n) = g(n)$. Таким образом, g совпадает с f на области определения последней, но может также быть определена и для других значений n.

Итак, давайте попробуем доказать следующее утверждение (которое, подчеркнём ещё раз, на самом деле неверно): *для всякой вычислимой функции* $f\colon \mathbb{N} \to \mathbb{N}$ *существует всюду определённое вычислимое продолжение* $g\colon \mathbb{N} \to \mathbb{N}$. Кажется, что это утверждение вполне разумно и доказать его можно так. Построим алгоритм для g. Получив на вход n, запустим алгоритм для вычисления f. Если алгоритм выдал какой-то результат, то подаём его на выход. Если нет, то выдаём произвольное значение (скажем, 0). Может показаться, что действительно это рассуждение показывает, что всякую вычислимую функцию можно вычислимо продолжить на все натуральные числа. Но на самом деле в этом рассуждении есть ошибка. Прежде чем читать дальше (ниже звёздочек), полезно попробовать найти ошибку самостоятельно.

* * *

Ошибка в этом рассуждении состоит в том, что алгоритм для вычисления f может не выдать ответа довольно неприятным для нас образом: алгоритм может продолжать работать, так и не выдавая ответа. И тогда в рассуждении выше неясно, в какой момент следует решить, что алгоритм для f не выдал ответа и прервать его: то ли алгоритм уже зациклился и не остановится никогда, то ли он ещё делает содержательную работу и через какое-то время выдаст результат.

Обдумывая это затруднение, хочется немного поправить рассуждение и как-то решить эту проблему: может быть, надо просто дать алгоритму поработать достаточно долго, и тогда станет ясно, выдаст он ответ или нет. В действительности все эти попытки обречены на неудачу: не только наше доказательство содержит ошибку, но и утверждение, которое мы пытаемся доказать, неверно! А именно, существует вычислимая функция, у которой нет всюду определённого вычислимого продолжения. Мы докажем это немного позже, см. теорему 15.16 ниже.

Но важный вывод, который мы должны сделать из разобранного ошибочного доказательства, состоит в том, что алгоритм, который не выдаёт ответ, может делать это, продолжая работать, так что по нему может быть не ясно, планирует он выдать ответ в будущем или нет. Так что конструкции вида «запустим некоторый алгоритм, и если он ничего не выдаст, то сделаем то-то и то-то» опасны. Если мы хотим воспользоваться такой конструкцией, то нам нужно дополнительно гарантировать, что запускаемый алгоритм остановится (пусть и не выдавая ответа).

Как мы уже упоминали выше, привести пример невычислимой функции не так уж легко. Тем не менее, совсем нетрудно понять, что невычислимые функции существуют. Мы это уже сделали в предыдущей главе, см. раздел 14.9. Кратко повторим это рассуждение в силу его важности.

Лемма 15.3. *Существует невычислимая функция* $f\colon \mathbb{N} \to \mathbb{N}$.

Доказательство. В главе 8 мы обсуждали, что множество функций из натуральных чисел в натуральные числа континуально. С другой стороны, как мы договорились выше, каждый алгоритм имеет конечное описание, то есть является конечной последовательностью символов в фиксированном конечном алфавите. Множество конечных последовательностей, элементы которых принадлежат конечному множеству символов, счётно. Это мы также проверили в главе 8.

Таким образом, множество функций континуально, а множество алгоритмов счётно. При этом каждый алгоритм вычисляет не более одной функции. Так что алгоритмов просто не хватит на все функции, а значит, существует невычислимая функция. □

В этом доказательстве мы вновь применили мощностной аргумент. Он позволил нам доказать существование невычислимой функции, но не позволяет предъявить никакую конкретную невычислимую функцию. Для этого нам потребуется другое рассуждение, см. ниже теоремы 15.17 и 15.19 или обсуждение проблемы остановки в предыдущей главе.

15.3 Разрешимые множества

Вторым базовым объектом, кроме функций, для нас являются множества. В предыдущей главе уже появлялись разрешимые множества. Поговорим о них подробнее.

Определение 15.4. Множество $A \subseteq \mathbb{N}$ называется *разрешимым*, если существует алгоритм, который для всякого входа $n \in \mathbb{N}$ выдаёт 1, если $n \in A$, и выдаёт 0, если $n \notin A$.

Заметим, что алгоритм, разрешающий какое-либо множество, всегда выдаёт что-то в качестве ответа.

Разрешимость множеств можно сформулировать в терминах вычислимости функций. Для этого напомним определение характеристической (или индикаторной) функции, которое уже появлялось в главе 5.

Определение 15.5. *Характеристической функцией* множества $A \subseteq \mathbb{N}$ называется функция $\chi_A\colon \mathbb{N} \to \{0, 1\}$, для которой

$$\chi_A(n) = \begin{cases} 1 & n \in A, \\ 0 & n \notin A. \end{cases}$$

Лемма 15.6. *Множество $A \subseteq \mathbb{N}$ разрешимо тогда и только тогда, когда функция χ_A вычислима.*

Доказательство. Действительно, алгоритм, разрешающий множество A, по определению есть в точности алгоритм, вычисляющий функцию χ_A. □

Свойство множества «быть разрешимым» замкнуто относительно основных теоретико-множественных операций.

Лемма 15.7. *Если множества $A, B \subseteq \mathbb{N}$ разрешимы, то разрешимы и множества $A \cap B$, $A \cup B$, $\mathbb{N} \setminus A$.*

Доказательство. Предъявим алгоритм, разрешающий $A \cap B$. Получив на вход n, запустим сначала алгоритм, разрешающий A. Когда он выдаст ответ (а он сделает это по определению разрешимости), запустим алгоритм, разрешающий B, на том же входе. Когда и второй алгоритм выдаст ответ, то мы выдадим ответ 1, если оба алгоритма выдали ответ 1, и выдадим ответ 0 иначе.

Алгоритм, разрешающий множество $A \cup B$, устроен почти так же, нужно лишь поменять правило, по которому выдаётся выход. Если хотя бы один из промежуточных алгоритмов выдаёт 1, то и мы выдаём 1, а если оба выдают 0, то мы также выдаём 0.

Наконец, алгоритм, разрешающий множество $\mathbb{N} \setminus A$, устроен совсем просто. На входе n мы запускаем алгоритм для разрешения множества A и выдаём 1, если он выдал 0, и выдаём 0, если он выдал 1. □

Замечание 15.4. Альтернативно, можно доказывать разрешимость нужных нам множеств через вычислимость характеристических функций. Для этого достаточно заметить, что $\chi_{A\cap B}(n) = \chi_A(n) \cdot \chi_B(n)$, $\chi_{A\cup B}(n) = 1 - (1 - \chi_A(n)) \cdot (1 - \chi_B(n))$, $\chi_{\mathbb{N}\setminus A}(n) = 1 - \chi_A(n)$. Теперь вычислимость этих функций легко следует из вычислимости функций χ_A и χ_B.

Задача 15.11. Докажите, что если множества $A, B \subseteq \mathbb{N}$ разрешимы, то разрешимо и множество $A \triangle B$.

15.4 Перечислимые множества

Наряду с разрешимыми множествами оказывается важным также понятие перечислимого множества. Это понятие также возникало в предыдущей главе, сейчас мы его уточним и обсудим основные свойства перечислимых множеств.

Определение 15.8. Множество $A \subseteq \mathbb{N}$ называется *перечислимым*, если существует алгоритм P, который при запуске на пустом входе (то есть не получая на вход ничего) выдаёт все элементы множества A. Алгоритм выдаёт элемент «целиком», а не бит за битом, то есть в какой-то момент сообщает миру очередной результат перечисления (иногда для наглядности говорят «печатает», подразумевая, что напечатанное сообщение уже невозможно изменить в ходе дальнейшей работы). Другими

словами, по мере своей работы алгоритм P постепенно выдаёт последовательность $p_0, p_1, p_2, \ldots$ (которая может быть как конечной, так и бесконечной), причём все элементы последовательности лежат в A и, наоборот, для всякого $a \in A$ существует k, для которого $p_k = a$.

Замечание 15.5. Обратите внимание на отличие от предыдущих определений: алгоритм P может не останавливаться (и, более того, заведомо не останавливается, если он перечисляет бесконечное множество). Важно лишь, чтобы всякий элемент A был выдан через конечное число шагов работы алгоритма.

Такое определение перечисляющего алгоритма не единственно возможное, ниже у нас появятся характеризации перечислимых множеств с помощью обычных вычислимых функций. То есть, понятие перечисляющего алгоритма избыточно. Но оно даёт наглядную картину происходящего, поэтому мы его будем использовать.

Если мы говорим о перечислении каких-то объектов, отличных от натуральных чисел, то предполагаем, как уже объяснялось, что речь идёт о перечислении кодировок этих объектов натуральными числами.

В качестве несложного упражнения предлагаем читателю решить следующие задачи, чтобы освоиться с понятием перечисляющего алгоритма и нашим соглашением о кодировках.

Мы рекомендуем читателю самостоятельно решать задачи, предлагаемые по ходу рассказа. Однако вычислимость — трудная тема. Поэтому мы приводим решения некоторых задач в последнем разделе этой и следующей глав. Настоятельно рекомендуем читать решения лишь после того, как задача решена самостоятельно или, по крайней мере, после длительных неудачных попыток решения.

Задача 15.12. Приведите явный алгоритм перечисления множества $\mathbb{N} \times \mathbb{N}$.

(См. решение в последнем разделе главы.)

Задача 15.13. Докажите, что множество, состоящее из таких наборов $(v, w, x, y, z) \in \mathbb{N}^5$, что $v^5 + w^5 + x^5 + y^5 = z^5$ перечислимо. Предъявите явный алгоритм перечисления.

(См. решение в последнем разделе главы.)

Замечание 15.6. Множество пятёрок из последней задачи непусто:

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5.$$

Обратите внимание, что в определении перечислимого множества мы не требуем, чтобы элементы множества A не повторялись в выдаче алгоритма P. Нужно лишь, чтобы все элементы A в последовательности встретились, а ничего, кроме элементов A, не встретилось. Впрочем, эта свобода ничего не меняет.

Задача 15.14. Докажите, что если множество $A \subseteq \mathbb{N}$ перечислимо, то есть алгоритм, перечисляющий элементы множества A без повторений.

(См. решение в последнем разделе главы.)

Понятие перечислимости оказывается более широким, нежели понятие разрешимости.

Лемма 15.9. *Если множество $A \subseteq \mathbb{N}$ разрешимо, то оно перечислимо.*

Доказательство. Пусть P — алгоритм, разрешающий A. Построим алгоритм, перечисляющий A. Ничего не получая на вход, этот алгоритм перебирает последовательно натуральные числа $0, 1, 2, \ldots$ и для каждого из них запускает алгоритм P для проверки, лежит ли это число в A. Когда алгоритм P выдаёт результат (а он это обязательно делает по определению разрешимости), то мы выдаём наше число на выход, если P выдал 1, и не выдаём в противном случае. После этого мы переходим к следующему числу. Результирующий алгоритм печатает в точности все элементы множества A (причём в порядке возрастания). □

Алгоритм в доказательстве предыдущей леммы никогда не остановится. Но это и не требуется от алгоритма, перечисляющего множество.

Как и разрешимость, свойство перечислимости сохраняется при применении к множествам некоторых базовых операций.

Лемма 15.10. *Если множества $A, B \subseteq \mathbb{N}$ перечислимы, то перечислимы и множества $A \cap B$ и $A \cup B$.*

Доказательство этой леммы несколько сложнее, чем доказательство аналогичной леммы для разрешимых множеств. По сути, здесь мы первый раз используем одно из базовых свойств алгоритмов — пошаговую работу.

Доказательство. Сначала построим алгоритм, перечисляющий $A \cup B$. Для этого рассмотрим алгоритмы P_A и P_B, перечисляющие A и B соответственно.

Естественной идеей было бы попробовать сделать то же самое, что и в случае разрешимых множеств: сначала запустить алгоритм P_A, а затем запустить алгоритм P_B. Но в текущей ситуации у нас есть существенное отличие от случая разрешимых множеств: там у нас была гарантия, что алгоритмы завершат свою работу. Здесь же у нас такой гарантии нет, и если алгоритм P_A свою работу не заканчивает, то мы никогда не перейдём к выполнению алгоритма P_B.

Вместо этого наш алгоритм будет работать следующим образом. Сначала мы запустим алгоритм P_A на один шаг его работы. После этого мы временно прерываем работу алгоритма P_A и запускаем алгоритм P_B на один шаг. После этого мы прерываем работу алгоритма P_B и запускаем P_A на второй шаг его работы. После этого мы делаем второй шаг работы алгоритма B, и так далее.

Таким образом работа нашего алгоритма состоит из следующих итераций: для каждого $k = 1, 2, 3, \ldots$ на итерации с номером k мы запускаем сначала алгоритм P_A на ещё один шаг, а затем алгоритм P_B на ещё один шаг. После k-й итерации в обоих алгоритмах P_A и P_B у нас будет сделано k шагов. По сути, мы запускаем наши алгоритмы параллельно по шагам.

Если какой-то из алгоритмов заканчивает свою работу после некоторого числа ходов, то ничего страшного, на всех последующих итерациях шаг в нем делаться уже не будет.

Осталось понять, зачем же мы делаем все эти итерации. На каждой итерации, если на очередном шаге какой-то из алгоритмов P_A и P_B выдаёт на выход какое-то число, то и мы тоже выдаём это число на выход.

Ясно, что построенный таким образом алгоритм будет печатать только те числа, которые лежат в $A \cup B$. С другой стороны, все элементы этого множества будут напечатаны. Действительно, если $x \in A \cup B$, то $x \in A$ или $x \in B$. Значит, один из алгоритмов P_A и P_B выдаст на выход число x на каком-то шаге k. Тогда наш алгоритм выдаст x на своей k-й итерации.

Таким образом, построенный алгоритм действительно перечисляет $A \cup B$.

Алгоритм, перечисляющий $A \cap B$ строится похожим образом. Только нам не следует сразу выдавать на выход всё, что выдают алгоритмы P_A и P_B. Вместо этого мы отдельно храним списки элементов, уже выданных P_A и уже выданных P_B (храним в массиве, если под алгоритмами понимаем программы, и записываем на отдельных листочках, если алгоритм выполняется «вручную»). Как только один из алгоритмов P_A и P_B на очередном шаге подаёт что-то на выход, мы проверяем, не было ли это же число раньше выдано другим алгоритмом (просто сравниваем его со всеми числами, ранее выданными другим алгоритмом). Если оно уже было выдано другим алгоритмом, то мы подаём его на выход, иначе нет.

Тогда в выходе нашего алгоритма будут встречаться только те числа, которые лежат в обоих множествах A и B. С другой стороны, всякое такое число будет выдано нашим алгоритмом. Действительно, пусть $x \in A \cap B$. Тогда P_A и P_B выдают x на выход на каких-то шагах, пусть это шаги k и n соответственно. Тогда наш алгоритм выдаст x на итерации $\max(k, n)$. □

Замечание 15.7. В связи с конструкцией в нашем доказательстве может возникнуть такой вопрос. Почему мы решили запускать алгоритмы параллельно по шагам? Казалось бы гораздо естественнее запустить алгоритм P_A, подождать пока он выдаст какое-нибудь число, после этого запустить алгоритм P_B, подождать пока он выдаст своё число, затем снова выполнять алгоритм P_A до появления следующего числа и так далее. Иными словами, почему бы не запускать алгоритмы параллельно, но прерывать их после выдачи очередного числа?

Так действительно можно сделать, если известно, что множества A и B оба являются бесконечными. Если же этого заранее не известно, то может так оказаться, что одно из множеств конечно, но алгоритм, перечисляющий его, работает бесконечно: сначала перечисляет все элементы множества, а потом бесконечно работает и ничего не печатает. Такое поведение допускается определением перечислимости. И тогда у нас может возникнуть проблема, что на очередной итерации мы запустим этот алгоритм, ожидая от него следующее число, алгоритм будет продолжать работать, и мы никогда не перейдём к следующим итерациям. А на следующих итерациях другой алгоритм мог напечатать что-то новое, что в нашей конструкции будет утеряно.

Есть несколько способов преодолеть эту трудность. Одним из них мы воспользовались в нашем доказательстве: если запускать алгоритмы параллельно по шагам, то такой проблемы не возникает, для всякого k каждый алгоритм дойдёт до шага k независимо от того, как ведёт себя другой алгоритм. Другой способ состоит в том, чтобы аккуратнее обработать случай конечных множеств. Подумайте, как именно это сделать, после прочтения раздела 15.5.

Можно заметить, что когда мы обсуждали замкнутость свойства разрешимости относительно операций с множествами, мы рассмотрели операцию взятия дополнения. Когда же мы доказывали аналогичные утверждения для перечислимых множеств, мы о дополнении множеств умолчали. И действительно, если вспомнить доказательство для разрешимых множеств, то неясно, как его переносить на случай перечислимых множеств.

Оказывается, что и само утверждение неверно, существует перечислимое множество $A \subseteq \mathbb{N}$, такое что $\mathbb{N} \setminus A$ не является перечислимым. Мы докажем это позже, см. следствие 15.20.

Когда мы говорили о вычислимых функциях и разрешимых множествах, всё было довольно естественно. Алгоритм вычислял функцию и разрешал множество в понятном и естественном смысле. Но затем мы начали обсуждать перечислимые множества, и здесь уже определение у нас довольно странное и непривычное. Зачем же мы стали изучать такой странный объект? Оказывается, что перечислимые множества очень тесно связаны с вычислимыми функциями. И если мы хотим изучать вычислимые функции, то нам в любом случае придётся изучать и перечислимые множества тоже.

Мы показываем связь между вычислимыми функциями и перечислимыми множествами в следующей теореме.

Теорема 15.11. *Пусть $A \subseteq \mathbb{N}$ — некоторое множество натуральных чисел. Следующие утверждения эквивалентны:*

1. *A перечислимо;*
2. *A является областью значений некоторой вычислимой функции;*
3. *A является областью определения некоторой вычислимой функции;*
4. *полухарактеристическая функция A*

$$\chi^*_A(n) = \begin{cases} 0, & \text{если } n \in A, \\ \text{не определена}, & \text{если } n \notin A. \end{cases}$$

вычислима.

Доказательство. Начнём с простых соотношений между этими утверждениями.

Совсем нетрудно увидеть, что из 4 следует 3. Действительно, областью определения функции $\chi^*_A(n)$ как раз и является множество A. Раз эта функция вычислима, то A является областью определения вычислимой функции.

Докажем, что из 1 следует 4. Действительно, пусть множество A перечисляется алгоритмом P_A. Предъявим алгоритм, вычисляющий функцию χ_A^*. Получив на вход n, этот алгоритм запускает алгоритм P_A. Когда P_A выдаёт что-то на выход, мы сравниваем это число с n. Если оно совпадает с n, то мы останавливаемся и выдаём 0. Если очередное выданное P_A число не совпадает с n, то мы продолжаем работу алгоритма P_A. Если $n \in A$, то на каком-то шаге алгоритм P_A выдаст это число, и наш алгоритм выдаст 0. Если же $n \notin A$, то P_A никогда не выдаст n и наш алгоритм не выдаст никакого результата. Таким образом, наш алгоритм вычисляет в точности функцию χ_A^*.

Покажем теперь, что из 3 следует 2. Пусть A является областью определения вычислимой функции f и пусть P — алгоритм, вычисляющий f. Рассмотрим следующий алгоритм. Получив на вход n, мы запускаем алгоритм P на n. Если P выдаёт какой-то выход, то мы подаём на выход n. Заметим, что на входе $n \in A$ наш алгоритм выдаст n, поскольку n входит в область определения f. Если же $n \notin A$, то наш алгоритм не выдаст никакого выхода на входе n, поскольку выхода не выдаёт и P. Таким образом, наш алгоритм вычисляет некоторую функцию, и область её значений совпадает с A.

Нам осталось доказать, что из 2 следует 1. Тогда мы установим эквивалентность всех четырёх утверждений. Эта часть доказательства самая сложная, здесь нам вновь потребуется запускать несколько алгоритмов параллельно, но в несколько более сложной ситуации.

Пусть A является областью значений функции f, вычисляемой алгоритмом P. Построим алгоритм, перечисляющий A. Наш алгоритм вновь будет последовательно выполнять итерации. Сначала мы запускаем P на входе 0 на один шаг. Затем мы запускаем P на входе 1 на один шаг, после чего делаем второй шаг P на входе 0. Затем мы запускаем P на входе 2 на один шаг, затем делаем второй шаг P на входе 1, затем третий шаг P на входе 0. В общем случае, на итерации $k = 1, 2, 3, \ldots$ мы последовательно запускаем P на входе $k-1$ на один шаг, на входе $k-2$ на второй шаг, и так далее, на входе 0 на k-й шаг.

Как только на каком-то входе на очередном шаге алгоритм P выдаёт какой-то выход, мы передаём его на выход нашего алгоритма (и продолжаем работу).

Ясно, что среди выходов нашего алгоритма встречаются только числа из области значений f (то есть из множества A). С другой стороны, всякий элемент области значений f встретится среди выходов нашего алгоритма. Действительно, пусть $a = f(n)$. Тогда P на входе n выдаёт a на некотором шаге k. Тогда на итерации $n + k$ мы (среди прочего) запустим P на входе n на k шагов и выдадим a на выход. Таким образом, наш алгоритм перечисляет в точности множество A. □

Задача 15.15. Докажите напрямую, что область определения вычислимой функции перечислима.

Таким образом, оказывается, что перечислимые множества — это в точности области определения и области значений вычислимых функций. С другой стороны, мы получили удобное эквивалентное определение перечислимых множеств: множе-

ство перечислимо тогда и только тогда, когда его полухарактеристическая функция вычислима. Поэтому в принципе можно обойтись без понятия перечисляющего алгоритма, заплатив за это наглядностью доказательств.

Ещё два факта, связывающих перечислимость и вычислимость приведём в качестве задач.

Проекцией множества $A \subseteq \mathbb{N} \times \mathbb{N}$ назовём множество $\operatorname{Пр} A = \{x : \exists y (x, y) \in A\}$. Название становится понятным, если нарисовать условную картинку на декартовой плоскости.

Задача 15.16. Докажите, что перечислимые множества натуральных чисел — это в точности проекции разрешимых множеств пар натуральных чисел.

(См. решение в последнем разделе главы.)

Задача 15.17. Докажите, что функция вычислима тогда и только тогда, когда её график перечислим.

(См. решение в последнем разделе главы.)

Выше мы показали, что разрешимые множества являются перечислимыми. Но может быть, верно и обратное? Может быть, мы зря мучаем себя сложным определением и от него можно избавиться? Оказывается, что это не так: существуют перечислимые неразрешимые множества. Мы докажем это позже, см. теорему 15.17.

Однако к свойству перечислимости можно добавить ещё одно естественное свойство, чтобы получить условие, эквивалентное разрешимости.

