

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Белгородский государственный технологический университет
им. В.Г. Шухова

Ю.Д. Рязанов

ДИСКРЕТНАЯ МАТЕМАТИКА

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Московский физико-технический институт (государственный университет)» рекомендует данное учебное издание к публикации

*Регистрационный номер рецензии 3060 от «29» июня 2015 г.
МГУП имени Ивана Федорова*

Белгород
2016

УДК 519.1
ББК 22.12
Р99

Рецензенты:

Доктор физико-математических наук, профессор кафедры программного обеспечения вычислительной техники и автоматизированных систем Белгородского государственного технологического университета им. В.Г. Шухова *А.Г. Брусенцев*

Доктор физико-математических наук, профессор кафедры информатики и информационных технологий Белгородского государственного аграрного университета имени В.Я. Горина *В.А. Ломазов*

Рязанов, Ю.Д.

Р99 Дискретная математика: учеб. пособие. / Ю.Д. Рязанов. — 2-е изд., перераб. и доп. — Белгород: Изд-во БГТУ, 2016. — 298 с.

ISBN 978-5-361-00108-8

В учебном пособии рассмотрены вопросы пяти разделов, изучаемых в курсе дискретной математики: теории множеств, комбинаторных объектов, отношений, графов и булевых функций. Особое внимание уделяется применению метода поиска с возвратом для решения различных задач дискретной математики. Пособие содержит теоретические сведения, задания для самостоятельной работы и перечень контрольных вопросов.

Содержание учебного пособия соответствует основным разделам дисциплины “Дискретная математика” и предназначено для студентов обучающихся по направлению 230100 “Информатика и вычислительная техника”

УДК 519.1
ББК 22.12

ISBN 978-5-361-00108-8

© Белгородский государственный
технологический университет
(БГТУ) им. В.Г. Шухова, 2016

Оглавление

Предисловие ко второму изданию	7
Введение к первому изданию	8
1. Множества	12
1.1. Основные понятия	12
1.2. Способы задания множеств	13
1.3. Операции над множествами	13
1.4. Свойства операций над множествами	16
1.5. Нормальные формы Кантора	18
1.6. Преобразование теоретико-множественного выражения в нормальную форму Кантора	20
1.7. Методы доказательства теоретико-множественных тождеств	22
1.7.1. Метод двух включений	22
1.7.2. Метод эквивалентных преобразований	23
1.7.3. Метод характеристических функций	24
Арифметические характеристические функции	25
Логические характеристические функции	26
1.7.4. Теоретико-множественный метод	27
1.7.5. Метод симметрической разности	29
1.8. Теоретико-множественные уравнения	30
1.9. Способы представления множества в памяти ЭВМ	32
1.10. Алгоритмы реализации операций над множествами	32
Практическое занятие 1.1. Операции над множествами.....	47
Практическое занятие 1.2. Нормальные формы Кантора	50
Практическое занятие 1.3. Теоретико-множественные тождества	51
Практическое занятие 1.4. Теоретико-множественные уравнения	53
Контрольные вопросы и задания	57
2. Комбинаторные объекты.....	58
2.1. Введение.....	58
2.2. Метод поиска с возвращением.....	58
2.3. Подмножества.....	61
2.4. Перестановки.....	64
2.5. Размещения.....	67
2.6. Размещения с повторениями.....	70
2.7. Сочетания.....	73
2.8. Перестановки с повторениями.....	76
2.9. Сочетания с повторениями.....	79

2.10. Упорядоченные разбиения множества.....	82
2.10.1. Упорядоченные разбиения множества M на k подмножеств с мощностями (n_1, n_2, \dots, n_k)	82
2.10.2. Упорядоченные разбиения множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$	84
2.10.3. Все упорядоченные разбиения множества M на k подмножеств.....	86
2.10.4. Все упорядоченные разбиения множества M	88
2.11. Разбиения множества.....	89
2.11.1. Разбиения множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$	89
2.11.2. Все разбиения множества M на k подмножеств.....	93
2.11.3. Все разбиения множества M	100
2.12. Использование алгоритмов порождения комбинаторных объектов при проектировании полнопереборных алгоритмов решения задач выбора.....	102
2.13. О неэффективности полнопереборных алгоритмов. Пример.....	105
Практическое занятие 2.1. Алгоритмы порождения комбинаторных объектов.....	110
Практическое занятие 2.2. Разбиения множеств.....	111
Практическое занятие 2.3. Задачи выбора.....	113
Контрольные вопросы и задания.....	116
3. Отношения.....	119
3.1. Основные понятия.....	119
3.2. Способы задания отношений.....	121
3.3. Операции над отношениями.....	122
3.4. Алгоритмы реализации операций над отношениями.....	125
3.5. Свойства отношений.....	131
3.6. Замыкание отношений.....	134
3.7. Классы эквивалентности и фактормножества.....	138
3.8. Упорядоченные множества.....	139
Практическое занятие 3.1. Отношения и их свойства.....	143
Практическое занятие 3.2. Транзитивное замыкание отношения.....	146
Практическое занятие 3.3. Фактормножества.....	147
Практическое занятие 3.4. Упорядоченные множества.....	148
Контрольные вопросы и задания.....	149
4. Графы.....	151
4.1. Основные понятия.....	151
4.2. Матричные способы представления графов.....	154
4.3. Изоморфизм графов.....	156

4.4. Маршруты.....	157
4.5. Метрические характеристики графа.....	163
4.6. Циклы.....	164
4.7. Связность.....	168
4.8. Деревья.....	170
4.9. Покрывающие деревья графа.....	173
4.10. Покрывающие деревья минимальной стоимости.....	178
4.11. Поиск в орграфе.....	182
4.12. Связность в орграфе. Компоненты сильной связности.....	188
4.13. Кратчайший путь в орграфе.....	193
4.14. Кратчайшие пути между каждой парой вершин в орграфе.....	201
4.15. Центры и медианы во взвешенном орграфе.....	203
4.16. Клики и независимые множества.....	206
4.17. Раскраска графа.....	215
Практическое занятие 4.1. Маршруты	219
Практическое занятие 4.2. Циклы.....	235
Практическое занятие 4.3. Связность.....	236
Практическое занятие 4.4. Анализ алгоритмов построения покрывающего дерева минимальной стоимости.....	237
Практическое занятие 4.5. Анализ алгоритмов поиска кратчайшего пути во взвешенном орграфе между заданной парой вершин.....	238
Практическое занятие 4.6. Кратчайшие пути во взвешенном орграфе.....	239
Практическое занятие 4.7. Анализ алгоритмов поиска кратчайших путей во взвешенном орграфе между каждой парой вершин.....	241
Практическое занятие 4.8. Кратчайшие пути между каждой парой вершин во взвешенном орграфе.....	242
Контрольные вопросы и задания.....	244
5. Булевые функции.....	246
5.1. Табличные способы задания булевых функций.....	246
5.2. Элементарные булевые функции.....	247
5.3. Функциональная полнота систем булевых функций.....	249
5.4. Аналитические способы задания булевых функций.....	251
5.5. Основные законы булевой алгебры и их следствия.....	255
5.6. Минимизация полностью определенных булевых функций.....	257
5.6.1. Основные понятия.....	257
5.6.2. Получение сокращенной ДНФ булевой функции.....	259
Метод Квайна.....	259
Метод Квайна — Мак-Класки.....	260
Метод Нельсона.....	261

5.6.3. Получение минимальной ДНФ булевой функции.....	261
Метод Петрика.....	262
Эвристический метод.....	263
5.6.4. Скобочная минимизация булевых функций.....	263
5.7. Минимизация частично определенных булевых функций.....	265
5.8. Минимизация систем полностью определенных булевых функций.....	268
5.9. Минимизация систем частично определенных булевых функций.....	271
5.10. Графовые способы задания булевых функций.....	276
5.10.1. Бинарные деревья и бинарные графы	276
5.10.2. Синтаксические деревья	281
5.11. Программная реализация булевых функций	284
5.11.1. Вычисление значения булевой функции по таблице истинности.....	284
5.11.2. Вычисление значения булевой функции по ДНФ.....	284
5.11.3. Вычисление значения булевой функции по бинарному графу.....	285
5.11.4. Вычисление значения булевой функции по синтаксическому дереву.....	287
Практическое занятие 5.1. Полностью определенные булевые функции.....	289
Практическое занятие 5.2. Частично определенные булевые функции.....	289
Практическое занятие 5.3. Минимизация систем полностью определенных булевых функций.....	291
Практическое занятие 5.4. Вычисление систем булевых функций.....	292
Контрольные вопросы и задания.....	293
Заключение.....	295
Библиографический список.....	296

Предисловие ко второму изданию

Учебное пособие «Дискретная математика» (издание первое) было издано в 2010 году и активно использовалось в учебном процессе. Со-вершенствование учебного процесса послужило поводом к переработке некоторого материала пособия и добавлению нового материала.

В первую главу добавлен материал по представлению множеств в нормальных формах Кантора и соответствующее практическое занятие. Переработан материал по теоретико-множественным тождествам. Изложение теоретико-множественного метода доказательства тождеств стало более понятным и очевидным. Методы арифметических и логических характеристических функций теперь описаны в одном стиле. Дополнен метод эквивалентных преобразований. Показан способ доказательства тождества этим методом с использованием совершенной нормальной формы Кантора, не требующий угадывания хода доказательства. Добавлен материал по решению теоретико-множественных уравнений, который практически не встречается в учебной литературе и в сети Интернет.

Третья глава пополнилась примерами выполнения операций над реальными, имеющими смысл отношениями, в результате чего получаются отношения, имеющие другой смысл, что позволило приблизить теорию к реальности и сделать изложение материала более понятным. Добавлена информация об упорядоченных множествах.

В пятую главу добавлен графовый способ задания булевых функций — синтаксическое дерево. Представлены правила построения синтаксического дерева по аналитической форме булевой функции и алгоритм вычисления значения функции по дереву. Алгоритмы построения синтаксических деревьев обычно рассматриваются при изучении вопросов трансляции языков программирования.

Исправлены обнаруженные опечатки, дополнен библиографический список.

Введение к первому изданию

Учебное пособие предназначено для студентов технических вузов, изучающих дискретную математику и программирование и состоит из пяти глав: “Множества”, “Комбинаторные объекты”, “Отношения”, “Графы” и “Булевые функции”. В каждой главе содержатся теоретические сведения, задания для самостоятельной работы в виде практических занятий и перечень контрольных вопросов.