Теорема 15.12 (теорема Поста)**.** *Множество $A \subseteq \mathbb{N}$ разрешимо тогда и только тогда, когда оба множества A и $\mathbb{N} \setminus A$ перечислимы.*

Доказательство. В одну сторону доказать утверждение совсем просто. Пусть A разрешимо. Тогда по уже доказанной лемме 15.9 множество A перечислимо. С другой стороны, по лемме 15.7 множество $\mathbb{N} \setminus A$ также разрешимо. И вновь по лемме 15.9 множество $\mathbb{N} \setminus A$ перечислимо.

Для доказательства в другую сторону мы используем приём, который у нас уже встречался. Пусть множества A и $\mathbb{N} \setminus A$ перечислимы, и пусть P_A и $P_{\mathbb{N}\setminus A}$ — перечисляющие их алгоритмы. Построим алгоритм, разрешающий A. Получив на вход число n, наш алгоритм запускает алгоритмы P_A и $P_{\mathbb{N}\setminus A}$ параллельно по шагам. На k-й итерации (для $k = 0, 1, 2, \ldots$) алгоритм выполняет k-й шаг алгоритма P_A, а затем k-й шаг алгоритма $P_{\mathbb{N}\setminus A}$. Если какой-то из алгоритмов P_A и $P_{\mathbb{N}\setminus A}$ после очередного шага выдаёт какое-то число на выход, то наш алгоритм сравнивает его с n. Если выданное число совпадает с n, то мы выдаём на выход 1, если число было выдано алгоритмом P_A, и выдаём 0, если число было выдано алгоритмом $P_{\mathbb{N}\setminus A}$. Если же выданное число не совпадает с n мы продолжаем работу.

Почему наш алгоритм работает правильно? Заметим, что всякое число $n \in \mathbb{N}$ лежит в ровно одном из двух множеств A и $\mathbb{N} \setminus A$. Так что для всякого n на какой-то итерации один из алгоритмов выдаст n, причём это будет P_A, если $n \in A$, и это будет $P_{\mathbb{N}\setminus A}$, если $n \notin A$. Так что описанный алгоритм действительно является разрешающим алгоритмом для множества A. □

Приведём ещё одну характеризацию разрешимых множеств через перечисляющие алгоритмы.

Задача 15.18. Докажите, множество $S \subseteq \mathbb{N}$ разрешимо тогда и только тогда, когда существует алгоритм перечисления элементов множества S в возрастающем порядке.

(См. решение в последнем разделе главы.)

15.5 Вычислимость и конечные объекты

Прежде чем двигаться дальше, стоит остановиться на одной особенности, которая существенно отличает наше понятие вычислимости от того, к чему мы привыкли в реальной жизни.

Лемма 15.13. *Если множество $A \subseteq \mathbb{N}$ конечно, то оно разрешимо.*

Это не вполне согласуется со здравым смыслом: если мы хотим на практике решать задачу о принадлежности числа какому-то произвольному множеству X, то нас обычно интересуют небольшие числа, скажем, такие у которых количество цифр не превышает числа атомов в Солнечной Системе. Возьмём пересечение X с подмножеством чисел, у которых длина записи ограничена какой-нибудь, пусть очень большой, константой. Получаем конечное множество, и наша лемма говорит о том, что это конечное множество разрешимо. Как его разрешать, по-прежнему непонятно. Такое положение дел — следствие двух вещей. Во-первых, мы объявили, что нас не интересует время работы разрешающего алгоритма и длина его описания. Важно лишь, чтобы он был конечным и заканчивал работу за конечное время. Во-вторых, если посмотреть внимательно на определение разрешимости, то в нем требуется доказать, что разрешающий алгоритм *существует*. При этом от нас не требуется его предъявлять.

Доказательство. Пусть множество конечно: $A = \{a_1, \ldots, a_k\}$. Рассмотрим следующий алгоритм. Получив на вход n, алгоритм сравнивает его с числом a_1, затем сравнивает n с числом a_2, затем с a_3 и так далее до a_k. Если n совпадает с каким-то из этих чисел, то мы выдаём 1, иначе выдаём 0.

Наш алгоритм хранит числа $a_1, \ldots, a_k$ как константы и просто сравнивает с ними данный ему вход. Таким образом, мы показали, что алгоритм, разрешающий множество A, существует: достаточно взять алгоритм, который сравнивает вход с правильным набором констант. При этом остаётся совершенно не ясным, как построить такой алгоритм. Ведь для него нужен список всех элементов множества A, а откуда его взять? Но от нас и не требовалось объяснять, как построить алгоритм для A. По определению разрешимости достаточно доказать, что такой алгоритм есть, — а это мы сделали. □

Аналогичное утверждение можно доказать и для функций на конечном множестве.

Лемма 15.14. *Пусть функция $f\colon \mathbb{N} \to \mathbb{N}$ имеет конечную область определения. Тогда f вычислима.*

Доказательство. Пусть областью определения f является множество $\{a_1, \ldots, a_k\}$. Обозначим $b_i = f(a_i)$ для всех $i = 1, \ldots, k$. Рассмотрим следующий алгоритм. Получив на вход число n, алгоритм сравнивает его с числом a_1, затем сравнивает n с числом a_2, затем с a_3, и так далее до a_k. Если n совпадает с каким-то числом a_i, то мы выдаём b_i на выход. Если нет — не выдаём ничего.

Здесь алгоритм хранит в качестве констант числа $a_1, \ldots, a_k$, а также числа $b_1, \ldots, b_k$. Видно, что предъявленный алгоритм вычисляет функцию f. □

Задача 15.19. Докажите, что множество таких натуральных чисел n, что в десятичной записи числа π встречается n идущих подряд девяток, разрешимо.

(См. решение в последнем разделе главы.)

Замечание 15.8. Есть гипотеза, что в десятичной записи числа π встречается любая строка, составленная из десятичных цифр. Если эта гипотеза верна, то множество в предыдущей задаче состоит из всех натуральных чисел. В этом случае оно по тривиальным причинам разрешимо.

Однако задачу нужно решить, не предполагая справедливость этой гипотезы.

Из аналогичных соображений иногда удаётся доказать вычислимость функций, определённых на всём множестве натуральных чисел.

Задача 15.20. Докажите, что любая всюду определённая невозрастающая функция из натуральных чисел в натуральные числа вычислима.

(См. решение в последнем разделе главы.)

15.6 Универсальная вычислимая функция

Мы уже анонсировали много утверждений о вычислимых функциях, перечислимых и разрешимых множествах и обещали их доказать. Пришла пора выполнять обещания.

Для того нам потребуется ещё одно фундаментальное свойство алгоритмов, формулировка которого использует следующее определение.

Определение 15.15. Функция $U\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ называется *универсальной вычислимой функцией* для класса вычислимых функций от одной переменной, если

1. U вычислима;
2. для всякой вычислимой функции $f\colon \mathbb{N} \to \mathbb{N}$ существует такое n, что для всякого x верно $f(x) = U(n, x)$.

Здесь и далее мы понимаем равенство (частичных) функций в расширенном смысле: $f(x) = U(n, x)$ означает, что либо обе части равенства определены и равны, либо обе части равенства не определены.

Иными словами, универсальная вычислимая функция U должна быть вычислима сама; кроме того, если фиксировать её первый аргумент n всеми возможными способами ($n = 0, 1, 2, \ldots$), то среди получающихся функций от второго аргумента должны встречаться все возможные вычислимые функции (заметим, что тут не возникает трудности из-за мощностей: множество возможных значений n счётно, но счётно и множество вычислимых функций).

Удобно ввести обозначение $U_n \colon \mathbb{N} \to \mathbb{N}$ для функций, которые получаются фиксированием первого аргумента функции U. То есть для $n \in \mathbb{N}$ по определению полагаем $U_n(x) = U(n, x)$ для всякого $x \in \mathbb{N}$. (Опять-таки, равенство понимается в расширенном смысле, как объяснено выше.)

	0	$\cdots$	x	$\cdots$
0	$U(0,0)$	$\cdots$	$U(0,x)$	$\cdots$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$
n	$U(n,0)$	$\cdots$	$U(n,x)$	$\cdots$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$

Таблица 15.1: универсальная вычислимая функция

Универсальную функцию можно наглядно изобразить в виде таблицы (см таблицу 15.1). Строки таблицы нумеруются значениями первого аргумента функции, а столбцы — значениями второго аргумента. Тогда в ячейке на пересечении строки n и столбца x записывается значение $U(n, x)$ (и пишется «не определено», если это значение не определено). Если посмотреть на отдельную строку с номером n в этой таблице, то в ней будут записаны значения функции U_n. Тогда второе условие в определении универсальной функции означает, что среди строк таблицы встречаются все вычислимые функции от одной переменной.

Замечание 15.9. Универсальная функция имеет два натуральных аргумента. Вычислимые функции от двух натуральных аргументов, как уже говорилось, определяются как вычислимые функции от одного натурального аргумента, который кодирует пару натуральных чисел. Аналогично определяются вычислимые функции и от нескольких натуральных аргументов (подходящая биективная кодировка приведена в примере 15.9). В дальнейшем мы будем говорить о вычислимых функциях на $\mathbb{N}^k$ и о разрешимости подмножеств $\mathbb{N}^k$, не повторяя эту оговорку каждый раз.

Теперь мы готовы сформулировать ещё одно важное свойство алгоритмов.

Свойство 4. *Существует универсальная вычислимая функция U для класса вычислимых функций одной переменной.*

Это свойство уже не столь очевидно, как предыдущие. Его тем не менее легко объяснить ссылкой на интуитивное понимание алгоритмов и языков программирования. Вспомним, что в предыдущей главе мы обсуждали *интерпретатор*: такой алгоритм, который можно запустить на паре (p, i), и он будет вести себя как программа p на входе i. Там мы привели неформальные доводы в пользу существования такого интерпретатора (для каждого формального определения алгоритма существование интерпретатора нужно доказывать, что обычно не столь сложно, сколь утомительно).

Интерпретатор и является алгоритмом вычисления универсальная функция для класса вычислимых функций одной переменной. Сейчас у нас аргументы функций — натуральные числа. Поэтому перескажем как работает интерпретатор и зададим тем самым универсальную вычислимую функцию. Первый аргумент n универсальной функции мы представляем как кодировку натуральным числом программы (или описания алгоритма, что для нас одно и то же), а второй аргумент x как обычное натуральное число. Для определённости будем использовать кодировку, указанную в начале главы в примере 15.1, хотя это и неважно в дальнейших рассуждениях.

Итак, алгоритм-интерпретатор превращает первый свой аргумент в описание программы, а затем исполняет эту программу, выбрав в качестве входа свой второй аргумент. Если исполнение программы закончено, то результат её работы интерпретатор подаёт на выход. Если программа ничего не выдаёт, то и интерпретатор ничего не выдаёт.

Конечно, при преобразовании натурального числа в двоичную строку, может получиться бессмысленный с точки зрения данного языка программирования текст (подайте этот абзац на вход компилятора C, он вам подтвердит, что такое очень даже возможно). Что с таким текстом сделает интерпретатор? Договоримся, что он ничего в таком случае не делает. Поэтому первые аргументы универсальной функции, которые отвечают бессмысленным программам, задают нигде не определённые функции: во всех ячейках строки таблицы 15.1, занумерованной таким аргументом, стоит запись «не определено».

Для нас важнее, что каждой программе отвечает какое-то число и потому в таблице 15.1 найдётся строка, являющая таблицей значений функции, вычисляемой этой программой. Это и означает выполнение свойства универсальности.

Имея в своём распоряжении универсальную функцию, мы можем доказывать отрицательные результаты. Начнём с обещанного результата, противоположного неверному утверждению, которое мы «доказывали» в начале главы.

Теорема 15.16. *Существует вычислимая функция $f\colon \mathbb{N} \to \mathbb{N}$, не имеющая всюду определённого вычислимого продолжения.*

Доказательство. Пусть $U\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ — универсальная вычислимая функция для класса вычислимых функций одной переменной.

Рассмотрим её «диагональную» функцию $d(n) = U(n, n)$. Значения этой диагональной функции записаны на диагонали таблицы 15.1. Функция d вычислима, поскольку вычислима функция U.

Положим $f(n) = d(n) + 1$. Эта функция также вычислима как композиция двух вычислимых функций (функции d и прибавления единицы).

Мы хотим доказать, что у f нет всюду определённого вычислимого продолжения. Предположим противное. Пусть $g\colon \mathbb{N} \to \mathbb{N}$ является всюду определённым вычислимым продолжением f.

Поскольку g вычислима, то она встречается среди строк таблицы 15.1, то есть существует k, для которого $g = U_k$. Посмотрим на пересечение строки k с диагональю, то есть на клетку (k, k) таблицы. С одной стороны, там написано значение $d(k)$ (поскольку значения d написаны на диагонали). С другой стороны, там написано значение $g(k)$ (поскольку значения g записаны в k-й строке). Значит $d(k) = g(k)$ (в расширенном смысле, обе части равенства могут оказаться одновременно неопределёнными).

Рассмотрим два случая. Пусть $d(k)$ не определено. Тогда мы сразу получаем противоречие, ведь g — всюду определённая функция, а значит, $g(k)$ определено. Пусть теперь $d(k)$ определено. Тогда $f(k) = d(k) + 1$ и также определено. Поскольку g является продолжением f, то это означает, что $g(k) = f(k)$. В итоге получаем, что $g(k) = f(k) = d(k) + 1$, что также противоречит тому, что $g(k) = d(k)$.

В обоих случаях мы пришли к противоречию, а значит, функция f не имеет всюду определённого вычислимого продолжения. □

Контрольный вопрос 15.21. Существует ли всюду определённое вычислимое продолжение у самой диагональной функции $d(n) = U(n, n)$?

Доказательство теоремы 15.16 очень похоже на доказательство несчётности множества бесконечных последовательностей из нулей и единиц и на доказательство неразрешимости проблемы остановки в прошлой главе.

Точно так же, как раньше мы располагали последовательности, теперь мы располагаем вычислимые функции в виде таблицы. Затем мы рассматриваем диагональ этой таблицы (функция d). Затем мы изменяем последовательность на диагонали во всех точках. Мы сделали это в два действия: сначала поменяли значения там, где d определена (получилась функция f), затем, пользуясь предположением, поменяли значения во всех точках, где d не определена (получилась функция g). Затем рассмотрели строку, в которой располагается получившаяся последовательность значений функции и пришли к противоречию на пересечении этой строки и диагонали.

Теперь мы можем доказать, что существует перечислимое неразрешимое множество.

Теорема 15.17. *Существует перечислимое неразрешимое множество $K \subseteq \mathbb{N}$.*

Доказательство. Рассмотрим функцию d из доказательства теоремы 15.16 и возьмём в качестве K её область определения. (Это же множество является областью

определения функции f из доказательства той же теоремы.) Множество K перечислимо как область определения вычислимой функции.

Предположим, что множество K разрешимо. Рассмотрим функцию

$$g(n) = \begin{cases} f(n), & \text{если } n \in K, \\ 0, & \text{если } n \notin K. \end{cases}$$

Функция g является всюду определённым продолжением f. С другой стороны, легко понять, что функция g является вычислимой. Действительно, g вычисляется следующим алгоритмом. По входу n сначала (с помощью алгоритма, разрешающего K) проверяем, лежит ли n в K. Если не лежит, то сразу выдаём 0. Если лежит, то поскольку K является областью определения f, то $f(n)$ определено. Запустим алгоритм для вычисления f на входе n и выдадим результат.

Таким образом, мы построили всюду определённое вычислимое продолжение f, которого не существует по теореме 15.16. Противоречие, а значит, K неразрешимо. □

Другой вариант перечислимого неразрешимого множества возникает из «проблемы остановки». Повторим формулировку из главы 14 в наших терминах. Пусть $U(n, x)$ — универсальная вычислимая функция.

Определение 15.18. Рассмотрим множество $\mathsf{Halt} \subseteq \mathbb{N} \times \mathbb{N}$, состоящее из таких пар (n, x), что $U(n, x)$ определено. Проблема остановки состоит в выяснении того, принадлежит ли данная пара множеству Halt.

Неформально, в проблеме остановки требуется по данной программе и данному входу определить, выдаст ли программа результат на этом входе, то есть закончит ли она успешно свою работу[4].

Теорема 15.19. *Множество* Halt *неразрешимо.*

Доказательство. Предположим, что множество Halt разрешимо, то есть существует алгоритм P, который на данной паре (n, x) выдаёт 1, если $(n, x) \in \mathsf{Halt}$, и выдаёт 0 иначе. Но тогда множество K из доказательства теоремы 15.17 также разрешимо. Действительно, его можно было бы разрешать следующим алгоритмом. Получив на вход n запускаем алгоритм P на входе (n, n) и выдаём результат его работы на выход. Этот алгоритм разрешает множество K, поскольку Halt является областью определения функции $U(n, x)$, а K является областью определения функции $d(n) = U(n, n)$. Но по теореме 15.17 множество K неразрешимо, противоречие. □

Заметим, что множество Halt перечислимо, как область определения вычислимой функции. Вспомнив про теорему Поста, приходим к такому следствию из неразрешимости множества Halt.

Следствие 15.20. *Дополнение к множеству* Halt *неперечислимо.*

[4]Обычно в проблеме остановки спрашивается, остановится ли программа. Но легко понять, что эта постановка эквивалентна нашей.

15.7 Главная универсальная функция

Универсальных вычислимых функций много, уж никак не меньше, чем разных языков программирования. На самом деле, их ещё больше, причём некоторые заметно отличаются от того примера, который мы разбирали выше — интерпретатора «обычного» языка программирования.

На самом деле мы пока выделили не все принципиально важные свойства таких интерпретаторов. Начнём с неформального объяснения. Программы на «обычных» языках программирования допускают некоторые алгоритмические преобразования, которым отвечают преобразования вычисляемых ими функций.

Например, если в программе есть несколько аргументов (входных данных), то легко преобразовать эту программу, заменив один из аргументов константой, как показано на примере ниже.

Было

```
function f(a,b: integer);
....
....
```

Стало

```
function f10(b: integer);
integer a;
a<- 10;
....
....
```

Из программы, вычисляющей функцию $f(a, b)$ с двумя аргументами, мы сделали программу, вычисляющую функцию $f_{10}(b)$, которая в точности равна $f(10, b)$ для всех значений второго аргумента.

Что принципиально важно, такое преобразование можно реализовать алгоритмически, то есть написать программу, которая получает на вход два параметра: текст программы, вычисляющей функцию от двух аргументов $f(a, b)$ и значение первого аргумента n. Результатом работы такого преобразователя является текст программы, вычисляющей функцию $g(b)$ от одного аргумента, которая получается подстановкой вместо первого аргумента функции f значения n, то есть для всех b выполняется равенство

$$g(b) = f(n, b)$$

в расширенном смысле (обе части равенства определены на одних и тех же значениях b).

Возможность такого преобразования для языков наподобие C или Pascal представляется вполне очевидной. В приведённом выше примере нужно вставить в текст программы две константные строки сразу после первой строки. Синтаксис языков может заметно различаться, но обычно сразу понятно как выполнить замену аргумента программы на константу.

На первый взгляд кажется, что это какое-то незначительное свойство программ, не имеющее большого смысла.

Как ни странно, это не так. Свойство подстановки констант вместо аргументов принципиально важно и позволяет в очень большой общности доказывать многие результаты о неразрешимости. Да и о разрешимости тоже. Например, выяснится, что это свойство, точнее его аккуратный формальный аналог, позволяет доказать, что существует программа, печатающая свой собственный текст.[5]

Как сформулировать это свойство замены аргумента на константу в терминах вычислимых функций? В нашем формализме универсальная вычислимая функция $U(n,x)$ соответствует «языку программирования». Значение U (включая значение «не определено») на паре аргументов n и x совпадает с результатом работы программы n на входе x. (Напомним ещё раз, что n — это число, которое кодирует текст программы, для простоты рассказа всюду далее мы отождествляем программы и их коды.)

Программа с двумя аргументами задаёт вычислимую функцию $V\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ от этих аргументов. Мы хотим автоматически по первому аргументу n этой функции получать программы, которые вычисляют функцию от одного аргумента $v_n(y) = V(n,y)$. Слово «автоматически» означает, что должна быть программа, реализующая это преобразование. То есть должна существовать вычислимая функция $s\colon \mathbb{N}\to\mathbb{N}$, которая по числу n находит программу $s(n)$, вычисляющую $v_n(y)$. Это по определению универсальной функции означает, что строка с номером $s(n)$ таблицы универсальной функции является таблицей значений функции $v_n(y)$: $U(s(n),y) = v_n(y) = V(n,y)$ (как всегда, в расширенном смысле, области определения должны также совпадать).

Получаем такое определение.

Определение 15.21. Универсальная вычислимая функция $U\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ для класса вычислимых функций от одной переменной называется *главной* (или гёделевой), если для любой вычислимой функции $V\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ существует такая всюду определённая вычислимая функция $s\colon \mathbb{N}\to\mathbb{N}$, что для всякого $n\in\mathbb{N}$ и для всякого $x\in\mathbb{N}$ верно $U(s(n),x) = V(n,x)$, или, другими словами, для всякого $n\in\mathbb{N}$ верно $U_{s(n)} = V_n$.

Определение свойства «главности» получилось сложным. Его смысл можно объяснить ещё одним, более общим, способом, чем автоматизация замены аргумента на константу. Будем думать о вычислимой функции V от двух аргументов как о другом языке программирования. Этот язык может даже не быть универсальным, то есть не способным вычислить все вычислимые функции. Но первый аргумент n функции V мы будем представлять как программу, которую V запускает на своём втором аргументе x. Тогда свойство главности U говорит, что для всякого другого языка программирования V существует вычислимый *компилятор* s этого языка в U. Этот

[5] Для каждого конкретного языка программирования написание такой программы — несложное упражнение для начинающих. Однако мы приведём единое доказательство для **всех** языков программирования, удовлетворяющих указанному выше свойству.

компилятор получает на вход программу n в языке V и выдаёт в результате своей работы программу $s(n)$ в языке U, которая вычисляет ту же самую функцию: $U_{s(n)} = V_n$.

При этом существенно, что компилятор с V на U всюду определён: он любую программу преобразует в эквивалентную.

Хотя определение главности довольно трудное, но у него есть простое и понятное применение. Если смотреть на U как на язык программирования, то свойство главности означает, что мы можем по программе n в этом языке вычислять номера несколько модифицированных программ. Другими словами, к программам в языке U легко применять небольшие изменения. Разберём несколько простых примеров.

Пример 15.22. Пусть $U\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ — главная универсальная функция. Тогда существует всюду определённая вычислимая функция $s\colon \mathbb{N} \to \mathbb{N}$, которая по всякой программе n строит программу $s(n)$, выдающую ответ на 1 больше, чем программа n. Другими словами, $U_{s(n)}(x) = U_n(x) + 1$. Действительно, положим $V(n, x) = U(n, x) + 1$. Тогда функция V вычислима, и в силу того, что U главная, существует такая всюду определённая вычислимая функция $s\colon \mathbb{N} \to \mathbb{N}$, что $U_{s(n)}(x) = V_n(x) = U_n(x) + 1$ для всех n и x.

Понятно, что точно так же мы можем применять и другие преобразования к выходу программы (умножать на 2, возводить в квадрат и т.д.). Достаточно лишь немного иначе определить V.

Также мы можем применять изменения и ко входу программы. Например, можно доказать, что существует всюду определённая вычислимая функция $s\colon \mathbb{N} \to \mathbb{N}$, которая по всякой программе n строит программу $s(n)$, выдающую на входе x тот же ответ, что n выдаёт на входе $x + 1$, то есть, $U_{s(n)}(x) = U_n(x + 1)$. Для этого обозначим $V(n, x) = U(n, x + 1)$. Вновь функция V вычислима; поскольку U главная, существует такая всюду определённая вычислимая функция $s\colon \mathbb{N} \to \mathbb{N}$, что $U_{s(n)}(x) = V_n(x) = U_n(x + 1)$ для всех n и всех x.

Разберём более сложный пример.

Пример 15.23. Пусть $U\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ — главная универсальная функция. Тогда существует всюду определённая вычислимая функция $c\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, которая по всякой паре программ p и q строит программу $c(p, q)$, вычисляющую композицию программ p и q. Другими словами, $U(c(p, q), x) = U(q, U(p, x))$.

Действительно, рассмотрим функцию[6] $V([p, q], x) = U(q, U(p, x))$, где $[p, q]$ — это биекция $\mathrm{code}_2\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ в сокращённых обозначениях, введённых в примере 15.3. Эта функция составлена из вычислимых и потому вычислима. Значит, в силу главности U, существует такая всюду определённая вычислимая s, что $U(s(n), x) = V(n, x)$. В итоге получаем $U(q, U(p, x)) = V([p, q], x) = U(s([p, q]), x)$. Определяя $c(p, q) = s[p, q]$, мы получаем требуемое равенство.

[6] Мы здесь используем не вполне аккуратную запись. Раз уж мы определяем функцию V, следовало бы написать $V(n, x) = U(r(n), U(l(n), x))$, где $l(n)$ и $r(n)$ — компоненты пары $\mathrm{code}_2^{-1}(n)$, определённые в примере 15.5. Но поскольку $\mathrm{code}_2(\cdot, \cdot) = [\cdot, \cdot]$ — биекция, то по существу выбор записи — это вопрос обозначений (по сути мы ввели обозначения $p = l(n)$ и $q = r(n)$ и сделали замену переменных). Мы будем использовать такие обозначения, в которых лучше видно, что происходит.

Мы посмотрели, как можно использовать свойство главности. Но почему главные функции существуют? Оказывается, требовать отдельного свойства существования главной универсальной функции не нужно: достаточно уже постулированных свойств.

Теорема 15.22. *Существует главная универсальная функция* $U\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$.

Сначала мы неформально объясним, почему теорема верна.

Мы предполагаем, что какая-то универсальная функция U существует (свойство 4). Если она уже главная, то и хорошо. Пусть U не главная. Нужно как-то её подправить, чтобы удовлетворить этому свойству. То есть, наша цель — построить улучшенный язык программирования U', допускающий автоматическую замену аргументов на константы. Для этого мы поступим экстравагантным с точки зрения реального программирования[7] способом: будем считать программой в новом языке пару (программа в языке U, возможная константа на замену). Извлечение из этой пары требуемой константы вполне можно поручить компьютеру.

Доказательство теоремы 15.22. Доказательство будет состоять из двух частей.

Сначала мы докажем, что существует универсальная вычислимая функция

$$T\colon \mathbb{N}\ \times\mathbb{N}\times\mathbb{N}\to\mathbb{N}$$

для класса вычислимых функций от двух переменных. Это значит, что T сама вычислима, и для всякой функции $V\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ существует такое $n\in\mathbb{N}$, что $T(n,x,y)=V(x,y)$ для всех x,y.

Для этого рассмотрим универсальную функцию $U\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ и определим $T(n,x,y)=U(n,[x,y])$, тут квадратные скобки означают биективную кодировку пар натуральных чисел натуральными числами. Функция T составлена из вычислимых функций и потому вычислима по свойству 3. Пусть теперь $V\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ — произвольная вычислимая функция. Рассмотрим функцию $f([x,y])=V(x,y)$ (или, что то же самое, $f(z)=V(l(z),r(z))$, где $l(n)$ и $r(n)$ — компоненты пары $\mathrm{code}_2^{-1}(n)$, как объясняется в примере 15.5[8]). Функция f является вычислимой функцией одной переменной. Поскольку U является универсальной, существует такое n, что $U(n,z)=f(z)$ для всякого z. Суммарно получаем

$$V(x,y)=f([x,y])=U(n,[x,y])=T(n,x,y),$$

что и требовалось. Следовательно, T действительно является универсальной вычислимой функций для класса вычислимых функций двух переменных.

Теперь мы готовы построить главную универсальную вычислимую функцию. Положим $U'([n,x],y)=T(n,x,y)$ (или что то же самое, $U'(z,y)=T(l(z),r(z),y)$).

[7] Впрочем, как мы уже говорили, обычные языки программирования и так соответствуют главным функциям. Мы сейчас преодолеваем трудность, возникающую из логики математического исследования вычислимости.

[8] Напомним, что $\mathrm{code}_2(x,y)=[x,y]$ по нашему соглашению, это два разных обозначения одной и той же функции.

Функция U' составлена из вычислимых и потому вычислима. Покажем, что для U' выполняется свойство главности. Пусть $V\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ — произвольная вычислимая функция. В силу универсальности T существует такое k, что $T(k,n,x)=V(n,x)$. Это значит, что $V(n,x)=T(k,n,x)=U'([k,n],x)$. Полагая $s(n)=[k,n]$, получаем то, что и требовалось.

Но почему построенная функция U' будет универсальной? Докажем, что свойство универсальности легко вытекает из свойства главности. Действительно, для произвольной вычислимой функции $f\colon\mathbb{N}\to\mathbb{N}$ рассмотрим вычислимую функцию $V(n,x)=f(x)$. В силу главности U' получаем, что существует всюду определённая вычислимая s, такая что $U(s(n),x)=V(n,x)=f(x)$ для всех n. В частности, $U'_{s(0)}=f$. □

15.8 Теорема Райса – Успенского

Теперь мы можем воспользоваться главными универсальными функциями для получения новых отрицательных результатов.

Теорема 15.23. *Пусть $U\colon\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ — главная универсальная функция. Тогда множество*

$$N_\varnothing=\{n\mid U_n \text{ является нигде не определенной функцией}\}$$

неразрешимо.

На языке нашей программистской интуиции, эта теорема означает неразрешимость следующей задачи: по тексту программы определить, выдаёт ли эта программа выход хоть на каком-нибудь входе.

Доказательство. Пусть $K\subseteq\mathbb{N}$ — перечислимое неразрешимое множество. Такое множество существует по теореме 15.17. Рассмотрим функцию

$$V(n,x)=\begin{cases}0, & \text{если } n\in K,\\ \text{не определена}, & \text{если } n\notin K.\end{cases}$$

Функция V вычислима. Действительно, получив на вход n и x, можно запустить алгоритм, перечисляющий K. Когда этот алгоритм выдаёт какое-то число на выход, мы сравниваем его с n. Если числа совпадают, выдаём 0 и заканчиваем работу. Иначе продолжаем перечисление K. Если n лежит в K, то на каком-то шаге оно будет выдано алгоритмом, перечисляющим K, и мы выдадим 0. Если же n не лежит в K, то мы не выдадим никакого ответа, что и требуется.

Поскольку V вычислима, а U — главная, то существует такая вычислимая всюду определённая функция $s\colon\mathbb{N}\to\mathbb{N}$, что $U_{s(n)}=V_n$.

Заметим теперь, что функция V_n (а значит, и $U_{s(n)}$) тождественно равна 0, если $n\in K$, и является нигде не определённой, если $n\notin K$. Таким образом, $s(n)\notin N_\varnothing$ тогда и только тогда, когда $n\in K$.

Если бы множество $N_\varnothing$ было разрешимо, то мы бы могли построить алгоритм, разрешающий K: по данному n сначала вычисляем $s(n)$ (вычисление s всегда заканчивается, поскольку это всюду определённая вычислимая функция), а затем запускаем алгоритм, проверяющий, принадлежит ли $s(n)$ множеству $N_\varnothing$. Если алгоритм выдаёт 0, то мы выдаём 1, и наоборот.

Но поскольку множество K неразрешимо, то мы пришли к противоречию, а значит неразрешимо и множество $N_\varnothing$. □

Доказанное только что утверждение можно существенно обобщить. Обозначим через $\mathcal{F}$ класс всех вычислимых функций на натуральных числах. *Свойством вычислимых функций* будем называть произвольное подмножество $\mathcal{A} \subseteq \mathcal{F}$. Свойство вычислимых функций $\mathcal{A}$ называется нетривиальным, если $\mathcal{A} \neq \varnothing$ и $\mathcal{A} \neq \mathcal{F}$.

Теорема 15.24 (Теорема Райса – Успенского)**.** *Пусть $U\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ — главная универсальная функция. Пусть $\mathcal{A}$ — нетривиальное свойство вычислимых функций. Тогда множество*

$$N = \{n \mid U_n \in \mathcal{A}\}$$

не разрешимо.

В нашей программистской интуиции эта теорема означает, что по данному тексту программы на каком-либо языке программирования нельзя алгоритмически распознать никакое нетривиальное свойство функции, вычисляемой данной программой. Заметим, что когда мы ограничились нетривиальными свойствами, мы исключили действительно только самые тривиальные случаи: когда все функции обладают свойством, и когда ни одна функция не обладает этим свойством.

Доказательство. Рассмотрим нигде не определённую функцию $f_\varnothing$ на натуральных числах. Эта функция либо лежит в $\mathcal{A}$, либо лежит в $\mathcal{F}\setminus\mathcal{A}$. Пусть для определённости она лежит в $\mathcal{A}$. Поскольку $\mathcal{A}$ — нетривиальное свойство, существует функция $g \in \mathcal{F} \setminus \mathcal{A}$.

Пусть $K \subseteq \mathbb{N}$ перечислимое неразрешимое множество. Такое множество существует по теореме 15.17. Рассмотрим функцию

$$V(n,x) = \begin{cases} g(x), & \text{если } n \in K, \\ \text{не определена}, & \text{если } n \notin K. \end{cases}$$

Докажем, что функция V вычислима. Действительно, получив на вход n и x, можно запустить алгоритм, перечисляющий K. Когда этот алгоритм выдаёт какое-то число на выход, мы сравниваем это число с n. Если числа совпадают, мы запускаем алгоритм для вычисления g на x, и если он выдаёт какой-то результат, подаём его на выход. Если же числа не совпадают, продолжаем перечисление K. Если n лежит в K, то на каком-то шаге оно будет выдано алгоритмом, перечисляющим K, и мы дальше будем вычислять функцию g на входе x. Если же n не лежит в K, то мы не выдадим никакого ответа, что и требуется.

Поскольку V вычислима, а U главная, то существует такая вычислимая всюду определённая функция $s\colon \mathbb{N} \to \mathbb{N}$, что $U_{s(n)} = V_n$.

Заметим теперь, что функция V_n (а значит, и $U_{s(n)}$) совпадает с g, если $n \in K$, и совпадает с $f_\varnothing$, если $n \notin K$. Таким образом, $s(n) \notin N$ тогда и только тогда, когда $n \in K$.

Если бы множество N было разрешимо, то мы бы могли построить алгоритм, разрешающий K: по данному n сначала вычисляем $s(n)$, а затем запускаем алгоритм, проверяющий, принадлежит ли $s(n)$ множеству N. Если алгоритм выдаёт 0, то мы выдаём 1 и наоборот.

Но поскольку множество K неразрешимо, то мы пришли к противоречию, а значит, неразрешимо и множество N. □

Замечание 15.10. Важно не запутаться в формулировке теоремы Райса – Успенского. Стоит обратить внимание на то, что в формулировке теоремы обсуждаются два разных объекта. С одной стороны, мы рассматриваем программы в каком-то языке программирования. С другой стороны, мы рассматриваем функции, которые этими программами вычисляются. И это не одно и то же. Например, несколько разных программ вполне могут вычислять одну и ту же функцию. И вся соль теоремы состоит в том, что по программе трудно понять, что за функцию она вычисляет. Теорема утверждает, что никакое свойство *функции* алгоритмически не распознаётся по вычисляющей её *программе*.

Чуть раньше мы определили понятие главной универсальной функции, объяснили, что это естественное свойство и даже доказали, что главные универсальные функции существуют. Но, может, быть ситуация обратная и все универсальные функции на самом деле главные? Бывают ли вообще не главные универсальные функции? Оказывается бывают, и мы можем доказать это с помощью теоремы Райса – Успенского.

Теорема 15.25. *Существует неглавная универсальная функция для класса вычислимых функций одной переменной.*

Доказательство. Рассмотрим какую-нибудь универсальную функцию

$$U\colon \mathbb{N} \times \mathbb{N} \to \mathbb{N}.$$

Введём обозначение

$$D = \{n \mid U_n \text{ определена хотя бы при одном значении аргумента}\}.$$

По теореме Райса-Успенского множество D не является разрешимым. Но при этом оно перечислимо. Действительно, покажем, что полухарактеристическая функция χ^*_D вычислима. При данном на вход n запустим вычисление функции U_n на всех возможных входах параллельно по шагам. Это можно сделать так же, как в доказательстве теоремы 15.11: вычисление проходит итерацией по $k = 1, 2, 3, \ldots$. На итерации k мы вычисляем U_n на входе $k - 1$ на один шаг, на входе $k - 2$ на два

шага, и так далее, на входе 0 на k шагов. Если на какой-то итерации хотя бы на одном входе алгоритм, вычисляющий U_n, выдал что-то на выход, то наш алгоритм выдаёт на выход 0 и заканчивает работу. Если функция U_n определена хотя бы на одном значении аргумента, то на какой-то итерации мы закончим вычисление на этом входе и выдадим 0. Если же U_n является нигде не определённой функцией, то мы никогда не выдадим ничего на выход.

Таким образом, множество D перечислимо. Зафиксируем некоторый алгоритм, перечисляющий D, и пусть $d_1, d_2, d_3 \ldots$ — последовательность элементов D, которую выдаёт этот алгоритм.

Определим функцию

$$U'(n,x) = \begin{cases} \text{не определено}, & \text{если } n = 0, \\ U(d_n, x), & \text{если } n > 0. \end{cases}$$

Во-первых, U' вычислима. Действительно, алгоритм, вычисляющий U', получает на вход n и x, и если $n = 0$, сразу зацикливается и не выдаёт ничего на выход. Если же $n > 0$, то наш алгоритм сначала запускает алгоритм, перечисляющий множество D до тех пор, пока не будет выдан n-й элемент d_n в выходе этого алгоритма. Затем наш алгоритм запускает алгоритм для вычисления U на входе d_n и x.

Далее, U' является универсальной. Действительно, пусть $f\colon \mathbb{N} \to \mathbb{N}$ — произвольная вычислимая функция. Если f является нигде не определённой, то $f = U'_0$. Если же f определена хотя бы в одной точке, то рассмотрим такое k, что $f = U_k$ (такое k существует в силу универсальности U). Заметим, что k входит в D, так что существует n, для которого $k = d_n$. В итоге получаем, что $f = U_k = U_{d_n} = U'_n$. Таким образом, в любом случае для f существует n, при котором $f = U'_n$.

Наконец, осталось показать, что функция U' не является главной. Для этого рассмотрим множество

$$N'_\varnothing = \{n \mid U'_n \text{ является нигде не определенной функцией}\}.$$

По построению функции U' мы знаем, что $N'_\varnothing = \{0\}$. Это множество конечно, а значит по лемме 15.13 оно разрешимо. Но по теореме 15.23 для главных универсальных функций это множество неразрешимо. Следовательно, построенная универсальная функция не является главной. □

В этой теореме мы построили универсальную функцию, которая не удовлетворяет свойству главности, про которое мы объясняли, что оно выполняется для «разумных» языков программирования. Таким образом, мы построили «неразумный» язык программирования. И действительно, получившийся язык программирования довольно странный. Например, почти любая программа в нем будет осмысленной. Если набрать случайный набор символов и запустить его как программу, то на каком-то входе будет выдан какой-то результат.

С практической точки зрения этот язык бесполезен: чтобы написать на нём программу, вычисляющую где-то определённую функцию, нужно вначале написать программу на «обычном» языке программирования U, а затем преобразовать её с помощью процедуры перечисления останавливающихся хотя бы однажды программ.

Причём, если мы ошиблись при составлении программы и получили программу, которая никогда не останавливается, это преобразование никогда не даст ответа.

15.9 Теорема о неподвижной точке

Теперь с помощью свойства главности универсальных функций мы докажем второй важный результат: так называемую теорему о неподвижной точке или теорему о рекурсии. Из него мы легко получим в общем виде решение такой классической задачи по программированию: написать программу, печатающую свой собственный текст (см. ниже теорему 15.29 на с. 403).

Теорема 15.26. *Пусть $U\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ — главная универсальная функция. Тогда для всякой всюду определённой вычислимой функции $h\colon \mathbb{N}\to\mathbb{N}$ существует $n\in\mathbb{N}$, при котором $U_n = U_{h(n)}$.*

В нашей программистской интуиции эта теорема утверждает, что для любого вычислимого преобразования программ в каком-либо языке программирования существует программа, которая до и после преобразования работает одинаково.

Доказательство. В доказательстве этой теоремы мы используем по существу ту же конструкцию, что и в доказательстве теоремы 15.16 о том, что существует вычислимая функция без всюду определённого вычислимого продолжения. Но нам потребуется немного обобщить эту конструкцию.

Сформулируем обобщённый вариант конструкции в виде леммы.

Лемма 15.27. *Пусть $\equiv$ — некоторое отношение эквивалентности на $\mathbb{N}$. Тогда следующие два утверждения не могут выполняться одновременно*:

1. *Для любой вычислимой $f\colon \mathbb{N}\to\mathbb{N}$ существует всюду определённое вычислимое $\equiv$-продолжение (то есть такая всюду определённая функция $g\colon \mathbb{N}\to\mathbb{N}$, что для всякого x из области определения f верно $f(x)\equiv g(x)$).*
2. *Существует всюду определённая вычислимая функция $h\colon \mathbb{N}\to\mathbb{N}$, не имеющая $\equiv$-неподвижной точки (это значит, что $h(n)\not\equiv n$ для всех n).*

Доказательство леммы. Доказательство по существу повторяет доказательство теоремы 15.16 (и здесь нам не требуется главность U). Предположим, что для некоторого отношения эквивалентности $\equiv$ оба утверждения верны. Рассмотрим функцию $d(n) = U(n,n)$. По утверждению 1 у функции d есть всюду определённое вычислимое $\equiv$-продолжение f, то есть $d(n)\equiv f(n)$ для всех n из области определения d. Рассмотрим функцию $h(f(n))$. Это всюду определённая вычислимая функция. Поскольку U является универсальной, то существует такое k, что $h(f(n)) = U(k,n)$ для всякого n. Рассмотрим теперь $n=k$. Тогда $h(f(k)) = U(k,k) = d(k)$. Если $d(k)$ не определено, то мы сразу получаем противоречие, поскольку $h(f(k))$ определено (композиция двух всюду определённых функций). Если же $d(k)$ определено, то $d(k)\equiv f(k)\not\equiv h(f(k))$, что также противоречит предыдущему равенству. □

Если в этой лемме в качестве отношения эквивалентности взять обычное равенство, то второе утверждение будет верно: достаточно взять функцию $h(n) = n + 1$. Значит первое утверждение неверно, и мы передоказали теорему 15.16.

Рассмотрим теперь следующее отношение эквивалентности: $k \equiv n$ тогда и только тогда, когда $U_k = U_n$. Легко убедиться, что это отношение действительно является отношением эквивалентности.

Для этого отношения, напротив, верно первое утверждение. Действительно, пусть $f\colon \mathbb{N} \to \mathbb{N}$ — вычислимая функция. Рассмотрим функцию $V(n, x) = U(f(n), x)$. Функция V составлена из вычислимых и потому вычислима. Следовательно, в силу главности U существует такая всюду определённая вычислимая функция $s\colon \mathbb{N} \to \mathbb{N}$, что $U_{s(n)} = V_n$. Поскольку для всякого n из области определения f верно $V_n = U_{f(n)}$, мы получаем, что s является всюду определённым вычислимым $\equiv$-продолжением f.

Контрольный вопрос 15.24. Функция $s(n)$ всюду определена. Пусть $f(k)$ не определено. Найдите $U_{s(k)}$.

Раз первое утверждение верно для нашего отношения эквивалентности, то второе должно быть ложно. Следовательно, для всякой всюду определённой $h\colon \mathbb{N} \to \mathbb{N}$ существует n, такое что $U_{h(n)} = U_n$. А это как раз то, что требовалось доказать. □

Полезно проследить, как выглядит неподвижная точка, существование которой мы только что доказали. Для этого нужно развернуть доказательство теоремы в другом порядке. Пусть нам дана всюду определённая вычислимая функция $h\colon \mathbb{N} \to \mathbb{N}$. Мы начинаем с диагональной функции d и рассматриваем функцию $V(n, x) = U(d(n), x) = U(U(n, n), x)$ (все действия у нас проходят внутри первого аргумента универсальной функции, так уж у нас определено наше отношение эквивалентности, и такая уж неподвижная точка нас интересует). В силу главности U существует $s\colon \mathbb{N} \to \mathbb{N}$, для которой

$$U(s(n), x) = V(n, x) = U(d(n), x) = U(U(n, n), x).$$

Затем мы применяем к s данную нам функцию h и, таким образом, рассматриваем функцию $U(h(s(n)), x)$. Дальше мы пользуемся универсальностью U и говорим, что существует k, для которого $U(k, n) = h(s(n))$ при всех n. Таким образом,

$$U(h(s(n)), x) = U(U(k, n), x)$$

при этом k и при любых n и x. Осталось лишь положить $n = k$ и объединить две цепочки равенств. Получаем

$$U(h(s(k)), x) = U(U(k, k), x) = U(d(k), x) = V(k, x) = U(s(k), x).$$

Сравнивая левую и правую часть цепочки, получаем $U_{h(s(k))} = U_{s(k)}$. И $s(k)$ является неподвижной точкой функции h. Полезно также проследить, что из себя представляют s и k. Оказывается, что s вовсе не зависит от h. Это функция, внутри которой спрятана диагональ нашей универсальной функции. А вот k уже связана с h — это номер программы, вычисляющей функцию $h \circ s$.

У теоремы о неподвижной точке есть альтернативная форма, пользоваться которой иногда бывает удобнее.