В первой главе даны основные понятия теории множеств, рассмотрены различные методы доказательства теоретико-множественных тождеств. Представлены алгоритмы реализации операций над множествами при различных способах их хранения в памяти ЭВМ. Показана зависимость структуры алгоритмов реализации операций над множествами от способа хранения множества в памяти ЭВМ.

Во второй главе рассматриваются основные комбинаторные объекты: подмножества, перестановки (без повторений и с повторениями), размещения (без повторений и с повторениями), сочетания (без повторений и с повторениями) и различные виды разбиений множества. При рассмотрении каждого типа объектов дается его определение, формулируется и доказывается теорема о количестве различных комбинаторных объектов и приводится алгоритм порождения всех объектов. Для порождения комбинаторных объектов различных типов используется один и тот же метод — метод поиска с возвращением, поэтому алгоритмы порождения всех типов объектов имеют одинаковую структуру.

Далее рассматриваются вопросы использования комбинаторных объектов и алгоритмов их порождения при проектировании полнопереборных алгоритмов решения дискретных задач выбора. Описаны этапы проектирования полнопереборных алгоритмов с использованием комбинаторных объектов и показано, как преобразовать алгоритм порождения комбинаторных объектов в алгоритм решения задачи. Приведены примеры проектирования полнопереборных алгоритмов. Показана простота проектирования и неэффективность (с точки зрения времени выполнения) полнопереборных алгоритмов. Приводится эффективный алгоритм решения одной задачи выбора.

В третьей главе рассматриваются бинарные отношения на множестве. Определены операции над отношениями и свойства отношений. Представлены алгоритмы реализации операций, нахождения транзитивного замыкания и фактормножества по заданному отношению эквивалентности.

В четвертой главе даны основные понятия теории графов, графические и матричные представления графов. Определено понятие маршрута

в графе и его разновидности: различные цепи и циклы. Представлены алгоритмы получения всех маршрутов, обладающих определенными свойствами: маршруты заданной длины, простые цепи и циклы, гамильтоновы и эйлеровы циклы. Эти алгоритмы основаны на методе поиска с возвращением и поэтому все они имеют одинаковую структуру, что наглядно представлено блок-схемами алгоритмов.

Далее рассматривается понятие связность графа. Определено отношение связности вершин и способы его вычисления, основанные на методе поиска с возвращением и нахождения транзитивного замыкания. Показано, что отношение связности вершин обладает свойством эквивалентности, а классы эквивалентности представляют собой связные компоненты графа.

Затем рассматривается особый подкласс графов — деревья и определяются их свойства. Рассматриваются покрывающие деревья графов и алгоритм Краскала построения покрывающего леса. Для построения всех покрывающих деревьев представлен алгоритм, основанный на методе поиска с возвращением. Для нахождения покрывающих деревьев минимальной стоимости представлены алгоритмы Краскала и Прима, даны рекомендации по программной реализации этих алгоритмов. Рассмотрены задачи, решение которых сводится к нахождению связных компонент и покрывающих деревьев минимальной стоимости.

Потом описываются алгоритмы поиска в орграфе и примеры задач, решение которых сводится к поиску в орграфе.

Затем рассматривается связность в орграфе. Определяются различные виды связности: сильная, односторонняя и слабая, и компоненты сильной связности. Определяются отношения достижимости, контрудостигимости и взаимодостижимости на множестве вершин орграфа. Представлены различные способы нахождения отношения достижимости: с использованием алгоритмов поиска в орграфе и вычисления транзитивного замыкания отношения смежности вершин орграфа. Показано, что отношение взаимодостижимости обладает свойством эквивалентности, а классы эквивалентности представляют собой сильносвязные компоненты. Даются определения понятиям конденсация, база и антибаза и приводятся задачи, для решения которых используются эти понятия.

Затем рассматриваются кратчайшие пути во взвешенном орграфе. Для нахождения кратчайшего пути от одной заданной вершины до другой рассматривается возможность применения алгоритма перебора простых цепей, основанного на методе поиска с возвращением, и его усовершенствование с использованием идеи метода ветвей и границ. Описан алгоритм Дейкстры нахождения кратчайших путей и даны рекомендации по его программной реализации. Для нахождения кратчайших

путей между каждой парой вершин взвешенного орграфа описан способ применения алгоритма Дейкстры, алгоритм Шимбелла и алгоритм Флойда. Даются определения центров и медиан взвешенных орграфов. Приводятся примеры задач, решение которых сводится к нахождению центров и медиан взвешенного орграфа.

Далее рассматриваются клики и независимые множества графов. Представлен алгоритм поиска всех максимальных клик, основанный на методе поиска с возвращением, показаны его недостатки и более эффективный алгоритм Брана — Кербоша. Представлены различные эвристические алгоритмы поиска наибольшей клики. Определена связь клик и независимых множеств и показано, как алгоритмы поиска клик использовать для поиска независимых множеств. Приводятся примеры задач, решение которых сводится к поиску клик и независимых множеств.

Заканчивается глава определением понятия раскраска графа. Приводятся алгоритмы, основанные на методе поиска с возвращением, позволяющие получить все или одну правильную раскраску графа не более чем в заданное число цветов. Показано, как эти алгоритмы применить для нахождения минимальной раскраски графа. Приводятся примеры задач, решение которых сводится к нахождению минимальной раскраски графа.

В пятой главе рассматриваются различные способы представления булевых функций, методы минимизации полностью и частично определенных булевых функций и их систем и алгоритмы вычисления булевых функций при различных способах представления.

В пособии все рассматриваемые задачи дискретной математики доводятся до алгоритмов их решения. Для решения многих задач используются общие методы, например, метод поиска с возвращением, метод перемножения матриц и т.д. Это позволяет строить алгоритмы одинаковой структуры для решения различных задач. Степень детализации алгоритмов выбрана такой, что даже “сложные” алгоритмы представлены в простой и доступной для понимания форме. Выбор структур данных для реализации алгоритмов отходит на второй план и оставлен для самостоятельной проработки, но, все же, рассматриваются способы хранения объектов дискретной математики в памяти ЭВМ и даются рекомендации по их использованию в программной реализации алгоритмов.

В пособии приведено много примеров реальных задач, решение которых сводится к решению рассмотренных задач дискретной математики.

Практические занятия носят творческий и исследовательский характер. В одних предлагается грамотно применить рассмотренные алгоритмы для решения конкретных задач, в других — выполнить сравнительный анализ различных алгоритмов решения одной и той же задачи.

Контрольные вопросы, приведенные в конце каждой главы, охватывают весь изложенный материал.

Предполагается, что изучающие материал пособия владеют основами алгоритмизации и программирования на каком-либо языке высокого уровня. Никаких предварительных знаний по дискретной математики не требуется. Все вводимые понятия детально поясняются на примерах.

При самостоятельном изучении рекомендуется читать материал последовательно, так как в каждой главе, за исключением первой, используются понятия и алгоритмы, рассмотренные в предыдущих главах.

В конце пособия приведен небольшой библиографический список. В качестве основной литературы рекомендуются классические труды и современные издания. Много дополнительной литературы можно найти в сети Интернет.

1. МНОЖЕСТВА

1.1. Основные понятия

Создатель теории множеств Г. Кантор определил *понятие множества и элемента множества* следующим образом: "Под множеством мы понимаем собрание определенных отличных друг от друга объектов (реальных или воображаемых), называемых элементами множества, в их общности". Обычно множества обозначают прописными буквами латинского алфавита, а их элементы — строчными буквами или числами.

Количество элементов в множестве A называется *мощностью множества A* и обозначается $|A|$. Если каждому элементу множества A можно поставить в соответствие единственный элемент множества B и каждому элементу множества B можно поставить в соответствие единственный элемент множества A , то множества A и B называются *равномощными* и обозначаются $|A| = |B|$.

Множество, состоящее из конечного числа элементов, называется *конечным*.

Множество, не являющееся конечным, называется *бесконечным*. Бесконечное множество называется *счетным*, если оно равномощно множеству N всех натуральных чисел. Говорят, что все элементы счетного множества можно пронумеровать. В противном случае бесконечное множество называется *несчетным*. Кантор доказал, что множество точек, расположенных на отрезке между 0 и 1, несчетно. Счетными являются, например, множества четных и нечетных чисел, множество целых решений неравенства $x > 5$. Множество же действительных решений неравенства $x > 5$ является несчетным. Мощность любого счетного множества меньше мощности несчетного. (Счетными являются множества целых Z и натуральных N чисел, поэтому они равномощны $|Z| = |N|$.)

Множество, не содержащее элементов, называется *пустым* и обозначается \emptyset или $\{ \}$.

Обычно в конкретных рассуждениях элементы всех множеств берутся из некоторого одного, достаточно широкого множества U (своего для каждого случая), которое называется *универсумом*.

Два множества A и B называются *равными*, если они состоят из одних и тех же элементов ($A = B$).

Если M — множество, а a — его элемент, то a *принадлежит* M ($a \in M$). Если же a не есть элемент множества M , то a *не принадлежит* M ($a \notin M$).

1.2. Способы задания множеств

1. *Перечислением всех элементов.*

$A = \{a, b, c\}$, $B = \{b, a, c\}$, $C = \{a\}$, $D = \{1, 2, 3, 5, 9\}$. Из определения равенства множеств вытекает, что $A = B$.

2. Заданием *характеристического свойства*, выделяющего элементы данного множества среди элементов указанного или указанных других множеств.

$$A = \{x / x \in N \text{ и } x < 10\} \quad (N - \text{множество натуральных чисел}), \\ B = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}, \quad A = B.$$

3. Описанием *порождающей процедуры* с указанием множества (или множеств), которое “пробегает” параметр (или параметры) процедуры.

$$A = \{x^2 / x \in N\} — \text{множество всех квадратов натуральных чисел.}$$

Перечислением можно задать только конечное множество, а с помощью характеристического свойства или порождающей процедуры — конечное или бесконечное множество.