Следствие 15.28. *Пусть $U\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ — главная универсальная функция. Тогда для всякой вычислимой функции $V\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ существует n, при котором $U_n = V_n$.*

В нашей программистской интерпретации это следствие утверждает, что если U обладает свойством главности, то для любого другого языка программирования V (не обязательно даже универсального) есть программа, которая в обоих языках работает одинаково.[9]

Доказательство. В силу главности U существует всюду определённая такая вычислимая функция $s\colon \mathbb{N}\to\mathbb{N}$, что $V_n = U_{s(n)}$ для любого n. По теореме о неподвижной точке существует n, при котором $U_{s(n)} = U_n$. В итоге получаем $V_n = U_{s(n)} = U_n$, что и требовалось. □

Наконец, докажем, что существует программа, печатающая свой собственный текст.

Теорема 15.29. *Пусть $U\colon \mathbb{N}\times\mathbb{N}\to\mathbb{N}$ — главная универсальная функция. Тогда существует такое $n\in\mathbb{N}$, что $U(n,x) = n$ для всякого x.*

Доказательство. Рассмотрим вычислимую функцию $V(n,x) = n$. По предыдущему следствию существует такое n, что $U(n,x) = V(n,x) = n$ для всех x, что и требовалось доказать. □

Ещё несколько задач, в которых применяется теорема о неподвижной точке. В условиях этих задач предполагается, что $U(n,x)$ — главная универсальная функция.

Задача 15.25. Докажите, что для некоторого m функции $U(m,x)$ и $U(m+1,x)$ совпадают.

(См. решение в последнем разделе главы.)

Задача 15.26. Докажите, что каждая всюду определённая функция $s(n)$ имеет бесконечно много неподвижных точек: $\{m : U_m = U_{s(m)}\} = \infty$.

(См. решение в последнем разделе главы.)

Задача 15.27. Докажите, что каждая вычислимая функция имеет бесконечно много номеров в главной нумерации.

(См. решение в последнем разделе главы.)

[9] В нашем формализме программа — это число. Но, как уже объяснялось, вместо чисел можно рассматривать более привычное представление программ, например, как текстов (последовательностей ASCII символов). «Работает одинаково» означает, что один и тот же текст в двух языках программирования задаёт одну и ту же функцию.

15.10 Решения задач

Для удобства чтения мы повторяем условия задач перед текстом решений.

Задача 15.12. Приведите явный алгоритм перечисления множества $\mathbb{N} \times \mathbb{N}$.

Решение. Как говорилось, перечисление пар натуральных чисел состоит в перечислении их кодов в некоторой кодировке. Алгоритм перечисления последовательно выдаёт результаты 0, 1, 2, …, n, …

На всякий случай объясним, как именно это происходит. Алгоритм перечисления хранит в памяти число n (начальное значение 0). На очередном этапе алгоритм прибавляет к этому числу 1. После завершения этой операции он выдаёт текущее значение n на выход. □

Задача 15.13. Докажите, что множество, состоящее из таких наборов $(v, w, x, y, z) \in \mathbb{N}^5$, что $v^5 + w^5 + x^5 + y^5 = z^5$ перечислимо. Предъявите явный алгоритм перечисления.

Решение. Как указано в тексте, это множество непусто. Следовательно тривиальный (ничего не выводящий) алгоритм перечисления не годится.

Перечисляющий алгоритм перебирает натуральные числа в порядке возрастания. Для каждого числа n он находит пятёрку $(v, w, x, y, z) \in \mathbb{N}^5$, кодом которой является число n, выводит n, только в том случае, когда $v^5 + w^5 + x^5 + y^5 = z^5$ (при проверке этого равенства мы предполагаем, что вычислимы арифметические операции и операция сравнения натуральных чисел). □

Задача 15.14. Докажите, что если множество $A \subseteq \mathbb{N}$ перечислимо, то есть алгоритм, перечисляющий элементы множества A без повторений.

Решение. Пусть P — алгоритм, перечисляющий множество A. Алгоритм Q исполняет алгоритм P и дополнительно хранит список элементов, перечисленных алгоритмом P. Всякий раз, когда P выдаёт x, алгоритм Q проверяет, есть ли x в списке. Если нет, то алгоритм Q выдаёт x и добавляет его в список.

Из построения очевидно, что Q выводит только элементы из множества, перечисляемого P. При этом Q выводит все элементы без повторений: если P выдаёт какой-то элемент x больше одного раза, то Q выдаст x на выход только в первый раз и добавит его в список. Далее всякий раз, когда P выдаёт x, алгоритм Q обнаружит его в списке и потому не выдаст его повторно. □

Задача 15.16. Докажите, что перечислимые множества — это в точности проекции разрешимых.

Решение. Сначала рассмотрим конечные множества. Они все разрешимы (формально, это утверждение леммы 15.13). Значит, по лемме 15.9 все они перечислимы.

С другой стороны, любое конечное множество S является проекцией конечного множества $S \times S$, которое разрешимо.

Далее рассматриваем только бесконечные множества.

Пусть $A \subseteq \mathbb{N} \times \mathbb{N}$ бесконечно и разрешимо. Выберем какой-нибудь его элемент $(a, b) \in A$. По определению, характеристическая функция χ_A вычислима. Но тогда вычислима и функция

$$f(x, y) = \begin{cases} x, & \text{если } \chi_A(x, y) = 1, \\ a, & \text{в противном случае,} \end{cases}$$

для которой $f(\mathbb{N} \times \mathbb{N}) = \text{Пр } A$. Поэтому Пр A перечислимо как множество значений вычислимой функции.

Пусть бесконечное множество S перечислимо. Выберем какой-нибудь перечисляющий его алгоритм P. Пусть этот алгоритм выдаёт последовательность $s_1, s_2, \ldots$, элементы которой образуют множество S. Тогда S — проекция множества

$$A = \{(x, t) : x = s_t,\ t \in \mathbb{N}\}$$

(проекцией такого множества являются в точности те элементы, которые встречаются в последовательности $s_1, s_2, \ldots$).

Докажем, что A разрешимо. Алгоритм разрешения получает на вход пару (x, t) и исполняет перечисляющий множество S алгоритм P, подсчитывая попутно, сколько элементов множества S обнаружено. Всякий раз, когда P выдаёт следующий элемент, алгоритм разрешения увеличивает количество выданных элементов k на единицу и сравнивает его с входным значением t. Если $k = t$, то алгоритм сравнивает x с обнаруженным только что элементом s. Если $x = s$, то результат работы алгоритма равен 1. В противном случае результат равен 0.

Корректность этого алгоритма очевидна: пара (x, t) принадлежит множеству A тогда и только тогда, когда t-й по счёту обнаруженный алгоритмом перечисления элемент множества S равен x. □

Задача 15.17. Докажите, что функция вычислима тогда и только тогда, когда её график перечислим.

Решение. Ясно, что график является образом $f(\mathbb{N})$ вычислимой функции из $\mathbb{N}$ в $\mathbb{N} \times \mathbb{N}$, которая определена как $x \mapsto (x, f(x))$. Поэтому график вычислимой функции перечислим как множество значений вычислимой функции.

Пусть график $\Gamma = \{(x, y) : y = f(x)\}$ перечислим. Тогда следующая программа вычисляет f: на входе x запускаем перечисление элементов графика; когда найден очередной элемент (y, z), проверяем $x = y$; в случае равенства выдаём результат z.

Из определения графика ясно, что на входе x такая программа может выдать лишь результат $f(x)$. С другой стороны, если x принадлежит области определения f, то пара $(x, f(x))$ рано или поздно будет перечислена. В этот момент программа и выдаст результат $f(x)$. □

Задача 15.18. Докажите, множество $S \subseteq \mathbb{N}$ разрешимо тогда и только тогда, когда существует алгоритм перечисления элементов множества S в возрастающем порядке.

Решение. Если S — конечное множество, то оно разрешимо по лемме 15.13.

Алгоритм, который перечисляет элементы конечного множества S в возрастающем порядке, хранит список элементов S как константу. Он по очереди выдаёт на выход элементы этого списка.

Далее рассматриваем только бесконечные множества.

Пусть бесконечное множество $S \subseteq \mathbb{N}$ разрешимо. Алгоритм, перечисляющий его элементы в возрастающем порядке, перебирает по очереди все натуральные числа в порядке возрастания: $0, 1, \ldots$. Для каждого очередного числа n алгоритм вычисляет характеристическую функцию $\chi_S(n)$ (она вычислима, так как S разрешимо). Если значение характеристической функции равно 1, то алгоритм выдаёт число n на вход.

Ясно, что такой алгоритм выдаёт на вход числа в порядке возрастания и эти числа принадлежат S. Ни один элемент $n \in S$ не будет пропущен: когда алгоритм дойдёт до натурального числа n и вычислит $\chi_S(n)$, то получит значение 1.

Предположим теперь алгоритм P перечисляет элементы бесконечного множества S в возрастающем порядке. Построим алгоритм Q разрешения S.

На вход Q подаётся число x. Алгоритм Q исполняет алгоритм P и ждёт, пока тот не напечатает либо x, либо число, большее x (не напечатав x ранее). В первом случае Q выводит 1, поскольку x был перечислен P, а значит $x \in S$. Во втором случае Q выводит 0: раз было напечатано число, большее x, то x уже никогда не будет перечислено алгоритмом P, так как тот перечисляет элементы S в порядке возрастания. Значит, $x \notin S$. □

Задача 15.19. Докажите, что множество таких натуральных чисел n, что в десятичной записи числа π встречается n идущих подряд девяток, разрешимо.

Решение. Это множество «замкнуто вниз»: если есть n идущих подряд девяток, то (в том же месте десятичной записи π) найдётся и k идущих подряд девяток для любого $k < n$.

Значит, такое множество либо содержит все натуральные числа и разрешимо очевидным образом, либо оно конечно. Тогда оно разрешимо по лемме 15.13. □

Задача 15.20. Докажите, что любая всюду определённая невозрастающая функция из натуральных чисел в натуральные числа вычислима.

Решение. Множество значений невозрастающей функции f из натуральных чисел в натуральные числа конечно: все значения не превосходят $f(0)$. Поэтому, начиная с некоторого n_0, функция становится постоянной: $f(n) = C$ при всех $n \geqslant n_0$.

Алгоритм вычисления хранит в качестве констант C, n_0 и таблицу значений f на отрезке от 0 до n_0. Получив на вход x, он сравнивает x с n_0. Если $x \geqslant n_0$, алгоритм выдаёт результат C. В противном случае алгоритм находит в таблице значений пару (x, f_x) и выдаёт результат f_x.

Замечание: мы доказали *существование* алгоритма вычисления f. Из этого доказательства остаётся совершенно неясным, как *построить* такой алгоритм для конкретной функции. □

Напомним, что в следующих задачах $U(n, x)$ — главная универсальная функция.

Задача 15.25. Докажите, что для некоторого m функции $U(m, x)$ и $U(m+1, x)$ совпадают.

Решение. Функция $s(n) = n + 1$ является вычислимой и всюду определённой. Применив к ней теорему о неподвижной точке, получаем, что для некоторого числа m и любого x выполняются равенства $U(m, x) = U(s(m), x) = U(m + 1, x)$ (в расширенном смысле). □

Задача 15.26. Докажите, что каждая всюду определённая функция $s(n)$ имеет бесконечно много неподвижных точек: $\{m : U_m = U_{s(m)}\} = \infty$.

Решение. Доказательство от противного. Допустим, что функция s имеет конечное число неподвижных точек $m_1, \ldots, m_k$. Пусть $n_\uparrow$ — номер нигде не определённой функции в U, а $n_\downarrow$ — номер некоторой всюду определённой функции.

Разобьём неподвижные точки на две группы: $M_\uparrow$ состоит из тех неподвижных точек m, для которых функция $U_m(x) = U(m, x)$ нигде не определена; $M_\downarrow$ состоит из тех неподвижных точек m, для которых функция $U_m(x) = U(m, x)$ определена хотя бы для одного значения x.

Определим функцию

$$h(n) = \begin{cases} s(n), & \text{если } n \notin \{m_1, \ldots, m_k\}; \\ n_\uparrow, & \text{если } n \in M_\downarrow; \\ n_\downarrow, & \text{если } n \in M_\uparrow. \end{cases}$$

Функция $h(n)$ всюду определена и вычислима. Заметим, что $U_{m_i} \neq U_{m_i}$ в силу определения h: если одна из функций U_{m_i}, U_{m_i} определена хотя бы для одного значения аргумента, то вторая нигде не определена. Поэтому ни одно число из множества $m_1, \ldots, m_k$ не является неподвижной точкой $h(n)$.

Применим для h теорему о неподвижной точке. Получаем, что для некоторого $m \notin \{m_1, \ldots, m_k\}$ выполнено $U_{h(m)} = U_m$. Но для такого m выполняется равенство $h(m) = s(m)$. Поэтому $U_{s(m)} = U_m$ и мы предъявили для функции s ещё одну неподвижную точку, что приводит нас к противоречию. □

Задача 15.27. Докажите, что каждая вычислимая функция имеет бесконечно много номеров в главной нумерации.

Решение. Пусть f — произвольная вычислимая функция. Построим вычислимую функцию двух аргументов $V(n, x) = f(x)$. В силу определения главной нумерации для V существует такая всюду определённая вычислимая функция $s\colon \mathbb{N} \to \mathbb{N}$, что $U(s(n), x) = V(n, x) = f(x)$ для всех x.

По предыдущей задаче у функции s бесконечно много неподвижных точек. Это означает, что для бесконечно многих m выполнено $U_m = U_{s(m)} = f$. □

Лекция 16

Машины Тьюринга

В этой главе мы рассмотрим классическую модель вычислений — *машины Тьюринга* (МТ), на основе которой дадим точное определение класса вычислимых функций.

В главе 14 при объяснении алгоритмической неразрешимости задачи остановки игры FRACTRAN мы использовали специальный бедный язык программирования и неформально объяснили, почему этот язык настолько мощный, что в нём можно записать любой алгоритм. Можно уточнить описание этого языка и получить другое определение класса вычислимых функций. Ещё одна возможность — обобщить примеры вычислимых функций из раздела 15.1 и получить точное определение класса рекурсивных функций.

Все эти определения оказываются эквивалентными. Доказательства эквивалентности идейно несложные, но громоздкие. Мы их не приводим. Вместо этого мы подробно изучим только определение, основанное на машинах Тьюринга, и покажем, как доказывать утверждения об алгоритмах в этой модели, на примерах тех свойств вычислимых функций, которые мы предполагали в предыдущей главе 15.

Машины Тьюринга часто оказываются удобными в доказательствах алгоритмической неразрешимости конкретных задач. В конце главы мы приводим один пример такого доказательства.

16.1 Определения

Мы начнём со стандартных определений.

МТ состоит из

– бесконечной в две стороны ленты, в ячейках которой могут быть записаны символы *алфавита* A (некоторого конечного множества);

– *головки*, которая может двигаться вдоль ленты, обозревая в каждый данный момент времени одну из ячеек;

– *оперативной памяти*, которая имеет конечный размер (другими словами, состояние оперативной памяти — это элемент некоторого конечного множества Q,

которое называется *множеством состояний MT*);

– *таблицы переходов* (или программы), которая задаёт функцию

$$\delta\colon A \times Q \to A \times Q \times \{-1, 0, +1\}.$$

Поскольку таблица переходов — это функция на конечном множестве, её возможно задать таблицей. Каждая строка таблицы — это пять значений

$$a,\ q,\ a',\ q', d, \quad \text{(другие способы записи: } \delta(a,q) = (a', q', d) \text{ или } \delta\colon (a,q) \mapsto (a', q', d)).$$

которые описывают следующий порядок действий МТ: если головка МТ находится над ячейкой, содержащей символ a, а состояние МТ равно q, то на очередном такте работы МТ записывает в текущую ячейку символ a', изменяет состояние на q' и сдвигает головку на d ячеек (отрицательное значение отвечает сдвигу влево, положительное — сдвигу вправо). Пример такта работы МТ изображён на рис. 16.1.

Работа МТ состоит из последовательного выполнения тактов в соответствии с таблицей переходов. Может так случиться, что для текущей пары значений (a, q) функция переходов не определена. В этом случае работа машины заканчивается (машина останавливается). Обычно среди состояний МТ выделяют множество Q_f *финальных состояний* — таких состояний q_f, что таблица переходов не определена для всех пар (a, q_f). Попав в финальное состояние, машина обязательно остановится, отсюда и название.

Лента МТ бесконечна и это не соответствует нашей интуиции об алгоритмах: алгоритм на каждом шаге работы оперирует лишь данными конечного размера. Чтобы учесть это обстоятельство, мы предполагаем, что в алфавите машины есть специальный символ Λ (пробел или *пустой символ*) и все ячейки ленты за исключением конечного числа содержат пустые символы. Это свойство ленты сохраняется при работе МТ, поскольку за такт работы меняется содержимое не более одной ячейки ленты.

Работу такой машины уже можно описать конечными данными. Вместо картинок как на рис. 16.1 мы будем использовать *конфигурации*. Конфигурация — это слово в алфавите $A \cup Q$, в котором первый и последний символы непустые, и ровно один символ принадлежит множеству состояний. Договоримся считать, что символ состояния записывается слева от символа в той ячейке, над которой находится головка МТ. На ленте слева и справа от символов конфигурации стоят только пустые

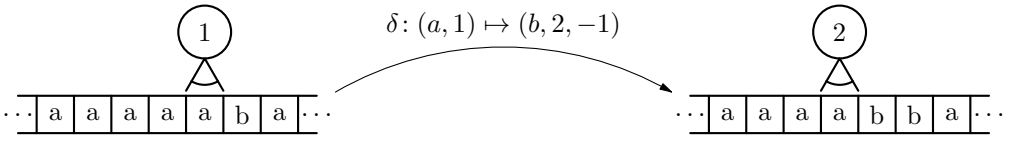


Рис. 16.1: Такт работы машины Тьюринга

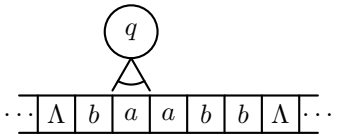


Рис. 16.2: Конфигурация МТ: слово *bqaabb*

символы. На рис. 16.2 проиллюстрировано соответствие между лентой с головкой и конфигурациями.

Мы также предполагаем, что работа МТ начинается из конфигурации вида q_0u, где q_0 — особое *начальное* состояние машины, а слово u — входные данные (или *вход*) машины. Конфигурации МТ преобразуются такт за тактом, порождая последовательность конфигураций

$$c_0 = q_0u,\ c_1,\ c_2, \ldots, c_t,\ \ldots$$

Эта последовательность бесконечна, если машина не останавливается, и конечна в противном случае. *Результатом* работы является та часть финальной конфигурации, которая расположена между символом состояния и ближайшим к нему пустым символом справа (см. рис. 16.3).

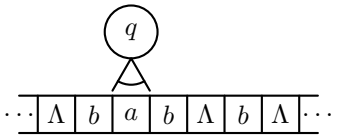


Рис. 16.3: Результат МТ: слово *ab*

Определение 16.1. МТ M *вычисляет* функцию $f\colon B^* \to B^*$ (где B — подмножество алфавита машины, не содержащее пустого символа, а B^* — множество слов в алфавите B), если для каждого w из области определения функции f результат работы машины M, запущенной с начальными данными w, равен $f(w)$, а для каждого w не из области определения f машина M не останавливается на входе w.

Функция f называется *вычислимой по Тьюрингу*, если есть такая МТ, которая вычисляет f.

Замечание 16.1. В литературе встречаются различные определения машин Тьюринга и функций, вычисляемых машинами Тьюринга. Эти различия несущественны, хотя от деталей определений зависят рассуждения о МТ. Всюду далее мы предполагаем данные выше определения.

Пример 16.1. Рассмотрим машину M_b с алфавитом A, множеством состояний $\{0,1\}$, где 0 — начальное состояние, и таблицей переходов

$$\delta\colon (a,0) \mapsto \begin{cases} (a,0,+1), & \text{если } a \neq b, \\ (b,1,0), & \text{если } a = b. \end{cases}$$

Это описание машины, которое задаёт её однозначно. И ничего больше — в описании не содержится никаких утверждений о такой машине.

Мы утверждаем, что такая машина переводит головку на ближайший справа символ b и после этого останавливается в состоянии 1 (и не останавливается, если справа нет ни одного символа b). Справедливость такого утверждения нужно доказывать.

Заметим, что если головка находится над символом b, то машина остановится после первого такта: это сразу следует из описания таблицы переходов.

Если же машина находится над каким-то другим символом, то из таблицы переходов видно, что головка сдвигается вправо без изменения состояния и символа в ячейке. Формальное завершение этого рассуждения требует применения индукции (по количеству символов между начальным положением головки и ближайшим справа символом b).

Машину из примера легко модифицировать, чтобы перемещать головку влево до первого символа b. Дальше в описании более сложных машин мы будем ссылаться на машину M_b и её левый аналог как на инструкцию «сдвинуться вправо до символа b» («сдвинуться влево ...»).

Пример 16.2. Построим машину, которая вычисляет функцию $f\colon w \mapsto wa$, где w — слово в алфавите $\{a\}$. Другими словами, эта машина реализует функцию $x \mapsto x+1$ на натуральных числах, если считать, что числа заданы в *унарной системе* (число n представлено словом, состоящим из n букв a; в детстве всех учат считать в унарной системе, только вместо букв a используют палочки).

Зададим машину таблицей переходов (0 — начальное состояние):

$$\begin{aligned} &\delta\colon (a,0) \mapsto (a,0,+1) \\ &\delta\colon (\Lambda,0) \mapsto (a,1,-1) \\ &\delta\colon (a,1) \mapsto (a,1,-1) \\ &\delta\colon (\Lambda,1) \mapsto (\Lambda,2,+1). \end{aligned}$$

Докажем, что эта машина и впрямь вычисляет заданную функцию.

Аналогично предыдущему примеру доказывается, что конфигурацию[1] $0a^n$ машина за n тактов переводит в конфигурацию a^n0 (исполняется только первая строчка таблицы переходов). На следующем такте работы исполняется вторая строчка и конфигурация становится $a^{n-1}1a^2$, если $n > 0$, и $1\Lambda a$, если $n = 0$. В первом случае

[1]Здесь, как и в остальных местах мы используем обозначение a^n для слова, состоящего из n символов a (или из n повторений слова a, если это не одиночный символ).

на последующих шагах работы исполняется третья строчка до тех пор, пока машина не увидит символ Λ, т.е. получится конфигурация $1\Lambda a^{n+1}$ (это утверждение легко доказывается по индукции). Теперь исполняется четвёртая строчка и в конфигурации $2a^{n+1}$ машина заканчивает работу, так как из описания машины видно, что таблица переходов не определена для всех пар $(x, 2)$. (Напомним, что в этом случае по нашему определению машина останавливается.)

Результат работы такой машины по определению равен $a^{n+1} = a^n a$.

16.2 Тезис Чёрча – Тьюринга

Тезис Чёрча – Тьюринга: *всякая вычислимая функция вычислима машиной Тьюринга.*

Другими словами, мы считаем машины Тьюринга формальным определением понятия алгоритма.

Тезис Чёрча – Тьюринга не является математическим утверждением. Это математическое определение. Его можно также воспринимать как закон природы: утверждение об окружающем нас мире, основанное на опыте.

Обсудим неформальные обоснования для этого тезиса. Во-первых, почему работа МТ реализуется алгоритмом? Это почти очевидно: каждый такт работы МТ состоит в выполнении очень простого набора действий: найти в таблице переходов строчку, которая начинается с текущего символа и текущего состояния; если таковой не нашлось, закончить работы и выделить из конфигурации результат; в противном случае заменить текущий символ на тот, который указан в найденной строчке, изменить состояние на указанное в найденной строчке, переместить головку влево, оставить на месте или переместить вправо в зависимости от указанной в найденной строчке команды движения. Фактически мы описали простые и однозначно понимаемые инструкции действий, что и есть алгоритм в неформальном понимании слова.

В другую строну тезис куда менее очевиден. Мы знаем алгоритмы, которые работают со сложными структурами данных и действия которых организованы куда более сложным образом, чем описанная выше циклическая последовательность локальных замен в слове. Почему ни один такой алгоритм не даёт больше возможностей, чем машина Тьюринга?

Объяснение, восходящее к Тьюрингу, тут такое. Любой алгоритм потенциально может быть исполнен человеком. Конечно, время такого исполнения может оказаться намного больше времени человеческой жизни, но нас интересует лишь потенциальная возможность.

А исполнение алгоритма человеком в предельно упрощённом виде можно представить так. У человека есть карандаш, ластик, неограниченный запас листов бумаги и книжечка с инструкциями (собственно алгоритм). Бумага лежит в двух стопках, человек может выполнять действия лишь на одной стороне листа. Это аналог ленты машины. Действия определяются номером страницы в книге инструкций (состояние МТ) и содержимым верхнего листа бумаги (символ алфавита в ячейке, на которую смотрит головка). Чтобы действие было однозначно понимаемым, разных

содержимых листа бумаги должно быть конечное число. Вот таким образом и приходим к формальному определению.

Если оставаться на чисто математических позициях, то нужно рассуждать так: у нас появилось определение алгоритма, так что теперь нужно его использовать для доказательства утверждений. В частности, все утверждения, которые мы раньше доказывали в рамках «наивной» теории алгоритмов, нужно теперь доказывать аккуратно, используя определения на основе машин Тьюринга.

16.3 Машины Тьюринга и свойства вычислимых функций

Давайте вспомним, какие свойства алгоритмов и вычислимых функций мы постулировали в предыдущей главе.

1. Алгоритм имеет конечное описание.
2. Алгоритм выполняется по шагам.
3. На каждом шаге алгоритма должно быть чётко и однозначно определено, что нужно делать на следующем шаге.
4. Пусть имеются алгоритмы A, B, C, причём результат работы алгоритма C «логический» (равен 0 или 1). Тогда их можно использовать при составлении других алгоритмов следующими способами.

   Вызов подпрограммы: допустима инструкция «исполнить алгоритм A и сохранить результат его работы для последующих вычислений».

   Условный переход: допустима инструкция «исполнить алгоритм C; если результат его работы равен 1, то исполнить алгоритм A; в противном случае исполнить алгоритм B».
5. Существует универсальная вычислимая функция U для класса вычислимых функций одной переменной.

Этот список неполный. Помимо этого мы использовали в рассуждениях утверждения о вычислимости некоторых конкретных функций (прежде всего, арифметических операций) и утверждения о возможности параллельного исполнения двух алгоритмов или даже неограниченного их числа.

Раз мы признали машины Тьюринга формальным определением алгоритма, эти свойства становятся математическими утверждениями, которые требуют доказательства. Это не очень простое дело, и почти вся данная глава посвящена обсуждению этих доказательств. Попутно нам придётся уточнить данные выше определения.

Начнём с первого свойства. Казалось бы, оно очевидно выполняется для машин Тьюринга. Машина полностью описывается указанием конечного множества состояний, конечного алфавита, начального состояния, заключительных состояний,

пустого символа и таблицы переходов. Все указанные объекты конечны. В чём тут может быть проблема?

Некоторая проблема тут всё-таки имеется. Конечных множеств слишком много. Скажем, не очень понятно, как задать конечным описанием произвольное конечное множество действительных чисел.[2]

Однако в отношении машин Тьюринга эта проблема не очень важна и поэтому игнорируется в стандартном определении, приведённом выше. Дело в том, что если переименовать все символы алфавита и все состояния, по существу работа машины не изменится.

Таким образом, можно договориться заранее, что и множество состояний машины, и множество символов занумерованы начальными отрезками натурального ряда. Причём можно также сразу договориться, что начальное состояние и пустой символ занумерованы нулями. Мы будем подразумевать эту договорённость всюду далее и будем обозначать состояния как q_i, а символы алфавита как a_j. Более того, если не возникает путаницы будем обозначать и состояния, и символы алфавита их номерами. Наряду с этим будем при необходимости использовать и другие обозначения для символов алфавита и состояний, подразумевая принципиальную возможность их нумерации и приведения к указанному стандартному виду.

Со вторым свойством всё хорошо: как и требовалось, работа машины Тьюринга происходит по шагам (или тактам, как обычно говорят в отношении машины Тьюринга).

С третьим свойством всё также прекрасно: действия машины Тьюринга на каждом такте работы полностью определяются её конфигурацией и изменение конфигурации однозначно определено.[3]

С двумя последними свойствами есть технические трудности. Интуитивно кажется ясным, что машина Тьюринга может осуществлять условные переходы, но очень ограниченного вида: в зависимости от содержания текущей ячейки памяти или своего состояния. Запустить другую машину Тьюринга как подпрограмму также кажется несложным, но тут также есть ограничения, связанные с крайней скудостью возможностей МТ по хранению и обработке данных.

И уж совсем неясно на первый взгляд, как доказать, не обращаясь к тезису Чёрча–Тьюринга, что для МТ возможно сохранение промежуточных результатов вычислений, параллельный запуск двух МТ или даже неограниченного их числа. Непонятно также, что такое та универсальная функция, о которой идёт речь в последнем из процитированных выше свойств.

[2]Если непонятна суть проблемы, вернитесь к обсуждению в главе 14, с. 356.

[3]Вам могут встретиться в книгах так называемые «недетерминированные машины Тьюринга». Это другая модель вычислений, когда действия алгоритма на шаге работы не обязательно однозначно определены. Для простоты мы здесь не касаемся таких алгоритмов.

16.4 Использование машин Тьюринга в доказательствах

Машины Тьюринга придуманы не для того, чтобы писать на них программы, а для того, чтобы доказывать математические утверждения. Поэтому обсудим важный вопрос: насколько подробно нужно описывать машину Тьюринга в рассуждении?

Прежде всего заметим, что само по себе описание машины Тьюринга, как и любое другое описание алгоритма, не является доказательством какого-либо утверждения. Это просто набор инструкций, который может исполнять человек или специально выдрессированный компьютер.

Конечно, в полном описании машины Тьюринга имеется вся информация, необходимая для доказательства свойств этой машины. Однако использовать столь подробную информацию зачастую неудобно.

С другой стороны, если нас интересует лишь существование машины, которая выполняет то или иное преобразование, то достаточно описать машину с той степенью подробности, которая нужна в доказательстве корректности её работы.

Для облегчения рассуждений с МТ есть несколько приёмов, которые заимствованы из практики написания реальных программ.

Блочная структура. Как обычно в программировании, при описании МТ удобно разбивать всю программу (в данном случае таблицу переходов) на блоки (подпрограммы, процедуры и т.п.)

Простейший вариант разбиения на блоки — это последовательное соединение машин. Опишем последовательное соединение двух машин, случай нескольких машин аналогичен.

Пусть есть машины Тьюринга M_1 и M_2. Их последовательное соединение — это МТ, которая получается после следующих преобразований таблиц переходов машин M_1 и M_2:

– переименовываем состояния машин так, чтобы они не пересекались (легко видеть, что от переименования состояний результат работы машины не меняется);

– для каждой пары (a, q), на которой не определена таблица переходов машины M_1, добавляем в таблицу переходов машины–соединения строчку

$$\delta(a, q) = (a, q_0^{(2)}, 0),$$

здесь $q_0^{(2)}$ — начальное состояние машины M_2.

Таким образом, если машина M_1 останавливается, то обязательно начинает работу машина M_2.

Контрольный вопрос 16.3. Почему необходимо переименование состояний?

Метки на ленте. Алфавит МТ конечен, но если нас интересует лишь существование машины, он может быть сколь угодно велик. Этим обстоятельством удобно пользоваться при описании машин, говоря о «метках» на ячейках ленты. Формально

использование меток состоит в том, что мы заменяем алфавит A на расширенный алфавит $A \times L$, где L — то вспомогательное множество меток. Договоримся, что среди меток всегда есть пустой символ Λ, который обозначает отсутствие метки. Символы исходного алфавита отождествим с парами (a, Λ), это позволяет говорить о вычислении функций в алфавите, который содержится в исходном алфавите A. В таблице переходов МТ метки позволяют определять команды вида «если данная ячейка помечена, сделай то-то, а если нет, то сделай иное».

Структурирование оперативной памяти. Аналогично алфавиту удобно расширять и множество состояний, разделив его на две части: «управляющие состояния» и «оперативная память». Управляющие состояния используются для организации условных переходов и соединения машин (как это было описано выше). Оперативная память используется для хранения информации. Скажем, машине потребовалось «запомнить» символ текущей ячейки. Для этого она меняет состояние на пару («управляющее состояние», «значение символа»).

Важно помнить, что множество состояний МТ конечно. Поэтому и «оперативная память» конечна — машина в состоянии запомнить лишь элемент какого-то конечного множества.

16.5 Композиция функций, вычислимых по Тьюрингу, и уборка мусора

После этих предварительных соглашений можно перейти к доказательствам утверждений о МТ.

Как мы помним, одно из важных свойств вычислимых функций — замкнутость относительно композиции. Докажем это свойство для функций, вычислимых по Тьюрингу.

Теорема 16.2. *Пусть $f\colon B^* \to B^*$, $g\colon B^* \to B^*$ вычислимы по Тьюрингу. Тогда $f \circ g$ также вычислима по Тьюрингу.*

Идея доказательства очевидна. Как мы говорили выше, результат работы одного алгоритма можно подать на вход другого алгоритма.

Поэтому предложим такое рассуждение. Пусть M_1 вычисляет g, а M_2 вычисляет f. Тогда МТ, которая состоит из последовательного соединения блоков M_1 и M_2 вычисляет $f \circ g$.

К сожалению, это рассуждение неверно. Дело в том, как определён результат вычисления МТ. Помимо собственно результата на ленте могут оказаться посторонние символы — «мусор» — и они изменят работу машины M_2. Поэтому такое соединение машин, вообще говоря, вычисляет какую-то другую функцию, а не композицию функций f и g.

Тем не менее, если первая машина M_1 заканчивает работу, оставляя на ленте только полезный результат и ничего больше, рассуждение становится корректным. Поэтому для завершения доказательства теоремы 16.2 нам нужно решить проблему «уборки мусора».

Заметим, что убирать мусор в конце работы поздно: непустые символы могут быть отделены от текущего положения головки сколь угодно длинными последовательностями из пустых символов.

Задача 16.4. Докажите, что не существует МТ с таким «волшебным состоянием» q_m, что любую конфигурацию, содержащую q_m, машина переводит в пустую конфигурацию q_f, где q_f — финальное состояние.

(См. решение в последнем разделе главы.)

Решить проблему уборки мусора можно, предусмотрев заранее некоторые санитарные процедуры. А именно, добавим к алфавиту ещё два символа $\triangleleft$, $\triangleright$, которые будут ограничивать рабочую зону на ленте (в процессе работы машины слева от $\triangleleft$ и справа от $\triangleright$ всегда будут только пустые символы). При разрастании рабочей зоны эти символы нужно сдвигать. В конце работы нужно будет выделить результат и «стереть» (то есть заменить на пустые) все символы внутри рабочей зоны, которая ограничена символами $\triangleleft$, $\triangleright$.

Лемма 16.3 (об уборке мусора)**.** *Пусть машина M вычисляет функцию f. Тогда существует такая машина M', которая вычисляет ту же функцию, но финальная конфигурация которой на любом входе w из области определения f имеет вид $q_f f(w)$.*

Доказательство. Машина M' будет последовательным соединением четырёх машин.

Первая машина M_1 преобразует начальную конфигурацию $q_0' w$ в конфигурацию $\triangleleft q_0 w \triangleright$, где q_0 — начальное состояние машины M.

Вторая машина M_2 работает так же, как исходная машина M, но сохраняет окаймление конфигурации символами $\triangleleft$, $\triangleright$.

Третья машина M_3 стирает символы слева от положения головки в финальной конфигурации машины M_2.

Четвёртая машина M_4 стирает символы справа от последнего символа результата работы M_2 и возвращает головку в ту ячейку, в которой она была при остановке машины M_2.

Как ясно из этого описания, при корректной реализации каждой из этих четырёх машин их соединение M' удовлетворяет искомому свойству.

Опишем реализации этих четырёх машин.

Машина M_1 последовательно выполняет следующие шаги:

1. сдвинуться на одну ячейку влево, записать в неё символ $\triangleleft$;
2. сдвинуться до первого пустого символа справа;
3. записать символ $\triangleright$;
4. сдвинуться до символа $\triangleleft$ слева;

Шаги 1, 3, 4 требуют выполнения конечного числа действий и потому реализуемы подходящей МТ. Шаг 2 исполняется МТ из примера 16.1. Корректность такой машины очевидна.

Таблица переходов машины M_2 совпадает с таблицей переходов исходной машины M за исключением работы на добавленных символах $\triangleleft$, $\triangleright$. На символе $\triangleleft$ машина выполняет следующие такты работы:

1. записать пустой символ, сдвинуться влево и перейти в состояние $\tilde{q}$;
2. записать символ $\triangleleft$, сдвинуться вправо перейти в состояние q.

На рис. 16.4 показано выполнение этих шагов. Работа M_2 на символе $\triangleright$ устроена аналогично (разумеется, символ переносится вправо).

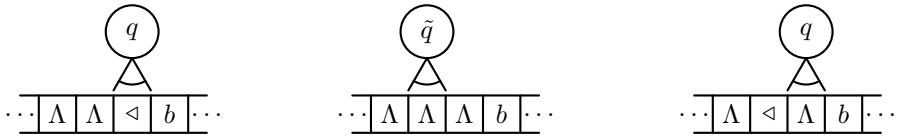


Рис. 16.4: Сдвиг левого ограничителя рабочей зоны

Как видно из описания, количество состояний у машины M_2 в два раза больше, чем у исходной машины M, а работа при чтении добавленных символов окаймления устроена так, что машина переносит символ окаймления на одну ячейку вне рабочей зоны, возвращается назад и продолжает работу как исходная машина M. Поэтому машина M_2 закончит работу, имея слева и справа от рабочей зоны символы $\triangleleft$, $\triangleright$.

Опишем теперь устройство машины M_3. Она начинает работу в одном из финальных состояний q_f исходной машины M. Работа машины M_3 разбивается на следующие шаги:

1. пометить текущую ячейку, сдвинуться влево и перейти в состояние q_ℓ;
2. идти влево до символа $\triangleleft$, заменяя символ в каждой ячейке на пустой символ;
3. на символе $\triangleleft$ записать пустой символ и перейти в финальное состояние q_r.

Второй шаг реализуется такой таблицей переходов:

$$\delta(a, q_\ell) = (\Lambda, q_\ell, -1), \quad a \neq \triangleleft.$$

Ясно, что в результате работы машины M_3 все символы слева от результата работы исходной машины M будут заменены на пустые.

Последняя машина M_4 должна убирать мусор справа от результата работы исходной машины. Она начинает работу в состоянии q_r (финальном состоянии предыдущей машины) и исполняет следующие шаги:

1. сдвинуться вправо до помеченной ячейки;

2. сдвинуться вправо до пустого символа (быть может, помеченного);
3. идти вправо до символа $\triangleright$, заменяя символ в каждой ячейке на пустой символ;
4. на символе $\triangleright$ записать пустой символ;
5. идти влево до помеченной ячейки;
6. убрать пометку и остановиться.

Шаги 1, 2, 5 исполняются аналогично примеру 16.1. (Контрольный вопрос, почему необходим первый шаг?) Шаг 3 исполняется аналогично шагу 2 машины M_3. Реализация остальных шагов очевидна. Обратите также внимание на уточнение в шаге 2. Оно необходимо для корректности работы машины в случае пустого результата работы (машина M_3 в этом случае ставит пометку на пустой символ). □

Доказательство теоремы 16.2. По лемме об уборке мусора каждая вычислимая по Тьюрингу функция вычислима такой «чистой» машиной, которая в конце работы оставляет на ленте только результат работы.

Последовательное соединение «чистых» машин вычисляет композицию функций, вычислимых этими машинами. □

16.6 Многоленточные машины Тьюринга

Как видно из предыдущего раздела, рассуждения о машинах Тьюринга получаются длинными и не всегда отвечающими нашей интуиции об алгоритмах. Частично ситуацию можно исправить, рассмотрев чуть более мощную модель вычислений — многоленточные машины Тьюринга — и доказав эквивалентность этих моделей. В этом разделе мы дадим определения для многоленточных машин Тьюринга, а в следующем докажем эквивалентность многоленточных и обычных (одноленточных) МТ.

Как уже ясно из названия, у многоленточных машин не одна лента, а несколько (фиксированное число для конкретной машины). На каждой ленте есть своя головка. За такт работы головки могут перемещаться по всем лентам. Действие на такте работы зависит как от состояния машины, так и от всего набора символов, которые видят головки машины на всех лентах.

Как эти неформальные изменения отражаются в формальном определении многоленточной машины и функций, вычисляемых такими машинами? Чтобы задать машину с h лентами, нужно указать:

– алфавит A, в котором выделен пустой символ Λ;
– множество состояний Q, в котором выделено начальное состояние q_0;
– таблицу переходов, которая теперь является функцией вида

$$\delta\colon A^h \times Q \to A^h \times Q \times \{-1, 0, +1\}^h$$

(первый аргумент — символы, которые машина видит на ленте; последний — команды движения для головок на каждой ленте);

– выделить среди лент *ленту входа* и *ленту результата* (возможно, что это одна и та же лента).

Таблица переходов по прежнему является функцией на конечном множестве, поэтому её возможно задать таблицей.

Работа МТ состоит из последовательного выполнения тактов в соответствии с таблицей переходов. Может так случиться, что для текущего набора значений $(a_1, a_2, \ldots, a_h, q)$ функция переходов не определена. В этом случае работа машины заканчивается (машина останавливается). Как и раньше, можно ввести множество финальных состояний Q_f, т.е. тех состояний q_f, для которых таблица переходов не определена для всех значений $(a_1, a_2, \ldots, a_h, q_f)$. В финальном состоянии машина обязательно останавливается.

Мы предполагаем, что h-МТ начинает работу в состоянии q_0, а все ленты кроме ленты входа содержат только пустые символы. На ленте входа записано входное слово, и головка находится над первой слева ячейкой, содержащей символы этого слова.

Поскольку за такт работы меняется содержимое не более одной ячейки ленты, в процессе работы машины на каждой ленте будет записано лишь конечное количество непустых символов.

Конфигурация многоленточной машины может быть задана набором конфигураций на каждой ленте

$$(u_1 q v_1,\ u_2 q v_2,\ \ldots,\ u_h q v_h).$$

Символ состояния один и тот же, так как по нашим определениям состояние есть у машины, а не у головки.[4]

Далее нам будет удобен ещё способ представления конфигурации машины. Выровняем ленты и будем рассматривать «окно», в которое заведомо помещаются все непустые символы на каждой ленте. В таком случае конфигурация однозначно определяется матрицей размера $h \times N$, в которой записаны символы на лентах. Нужно ещё указать положения головок на лентах (они-то не обязательно выровнены — машина способна перемещать головки независимо). По этой причине будем помещать в матрицу не символы алфавита A, а пары $(\hat{q}, a)$, где $\hat{q}$ указывает, расположена ли на данной ленте головка над данной ячейкой. Если да, то $\hat{q} \in Q$ — текущее состояние машины. Если нет, то $\hat{q}$ — какой-то символ не из Q, который указывает, что над над данной ячейкой на данной ленте нет головки. Будем для единообразия использовать в качестве такого символа Λ. Такую матрицу в дальнейшем называем *матрицей конфигурации*.

[4]Разумеется, возможно и такое определение, в котором состояния головок различаются. В этом случае определение таблицы переходов нужно изменить (подумайте, каким именно образом); но полученная модель вычислений будет эквивалентна нашей.

Вот пример матрицы начальной конфигурации для двухленточной машины:

(q_0, Λ)	(Λ, Λ)	(Λ, Λ)	(Λ, Λ)	(Λ, Λ)
(q_0, a)	(Λ, b)	(Λ, a)	(Λ, a)	(Λ, b)

Как и для одноленточной машины, работа h-МТ порождает последовательность конфигураций

$$c_0 = (q_0 u, \underbrace{q_0, \ldots, q_0}_{X \text{ штук}}),\ c_1,\ c_2, \ldots, c_t,\ \ldots$$

Контрольный вопрос 16.5. Чему равно X в предыдущей формуле?

Эта последовательность бесконечна, если машина не останавливается, и конечна в противном случае. *Результатом* работы является та часть финальной конфигурации на ленте результата, которая расположена между положением головки и ближайшим к нему пустым символом справа. Например, если у двухленточной МТ лента результата — нижняя, то результатом работы МТ, остановившейся в конфигурации, заданной окном

(Λ, Λ)	(Λ, Λ)	(q_f, a)	(Λ, a)	(Λ, Λ)
(Λ, a)	(q_f, b)	(Λ, a)	(Λ, Λ)	(Λ, b)

,

будет ba.

Определение функции, вычислимой h-МТ, сохраняется дословно.

Определение 16.4. h-ленточная машина Тьюринга M *вычисляет* функцию $f\colon B^* \to B^*$ (где B — подмножество алфавита машины, не содержащее пустого символа), если для каждого w из области определения функции f результат работы M равен $f(w)$, а для каждого w не из области определения f машина M не останавливается на входе w.

Многие алгоритмические действия выполняются на многоленточных машинах проще, чем на одноленточных. Предлагаем читателю в этом убедиться, решив следующие задачи.

Задача 16.6. Постройте двухленточную машину, которая (а) копирует с одной ленты на другую символы от текущего положения головки до ближайшего справа символа-разделителя #, скопированное слово на второй ленте начинается с первоначального положения головки на ней; (б) переносит указанные символы (аналогично предыдущему, но на первой ленте символы заменяется на пустые).

(См. решение в последнем разделе главы.)

Задача 16.7. Постройте двухленточную машину, которая сравнивает слова на двух лентах от текущего положения головок до символа-разделителя. В случае равенства слов машина заканчивает работу в состоянии q_y, в случае неравенства — в состоянии q_n.

(См. решение в последнем разделе главы.)

Описанные в этих задачах действия легко модифицировать, если концом зоны, которую нужно скопировать (перенести или сравнить) является не один символ-разделитель, а какое-нибудь фиксированное слово или даже фиксированный набор слов.