1.3. Операции над множествами

1. *Включение* A в B ($A \subseteq B$ или $B \supseteq A$) истинно, если каждый элемент множества A принадлежит множеству B . В этом случае A называется *подмножеством* B , а B — *надмножеством* A . Множества A и B *равны* ($A = B$), если $A \subseteq B$ и $B \subseteq A$. Пустое множество является подмножеством любого множества. Универсум является надмножеством любого множества. Множество всех подмножеств множества M называется *булеаном* и обозначается 2^M :

$$2^M = \{A / A \subseteq M\}$$

Для конечного множества M $|2^M| = 2^{|M|}$. Каждому подмножеству A можно сопоставить $|M|$ -разрядный двоичный вектор C , в котором: $C_i = 1$, если $i \in A$ и $C_i = 0$, если $i \notin A$, где i — элемент множества M , C_i — i -й разряд вектора C . Количество различных $|M|$ -разрядных двоичных векторов равно $2^{|M|}$, следовательно $|2^M| = 2^{|M|}$.

2. *Строгое включение* A в B ($A \subset B$ или $B \supset A$) истинно, если $A \subseteq B$ и $A \neq B$. Если $A \neq \emptyset$ и $A \subset B$, то A есть *собственное подмножество* B .

3. *Объединение A и B* ($A \cup B$) есть множество, состоящее из всех тех и только тех элементов, которые принадлежат A или B , т.е.

$$A \cup B = \{x / x \in A \text{ или } x \in B\}.$$

4. *Пересечение A и B* ($A \cap B$) есть множество, состоящее из всех тех и только тех элементов, которые принадлежат каждому из множеств A и B , т.е.

$$A \cap B = \{x / x \in A \text{ и } x \in B\}.$$

5. *Разность A и B* ($A - B$) есть множество, состоящее из всех тех и только тех элементов множества A , которые не принадлежат множеству B , т.е.

$$A - B = \{x / x \in A \text{ и } x \notin B\}.$$

Для обозначения разности множеств в дискретной математике обычно используют символ “\”.

6. *Симметрическая разность A и B* ($A \Delta B$) есть множество, состоящее из всех тех и только тех элементов множества A , которые не принадлежат множеству B и только тех элементов множества B , которые не принадлежат множеству A , т.е.

$$A \Delta B = \{x / x \in A \text{ и } x \notin B \text{ или } x \in B \text{ и } x \notin A\}.$$

$$A \Delta B = (A \cup B) - (A \cap B) = (A - B) \cup (B - A).$$

7. *Дополнение A до универсума U* (\overline{A}) есть множество, состоящее из всех тех и только тех элементов универсума U , которые не принадлежат множеству A , т.е.

$$\overline{A} = \{x / x \notin A\}.$$

$$\overline{A} = U - A.$$

Для любых двух множеств A и B имеет место хотя бы один из следующих пяти случаев (возможностей):

1) $A = B$, ($|A| = |B|$) А равно B;

2) $A \subset B$, ($|A| < |B|$) А строго включено в B;

3) $B \subset A$, ($|B| < |A|$) B строго включено в A;

4) $A \cap B = \emptyset$, ($|A \cap B| = 0$ и $|A \cup B| = |A| + |B|$) А и B не пересекаются;

5) $A \cap B \neq \emptyset$ и $A \neq B$, ($|A \cup B| < |A| + |B|$ и $|A| < |A \cup B| > |B|$)

А и B в общем положении.

Если оба множества A и B не пусты, то имеет место только один случай.

На рис.1.1 приведены *диаграммы Эйлера* (Эйлер (1707—1783) предложил изображать множества в виде кругов еще до создания теории множеств Кантором (1845—1918)), иллюстрирующие операции над множествами. Множества изображаются фигурами (овалами), а результат выделяется графически.

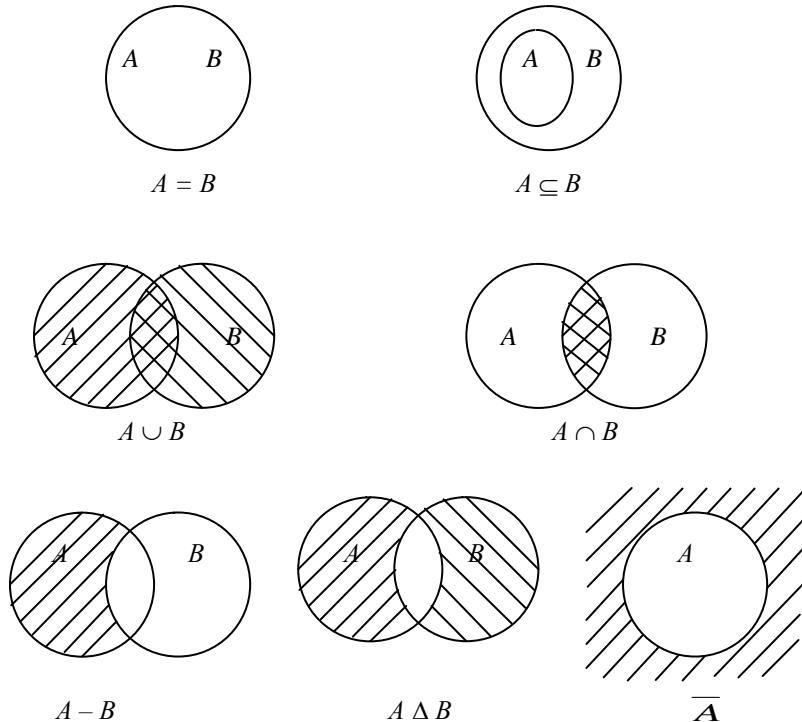


Рис.1.1. Операции над множествами

Выражение, составленное из некоторых, исходных, подмножеств универсума U и операций над множествами, называется *теоретико-множественным выражением*. Исходные подмножества обозначаются либо именами, либо конкретными значениями. Значением теоретико-множественного выражения является подмножество универсума U . Теоретико-множественное выражение может содержать скобки, определяющие порядок выполнения операций. Если скобки не указаны, то порядок выполнения операций определяется их приоритетами (табл.1.1). Последовательность операций одного приоритета выполняется слева направо.

Таблица 1.1
Приоритеты операций над множествами

Название операции	Обозначение	Приоритет
Дополнение A	\overline{A}	1
Пересечение A и B	$A \cap B$	2
Объединение A и B	$A \cup B$	3
Разность A и B	$A - B$	3
Симметрическая разность A и B	$A \Delta B$	3
Равенство A и B	$A = B$	4
Включение A в B	$A \subseteq B$ или $B \supseteq A$	4
Строгое включение A в B	$A \subset B$ или $B \supset A$	4

1.4. Свойства операций над множествами

1. Идемпотентность:

$$A \cup A = A \quad A \cap A = A$$

2. Коммутативность:

$$A \cup B = B \cup A \quad A \cap B = B \cap A \quad A \Delta B = B \Delta A$$

3. Ассоциативность:

$$A \cup (B \cup C) = (A \cup B) \cup C \quad A \cap (B \cap C) = (A \cap B) \cap C \\ A \Delta (B \Delta C) = (A \Delta B) \Delta C$$

4. Дистрибутивность для пересечения и объединения:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \\ A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

5. Дистрибутивность для пересечения и разности:

$$A \cap (B - C) = (A \cap B) - (A \cap C) \\ (A \cap B) - C = (A - C) \cap (B - C)$$

6. Дистрибутивность для пересечения и симметрической разности:

$$A \cap (B \Delta C) = (A \cap B) \Delta (A \cap C)$$

7. Дистрибутивность для разности:

$$(A - B) - C = (A - C) - (B - C) \\ (A - (B - C) \text{ не всегда равно } (A - B) - (A - C))$$

8. Законы поглощения:

$$(A \cap B) \cup A = A \quad (A - B) \cup A = A \quad (A \cup B) \cap A = A$$

9. Законы склеивания:

$$A \cap \overline{B} \cup A \cap B = A$$

$$(A \cup \overline{B}) \cap (A \cup B) = A$$

10. Свойства нуля:

$$A \cup \emptyset = A$$

$$A \cap \emptyset = \emptyset$$

$$A - \emptyset = A$$

$$\emptyset - A = \emptyset$$

$$A \Delta \emptyset = A$$

11. Свойства единицы:

$$A \cup U = U$$

$$A \cap U = A$$

$$A - U = \emptyset$$

$$U - A = \overline{A}$$

$$A \Delta U = \overline{A}$$

12. Инволютивность: $\overline{\overline{A}} = A$

13. Законы де Моргана:

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

14. Законы де Моргана для разности, пересечения и объединения:

$$A - (B \cup C) = (A - B) \cap (A - C)$$

$$A - (B \cap C) = (A - B) \cup (A - C)$$

15. Свойства дополнения:

$$A \cup \overline{A} = U$$

$$A \cap \overline{A} = \emptyset$$

$$A \Delta \overline{A} = U$$

$$A - \overline{A} = A$$

16. Определения разности:

$$A - B = A \cap \overline{B} = (A \cap B) \Delta A$$

17. Определения симметрической разности:

$$A \Delta B = (A \cup B) - (A \cap B) = (A - B) \cup (B - A) = (A \cap \overline{B}) \cup (B \cap \overline{A})$$

18. Определения объединения:

$$A \cup B = (A \Delta B) \Delta (A \cap B) = (A - B) \Delta B = (B - A) \Delta A$$

19. Определения пересечения:

$$A \cap B = (A \cup B) - (A \Delta B) = A - (A - B) = B - (B - A)$$

20. Разложение Шеннона:

$$f(X, A_1, \dots, A_n) = \overline{X} \cap f(\emptyset, A_1, \dots, A_n) \cup X \cap f(U, A_1, \dots, A_n),$$

где $f(X, A_1, \dots, A_n)$ — теоретико-множественное выражение, содержащее множества X, A_1, \dots, A_n ; $f(\emptyset, A_1, \dots, A_n)$ — выражение, получаемое из $f(X, A_1, \dots, A_n)$ путем подстановки вместо множества X пустого множества \emptyset ; $f(U, A_1, \dots, A_n)$ — выражение, получаемое из $f(X, A_1, \dots, A_n)$ путем подстановки вместо множества X универсума U .

1.5. Нормальные формы Кантора

Множество M может быть определено исходя из других, *порождающих* множеств M_1, M_2, \dots, M_n с помощью теоретико-множественного выражения. Выражения M_i и \overline{M}_i называют *первичными термами*. Пересечение попарно различных первичных термов называют *элементарным*. Выражение, задающее множество M в виде объединения различных элементарных пересечений, называется *нормальной формой Кантора (НФК) множества M* . Любое множество, которое может быть получено из порождающих множеств с помощью операций над множествами, может быть представлено в НФК.

Элементарное пересечение, содержащее все порождающие множества, называется *конституентой*. Любое множество, которое может быть получено из порождающих множеств с помощью операций над множествами, может быть представлено объединением некоторого количества различных конституент. Такое представление множества называется *совершенной НФК*.