16.7 Моделирование многоленточной МТ на одноленточной

Вернёмся к обсуждению общих свойств алгоритмов. Мы использовали в доказательствах возможность параллельного пошагового исполнения алгоритмов. Для одноленточных машин Тьюринга это легко осуществить, если использовать двухленточную машину. На одной ленте работает одна машина, на другой — другая. В качестве множества состояний такой двухленточной машины можно взять пары состояний первой и второй машины. Тогда таблица переходов двухленточной машины получается из таблиц переходов одноленточных машин объединением всех пар строк: паре строк

$$\delta_1(a_1, q_1) = (a_1', q_1', d_1), \qquad \delta_2(a_2, r_2) = (a_2', r_2', d_2)$$

исходных машин будет отвечать строка

$$\delta_{1\otimes 2}((a_1, a_2), (q_1, r_2)) = ((a_1', a_2'), (q_1', r_2'), (d_1, d_2).$$

Обратите внимание, что (q_1, r_2) и (q_1', r_2') — это состояния двухленточной машины, которыми выбраны пары состояний исходных машин.

Всё это хорошо, но ведь в качестве определения алгоритма мы выбрали одноленточные машины? Поэтому нужно разобраться, как преобразовать многоленточную машину в одноленточную с сохранением результатов работы.

Теорема 16.5. *Любая функция, вычислимая на многоленточной МТ, вычислима и на одноленточной машине.*

Доказательство теоремы состоит в том, что по многоленточной машине строится одноленточная машина, которая моделирует работу многоленточной.

Чтобы понять идею такого моделирования, рассмотрим описание конфигурации многоленточной машины M_h в виде матрицы конфигурации размера $h \times N$. Каждый столбец такой матрицы может находиться в конечном числе состояний.

Задача 16.8. Проверьте, что возможных значений столбца матрицы конфигурации h-МТ не более $(A \cdot (Q+1))^h$, где A — размер алфавита, а Q — количество состояний.

Моделирующая машина M_1 использует расширенный алфавит из $A+(A\cdot(Q+1))^h$ (пустой символ и символы, отвечающие различным столбцам матрицы конфигурации). Она поддерживает описание матрицы конфигурации машины M_h в этом алфавите и изменяет его, моделируя работу M_h по тактам.

Поскольку действия M_h на каждом такте работы зависят от её состояния и символов под головками на каждой ленте, машина M_1 поддерживает также и эту информацию, записывая её в «оперативную память». Т.е. состояния M_1 представляются парами («управляющее состояние», «оперативная память»).

Такт работы машины M_h моделируется машиной M_1 в два этапа.

На первом этапе машина M_1 просматривает все непустые ячейки на своей ленте слева направо и определяет, какие символы расположены под текущими положениями головок машины M_h.

На втором этапе M_1 изменяет содержимое своей ленты в соответствии с таблицей переходов машины M_h.

Опишем более детально устройство машины M_1. Алфавит машины M_1 — это множество

$$A' = A \cup ((Q \cup \Lambda) \times A)^h,$$

пустой символ тот же, что и у моделируемой машины M_h, то есть Λ.

Машина M_1 является последовательным соединением трёх машин.

Первая машина M_s подготавливает содержимое ленты к двухэтапному моделированию тактов работы машины M_h. Машина M_s просматривает ячейки входного слова. Первый символ a_1 она заменяет на $((q_0, a_1), (q_0, \Lambda), \ldots, (q_0, \Lambda))$ (это первый столбец матрицы начальной конфигурации M_h), а каждый последующий символ входа a на $((\Lambda, a), (\Lambda, \Lambda), \ldots, (\Lambda, \Lambda))$ (это остальные столбцы матрицы начальной конфигурации — напомним, что в начальной конфигурации все ленты кроме входной пусты). Обнаружив пустой символ Λ, машина M_s возвращается в крайнее левое положение и останавливается.

Вторая машина M_w моделирует такты работы M_h описанным выше способом. Точнее говоря, она преобразует запись конфигурации M_h в «матричном виде» в запись конфигурации M_h после одного такта работы и повторяет этот процесс циклически, пока M_h не достигнет финального состояния (напомним, что состояния M_h хранятся в «оперативной памяти» M_1).

Для моделирования такта работы многоленточной машины, машина M_w проходит непустые ячейки ленты два раза. При движении слева направо машина M_w «запоминает» символы под головками машины M_h по следующему правилу: если в очередном столбце матрицы конфигурации машины M_h на i-й позиции находится пара (q, a), $q \in Q$, то i-я головка расположена над символом a.

К концу первого прохода в оперативной памяти M_w содержится полная информация о символах под головками и состоянии M_h, что однозначно определяет строчку таблицы переходов M_h, которую нужно применить на данном такте. Если такой строчки нет, то M_w заканчивает работу.

На втором проходе найденная строчка таблицы переходов M_h используется для обновления матрицы конфигурации. Информация о символах на лентах M_h обновляется по следующему правилу: если в очередном столбце матрицы конфигурации машины M_h на i-й позиции находится пара (q, a), $q \in Q$, то столбец меняется так, чтобы в этой позиции было написана пара (q, a'), где a' — символ, который M_h записывает на i-ую ленту. Те пары в столбце, которые соответствуют ячейкам, над которыми нет головки, не изменяются.

Кроме того нужно обновить информацию о положениях головок соответственно текущей команде движения. Для этого машина M_w переписывает вторые компоненты пар, из которых состоит столбец матрицы конфигурации M_h (т.е. текущий

столбец матрицы). Движение по каждой ленте может быть как влево, так и вправо. Поэтому для выполнения этого действия M_w перемещается из текущего положения на шаг вправо, записывая в этот столбец новые положения головок, и на шаг влево, выполняя аналогичное действие. При выполнении этих действий машина M_w может выйти за пределы рабочей зоны (матрицы конфигурации). Тогда она оказывается над пустым символом, который заменяется на подходящий столбец, описывающий пустые символы и положения головок машины M_h.

По завершении работы машины M_w начинает работу третья машина M_f, которая восстанавливает на ленте состояние ленты результата машины M_h. Она проходит по всем ячейкам рабочей зоны и заменяет столбец матрицы конфигурации на символ из алфавита A машины M_h, если головка M_h на ленте результата не находится в этом столбце. Затем она возвращается в ту ячейку, которая соответствует положению головки на ленте результата машины M_h, и производит ту же замену. После этого M_f останавливается.

Корректность работы этих машин вполне очевидна из сделанного описания.

16.8 Универсальная машина Тьюринга

Важнейшим свойством вычислимых функций является существование универсальной вычислимой функции. Выбрав в качестве формального определения алгоритма машины Тьюринга, мы должны доказать, что это свойство выполняется.

На языке машин Тьюринга удобнее говорить о существовании универсальной машины, способной исполнять работу любой другой МТ, если на вход универсальной машины подано описание МТ и её входа.

Используя неформальное обоснование тезиса Чёрча – Тьюринга, легко убедить себя в том, что универсальная машина существует. Однако более детальное доказательство этого факта само по себе даёт дополнительную уверенность в тезисе Чёрча – Тьюринга и позволяет пересказывать результаты о неразрешимости, аналогичные тем, что были получены в абстрактной постановке.

Самый короткий путь к построению универсальной машины лежит через построение многоленточной универсальной машины. Как мы уже знаем, многоленточные машины моделируются одноленточными, так что этого достаточно.

По аналогии с абстрактной теорией хотелось бы сказать, что универсальная машина Тьюринга U моделирует работу любой другой машины в том смысле, что получив на вход описание машины Тьюринга M и описание её входа x, она выдаёт тот же результат $M(x)$.

Однако такая машина невозможна: ведь её алфавит конечен, а существуют МТ со сколь угодно большим алфавитом. Поэтому нужно научиться описывать и МТ, и их входные слова в каком-то одном алфавите. После этого уже можно сформулировать требования к универсальной машине.

Описание машин и их входов в одном алфавите — несложная задача, хотя и требующая утомительных подробностей.

Зафиксируем формат описания МТ и входных слов. Как уже говорилось, считаем и алфавит, и множество состояний занумерованными начальными отрезками натурального ряда:

$$A = \{a_0, a_1, \ldots, a_n\}, \quad Q = \{q_0, q_1, \ldots, q_k\},$$

причём считаем, что начальное состояние это q_0, а пустой символ $\Lambda = a_0$. В описании МТ достаточно указывать только номера символов и состояний, поскольку они будут различаться местоположением в описании.

Таблица переходов — это множество пятёрок вида

$(\langle$старый символ$\rangle, \langle$старое состояние$\rangle, \langle$новый символ$\rangle, \langle$новое состояние$\rangle, \langle$команда движения$\rangle)$.

В команде движения будем считать, что 0 отвечает сдвигу влево, 2 — сдвигу вправо, а 1 означает, что головка должна остаться в прежнем положении.

Для простоты дальнейших конструкций числа мы будем кодировать равномерно: записи всех символов имеют одинаковую длину:

$$\langle i \rangle = \underbrace{00\ldots0}_{i \text{ символов}}\,1\,\underbrace{00\ldots00000}_{n-i \text{ символов}}.$$

Строка таблицы переходов будет задаваться пятью кодами такого вида, разделёнными символом #.

Итак, описание $\langle M \rangle$ машины Тьюринга M в таком формате выглядит как набор записей вида

$$\#\langle a\rangle\#\langle q\rangle\#\langle a'\rangle\#\langle q'\rangle\#\langle d\rangle\# \ .$$

Входное слово кодируется аналогично как последовательность кодов входящих в него символов:

$$\#\langle a_1\rangle\#\langle a_2\rangle\#\langle a_3\rangle\#\ldots\#\langle a_n\rangle\# \ .$$

Пустое слово будем кодировать как $\#10^n\#$ (код пустого символа). Поэтому пустое слово и слово Λ из одного пустого символа получают один код. Однако это не приведёт к трудностям при моделировании работы МТ: пустое слово на ленте означает, что на ленте написаны одни пустые символы. То же самое содержимое ленты возможно описать по-другому: на ленте записано слово Λ. Мы будем пользоваться этим вторым вариантом.

Таким образом, для описания МТ и их входов мы используем алфавит $\{0, 1, \#\}$. В абстрактной теории алгоритмов мы рассматривали универсальную функцию $U(p, x)$ от пар чисел (программа вычисления функции одного аргумента, аргумент). Теперь нам удобнее рассматривать функции от пар слов в алфавите $\{0, 1, \#\}$, что не изменяет абстрактной теории благодаря существованию вычислимой биекции между словами и натуральными числами, как это уже объяснялось раньше.

Универсальной будем называть функцию $U\colon \{0,1,\#\}^* \times \{0,1,\#\}^* \to \{0,1,\#\}^*$, значение которой на входах $(\langle M\rangle, \langle x\rangle)$, где $\langle M\rangle$ — описание машины M, а $\langle x\rangle$ — описание входного слова x в алфавите этой машины, равно $\langle M(x)\rangle$ (или не определено, если M не останавливается на x).

Универсальная машина (УМТ) должна вычислять какую-нибудь универсальную функцию. Мы ещё не определили, как МТ вычисляет функции от нескольких аргументов. Наиболее естественный способ — разделять аргументы каким-нибудь специальным символом $\odot$.

Определение 16.6. МТ M *вычисляет* функцию $f\colon B^* \times B^* \to B^*$, если для каждой пары (u, v) из области определения функции f результат работы M на входе $u \odot v$ равен $f(u, v)$, а для каждой пары (u, v) не из области определения f машина M не останавливается на входе $u \odot v$.

Аналогично определяется вычисление функции от нескольких аргументов.

Замечание 16.2. Обратите внимание, что поведение универсальной функции определено только на входах, которые являются кодами машины и входного слова.

Если есть УМТ, которая вычисляет какую-нибудь универсальную функцию на таких входах, она также как-то работает и на остальных парах слов. Именно такая вычислимая функция и будет точным аналогом универсальной функции из абстрактной теории.

Другими словами, мы допускаем «нестандартное» кодирование некоторых алгоритмов. Это не создаёт проблему, хотя при желании можно построить вычислимые биекции между кодами машин и всеми словами в алфавите $\{0, 1, \#\}$, а также между кодами входных слов и всеми словами в алфавите $\{0, 1, \#\}$.

16.9 Универсальная 3-ленточная машина для 1-ленточных машин

Мы построим 3-ленточную машину, которая вычисляет универсальную функцию. Поскольку любую 3-ленточную машину можно преобразовать в 1-ленточную, получим отсюда существование искомой 1-ленточной универсальной МТ.

Для вычисления универсальной функции 3-ленточная УМТ U моделирует работу произвольной МТ на произвольном входе. Идея моделирования достаточно проста. На одной ленте (лента машины) машина U хранит описание моделируемой машины Тьюринга M, на второй (лента состояния) держит описание текущего состояния и текущего символа, на третьей (рабочая лента, она же входная и лента результата) поддерживает описание ленты моделируемой машины. УМТ начинает работу, приводя начальную конфигурацию к указанному виду. После этого работа машины M моделируется такт за тактом.

Опишем более подробно эти действия. Все они основаны на МТ, которые копируют или переносят часть одной ленты на другую, а также МТ, которые сравнивают слова на двух лентах.

Задача 16.9. Постройте (многоленточные) МТ, которые выполняют следующие действия.

(а) «Переход к указателю»: машина двигает головку до тех пор, пока головка не оказывается над заданным символом (указателем) #. В терминах конфигураций

это означает, что конфигурацию $\dots q_0 x \# \dots$ машина должна переводить в конфигурацию $\dots x q_1 \# \dots$. Здесь q_0, q_1 — состояния МТ, а x — слово, которое не содержит указателя.

(Будет также использоваться и переход влево к указателю.)

(б) «Копирование»: машина копирует содержимое области одной ленты на другую ленту. В терминах конфигураций это означает, что машина переводит конфигурацию

$$(\dots q_0 \triangleleft u \triangleright \dots \ , \ \dots q_0 \# \dots)$$

в конфигурацию

$$(\dots q_1 \triangleleft u \triangleright \dots \ , \ \dots q_1 \# u \dots).$$

Здесь q_0, q_1 — состояния МТ, символы-ограничители $\triangleleft$, $\triangleright$ задают область, которую нужно скопировать, $\#$ — указатель, а слово u не содержит ни указателя, ни ограничителей.

(Напомним, что конфигурация двухленточной машины — это пара слов.)

(в) «Перенос»: машина переносит содержимое области одной ленты на другую ленту. В терминах конфигураций это означает, что машина переводит конфигурацию

$$(\dots q_0 u \triangleright \dots \ , \ \dots q_0 \dots)$$

в конфигурацию

$$(\dots q_1 \triangleright \dots \ , \ \dots q_1 u \dots).$$

Здесь q_0, q_1 — состояния МТ, символы-ограничители $\triangleleft$, $\triangleright$ задают область, которую нужно скопировать, $\#$ — указатель, а слово u не содержит ограничителя $\triangleright$.

(г) «Сравнение слов»: машина сравнивает две области на двух лентах. В терминах конфигураций это означает, что машина переводит конфигурацию

$$(\dots q_0 \triangleleft u \triangleright \dots \ , \ \dots q_0 \triangleleft v \triangleright \dots)$$

в конфигурацию

$$(\dots q' \triangleleft u \triangleright \dots \ , \ \dots q' \triangleleft v \triangleright \dots),$$

где $q' = q_1$ если $u = v$, а в противном случае $q' = q_2$. Здесь q_0, q_1, q_2 — состояния МТ, символы-ограничители $\triangleleft$, $\triangleright$ задают сравниваемые области, а слова u, v не содержат ограничителей.

Указание: примените конструкции из примера 16.1 и задач 16.6, 16.7.

Кроме того универсальная машина должна уметь считать до пяти (строки таблицы переходов — это пятёрки). Мы не указываем явно расширенного множества состояний, которое позволяет это сделать, см. общие замечания об использовании «оперативной памяти» в МТ.

Универсальная машина получается последовательным соединением трёх машин: (1) подготовительной, (2) моделирующей такт работы, (3) финальной.

Подготовительная машина переносит с входной ленты на ленту описания МТ часть входа до символа $\odot$ (см. задачу 16.9в). Символ $\odot$ удаляется на первой ленте

и головка сдвигается вправо. Если описание машины пусто, то подготовительная машина останавливается без запуска машины, моделирующей такт работы.

Далее подготовительная машина помещает на ленту состояния код начального состояния 10^k. Для этого она использует второй код в первой строке таблицы переходов и машину из задачи 16.9в. Однако скопированное состояние может и не быть начальным (мы не оговаривали порядок строк в таблице переходов). Поэтому подготовительная машина пишет на первую позицию кода 1, а на все последующие позиции вплоть до символа # она записывает 0, чтобы на второй ленте и впрямь появился код начального состояния[5], после чего переходит влево до символа # (задача 16.9а). На ленте описания МТ подготовительная машина переводит головку в крайнее левое положение (т.е. переходит влево по указателю #).

На том работа подготовительной машины завершается.

Корректность работы машины очевидна из сделанного описания за одним исключением. Подготовительная машина берётся определить результат работы *пустой машины* — то есть такой МТ, таблица переходов которой не определена ни для одной пары (символ, состояние). Из построения видно, что результат оказывается равным входу. Это соответствует работе пустой машины — та останавливается на первом же такте и не сдвигает головку, поэтому вычисляет тождественную функцию.

Машина, моделирующая такт работы. Подготовительная машина заканчивает работу в такой конфигурации: на первой ленте (описания МТ) и второй ленте (описание текущего состояния) головки находятся над самой левой непустой ячейкой (содержащей разделитель #), а на третьей ленте головка находится над разделителем перед кодом символа в текущей ячейке машины M.

Моделирование такта работы машины M, описание которой лежит на ленте описания МТ, будет начинаться и заканчиваться именно в таких конфигурациях.

Моделирование такта работы состоит в выполнении следующих шагов, которые, как и выше, используют МТ из задачи 16.9 и переходы между разделителями на заданное число позиций (итерация МТ из задачи 16.9а нужное количество раз, которое не превосходит 5):

(1) Перейти вправо до # по ленте состояния и скопировать на эту ленту код текущего символа.

(2) Найти строчку таблицы переходов, которая соответствует текущему состоянию и символу моделируемой машины. Для этого последовательно проверять пятёрки кодов на ленте описания МТ (это строчки таблицы переходов моделируемой машины M), для каждой пятёрки первые два кода сравнить с текущим символом и текущим состоянием соответственно (задача 16.9г применительно к рабочей ленте и ленте состояния).

   – В случае совпадения искомая строчка найдена, перейти влево на первый символ # этой строчки.

[5]Построение такой вспомогательной машины оставляем читателю в качестве упражнения.

– В случае несовпадения перейти вправо на первый символ # следующей строчки.

Если совпадения не найдено, управление передаётся финальной машине. Иначе выполняются следующие действия:

(3) скопировать четвёртый код в пятёрке на ленту состояния;

(4) скопировать третий код в пятёрке на рабочую ленту;

(5) выполнить команду движения: переход влево или вправо до символа # на рабочей ленте; если сдвиг выходит за границы закодированного слова, т.е. машина встретила пустой символ машины U, то добавить на рабочую ленту код пустого символа аналогично тому, как это делает подготовительная машина с кодом начального состояния.

Чтобы убедиться в корректности такой машины (т.е. проверить, что конфигурация изменяется в соответствии с тактом работы машины M), достаточно сравнить формат описания МТ, данный выше с указанной последовательностью действий.

Финальная машина помечает текущую позицию, после чего ищет справа первый код пустого символа. Поскольку этот код 10^n, то сравнение символа с пустым выполняется без труда. Далее машина пишет пустой символ (машины U) и возвращается в помеченную исходную позицию и убирает метку.

Итак, мы построили 3-ленточную машину, вычисляющую универсальную функцию. Поскольку 3-ленточные машины моделируются на 1-ленточных, существует и 1-ленточная машина, вычисляющая универсальную функцию.

16.10 Соответствие между абстрактной теорией алгоритмов и МТ

Мы уже реализовали на машинах Тьюринга два самых важных свойства алгоритмов: доказали, что композиция вычислимых машинами Тьюринга функций вычислима и построили универсальную машину Тьюринга.

Но этого пока недостаточно, чтобы доказать неразрешимость следующей *проблемы остановки машины Тьюринга*: даны описание машины Тьюринга и её входа, нужно узнать, останавливается ли эта машина на этом входе.

Действительно, мы доказывали, что неразрешимо множество

$$H = \{n : U(n, n) \text{ определена}\}.$$

Сейчас под n нужно понимать слово в алфавите $\{0, 1, \#\}$. Заметьте, что построенная нами универсальная функция вполне возможно даёт какой-то результат и для слов, которые не являются описаниями МТ и их входов. Вдруг окажется так, что вся «трудность» диагонального множества H прячется именно в таких «паразитных» парах?

Контрольный вопрос 16.10. Какой результат даёт построенная выше УМТ на парах $\langle M\rangle \odot \langle x\rangle$, где алфавит M состоит из 10 символов, а слово x в алфавите из 5 символов?

Для преодоления этой трудности необходимо использовать эффективность выбранного нами способа кодирования МТ.

Задача 16.11. Докажите, что разрешимы по Тьюрингу[6] следующие множества слов:

(а) слова вида $\langle M\rangle$, где M — машина Тьюринга;

(б) слова вида $\langle x\rangle$, где x — слово в некотором алфавите;

(в) слова вида $\langle M\rangle \odot \langle x\rangle$, где M — машина Тьюринга, x — входное слово для M.

Разрешимость множества слов X означает, что существует МТ, которая даёт результат 1 на словах из X и результат 0 на остальных словах.

(См. решение в последнем разделе главы.)

Теорема 16.7. *Проблема остановки алгоритмически неразрешима.*

Доказательство. Используем результат предыдущей задачи. Подправим универсальную функцию: на паре (u, v) подправленная функция даёт результат $\langle M(x)\rangle$, если $u = \langle M\rangle$, $v = \langle x\rangle$, и не определена иначе. Вычислимость такой подправленной функции следует из разрешимости множества пар (описание МТ, описание входа МТ), при этом она остаётся универсальной.

Если бы проблема остановки МТ была разрешима, то было бы разрешимым и диагональное множество для такой подправленной функции: алгоритм разрешения на входе w проверяет, является ли $w \odot w$ описанием МТ и её входного слова; если да, то применяет алгоритм разрешения для проблемы остановки, а если нет, то говорит, что w не принадлежит диагональному множеству. □

Впрочем, неразрешимость проблемы остановки МТ можно доказать и без высокой теории, прямым применением диагонального рассуждения.

Задача 16.12. Выведите напрямую из тезиса Чёрча–Тьюринга неразрешимость проблемы остановки МТ (без использования существования УМТ или абстрактной теории алгоритмов).

(См. решение в последнем разделе главы.)

Перед решением этой задачи стоит обдумать следующее наблюдение. Наше определение универсальной машины страдает некоторым изъяном: не всегда можно подать на вход машины её собственное описание.