Количество вхождений первичных термов в выражение, задающее множество, определяет *сложность представления множества. Минимальной НФК множества M* называется НФК, имеющая минимальную сложность по отношению ко всем другим НФК этого же множества. Количество вхождений первичных термов в элементарное пересечение называется *рангом* пересечения.

Элементарное пересечение, определяющее подмножество множества M , не являющееся надмножеством никакого другого подмножества множества M , которое может быть задано элементарным пересечением, называется *простой импликантой*. Можно получить все простые импликанты, используя совершенную НФК. Для этого необходимо выполнить все возможные склеивания конституент, а затем элементарных пересечений меньшего ранга, пока это возможно. Элементарные пересечения, не участвовавшие в склеивании, являются простыми импликантами. Для сокращения количества проверок на склеивание можно использовать следующий способ.

Каждую конституенту представим двоичным вектором, в котором i -й разряд равен 1, если конституента содержит множество M_i , и равен 0, если конституента содержит дополнение множества M_i , а совершенную НФК — в виде объединения двоичных векторов. Например, совершенная НФК $A \cap \bar{B} \cap \bar{C} \cup \bar{A} \cap B \cap C \cup \bar{A} \cap \bar{B} \cap \bar{C} \cup A \cap B \cap \bar{C} \cup A \cap B \cap C$ примет вид $100 \cup 011 \cup 000 \cup 110 \cup 111$. Конституенты, представленные двоичными векторами, разобьем на группы. В i -ю группу включаем двоичные вектора, содержащие i единиц. Результат запишем в таблицу:

Номер группы			
0	1	2	3
000	100	011 110	111

Проверяем на возможность склеивания только конституенты из соседних групп. Конституенты склеиваются, если соответствующие им двоичные вектора различаются только одним i -м разрядом. Результат склеивания — вектор, в котором i -й разряд заменяется прочерком. Конституенты, участвовавшие в склеивании, отмечаем крестиком справа, а результат склеивания дописываем в соответствующую группу:

Номер группы			
0	1	2	3
000+	100+	011+ 100+	111+
-00	1-0	-11 11-	

Аналогично проверяем на возможность склеивания в полученной части таблицы. Склеиваются вектора, различающиеся только в одном разряде (с учетом прочерка). В данном случае склеивания не возможны и все простые импликанты получены — они соответствуют неотмеченным крестиком векторам. Если в i -м разряде прочерк, то множество M_i не записывается в простую импликанту. Объединение всех простых импликант образует *сокращенную НФК*: $\bar{B} \cap \bar{C} \cup A \cap \bar{C} \cup B \cap C \cup A \cap B$.

Сокращенная НФК может содержать простые импликанты, исключение которых не изменяет определяемого множества. Такие простые импликанты называются *лишними*. Если из сокращенной НФК исключить все лишние простые импликанты, то получим *туникую НФК*. Туниковых НФК может быть несколько. Для получения туниковой НФК используют импликантную матрицу Квайна, в которой строки соответствуют простым импликантам, а столбцы — конституентам. Клетка импликантной матрицы на пересечении строки i и столбца j отмечается крестиком, если i -я импликанта покрывает j -ю конституенту, т.е. является ее составной частью. Импликантная матрица Квайна для рассматриваемого примера представлена в табл. 1.2. Туниковой НФК соответствует *минимальное покрытие столбцов строками* в импликантной матрице Квайна — такое множество строк, при котором для каждого столбца найдется хотя бы одна строка из этого множества, на пересечении с которой этот столбец имеет крестик, причем при удалении хотя бы одной строки из этого множества указанное свойство не выполняется.

Таблица 1.2
Импликантная матрица Квайна

Простые импликанты	Конституенты				
	100	011	000	110	111
-00	+		+		
1-0	+			+	
-11		+			+
11-				+	+

Для нахождения всех тупиковых НФК обозначим строки матрицы буквами a, d, c и d . Каждому столбцу поставим в соответствие объединение строк, которые его покрывают, а всей таблице — пересечение этих объединений: $(a \cup b) \cap c \cap a \cap (b \cup d) \cap (c \cup d)$.

Используя свойства дистрибутивности, идемпотентности и поглощения преобразуем это выражение в объединение элементарных пересечений: $a \cap c \cap b \cup a \cap c \cap d$. Каждое элементарное пересечение определяет минимальное покрытие импликантной матрицы, т. е. тупиковую НФК. Итак, получили две тупиковые НФК: $\bar{B} \cap \bar{C} \cup B \cap C \cup A \cap \bar{C}$ и $\bar{B} \cap \bar{C} \cup B \cap C \cup A \cap B$. Тупиковая НФК, имеющая минимальную сложность, является минимальной НФК. В данном случае тупиковые НФК имеют одинаковую сложность, поэтому обе они являются и минимальными НФК. (Если сложность оценивать количеством операций, то минимальной НФК будет только вторая из полученных тупиковых НФК).

1.6. Преобразование теоретико-множественного выражения в нормальную форму Кантора

Теоретико-множественное выражение может содержать скобки, операции разности и симметрической разности, групповые дополнения, что не соответствует НФК. Любое теоретико-множественное выражение можно преобразовать в НФК. Для этого можно выполнить следующие действия:

1. Избавиться от разности и симметрической разности, используя соотношения:

$$A - B = A \cap \bar{B} \quad A \Delta B = (A \cap \bar{B}) \cup (B \cap \bar{A}).$$

2. Последовательно избавиться от всех групповых дополнений, используя законы де Моргана:

$$\overline{A \cap B} = \bar{A} \cup \bar{B} \quad \overline{A \cup B} = \bar{A} \cap \bar{B}.$$

3. Используя свойство дистрибутивности, раскрыть скобки.
4. Упростить выражение, используя свойства коммутативности, идемпотентности, поглощения, дополнения, нуля и единицы.

Преобразуем выражение $\overline{A \Delta B - C}$ в НФК.

$$\begin{aligned}
& \overline{A \Delta B - C} = \overline{(A \cap \overline{B} \cup \overline{A} \cap B) - C} = \overline{(A \cap \overline{B} \cup \overline{A} \cap B) \cap \overline{C}} = \\
&= \overline{(A \cap \overline{B} \cup \overline{A} \cap B) \cup C} = \overline{\overline{A} \cap \overline{B} \cap \overline{\overline{A} \cap B} \cup C} = \\
&= (\overline{A} \cup B) \cap (A \cup \overline{B}) \cup C = \overline{A} \cap A \cup \overline{A} \cap \overline{B} \cup B \cap A \cup B \cap \overline{B} \cup C = \\
&= \overline{A} \cap \overline{B} \cup B \cap A \cup C.
\end{aligned}$$

Произвольную НФК можно преобразовать в совершенную НФК. Пока в НФК есть элементарное пересечение, не являющееся конституентой, заменить его двумя элементарными пересечениями большего ранга, применяя склеивание в обратном порядке: $A = A \cap \bar{B} \cup A \cap B$. Затем упростить выражение, используя свойства коммутативности и идемпотентности.

Преобразуем НФК $\bar{A} \cap \bar{B} \cup B \cap A \cup C$ в совершенную НФК.

$$\begin{aligned} \bar{A} \cap \bar{B} \cup B \cap A \cap C &= \\ \bar{A} \cap \bar{B} \cap \bar{C} \cup \bar{A} \cap \bar{B} \cap C \cup B \cap A \cap \bar{C} \cup B \cap A \cap C \cup C \cap A \cup C \cap \bar{A} &= \\ \bar{A} \cap \bar{B} \cap \bar{C} \cup \bar{A} \cap \bar{B} \cap C \cup B \cap A \cap \bar{C} \cup B \cap A \cap C \cup \\ C \cap A \cap \bar{B} \cup C \cap A \cap B \cup C \cap \bar{A} \cap \bar{B} \cup C \cap \bar{A} \cap B &= \\ \bar{A} \cap \bar{B} \cap \bar{C} \cup \bar{A} \cap \bar{B} \cap C \cup B \cap A \cap \bar{C} \cup B \cap A \cap C \cup C \cap A \cap \bar{B} \cup C \cap \bar{A} \cap B. & \end{aligned}$$

Для преобразования произвольного теоретико-множественного выражения в совершенную НФК можно использовать разложение Шеннона по всем порождающим множествам:

$$f(M_1, M_2, \dots, M_n) = \overline{M_1} \cap \overline{M_2} \cap \dots \cap \overline{M_n} \cap f_0(\emptyset, \emptyset, \dots, \emptyset) \cup \\ \overline{M_1} \cap \overline{M_2} \cap \dots \cap M_n \cap f_1(\emptyset, \emptyset, \dots, U) \cup \\ \vdots \\ M_1 \cap M_2 \cap \dots \cap M_n \cap f_{n-1}(U, U, \dots, U).$$

Результат будет представлять собой объединение пересечений конституент с множеством f_i , которое принимает значения либо U , либо \emptyset . Если $f_i = U$, то конституента при f_i принадлежит совершенной НФК, иначе — нет.

Преобразуем выражение $\overline{A \Delta B - C}$ в совершенную НФК.

$$\begin{aligned}
\overline{A \Delta B - C} &= \overline{\overline{A} \cap \overline{B} \cap \overline{C} \cap (\overline{\emptyset \Delta \emptyset} - \overline{\emptyset})} \cup \\
&\quad \overline{\overline{A} \cap \overline{B} \cap C \cap (\overline{\emptyset \Delta \emptyset} - \overline{U})} \cup \\
&\quad \overline{\overline{A} \cap B \cap \overline{C} \cap (\overline{\emptyset \Delta U} - \overline{\emptyset})} \cup \\
&\quad \overline{\overline{A} \cap B \cap C \cap (\overline{\emptyset \Delta U} - \overline{U})} \cup \\
&\quad \overline{A \cap \overline{B} \cap \overline{C} \cap (\overline{U \Delta \emptyset} - \overline{\emptyset})} \cup \\
&\quad \overline{A \cap \overline{B} \cap C \cap (\overline{U \Delta \emptyset} - \overline{U})} \cup \\
&\quad \overline{A \cap B \cap \overline{C} \cap (\overline{U \Delta U} - \overline{\emptyset})} \cup \\
&\quad \overline{A \cap B \cap C \cap (\overline{U \Delta U} - \overline{U})} = \\
&= \overline{\overline{A} \cap \overline{B} \cap \overline{C} \cap U} \cup \\
&\quad \overline{\overline{A} \cap \overline{B} \cap C \cap U} \cup \\
&\quad \overline{A \cap B \cap \overline{C} \cap \emptyset} \cup \\
&\quad \overline{A \cap B \cap C \cap U} \cup \\
&\quad \overline{A \cap \overline{B} \cap \overline{C} \cap \emptyset} \cup \\
&\quad \overline{A \cap \overline{B} \cap C \cap U} \cup \\
&\quad \overline{A \cap B \cap \overline{C} \cap U} \cup \\
&\quad \overline{A \cap B \cap C \cap U} = \\
&= \overline{\overline{A} \cap \overline{B} \cap \overline{C}} \cup \overline{A} \cap \overline{B} \cap C \cup \overline{A} \cap B \cap \overline{C} \cup A \cap \overline{B} \cap C \cup A \cap B \cap \overline{C} \cup A \cap B \cap C.
\end{aligned}$$

1.7. Методы доказательства теоретико-множественных тождеств

Равенство, левая и правая части которого представляют собой теоретико-множественное выражение, верное для любых входящих в них множеств, называют *теоретико-множественным тождеством*. Рассмотрим различные методы доказательства теоретико-множественных тождеств.