Этот изъян легко устранить дополнительным соглашением об описании машин Тьюринга. Будем считать в дальнейшем, что все МТ имеют алфавит из не менее чем 4 символов. Три символа из алфавита сопоставим символам 0, 1 и # алфавита, в котором записываются описания МТ, а четвёртый символ, разумеется, пустой.

Для машин с меньшим алфавитом будем кодировать их входы в бóльшем алфавите. При этом в таблице переходов не будет никаких команд по действиям с

[6] То есть, что характеристические функции этих множеств вычислимы по Тьюрингу.

символами, не входящими в исходный алфавит. Но это и нестрашно: нас ведь не интересует работа машины на словах, содержащих такие символы.

Если придерживаться такого соглашения, то на вход любой машины возможно подать её собственное описание. Это существенно для решения задачи 16.12 (вспомните обсуждение проблемы самоприменимости в главе 14).

Построенная универсальная функция — интерпретатор машин Тьюринга на машине Тьюринга — является главной универсальной функцией. Сформулируем это свойство развёрнуто и заодно напомним, в чём оно состоит.

Теорема 16.8. *Пусть* $U\colon \{0,1,\#\}^* \times \{0,1,\#\}^* \to \{0,1,\#\}^*$ *— универсальная функция для машин Тьюринга, то есть её значение на входах* $(\langle M\rangle, \langle x\rangle)$*, где* $\langle M\rangle$ *— описание машины* M*, а* $\langle x\rangle$ *— описание входного слова* x *в алфавите этой машины, равно* $\langle M(x)\rangle$ *(или не определено, если* M *не останавливается на* x*).*

Тогда для любой вычислимой по Тьюрингу функции $V\colon \{0,1,\#\}^* \times \{0,1,\#\}^* \to \{0,1,\#\}^*$ *существует такая всюду определённая вычислимая по Тьюрингу функция* $s\colon \{0,1,\#\}^* \to \{0,1,\#\}^*$*, что для всякого* $w \in \{0,1,\#\}$ *и для всякого* $x \in \{0,1,\#\}$ *верно* $U(s(w),x) = V(w,x)$*.*

Доказательство. Пусть функция двух аргументов $V(w,x)$ вычислима МТ M, т.е. на входе вида $w \odot x$ машина M даёт результат $V(n,x)$.

Наша цель: построить МТ, которая на входе w даёт в качестве результата описание МТ, вычисляющей функцию $f_w(x) = V(w,x)$.

Одной из машин, вычисляющих f_n, является такая: ко входу x она дописывает слева разделитель $\odot$, а за ним первый аргумент w; далее она выполняет действия машины, вычисляющей $V(n,x)$.

Описание этой машины легко строится по входу w. Предлагаем читателю восстановить технические детали этого рассуждения самостоятельно, решив следующую задачу. □

Задача 16.13. Докажите существование МТ, которая на входе w даёт в качестве результата описание МТ, вычисляющей функцию $f_w(x) = V(w,x)$. Здесь $V(w,x)$ — функция, вычислимая некоторой фиксированной машиной Тьюринга M.

(См. решение в последнем разделе главы.)

При изучении главных нумераций мы активно использовали биекцию между парами чисел и числами.

Задача 16.14. Докажите, что существует МТ, которая вычисляет биекцию между парами натуральных чисел и натуральными числами, заданную правилом

$$\mathrm{code}_2\colon (x,y) \mapsto \binom{x+y+1}{2} + y.$$

Числа заданы в унарной записи (т.е. n записывается как слово 1^n).

Как определить перечислимые множества, если в качестве алгоритмов используются машины Тьюринга? Для этого нужно определить машину Тьюринга, которая порождает список элементов множества. Будем считать, что такая перечисляющая машина имеет специальную ленту результата, куда записывает элементы списка через разделитель #. При этом перечисляющая машина сдвигает головку на ленте результата только вправо (такое ограничение легко выражается в терминах таблицы переходов).

Используя такое определение, можно заново доказать утверждения о перечисляющих алгоритмах из главы 15. Например, мы использовали в той главе возможность параллельного исполнения алгоритма одновременно на всех его входах.

Напомним, что такое исполнение разбивается на стадии: первая стадия состоит в исполнении первого такта на первом входе, вторая — второго такта на первом входе и первого такта на втором входе и т.д.: на N-й стадии исполняется N-й такт на первом входе, $(N-1)$-й такт на втором входе, ..., первый такт на N-м входе.

Задача 16.15. Докажите существование (многоленточной) МТ, которая реализует параллельное исполнение некоторой МТ на всех входах.

(См. решение в последнем разделе главы.)

Для решения этой задачи нужно каким-то образом занумеровать все входы, достаточно это сделать для кодировок входных слов в алфавите $\{0, 1, \#\}$.

Упорядочим слова в алфавите $\{0, 1, \#\}$ по правилу: более короткое слово меньше более длинного, а слова одинаковой длины упорядочиваются лексикографически. Этот порядок задаёт биекцию между словами и натуральными числами. Будем для краткости также называть его лексикографическим.

Мы будем нумеровать входы в этом порядке. Предлагаем самостоятельно построить алгоритм, реализующий такую нумерацию. Точнее, предлагаем решить следующую задачу и использовать её в решении задачи 16.15.

Задача 16.16. Докажите существование МТ, которая по слову w в алфавите $\{0, 1, \#\}$ находит следующее слово в лексикографическом порядке.

Мы описали способ перенести все утверждения, доказанные для абстрактных вычислимых функций на машины Тьюринга. Если довести эту работу до конца, получим как следствия такие теоремы о машинах Тьюринга:

Следствие 16.9 (теорема Райса)**.** *Для любого нетривиального свойства $\mathcal{A}$ функций алгоритмически неразрешима задача: по описанию МТ проверить, вычисляет ли эта МТ функцию, обладающую свойством $\mathcal{A}$.*

Следствие 16.10. *Неперечислимо множество описаний МТ, которые не останавливаются на любом входе.*

Следствие 16.11. *Существует МТ, которая на любом входе выдаёт в качестве результата своё описание.*

16.11 Машины Тьюринга в доказательствах неразрешимости

16.11.1 Задача достижимости на графе подстановок слов

Как вам известно, задача достижимости (можно ли из одной вершины попасть в другую, двигаясь по рёбрам) на конечном ориентированном графе разрешима. Более того, есть весьма эффективные алгоритмы её решения.

Подходящее обобщение этой задачи на бесконечные графы даёт уже алгоритмическую неразрешимость. Начнём с описания такого обобщения.

Здесь мы будем рассматривать бесконечные ориентированные графы, причём такие, что выходная и входная степень каждой вершины конечна.

Нужно договориться о способе задания графа и его вершин: для бесконечного графа это можно делать существенно различными способами.

Мы выберем такой способ задания (ор)графа. Множество вершин — это множество слов в некотором алфавите Σ. А рёбра задаются *правилами подстановки*. Каждое правило имеет вид

$$L \to R,$$

где L, R — слова в алфавите Σ. Из слова x ведёт ребро в слово y по правилу подстановки $L \to R$, если $x = uLv$, $y = uRv$.

Пример 16.17. Рассмотрим слова (последовательности букв) в русском алфавите. Тогда из слова `входная` ведёт ребро в слово `выходная` по правилу подстановки `ход` $\to$ `ыход`.

Таким образом, одно правило подстановки даёт бесконечное количество рёбер. Мы разрешим использовать несколько правил подстановки:

$$\begin{aligned} &L_1 \to R_1,\\ &L_2 \to R_2,\\ &\dots\\ &L_n \to R_n, \end{aligned}$$

и множество рёбер графа будет состоять из всех рёбер, отвечающих какому-то из этих правил.

Будем также использовать обозначение $u \underset{\mathcal{R}}{\longrightarrow} v$ для пары вершин нашего графа (т.е. слов), связанных одним из правил подстановки из множества правил подстановки $\mathcal{R} = \{L_i \to R_i\}$. Если из слова u можно попасть в слово v, двигаясь по рёбрам орграфа, будем указывать это как $u \overset{*}{\underset{\mathcal{R}}{\longrightarrow}} v$ (другими словами, это обозначение для отношения достижимости).

Задача достижимости. Задан граф на множестве слов в алфавите Σ набором правил подстановки $\mathcal{R} = \{L_i \to R_i\}$ и два слова $u, v \in \Sigma^*$. Верно ли, что $u \overset{*}{\underset{\mathcal{R}}{\longrightarrow}} v$?

В некоторых случаях решить задачу достижимости легко.

Пример 16.18. Пусть есть ровно одно правило подстановки в алфавите $\{a, b\}$:

$$ab \to ba.$$

Тогда верно, что $aaaabbaab \xrightarrow{*} baabaabaa$. (Количество букв a и b одинаково, а правило подстановки позволяет поменять местами любую пару различных букв.)

Пример 16.19. Пусть правила замены слов в алфавите $\{a, b\}$ имеют вид:

$$aba \to baabbb,$$
$$bbba \to a,$$
$$abab \to bbbb.$$

Тогда неверно, что $ababbbaa \xrightarrow{*} bbbaaba$. (Подстановки не меняют чётности количества букв a в слове.)

Однако общего «рецепта» решения задачи достижимости, т.е. алгоритма, нет.

Чтобы говорить о существовании или несуществовании алгоритма решения задачи достижимости, нужно представить условия задачи словом в конечном алфавите. Будем использовать те же правила, что и при описании машин Тьюринга. Каждый символ алфавита $i \in \Sigma = \{0, 1, \ldots, n\}$ кодируется двоичным словом

$$\langle i \rangle = \underbrace{00\ldots0}_{i \text{ символов}}\, 1\, \underbrace{00\ldots00000}_{n-i \text{ символов}},$$

слово $u = u_1 \ldots u_k$, $u_i \in \Sigma$, кодируется словом

$$\langle u \rangle = \#\langle u_1 \rangle \# \langle u_2 \rangle \# \ldots \# \langle u_k \rangle \#$$

в алфавите $\{0, 1, \#\}$. Правило подстановки $L \to R$ записывается как коды слов L и R, разделённые символом $\to$ и окружённые разделителями $\triangleleft$, $\triangleright$:

$$\triangleleft \langle L \rangle \to \langle R \rangle \triangleright .$$

Код набора правил $\mathcal{R} = \{L_i \to R_i\}$ — это просто запись всех правил подстановки одного за другим.

16.11.2 Неразрешимость задачи достижимости для графа подстановок слов

Теорема 16.12. *Не существует алгоритма, который по слову $\langle \mathcal{R} \rangle \odot \langle u \rangle \odot \langle v \rangle$, где $\mathcal{R}$ — набор правил подстановки, u, v — слова, выдавал бы результат 1, если $u \xrightarrow[\mathcal{R}]{*} v$, и 0 в противном случае.*

Доказательство. Используем неразрешимость проблемы остановки машины Тьюринга. Общий план доказательства такой: предположим, что существует алгоритм для решения задачи достижимости. Тогда мы построим алгоритм решения задачи остановки МТ следующего вида: алгоритм преобразует вход $\langle M \rangle \odot \langle x \rangle$ в описание

набора правил $\mathcal{R}$ и двух слов u, v; после этого вызывает алгоритм решения задачи достижимости и выдаёт в качестве результата результат работы этого алгоритма.

Чтобы описанный алгоритм был корректным, нужно выполнение такого условия: машина M останавливается на входе x тогда и только тогда, когда $u \xrightarrow[\mathcal{R}]{*} v$.

Мы сейчас опишем такое преобразование машины и её входа в набор правил подстановки и два слова, для которого это свойство выполняется.

Итак, пусть имеется машина Тьюринга $M = (A, Q, \delta, q_0, \Lambda)$ и её входное слово x. Заметим, что за один такт работы конфигурация машины изменяется незначительно — изменения затрагивают лишь небольшую окрестность символа из множества состояний Q, который обязательно присутствует в конфигурации.

Мы хотим описать это изменение как результат применения правил подстановки. При этом удобно применять правила подстановки к словам чуть более общего вида, чем конфигурации.

В качестве алфавита для графа подстановок слов возьмём множество

$$\Sigma = A \cup Q \cup \{\Lambda_0, f\}, \qquad \{\Lambda_0, f\} \cap (A \cup Q) = \varnothing, \quad A \cap Q = \varnothing.$$

Дополнительный символ Λ_0 будет использоваться для расширенного описания конфигураций. А именно, конфигурация uqv машины M будет представляться любым словом вида $\Lambda_0\Lambda^k uqv\Lambda^s\Lambda_0$. Другими словами, мы разрешаем иметь слева и справа от символов конфигурации произвольное количество пустых символов, окаймлённое дополнительным символом Λ_0.

Дополнительный символ f будет играть вспомогательную роль, которая станет ясна позднее.

Достижимость будет проверяться для пары слов $u = \Lambda_0 q_0 x \Lambda_0$ и $v = \Lambda_0 f \Lambda_0$.

Осталось описать набор правил подстановки Δ. Правила будут двух типов.

Правила первого типа описывают такт работы МТ. Чтобы их задать, нужно аккуратно рассмотреть все возможные случаи положения и движения головки на ленте и выписать правила преобразования конфигурации.

Для удобства описания расширим таблицу переходов на множество $(A \cup \{\Lambda_0\}) \times Q$ по правилу $\delta(\Lambda_0, q) = \delta(\Lambda, q)$ для любого $q \in Q$, это формальная запись такого свойства «с точки зрения таблицы переходов символ Λ_0 не отличается от пустого символа Λ».

Пусть $\delta(a, q) = (b, q', 0)$ и q не является финальным состоянием МТ. По такой строчке таблицы переходов добавим правила

$$qa \underset{\Delta}{\longrightarrow} q'b, \qquad a \neq \Lambda_0, \tag{16.1}$$

$$q\Lambda_0 \underset{\Delta}{\longrightarrow} q'b\Lambda_0. \tag{16.2}$$

Если $\delta(a, q) = (b, q', 1)$ и q не является финальным состоянием МТ, то добавим правила

$$qa \underset{\Delta}{\longrightarrow} bq', \qquad a \neq \Lambda_0, \tag{16.3}$$

$$q\Lambda_0 \underset{\Delta}{\longrightarrow} bq'\Lambda_0. \tag{16.4}$$

Наконец, если $\delta(a, q) = (b, q', -1)$ и q не является финальным состоянием МТ, то добавим правила

$$xqa \underset{\Delta}{\longrightarrow} q'xb, \qquad \text{для всех } x \in A,\ a \neq \Lambda_0, \tag{16.5}$$

$$\Lambda_0 qa \underset{\Delta}{\longrightarrow} \Lambda_0 q' \Lambda b, \qquad a \neq \Lambda_0, \tag{16.6}$$

$$xq\Lambda_0 \underset{\Delta}{\longrightarrow} q'xb\Lambda_0, \qquad \text{для всех } x \in A, \tag{16.7}$$

$$\Lambda_0 q \Lambda_0 \underset{\Delta}{\longrightarrow} \Lambda_0 q' \Lambda b \Lambda_0. \tag{16.8}$$

Правила второго типа задают преобразования слов после окончания работы МТ.

Для каждой пары (a, q), $a \in A \cup \{\Lambda_0\}$, на которой таблица переходов не определена или q является финальным состоянием МТ, добавим правило

$$qa \underset{\Delta}{\longrightarrow} fa. \tag{16.9}$$

Добавим также правила

$$af \underset{\Delta}{\longrightarrow} f, \qquad \text{для всех } a \in A, \tag{16.10}$$

$$fa \underset{\Delta}{\longrightarrow} f, \qquad \text{для всех } a \in A. \tag{16.11}$$

На этом построение набора правил закончено. Проверим, что построенные Δ, u, v удовлетворяют сформулированному выше требованию: машина M останавливается на x тогда и только тогда, когда $\Lambda_0 q_0 x \Lambda_0 \overset{*}{\underset{\Delta}{\longrightarrow}} \Lambda_0 f \Lambda_0$.

Прежде всего заметим, что к слову вида

$$\Lambda_0 \Lambda^k w \Lambda^s \Lambda_0,$$

где w — конфигурация МТ, применимо не более одного правила первого типа, и его применение даёт слово такого же вида

$$\Lambda_0 \Lambda^{k'} w' \Lambda^{s'} \Lambda_0,$$

где w' — конфигурация после такта работы машины M на конфигурации w. (Для каждого правила (16.1–16.8) это утверждение проверяется непосредственным применением правила выполнения такта работы МТ и определения конфигурации.)

Значит, если машина Тьюринга M не останавливается на входе x, то преобразованиями по данной системе правил будут получаться слова, в которые обязательно входит символ из Q, т.е. слово $\Lambda_0 f \Lambda_0$ получить из начального слова невозможно.

Пусть машина Тьюринга M останавливается на входе x. Заметим, что возможна ровно одна подстановка для слова вида $\Lambda_0 \Lambda^k w \Lambda^s \Lambda_0$, где w — конфигурация МТ. Поэтому возможно выполнение подстановок до тех пор, пока не будет достигнута финальная конфигурация. В финальной конфигурации применимо правило (16.9). После этого применением правил (16.10, 16.11) возможно убрать из слова все символы алфавита A. Останется в точности искомое слово $\Lambda_0 f \Lambda_0$.

Описание правил подстановки было вполне конструктивным. Поэтому легко поверить, что существует алгоритм, который по описанию машины Тьюринга M и входа x строит описание Δ и слов u, v. Если говорить о (многоленточной) МТ, решающей такую задачу, нужно разбить её построение на несколько шагов:

– Определение размера алфавита Σ и построение МТ, которая преобразует код символа в описании M в код соответствующего символа в алфавите Σ.

– Преобразование описания x в описание слов $u = \Lambda_0 q_0 x \Lambda_0$ и $v = \Lambda_0 f \Lambda_0$.

– Запись описания правил первого типа по каждой строчке таблицы переходов.

– Перечисление всех пар (a, q), для которых таблица переходов не определена и запись правил второго типа, отвечающих таким парам.

– Запись остальных правил второго типа, для этого нужно перечислить все символы алфавита A.

□

Описанное преобразование МТ и её входа в набор правил подстановки обладает дополнительными интересными свойствами. Скажем, легко заметить, что правила зависят только от машины, а не от входа. Вспомнив о существовании универсальной машины, получаем такое следствие.

Следствие 16.13. *Существует такой граф подстановок, что задача достижимости для этого графа алгоритмически неразрешима.*

Неориентированные графы также можно задавать правилами подстановки. Но в этом случае подстановки двусторонние: можно не только заменять левую часть правила на правую, но и наоборот: правую на левую.

Задача 16.20. Докажите, что задача достижимости для неориентированных графов, заданных правилами подстановки, алгоритмически неразрешима.

Подсказка: используйте то же самое преобразование, что и в доказательстве теоремы, и докажите, что даже при двусторонних заменах слово v достижимо из слова u тогда и только тогда, когда машина M останавливается на входе x.

16.12 Решения задач

Задача 16.4. Докажите, что не существует МТ с таким «волшебным состоянием» q_m, что любую конфигурацию, содержащую q_m, машина переводит в пустую конфигурацию q_f, где q_f — финальное состояние.

Доказательство. Допустим, что такая машина существует. Тогда, получив на вход конфигурацию $q_m 0$, она совершает некоторое конечное число шагов t и переходит в пустую конфигурацию q_f. Покажем, что получив на вход конфигурацию $0\Lambda^t q_m 0$, МТ

переведёт её в конфигурацию $0\Lambda^n q_f$, где n — некоторое число. Стартуя из конфигурации $0\Lambda^t q_m 0$, МТ будет совершать те же переходы, что совершала бы, стартовав из $q_m 0$, поскольку за t шагов МТ не доберётся до нуля, стоящего в $(t+1)$-й клетке справа от головки, а содержимое остальных клеток одинаковое. А значит, ноль так и останется стоять в той клетке, а МТ при этом перейдёт в состояние q_f. □

Задача 16.6. Постройте двухленточную машину, которая (а) копирует с одной ленты на другую символы от текущего положения головки до ближайшего справа символа-разделителя #, скопированное слово на второй ленте начинается с первоначального положения головки на ней; (б) переносит указанные символы (аналогично предыдущему, но на первой ленте символы заменяется на пустые).

Доказательство. Построим обе МТ, задав их таблицы переходов, см. таблицы 16.5 и 16.6.

a_1	a_2	q	a_1'	a_2'	q'	d_1	d_2
0	0	q_C	0	0	q_C	+1	+1
0	1	q_C	0	0	q_C	+1	+1
0	Λ	q_C	0	0	q_C	+1	+1
0	#	q_C	0	0	q_C	+1	+1
1	0	q_C	1	1	q_C	+1	+1
1	1	q_C	1	1	q_C	+1	+1
1	Λ	q_C	1	1	q_C	+1	+1
1	#	q_C	1	1	q_C	+1	+1
Λ	0	q_C	Λ	Λ	q_C	+1	+1
Λ	1	q_C	Λ	Λ	q_C	+1	+1
Λ	Λ	q_C	Λ	Λ	q_C	+1	+1
Λ	#	q_C	Λ	Λ	q_C	+1	+1

Таблица 16.5: Таблица переходов МТ для задачи 16.4(а)

Докажем корректность построения. Мы полагаем, что в алфавите МТ четыре символа: $\Lambda, 0, 1, \#$. Рассмотрим первые четыре строки таблицы, отвечающие за копирование символа 0. Какой бы ни был символ на второй ленте, МТ в состоянии q_C заменит его на 0 и сдвинет обе головки вправо. Аналогичные рассуждения справедливы для символов 1 и Λ, за копирование которых отвечают по четыре следующие строки соответственно. Встретив символ # на первой ленте в состоянии q_C, МТ останавливает процесс копирования, поскольку переход из q_C по # не определён. Если

копирование требовалось использовать как подзадачу, можно определить переходы из q_C по # для последующих нужд.

a_1	a_2	q	a_1'	a_2'	q'	d_1	d_2
0	0	q_M	Λ	0	q_M	$+1$	$+1$
0	1	q_M	Λ	0	q_M	$+1$	$+1$
0	Λ	q_M	Λ	0	q_M	$+1$	$+1$
0	#	q_M	Λ	0	q_M	$+1$	$+1$
1	0	q_M	Λ	1	q_M	$+1$	$+1$
1	1	q_M	Λ	1	q_M	$+1$	$+1$
1	Λ	q_M	Λ	1	q_M	$+1$	$+1$
1	#	q_M	Λ	1	q_M	$+1$	$+1$
Λ	0	q_M	Λ	Λ	q_M	$+1$	$+1$
Λ	1	q_M	Λ	Λ	q_M	$+1$	$+1$
Λ	Λ	q_M	Λ	Λ	q_M	$+1$	$+1$
Λ	#	q_M	Λ	Λ	q_M	$+1$	$+1$

Таблица 16.6: Таблица переходов МТ для задачи 16.4(б)

Для второго пункта задачи доказательство корректности аналогично. □

Задача 16.7. Постройте двухленточную машину, которая сравнивает слова на двух лентах от текущего положения головок до символа-разделителя. В случае равенства слов машина заканчивает работу в состоянии q_y, в случае неравенства — в состоянии q_n.