1.7.1. Метод двух включений

Пусть левая часть теоретико-множественного тождества определяет множество X , а правая часть — множество Y . Чтобы доказать равенство множеств X и Y , достаточно доказать два включения $X \subseteq Y$ и $Y \subseteq X$, т.е. доказать, что из предположения $x \in X$ (для произвольного x) следует, что $x \in Y$, и, наоборот, из предположения $x \in Y$ следует, что $x \in X$.

Докажем этим методом тождество:

$$A \Delta B = (A \cup B) - (A \cap B).$$

Пусть $x \in A \Delta B$. Тогда, согласно определению симметрической разности, $x \in (A - B) \cup (B - A)$. Это означает, что $x \in (A - B)$ или $x \in (B - A)$. Если $x \in (A - B)$, то $x \in A$ и $x \notin B$, т.е. $x \in A \cup B$ и при этом $x \notin A \cap B$. Если же $x \in (B - A)$, то $x \in B$ и $x \notin A$, откуда $x \in A \cup B$ и $x \notin A \cap B$. Итак, в любом случае из $x \in (A - B) \cup (B - A)$ следует $x \in A \cup B$ и $x \notin A \cap B$, т.е. и $x \in (A \cup B) - (A \cap B)$. Таким образом, доказано, что $A \Delta B \subseteq (A \cup B) - (A \cap B)$.

Покажем обратное включение $(A \cup B) - (A \cap B) \subseteq A \Delta B$.

Пусть $x \in (A \cup B) - (A \cap B)$. Тогда $x \in A \cup B$ и $x \notin A \cap B$. Из $x \in A \cup B$ следует, что $x \in A$ или $x \in B$. Если $x \in A$, то с учетом $x \notin A \cap B$ имеем $x \notin B$, и поэтому $x \in A - B$. Если же $x \in B$, то опять-таки в силу $x \notin A \cap B$ получаем, что $x \notin A$ и $x \in B - A$. Итак, $x \in A - B$ или $x \in B - A$, т.е. $x \in (A - B) \cup (B - A)$. Следовательно,

$$(A \cup B) - (A \cap B) \subseteq A \Delta B.$$

Оба включения имеют место и тождество доказано.

Доказательство сложных теоретико-множественных тождеств методом двух включений часто бывает довольно громоздким, и при построении доказательства ход рассуждений не всегда очевиден.

1.7.2. Метод эквивалентных преобразований

Теоретико-множественные тождества можно доказывать, используя свойства операций над множествами (см. п. 1.4. “Свойства операций над множествами”). Для этого нужно преобразовать левую часть в правую, или правую — в левую.

Докажем этим методом тождество:

$$A \cap (B \Delta C) = (A \cap B) \Delta (A \cap C).$$

Преобразуем левую часть к правой.

1. По определению $B \Delta C = (B - C) \cup (C - B)$, поэтому

$$A \cap (B \Delta C) = A \cap ((B - C) \cup (C - B)).$$

2. Используя свойство дистрибутивности для \cap и \cup получим

$$A \cap (B - C) \cup A \cap (C - B).$$

3. Используя свойство дистрибутивности для \cap и $-$ получим

$$((A \cap B) - (A \cap C)) \cup ((A \cap C) - (A \cap B)).$$

Это есть определение симметрической разности для $(A \cap B)$ и $(A \cap C)$, т.е. $A \cap (B \Delta C) = (A \cap B) \Delta (A \cap C)$. Тождество доказано.

При использовании этого метода последовательность преобразований, применяемых при доказательстве, определяется только интуицией исследователя и, как правило, является сложной, творческой задачей (попробуйте преобразовать правую часть вышеприведенного тождества в левую).

Данный метод можно значительно упростить и формализовать, если левую и правую части тождества преобразовывать в некоторую третью, уникальную форму. Такой формой может быть совершенная НФК.

Равенство двух множеств $X = Y$, в котором множества X и Y представлены в совершенной НФК, является теоретико-множественным тождеством, если совершенные НФК множеств X и Y содержат одинаковое количество конституент и каждой конституенте из совершенной НФК множества X найдется тождественная ей конституента из совершенной НФК множества Y . Две конституенты тождественны, если они состоят из одних и тех же первичных термов.

Доказательство теоретико-множественного тождества с использованием совершенной НФК заключается в следующем: левую и правую части равенства представить в совершенной НФК, и, если будет получено теоретико-множественное тождество, то и исходное равенство представляет собой теоретико-множественное тождество.

Докажем этим методом тождество:

$$(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C).$$

Левая часть:

$$\begin{aligned} (A \Delta B) \cap C &= (A \cap \bar{B} \cup \bar{A} \cap B) \cap C = \\ &= A \cap \bar{B} \cap C \cup \bar{A} \cap B \cap C. \end{aligned}$$

Правая часть:

$$\begin{aligned} (A \cap C) \Delta (B \cap C) &= A \cap C \cap \overline{B \cap C} \cup \overline{A \cap C} \cap B \cap C = \\ &= A \cap C \cap (\bar{B} \cup \bar{C}) \cup (\bar{A} \cup \bar{C}) \cap B \cap C = \\ &= A \cap C \cap \bar{B} \cup \bar{A} \cap B \cap C. \end{aligned}$$

Совершенные НФК левой и правой части получены, каждая из них состоит из двух конституент, причем первой конституенте левой части тождественна первая конституента правой части, а второй конституенте левой части тождественна вторая конституента правой части, следовательно исходное равенство является теоретико-множественным тождеством.

1.7.3. Метод характеристических функций

Метод характеристических функций относится к методам, не требующим "угадывания" путем доказательства.

Арифметические характеристические функции

Арифметическая характеристическая функция (АХФ) χ_A множества A для $x \in U$ определяется следующим образом: $\chi_A(x) = 1$, если $x \in A$ и $\chi_A(x) = 0$, если $x \notin A$.

Для АХФ χ_A множества A справедливо тождество:

$$\chi_A^2(x) = \chi_A(x).$$

АХФ $\chi_{A \cap B}(x)$ пересечения множеств A и B определяется произведением АХФ множеств A и B :

$$\chi_{A \cap B}(x) = \chi_A(x) \cdot \chi_B(x).$$

Для получения АХФ $\chi_{A \cup B}(x)$ объединения множеств A и B сложим АХФ множеств A и B . Но в этом случае для элементов $x \in A \cap B$ такая сумма будет иметь значение 2, поэтому из этой суммы необходимо вычесть значение АХФ $\chi_{A \cap B}(x)$ пересечения множеств A и B :

$$\chi_{A \cup B}(x) = \chi_A(x) + \chi_B(x) - \chi_A(x) \cdot \chi_B(x).$$

АХФ $\chi_A^-(x)$ дополнения множества A определяется формулой

$$\chi_A^-(x) = 1 - \chi_A(x).$$

АХФ $\chi_{A-B}(x)$ разности множеств A и B имеет вид

$$\chi_{A-B}(x) = \chi_A(x) - \chi_A(x) \cdot \chi_B(x),$$

а для симметрической разности —

$$\chi_{A \Delta B}(x) = \chi_A(x) + \chi_B(x) - 2 \cdot \chi_A(x) \cdot \chi_B(x).$$

Метод характеристических функций доказательства справедливости теоретико-множественного тождества заключается в выражении характеристических функций обеих его частей через характеристические функции входящих в него множеств. Тождества верны тогда и только тогда, когда характеристические функции левой и правой частей совпадают.

Докажем этим методом тождество:

$$(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C).$$

С одной стороны,

$$\begin{aligned} \chi_{(A \Delta B) \cap C}(x) &= \chi_{(A \Delta B)}(x) \cdot \chi_C(x) = (\chi_A(x) + \chi_B(x) - 2 \cdot \chi_A(x) \cdot \chi_B(x)) \cdot \chi_C(x) = \\ &= \chi_A(x) \cdot \chi_C(x) + \chi_B(x) \cdot \chi_C(x) - 2 \cdot \chi_A(x) \cdot \chi_B(x) \cdot \chi_C(x). \end{aligned}$$

С другой стороны,

$$\begin{aligned} \chi_{(A \cap C) \Delta (B \cap C)}(x) &= \chi_{A \cap C}(x) + \chi_{B \cap C}(x) - 2 \cdot \chi_{A \cap C}(x) \cdot \chi_{B \cap C}(x) = \\ &= \chi_A(x) \cdot \chi_C(x) + \chi_B(x) \cdot \chi_C(x) - 2 \cdot \chi_A(x) \cdot \chi_C(x) \cdot \chi_B(x) \cdot \chi_C(x) = \\ &= \chi_A(x) \cdot \chi_C(x) + \chi_B(x) \cdot \chi_C(x) - 2 \cdot \chi_A(x) \cdot \chi_B(x) \cdot \chi_C(x). \end{aligned}$$

АХФ левой и правой частей тождества совпадают. Следовательно, тождество верно.

Логические характеристические функции

Логическая характеристическая функция (ЛХФ) λ_A множества A для $x \in U$ определяется следующим образом: $\lambda_A(x) = \text{TRUE}$, если $x \in A$ и $\lambda_A(x) = \text{FALSE}$, если $x \notin A$.

ЛХФ $\lambda_{A \cap B}(x)$ пересечения множеств A и B определяется конъюнкцией ЛХФ множеств A и B :

$$\lambda_{A \cap B}(x) = \lambda_A(x) \wedge \lambda_B(x).$$

ЛХФ $\lambda_{A \cup B}(x)$ пересечения множеств A и B определяется дизъюнкцией ЛХФ множеств A и B :

$$\lambda_{A \cup B}(x) = \lambda_A(x) \vee \lambda_B(x).$$

ЛХФ $\lambda_A^-(x)$ дополнения множества A определяется отрицанием ЛХФ множества A :

$$\lambda_A^-(x) = \overline{\lambda_A(x)}.$$

ЛХФ $\lambda_{A \Delta B}(x)$ симметрической разности множеств A и B определяется сложением по модулю 2 ЛХФ множеств A и B :

$$\lambda_{A \Delta B}(x) = \lambda_A(x) \oplus \lambda_B(x).$$

ЛХФ $\lambda_{A - B}(x)$ разности множеств A и B определяется формулой:

$$\lambda_{A - B}(x) = \lambda_A(x) \wedge \overline{\lambda_B(x)}$$

Доказательство методом ЛХФ проводится так же, как и методом АХФ. Для проверки равенства ЛХФ можно использовать их таблицы истинности.

Докажем этим методом тождество:

$$(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C).$$

Для левой части: $\lambda_{(A \Delta B) \cap C}(x) = \lambda_{A \Delta B}(x) \wedge \lambda_C(x) = (\lambda_A(x) \oplus \lambda_B(x)) \wedge \lambda_C(x).$

Для правой части: $\lambda_{(A \cap C) \Delta (B \cap C)}(x) = \lambda_{(A \cap C)}(x) \oplus \lambda_{(B \cap C)}(x) = \lambda_A(x) \wedge \lambda_C(x) \oplus \lambda_B(x) \wedge \lambda_C(x).$

Составим таблицу истинности для ЛХФ левой и правой части (табл.1.3). В этой таблице «1» соответствует значению TRUE, а «0» — значению FALSE. Таблицы истинности ЛХФ левой и правой части совпадают, следовательно, тождество верно.

Таблица 1.3
Таблица истинности

$\lambda_A(x)$	$\lambda_B(x)$	$\lambda_C(x)$	ЛХФ левой части	ЛХФ правой части
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

1.7.4. Теоретико-множественный метод

Метод основан на вычислении теоретико-множественного тождества при определенных значениях входящих в него множеств, и, если при этих значениях множеств теоретико-множественное тождество истинно, то оно истинно и при любых других значениях входящих в него множеств. Сначала рассмотрим этот метод на примере.

Пусть равенство содержит не более трех исходных множеств, например:

$$(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C).$$

Множества, определяемые левой и правой частью равенства, можно изобразить графически на кругах Эйлера, и, если эти множества изображаются одинаково, то равенство представляет собой теоретико-множественное тождество.

Круги Эйлера, соответствующие множествам A , B и C , находящимся в общем положении, разбивают плоскость на восемь непересекающихся областей (соответствует разбиению универсума на восемь подмножеств). Пронумеруем эти области так, как показано на рис.1.2.

Каждое из исходных множеств (A , B и C) теперь можно представить как множество, состоящее из четырех непересекающихся областей и считать, что $A = \{1, 3, 5, 7\}$, $B = \{2, 3, 6, 7\}$ и $C = \{4, 5, 6, 7\}$. Для того, чтобы найти множество областей, соответствующих множеству, определяемому левой частью равенства, подставим полученные значения множеств A , B и C в левую часть равенства и вычислим его значение:

$$\begin{aligned} (A \Delta B) \cap C &= (\{1, 3, 5, 7\} \Delta \{2, 3, 6, 7\}) \cap \{4, 5, 6, 7\} = \\ &= \{1, 2, 5, 6\} \cap \{4, 5, 6, 7\} = \{5, 6\}. \end{aligned}$$

Следовательно, множество, соответствующее левой части равенства, определяется на кругах Эйлера областями 5 и 6.

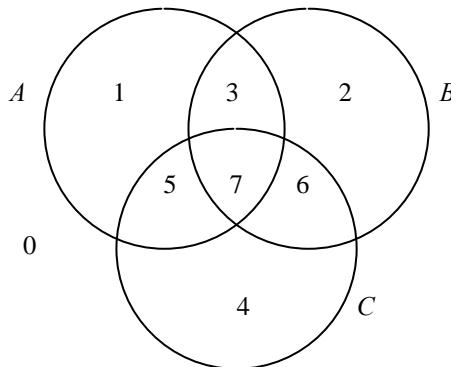


Рис.1.2. Разбиение универсума множествами A , B и C
на подмножества

Аналогично определим множество областей, соответствующих множеству, определяемому правой частью равенства:

$$\begin{aligned}
 & (A \cap C) \Delta (B \cap C) = \\
 & = (\{1, 3, 5, 7\} \cap \{4, 5, 6, 7\}) \Delta (\{2, 3, 6, 7\} \cap \{4, 5, 6, 7\}) = \\
 & = \{5, 7\} \Delta \{6, 7\} = \{5, 6\}.
 \end{aligned}$$

Множества, соответствующие левой и правой части равенства, определяются на кругах Эйлера одинаковыми областями, следовательно, равенство является теоретико-множественным тождеством.

Из рассмотренного примера следует, что для доказательства теоретико-множественного тождества достаточно вычислить значение соответствующего равенства на определенных значениях входящих в равенство множествах, и, если получим истинное значение, то теоретико-множественное тождество верно.

Если количество n входящих в равенство множеств больше трех, то использование кругов Эйлера для нахождения значений множеств становится громоздким и ненаглядным. Количество областей в этом случае равно 2^n и каждой области можно поставить во взаимно-однозначное соответствие n -разрядный двоичный вектор, в котором i -й разряд соответствует i -му множеству равенства и равен 1, если область принадлежит этому множеству и 0 — в противном случае. Двоичный вектор, соответствующий j -й области, можно рассматривать как двоичное представление числа j . Для трех множеств сопоставление областям двоичных векторов представлено в табл.1.4.

По такой таблице значение множества определяется следующим образом: просматривается столбец, соответствующий множеству, и, если встречается единица, то номер строки (номер области) добавляется в множество.

Из рассмотренного выше следует, что для доказательства теоретико-множественного тождества, содержащего n множеств, необходимо построить таблицу двоичных n -разрядных векторов, состоящую из 2^n строк, по таблице определить значения входящих в равенство множеств, вычислить значение равенства на этих множествах, и, если получим истинное значение, то теоретико-множественное тождество верно.

*Таблица 1.4
Сопоставление областям универсума двоичных векторов*

Номер области	Двоичный вектор		
	C	B	A
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

1.7.5. Метод симметрической разности

Пусть множества X и Y в равенстве $X = Y$ представлены теоретико-множественными выражениями. Множества X и Y равны тогда и только тогда, когда $X \Delta Y = \emptyset$, поэтому для доказательства теоретико-множественного тождества $X = Y$ достаточно доказать $X \Delta Y = \emptyset$.

Для доказательства $X \Delta Y = \emptyset$ можно множество $X \Delta Y$ представить в НФК (см. выше) или применять разложение Шеннона и выполнять упрощения, используя свойства нуля и единицы и дополнения.

Докажем этим методом тождество:

$$(A \Delta B) \cap C = (A \cap C) \Delta (B \cap C).$$

$$\begin{aligned} (A \Delta B) \cap C \Delta (A \cap C) \Delta (B \cap C) &= \\ &= A \cap ((U \Delta B) \cap C \Delta (U \cap C) \Delta (B \cap C)) \cup \\ &\cup \bar{A} \cap ((\bar{U} \Delta B) \cap C \Delta (\bar{U} \cap C) \Delta (B \cap C)) = \\ &= A \cap ((\bar{B} \cap C) \Delta C \Delta (B \cap C)) \cup \bar{A} \cap ((B \cap C) \Delta \bar{C} \Delta (B \cap C)) = \\ &= A \cap (B \cap ((\bar{U} \cap C) \Delta C \Delta (U \cap C))) \cup \bar{B} \cap ((\bar{U} \cap C) \Delta C \Delta (\bar{U} \cap C)) \cup \emptyset = \\ &= A \cap (B \cap (\bar{U} \Delta C \Delta C)) \cup \bar{B} \cap (C \Delta C \Delta \bar{U}) = A \cap (\emptyset \cup \emptyset) = \emptyset. \end{aligned}$$

1.8. Теоретико-множественные уравнения

Теоретико-множественным уравнением (уравнением с множествами) называется равенство

$$g(X, A_1, \dots, A_n) = h(X, A_1, \dots, A_n),$$

где A_1, \dots, A_n — известные исходные множества, $A_i \subseteq U$, U — конечный универсум; X — неизвестное искомое множество, $X \subseteq U$; $g(X, A_1, \dots, A_n)$ и $h(X, A_1, \dots, A_n)$ — теоретико-множественные выражения.

Множество X , при котором равенство принимает истинное значение, называется *частным решением уравнения*, а множество всех частных решений образует *общее решение*.

Для решения уравнения приведем его к виду $f(X, A_1, \dots, A_n) = \emptyset$.

Множество $f(X, A_1, \dots, A_n)$ пусто тогда и только тогда, когда множество $g(X, A_1, \dots, A_n)$ равно множеству $h(X, A_1, \dots, A_n)$, поэтому

$$g(X, A_1, \dots, A_n) \Delta h(X, A_1, \dots, A_n) = \emptyset.$$

Выполним разложение Шеннона левой части полученного уравнения по неизвестному множеству X :

$$\overline{X} \cap \varphi^\emptyset \cup X \cap \varphi^U = \emptyset,$$

где $\varphi^\emptyset = f(\emptyset, A_1, \dots, A_n)$, $\varphi^U = f(U, A_1, \dots, A_n)$. Множества φ^\emptyset и φ^U определяются универсумом U , пустым множеством \emptyset и известными исходными множествами, поэтому можно вычислить их значения.

Это уравнение имеет решение, если $\overline{X} \cap \varphi^\emptyset = \emptyset$ и $X \cap \varphi^U = \emptyset$. Множество $\overline{X} \cap \varphi^\emptyset$ пусто тогда и только тогда, когда $\overline{X} \subseteq \overline{\varphi^\emptyset}$, или, то же самое, $X \supseteq \varphi^\emptyset$, а множество $X \cap \varphi^U$ пусто тогда и только тогда, когда $X \subseteq \overline{\varphi^U}$, поэтому множество X должно быть подмножеством множества $\overline{\varphi^U}$ и надмножеством множества φ^\emptyset , т.е. $\varphi^\emptyset \subseteq X \subseteq \overline{\varphi^U}$.