Доказательство. Построим МТ, задав её таблицу переходов, см. таблицу 16.7.

Докажем корректность построения. Первые три строки таблицы переходов отвечают за проверку символов на совпадение. МТ начинает работу из состояния q_E. Если под головками на обеих лентах стоят одинаковые символы, то машина остаётся в состоянии q_E (и сдвигает обе головки вправо).

Заметим, что во всех остальных случаях машина изменяет состояние q_E. Поэтому следующая, четвёртая, строка в таблице выполняется лишь в том случае, когда головки на обеих лентах достигли разделителя одновременно, причём символы под головками были одинаковыми на всех предыдущих шагах. Это означает, что сравниваемые слова одинаковы. А машина переходит в состояние q_y, как и требовалось.

Остальные $4^2 - 4 = 12$ строк таблицы описывают все 12 возможностей, когда головки машины находятся над разными символами. В каждом таком случае сравниваемые слова различны. Машина переходит в состояние q_n, как и требовалось. □

a_1	a_2	q	a'_1	a'_2	q'	d_1	d_2
0	0	q_E	0	0	q_E	$+1$	$+1$
1	1	q_E	1	1	q_E	$+1$	$+1$
Λ	Λ	q_E	Λ	Λ	q_E	$+1$	$+1$
#	#	q_E	#	#	q_y	0	0
#	0	q_E	#	0	q_y	0	0
#	1	q_E	#	1	q_y	0	0
#	Λ	q_E	#	Λ	q_y	0	0
0	1	q_E	0	1	q_n	0	0
0	Λ	q_E	0	Λ	q_n	0	0
0	#	q_E	0	#	q_n	0	0
1	0	q_E	1	0	q_n	0	0
1	Λ	q_E	1	Λ	q_n	0	0
1	#	q_E	1	#	q_n	0	0
Λ	0	q_E	Λ	0	q_n	0	0
Λ	1	q_E	Λ	1	q_n	0	0
Λ	#	q_E	Λ	#	q_n	0	0

Таблица 16.7: Таблица переходов для МТ, реализующей сравнение.

Контрольный вопрос 16.24. Укажите номера строк в таблице 16.7, которыми завершается сравнение слов разной длины.

Задача 16.11. Докажите, что разрешимы по Тьюрингу следующие множества слов:
(а) слова вида $\langle M \rangle$, где M — машина Тьюринга;
(б) слова вида $\langle x \rangle$, где x — слово в некотором алфавите;
(в) слова вида $\langle M \rangle \odot \langle x \rangle$, где M — машина Тьюринга, x — входное слово для M.
Разрешимость множества слов X означает, что существует МТ, которая даёт результат 1 на словах из X и результат 0 на остальных словах.

Доказательство. Приведём решения для пунктов (а) и (б) и оставим читателю в качестве упражнение их комбинирование для решения пункта (в). Начнём с пункта (б).

Трудность тут в том, что размер алфавита для слова x, хотя и конечен, но может быть произвольно велик. Символы алфавита ограниченного размера, скажем, 2029, возможно поместить в «оперативную память» машины (в разделе 16.4 описано, что

это означает) и сделать задачу намного легче. Однако для построения универсальной машины Тьюринга требуется обрабатывать алфавиты сколь угодно большого размера.

Напомним в чём состоит описание $\langle x \rangle$ входного слова x над алфавитом $A_n = \{a_0, a_1, \ldots, a_{n-1}\}$. Код i-й буквы алфавита имеет вид

$$\langle i \rangle = \underbrace{00\ldots0}_{i} 1 \underbrace{00\ldots0}_{n-i}.$$

Таким образом, троичный алфавит A_3 имеет коды букв $\langle 0 \rangle = 100$, $\langle 1 \rangle = 010$, $\langle 2 \rangle = 001$.

Пусть $x = x_1 x_2 \ldots x_m$, где x_k — k-й символ слова x. Тогда код $\langle x \rangle$ представляет собой посимвольное кодирование каждой буквы с разделителем # между кодами букв:

$$\langle x \rangle = \#\langle x_1 \rangle \# \langle x_2 \rangle \# \ldots \# \langle x_k \rangle \# \ldots \# \langle x_m \rangle \#.$$

Задача состоит в том, чтобы проверить по слову w, состоящему из символов $0, 1, \#$, является ли это слово кодом некоторого слова x над алфавитом (заранее неоговоренного) размера n.

Сформулируем условия, которые необходимы и достаточны для того, чтобы слово w являлось кодом.

1. Слово w начинается и заканчивается символом-разделителем #.
2. Между любыми двумя соседними разделителями # находится одинаковое ненулевое количество символов.
3. Между любыми двумя соседними разделителями # находится ровно одна единица.

Контрольный вопрос 16.26. Почему эти условия выполняются для кода пустого слова?

Будем доказывать существование МТ, которые проверяют каждое из этих условий в том смысле, что в конце работы переходят в состояние q_y, если проверяемое условие выполнено, и в состояние $q_n \neq q_y$, если условие не выполнено. Преобразовать такую машину к нашему стандартному виду (результат на ленте равен 1 или 0), очень легко.

Контрольный вопрос 16.27. Постройте МТ, которая на любом входе даёт результат 1.

МТ для проверки первого условия на первом такте работы проверяет, что головка находится над символом #. После этого головка двигается слева направо до первого пустого символа (см. пример 16.1, если непонятно, что это означает). После этого головка сдвигается на один шаг влево и машина проверяет, что под головкой

стоит символ #. После этого машина останавливается в состоянии q_y, если обе проверки дали положительный результат, и в состоянии q_n во всех остальных случаях. Корректность работы этой машины очевидна.

МТ для проверки третьего условия перемещает головку между блоками $\#\ldots\#$, а внутри каждого блока двигается слева направо и подсчитывает количество встреченных единиц. Результат подсчёта хранится в «оперативной памяти», поэтому он не может принимать сколь угодно большие значения. Договоримся, что когда счётчик принимает значение 2, он уже не изменяется до конца движения по блоку.

Достигнув границы блока, то есть разделителя #, машина действует в зависимости от значения счётчика.

Если оно равно 1, то машина сбрасывает счётчик (устанавливает его значение 0) и выполняет ту же процедуру для следующего блока. Если головка оказывается над пустым символом Λ, то машина завершает работу с положительным результатом.

Если значение счётчика после обработки некоторого блока равно 0 или 2, то МТ завершает работу с отрицательным результатом.

Доказать существование МТ, проверяющей условие 2, сложнее. Тут удобно использовать двухленточные МТ и теорему об эквивалентности многоленточных МТ одноленточным.

Двухленточная МТ копирует первый блок на вторую ленту и проверяет при копировании, что длина блока не ноль (достаточно счётчика с двумя состояниями: 0 и $\geqslant 1$). После чего МТ возвращает головку второй ленты на первый разделитель #, а далее синхронно двигает головки по обеим лентам вперёд (после копирования головка на первой ленте как раз находится над вторым разделителем прямо перед вторым блоком). Как только на второй ленте достигнут #, МТ проверяет находится ли # под головкой на первой ленте. Если нет, то заканчивает работу с отрицательным результатом, а если да, то перемещает головку второй ленты на первый разделитель и продолжает проверять совпадение длины следующего блока с первым. В случае если совпадение было полным, пока на первой ленте головка не дошла до пустого символа Λ, МТ завершает работу с положительным результатом.

Перейдём к пункту (а). Описание $\langle M\rangle$ МТ M задано кодировкой таблицы переходов: подряд идущими словами вида

$$\#\langle a\rangle\#\langle q\rangle\#\langle a'\rangle\#\langle q'\rangle\#\langle d\rangle\#,$$

каждое из которых соответствует записи $\delta(a,q) = (a',q',d)$ в таблице переходов. Множество состояний конечно и кодируется той же кодировкой, что и алфавитные символы. То же относится и к кодировке переходов — каждый переход кодируется как элемент трёхэлементного множества. Напомним, что при кодировке разделители не накладываются: в кодовом слове нет двух разделителей подряд.

Таким образом, чтобы проверить, что слово w над алфавитом $0,1,\#$ является кодировкой некоторой МТ, необходимо и достаточно проверить, что w начинается и заканчивается на #, а также что при разбиении w на пятёрки по разделителям

$$\#w_1^1\#w_2^1\#w_3^1\#w_4^1\#w_5^1\#\ldots\#w_1^i\#w_2^i\#w_3^i\#w_4^i\#w_5^i\#\ldots\#w_1^m\#w_2^m\#w_3^m\#w_4^m\#w_5^m\#$$

- первые и третьи компоненты всех пятёрок являются кодами символов одного и того же алфавита;
- вторые и четвёртые компоненты всех пятёрок являются кодами символов одного и того же алфавита;
- пятые компоненты всех пятёрок являются кодами символов троичного алфавита A_3;
- при $i \neq j$ не выполняются одновременно оба равенства $w_1^i = w_1^j$ и $w_2^i = w_2^j$.

В пояснениях нуждается только последнее условие. Мы предполагаем, что в таблице переходов есть не более одной строки для каждой пары аргументов (a, q). Вообще-то, это необязательно. Но обычно при записи множеств элементы указываются без повторений и мы придерживаемся того же правила.

Проверка корректности кодировок осуществима аналогично первому пункту задачи. Для этого можно на дополнительную ленту скопировать соответствующие компоненты всех пятёрок, а затем проверить корректность кодировок тем же способом, что и в первом пункте. Проверить отсутствие одинаковых пар можно, используя дополнительную ленту, на которую при проходе входной ленты слева направо будут записываться компоненты $\#w_1^i\#w_2^i\#$, при этом при переходе к j-ой пятёрке на входной ленте головка на дополнительной ленте перемещается на начало и идёт проверка на совпадение $\#w_1^j\#w_2^j\#$ со всеми парами на дополнительной ленте. □

Задача 16.12. Выведите напрямую из тезиса Чёрча–Тьюринга неразрешимость проблемы остановки МТ (без использования существования УМТ или абстрактной теории алгоритмов).

Доказательство. Предположим, что проблема остановки МТ разрешима. Тогда тезис Чёрча–Тьюринга обещает, что существует МТ H, которая на входе $\langle M\rangle \odot \langle x\rangle$ даёт результат 1, если МТ M при работе на входе x останавливается, и результат 0 в противном случае.

Построим по машине H другую машину F так, чтобы обмануть алгоритм, решающий проблему остановки. Машина F получает на вход описание МТ M и запускает машину H на входе $\langle M\rangle \odot \langle\langle M\rangle\rangle$ (вторая компонента — это описание слова в троичном алфавите, которое является описанием машины M, в этом месте мы используем уточнение об описании машин с маленькими алфавитами, сделанное после формулировки данной задачи в основном тексте). Если машина H выдаёт результат 1, то машина F не останавливается, а если результат 0, то останавливается.

Единственный не вполне очевидный момент в этом построении, это преобразование описания M в описание её описания. Такая задача легко решается на двухленточной МТ (прочитав символ из множества $\{\#, 0, 1\}$, пишем на вторую ленту его код). По теореме об эквивалентности многоленточных и одноленточных машин Тьюринга, есть и одноленточная машина, выполняющая такое преобразование.

Что же теперь будет, если запустить машину F на своём собственном описании $\langle F\rangle$? Машина F запустит машину для проверки остановки H на входе $\langle F\rangle \odot \langle\langle F\rangle\rangle$.

Если эта машина H выдаст результат 1, то это означает по определению H, что F останавливается на своём описании. Но, получив от вызова H результат 1, F не останавливается по правилам своей работы. Значит, результат 1 работы машины проверки остановки на данном входе невозможен.

Если машина проверки остановки выдаст результат 0, то по определению машины остановки это означает, что F не останавливается на своём описании. Но, получив от вызова H результат 0, F останавливается по правилам своей работы. Значит, результат 0 работы машины проверки остановки на данном входе невозможен.

Но других результатов работы машины H не предусмотрено. Пришли к противоречию, которое показывает, что машина H не существует. □

Задача 16.13. Докажите существование МТ, которая на входе w даёт в качестве результата описание МТ, вычисляющей функцию $f_w(x) = V(w, x)$. Здесь $V(w, x)$ — функция, вычислимая некоторой фиксированной МТ M.

Доказательство. Как уже говорилось выше, нужно по входу w построить описание машины, которая пишет разделитель $\odot$ слева от своего входного слова, а потом слово w символ за символом в обратном порядке; после чего вызывает исполнение машины M на построенном таким образом входе.

Искомая машина M_w будет иметь $n+2+m$ состояний, где n — длина слова w, m — количество состояний машины M. Алфавит машины M_w состоит из $k+1$ символа, где k символов — это символы алфавита машины M, а дополнительный символ $\odot$ нужен как разделитель между парой аргументов во входе машины M.

Состояния от $n+2$ до $n+m+1$ мы отождествляем с состояниями машины M, причём состояние $n+2$ отождествляем с начальным состоянием M. Заметим, что само по себе описание машины M является константой для машины M_w. Однако при выбранном нами правиле (чуть более удобном в другом отношении), придётся эту константу подправлять, изменяя подходящим образом номера состояний (прибавляя к ним $n+2$).

Для краткости мы указываем только номера состояний и символов алфавита. Это не приведёт к путанице, так как в решении данной задачи мы не используем конфигурации МТ (для которых только и важно, чтобы алфавит и множество состояний МТ не переескались).

В состоянии 0 (начальном) машина сдвигается влево, не изменяя символа под головкой, и переходит в состояние 1.

Этому правилу отвечает в таблице переходов k строк вида

$$(i, 0, i, 1, -1), \quad 0 \leqslant i < k$$

(записываем строки пятёрками, как они будут входить в описание M_w).

В состоянии 1 машина пишет символ $\odot$, сдвигается влево и переходит в состояние 2. Этому правилу отвечает в таблице переходов машины M_w строка вида

$$(0, 1, k+1, 2, -1),$$

мы считаем, что код разделителя $k+1$.

В состоянии $1 < i < n+1$ машина пишет на ленту символ w_{n+2-i} (i-й от правого конца символ слова w) и сдвигается влево, переходя в состояние $i+1$. Этому правилу отвечает в таблице переходов $n-1$ строка вида

$$(0, i, w_{n+2-i}, i+1, -1), \quad 1 < i < n+1.$$

В состоянии $n+1$ машина пишет на ленту символ w_1, не перемещает головку и переходит в состояние $n+2$, которое мы отождествили с начальным состоянием машины M. Этому правилу отвечает в таблице переходов машины M_w строка вида

$$(0, n+1, w_1, n+2, 0).$$

Осталось понять, как устроена машина G, которая по слову w порождает такой список пятёрок, то есть описание машины M_w. Мы построим многоленточную машину, осуществляющую такое преобразование. Далее нужно воспользоваться теоремой об эквивалентности многоленточных и одноленточных машин. Машина G состоит из нескольких блоков.

Прежде всего G должна определить длину слова w и каким-то образом сохранить её для дальнейших вычислений. На самом деле, нужно число $n+2+m$ (длина кода состояния M_w).

Поэтому первый блок G_1 копирует со входной ленты на вторую слово w, заменяя все его символы символами 0 (из алфавита описания МТ). Когда слово w заканчивается, головка на входной ленте остаётся на месте (первый пустой символ после символов слова w), а на второй ленте дописывается $m+2$ символа 0. Поскольку m константа для данной машины, это нетрудно сделать, заведя для каждого очередного действия отдельное состояние.

Таким образом, на второй ленте хранится «заготовка» для записи состояния машины M_w в очередную пятёрку. Запись осуществляется копированием этой заготовки на ленту результата (это третья лента МТ G) и вставкой 1 в подходящем месте.

Второй блок G_2 пишет на ленту результате пятёрки, которые не зависят от символов слова w и от машины M. Таких пятёрок $k+1$ и их коды легко порождаются из строки длины $n+m+2$, состоящей из одних нулей (нужно вставить 1 на небольшом расстоянии от левого края). Заметим также, что размер алфавита машины M, который мы обозначили k, является константой для машины G и ей нетрудно написать в пятёрке код каждого конкретного символа алфавита.

Следующий блок G_3 пишет на ленту результата те пятёрки, которые не зависят от символов слова w, но зависят от машины M. Как уже говорилось, количество таких пятёрок — константа для машины G. Более того, коды символов, отвечающих состояниям машины M в машине M_w, то есть коды чисел от от $n+2$ до $n+m+1$, легко построить из нулевой строки длины $n+m+2$, вставляя 1 на небольшом расстоянии от правого края.

Наконец, блок G_4 пишет пятёрки, зависящие от символов слова w. Для этого машина обрабатывает символы слова на входной ленте справа налево (напомним,

что блок G_1 оставил головку на входной ленте справа от слова, за один такт можно переместить её на первый символ слова и далее действовать в цикле). Теперь удобно менять в цикле «заготовку» на второй ленте, переписывая 1 на каждой итерации цикла в следующую слева позицию.

Цикл заканчивается, когда под головкой на входной ленте стоит пустой символ.

Обратите внимание, что такой цикл неправильно записывает последнюю пятёрку. В ней стоит код команды движения -1, а должен стоять код команды движения 0. Последний блок G_5 исправляет эту ошибку, после чего двигается к левому концу списка пятёрок на ленте результата. (Это нужно, чтобы соблюсти наше формальное требование к определению результата работы МТ.) □

Задача 16.15. Докажите существование (многоленточной) МТ, которая реализует параллельное исполнение некоторой МТ на всех входах.

Доказательство. Мы предполагаем, что вычислима по Тьюрингу функция, которая по слову в алфавите $\{0, 1, \#\}$ находит следующее слово в лексикографическом порядке.

Мы также используем обозначение $\underline{\mathrm{Lex}}(n)$ для n-го в лексикографическом порядке слова.

Опишем как получить искомую МТ, модифицировав универсальную МТ. Мы назовём эту машину параллельно исполняющей машиной Тьюринга — ПМТ. Сначала мы опишем общий план, а потом перейдём к его реализации.

Добавим к алфавиту УМТ разделители $\{\lhd, \rhd\}$. Мы изменим ленту состояний УМТ так, что вместо описания одного состояния $\langle q\rangle$ она будет содержать описание конечного набора состояний через разделители

$$\lhd \langle q_\Lambda\rangle \rhd \lhd\langle q_0\rangle \rhd \lhd\langle q_1\rangle \rhd \ldots \lhd \langle q_{\underline{\mathrm{Lex}}(n)}\rangle \rhd . \tag{16.12}$$

Здесь q_w — состояние МТ, обрабатывающей вход w. Аналогично модифицируем рабочую ленту: вместо содержимого одной рабочей ленты мы будем поддерживать набор из содержимого нескольких рабочих лент:

$$\lhd w_\Lambda \rhd \lhd w_0 \rhd \lhd w_1 \rhd \ldots \lhd w_{\underline{\mathrm{Lex}}(n)} \rhd \lhd w_{\underline{\mathrm{Lex}}(n+1)} \rhd . \tag{16.13}$$

Лента с описанием машины остаётся без изменений. В начале работы ПМТ получает описание МТ на эту ленту и начинает параллельно её запускать на всех входах.

Опишем теперь схему работы ПМТ, начнём с инициализации. Обозначим через s начальное состояние МТ M, описание которой подаётся на вход ПМТ. ПМТ записывает на ленту состояний $\lhd\langle s\rangle\rhd$, а на рабочую ленту $\lhd\langle\Lambda\rangle \rhd \lhd\langle 0\rangle\rhd$.

Перейдём к описанию шага работы ПМТ. Лента состояний и рабочая лента в начале шага выглядит как описано в формулах (16.12) и (16.13) соответственно. При этом на рабочей ленте блоков на один больше, чем состояний в списке, и

$$w_{\underline{\mathrm{Lex}}(n+1)} = \underline{\mathrm{Lex}}(n+1).$$

ПМТ в начале шага перемещает головки ленты состояний и рабочей ленты до правого конца и дописывает к их содержимому блоки $\triangleleft\langle s\rangle\triangleright$ и $\triangleleft\langle\underline{\mathrm{Lex}}(n+2)\rangle\triangleright$; последнее реализуется нахождением следующего в лексикографическом порядке слова за $\underline{\mathrm{Lex}}(n+1)$, которое записано в последнем блоке. После добавления новых блоков, ПМТ перемещает головки обеих лент на левый конец и исполняет последовательно шаг работы УМТ для каждой пары блоков состояния и содержимого рабочей ленты, начиная с $\triangleleft\langle q_\Lambda\rangle\triangleright$, $\triangleleft w_\Lambda\triangleright$ и заканчивая $\triangleleft\langle q_{\underline{\mathrm{Lex}}(n+1)}\rangle\triangleright$, $\triangleleft w_{\underline{\mathrm{Lex}}(n+1)}\triangleright$.

Таким образом, на каждом шаге работы ПМТ добавляет новый вход и начинает исполнение на нём МТ M на входе, делая при этом по шагу для каждого предыдущего входа. И поскольку входы добавляются в лексикографическом порядке, то ПМТ в итоге исполняет работу M параллельно на всех входах. $\square$

Литература

[1] Н. Алон, Дж. Спенсер. Вероятностный метод. — М.: Бином, 2007.

[2] Б. Л. ван дер Варден. Алгебра. — М.: Наука, 1976.

[3] Н. К. Верещагин, А. Шень. Лекции по математической логике и теории алгоритмов. Начала теории множеств. — Изд. 4е. — М.: МЦНМО, 2012.

[4] Н. К. Верещагин, А. Шень. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. — Изд. 4е. — М.: МЦНМО, 2012.

[5] Э. Б. Винберг. Курс алгебры. — М.: Факториал, 1999.

[6] А. Л. Городенцев. Алгебра. Учебник для студентов-математиков. Часть 1. — М.: МЦНМО, 2013.

[7] Р. Грэхем, Д. Кнут и О. Паташник. Конкретная математика. — М.: Мир, 1998.

[8] С. Дасгупта, Х. Пападимитриу, У. Вазирани. Алгоритмы. — М.: МЦНМО, 2014.

[9] А. К Звонкин. Малыши и математика. Домашний кружок для дошкольников. — М.: МЦНМО, 2006,

[10] Д. Кнут. Искусство программирования для ЭВМ. Том 1. Основные алгоритмы. Том 2. Получисленные алгоритмы. Том 3. Сортировка и поиск. — М.: Мир, 1977.

[11] Т. Кормен, Ч. Лейзерсон, Р. Ривест. Построение и анализ алгоритмов. М.: МЦНМО, 2000.

[12] А. И. Кострикин. Введение в алгебру. — М.: Наука, 1977.

[13] А. И. Кострикин. Введение в алгебру. Основы алгебры. — М.: Физматлит, 1994.

[14] А. Г. Курош. Курс высшей алгебры. — М.: Наука, 1975.

[15] С. К. Ландо. Лекции о производящих функциях. — М.: МЦНМО, 2002.

[16] С. Ленг. Алгебра. — М.: Мир, 1968.

[17] Р. Стенли. Перечислительная комбинаторика. — М.: Мир, 1990.