Решение этого неравенства является частным решением исходного уравнения, а множество всех его решений — общим решением уравнения.

Уравнение имеет хотя бы одно решение, если $\varphi^\emptyset \subseteq \overline{\varphi^U}$. Если $\varphi^\emptyset = \emptyset$ и $\overline{\varphi^U} = U$, то частным решением будет любое подмножество универсума, а общим решением — булев универсум. Если уравнение имеет решение, то минимальным по мощности будет $X = \varphi^\emptyset$, а максимальным — $X = \overline{\varphi^U}$. Частным решением уравнения будет любое множество X , являющееся подмножеством множества $\overline{\varphi^U}$ и включающим в себя множество φ^\emptyset , т.е. $X = \varphi^\emptyset \cup \varphi_i$, где $\varphi_i \subseteq \overline{\varphi^U} - \varphi^\emptyset$. Следовательно, для получения общего решения уравнения необходимо породить все

подмножества множества $\overline{\varphi^U} - \varphi^\emptyset$ и каждое из них объединить с множеством φ^\emptyset .

Рассмотрим пример решения теоретико-множественного уравнения

$$B \cap (\overline{X} \cup \overline{C})\Delta A = (\overline{A} \cup X)\Delta B \cap X$$

при $A = \{3, 4, 5, 6, 8, 10\}$, $B = \{1, 2, 7, 8, 9, 10\}$, $C = \{1, 2, 3, 4, 5, 10\}$ и $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

1. Преобразуем исходное уравнение в уравнение с пустой правой частью:

$$B \cap (\overline{X} \cup \overline{C})\Delta A \Delta (\overline{A} \cup X)\Delta B \cap X = \emptyset.$$

2. Выполним разложение Шеннона левой части уравнения по неизвестному множеству X :

$$\overline{X} \cap \varphi^\emptyset \cup X \cap \varphi^U = \emptyset,$$

$$\text{где } \varphi^\emptyset = B \cap (\overline{\emptyset} \cup \overline{C})\Delta A \Delta (\overline{A} \cup \emptyset) \Delta B \cap \emptyset,$$

$$\varphi^U = B \cap (\overline{U} \cup \overline{C})\Delta A \Delta (\overline{A} \cup U) \Delta B \cap U.$$

3. Упростим φ^\emptyset и φ^U , используя свойства нуля, единицы и дополнения: $\varphi^\emptyset = B \Delta A \Delta \overline{A} = B \Delta U = \overline{B}$,

$$\varphi^U = B \cap \overline{C} \Delta A \Delta U \Delta B = B \cap \overline{C} \Delta A \Delta \overline{B}.$$

4. Вычислим φ^\emptyset и $\overline{\varphi^U}$ при заданных исходных множествах:

$$\varphi^\emptyset = \overline{\{1, 2, 7, 8, 9, 10\}} = \{3, 4, 5, 6\},$$

$$\begin{aligned} \overline{\varphi^U} &= \overline{\{1, 2, 7, 8, 9, 10\} \cap \{1, 2, 3, 4, 5, 10\} \Delta \{3, 4, 5, 6, 8, 10\} \Delta \{1, 2, 7, 8, 9, 10\}} = \\ &= \overline{\{1, 2, 7, 8, 9, 10\} \cap \{6, 7, 8, 9\} \Delta \{3, 4, 5, 6, 8, 10\} \Delta \{3, 4, 5, 6\}} = \\ &= \overline{\{7, 8, 9\} \Delta \{3, 4, 5, 6, 8, 10\} \Delta \{3, 4, 5, 6\}} = \overline{\{3, 4, 5, 6, 7, 9, 10\} \Delta \{3, 4, 5, 6\}} = \\ &= \overline{\{7, 9, 10\}} = \{1, 2, 3, 4, 5, 6, 8\}. \end{aligned}$$

5. Множество $\varphi^\emptyset = \{3, 4, 5, 6\}$ является подмножеством множества $\overline{\varphi^U} = \{1, 2, 3, 4, 5, 6, 8\}$, следовательно, уравнение имеет решения.

6. Минимальным по мощности частным решением является множество $\varphi^\emptyset = \{3, 4, 5, 6\}$, а максимальным — $\overline{\varphi^U} = \{1, 2, 3, 4, 5, 6, 8\}$.

7. Для получения общего решения необходимо каждое подмножество множества $\overline{\varphi^U} - \varphi^\emptyset = \{1, 2, 8\}$ объединить с $\varphi^\emptyset = \{3, 4, 5, 6\}$. Общее решение: $\{\{3, 4, 5, 6\}, \{3, 4, 5, 6, 1\}, \{3, 4, 5, 6, 2\}, \{3, 4, 5, 6, 1, 2\}, \{3, 4, 5, 6, 8\}, \{3, 4, 5, 6, 1, 8\}, \{3, 4, 5, 6, 2, 8\}, \{3, 4, 5, 6, 1, 2, 8\}\}$.

Самостоятельно выполните проверку решения.

1.9. Способы представления множества в памяти ЭВМ

Для представления конечного множества в памяти ЭВМ можно использовать различные способы. Будем считать, что универсум U представляет собой KU натуральных чисел: $U = \{x / x \in N \text{ и } x \leq KU\}$. Если это не так, то элементы множества U можно пронумеровать и определить однозначное соответствие между элементами множества U и их номерами. В этом случае элементами множеств можно считать натуральные числа, которые можно сравнивать и использовать в качестве индекса элемента массива.

1. Элементы множества A хранятся в переменной A типа массив, мощность множества A — в переменной KA . Количество элементов в массиве A равно мощности универсума. Элементы массива A неупорядочены.

2. Элементы множества A хранятся в переменной A типа массив, мощность множества A — в переменной KA . Количество элементов в массиве A равно мощности универсума. Элементы массива A упорядочены по возрастанию.

3. Элементы множества A хранятся в переменной A типа массив, элементы которого типа *boolean*. Если $i \in A$, то $A_i = true$, иначе $A_i = false$. Количество элементов в массиве A равно мощности универсума.

4. Множество A представляется двоичным кодом C , в котором: $C_i = 1$, если $i \in A$ и $C_i = 0$, если $i \notin A$, где C_i — i -й разряд кода C . В зависимости от мощности универсума код C может храниться в простой переменной или в массиве.

5. Для хранения множества можно использовать множественный тип.

1.10. Алгоритмы реализации операций над множествами

Алгоритмы реализации операций над множествами зависят от способа представления множества в памяти ЭВМ.

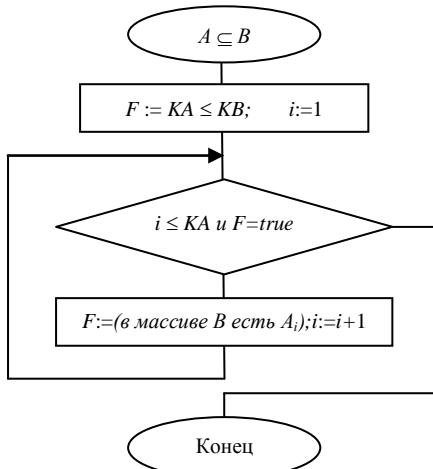
1. Элементы множества A хранятся в переменной A типа массив, мощность множества A — в переменной KA . Количество элементов в массиве A равно мощности универсума. Элементы массива A неупорядочены.

Алгоритм 1.1 (рис. 1.3) вычисления включения A в B ($A \subseteq B$).

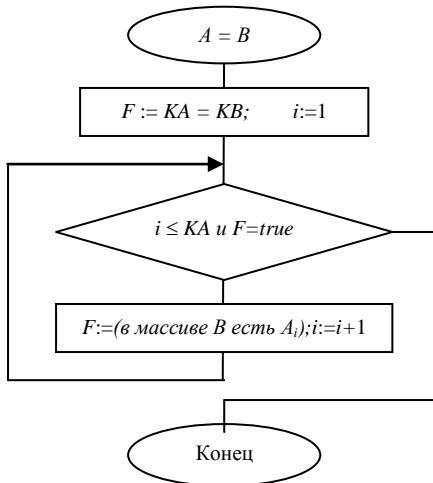
Вход: A — массив, хранящий элементы множества A , $KA = |A|$;

B — массив, хранящий элементы множества B , $KB = |B|$.

Выход: $F = true$, если $A \subseteq B$, иначе $F = false$.

Рис.1.3. Блок-схема алгоритма вычисления включения A в B

Алгоритм 1.2 (рис.1.4) вычисления равенства A и B ($A = B$).
 Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.
 Выход: $F = \text{true}$, если $A = B$, иначе $F = \text{false}$.

Рис.1.4. Блок-схема алгоритма вычисления равенства A и B

Алгоритм 1.3 (рис.1.5) вычисления объединения A и B ($A \cup B$).

Вход: A — массив, хранящий элементы множества A , $KA = |A|$;

B — массив, хранящий элементы множества B , $KB = |B|$.

Выход: C — массив, хранящий объединение множеств A и B , $KC = |C|$.

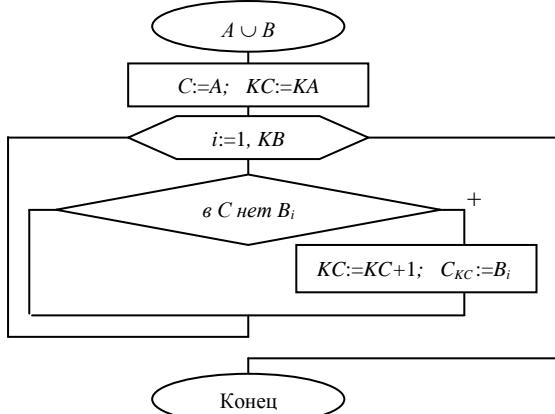


Рис.1.5. Блок-схема алгоритма вычисления объединения A и B

Алгоритм 1.4 (рис.1.6) вычисления пересечения A и B ($A \cap B$).

Вход: A — массив, хранящий элементы множества A , $KA = |A|$;

B — массив, хранящий элементы множества B , $KB = |B|$;

пусть $KA \leq KB$.

Выход: C — массив, хранящий пересечение множеств A и B , $KC = |C|$.

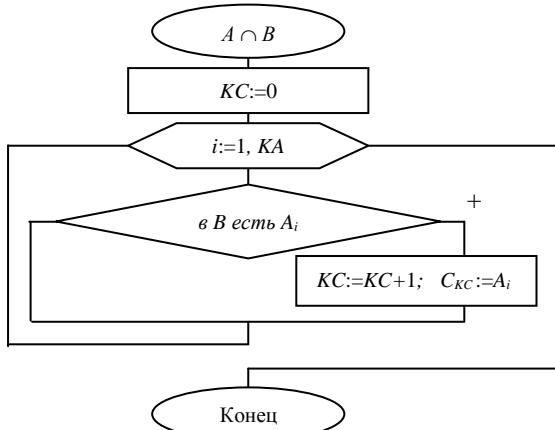


Рис.1.6. Блок-схема алгоритма вычисления пересечения A и B

Алгоритм 1.5 (рис.1.7) вычисления разности A и B ($A - B$).

Вход: A — массив, хранящий элементы множества A, $KA = |A|$;

B — массив, хранящий элементы множества B, $KB = |B|$.

Выход: C — массив, хранящий разность множеств A и B, $KC = |C|$.

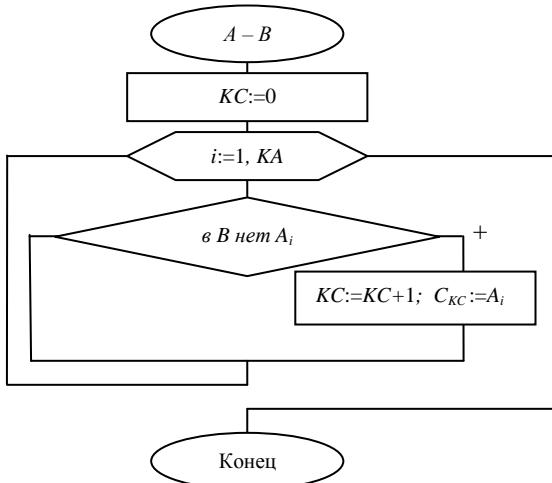


Рис.1.7. Блок-схема алгоритма вычисления разности A и B

Алгоритм 1.6 вычисления симметрической разности A и B ($A \Delta B$).

Вход: A — массив, хранящий элементы множества A, $KA = |A|$;

B — массив, хранящий элементы множества B, $KB = |B|$.

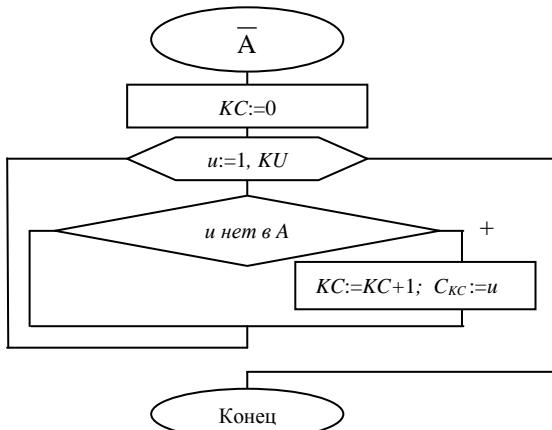
Выход: C — массив, хранящий симметрическую разность множеств A и B, $KC = |C|$.

1. $C := A - B \cup B - A$; {или $C := (A \cup B) - (A \cap B)$ }
2. Конец.

Алгоритм 1.7 (рис.1.8) вычисления дополнения A (\bar{A}).

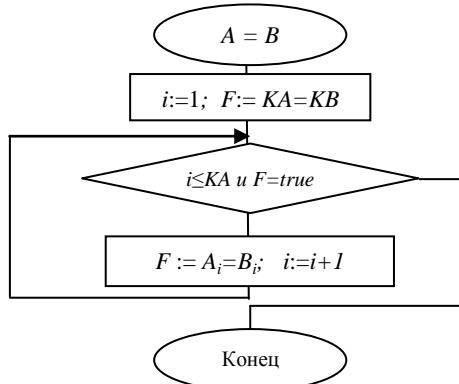
Вход: A — массив, хранящий элементы множества A, $KA = |A|$.

Выход: C — массив, хранящий дополнение множества A, $KC = |C|$.

Рис.1.8. Блок-схема алгоритма вычисления дополнения A

2. Элементы множества A хранятся в переменной A типа массив, мощность множества A – в переменной KA . Количество элементов в массиве A равно мощности универсума. Элементы массива A упорядочены по возрастанию.

Алгоритм 1.8 (рис.1.9) вычисления равенства A и B ($A = B$).
 Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.
 Выход: $F = \text{true}$, если $A = B$, иначе $F = \text{false}$.

Рис.1.9. Блок-схема алгоритма вычисления равенства A и B

Алгоритм 1.9 (рис.1.10) вычисления включения $A \subseteq B$ ($A \subseteq B$).

Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.

Выход: $F = true$, если $A \subseteq B$, иначе $F = false$.

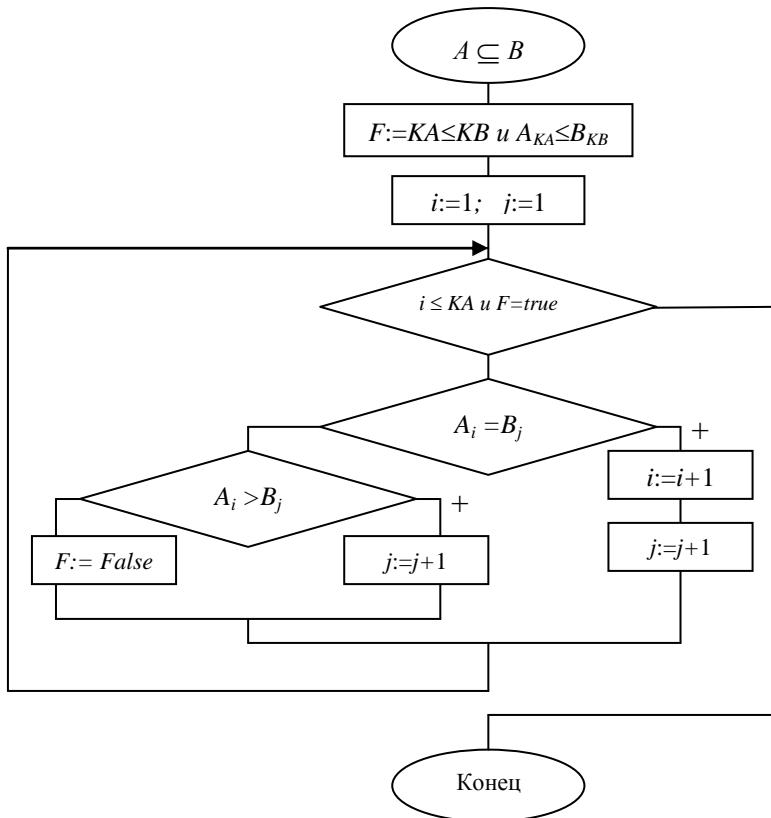


Рис.1.10. Блок-схема алгоритма вычисления включения $A \subseteq B$

*Алгоритм 1.10 (рис.1.11) вычисления объединения A и B ($A \cup B$).
Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.
Выход: C — массив, хранящий объединение множеств A и B , $KC = |C|$.*

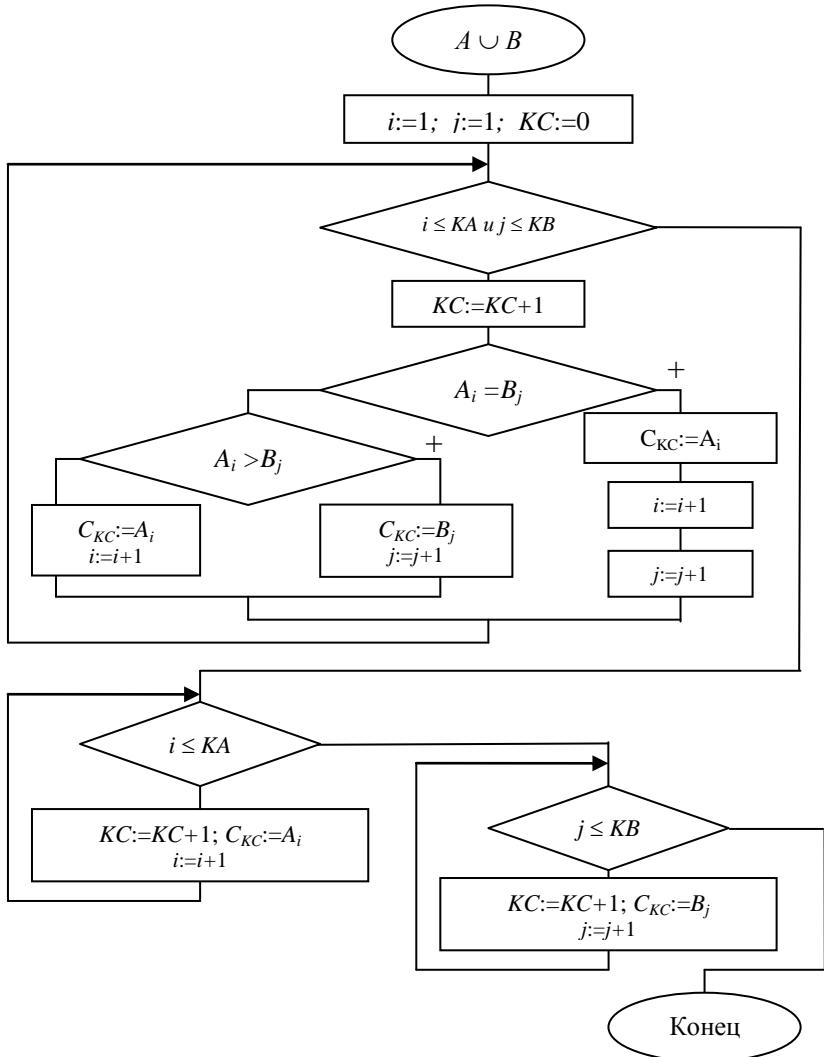


Рис.1.11. Блок-схема алгоритма вычисления объединения A и B

Алгоритм 1.11 (рис.1.12) вычисления пересечения A и B ($A \cap B$).

Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.

Выход: C — массив, хранящий пересечение множеств A и B , $KC = |C|$.

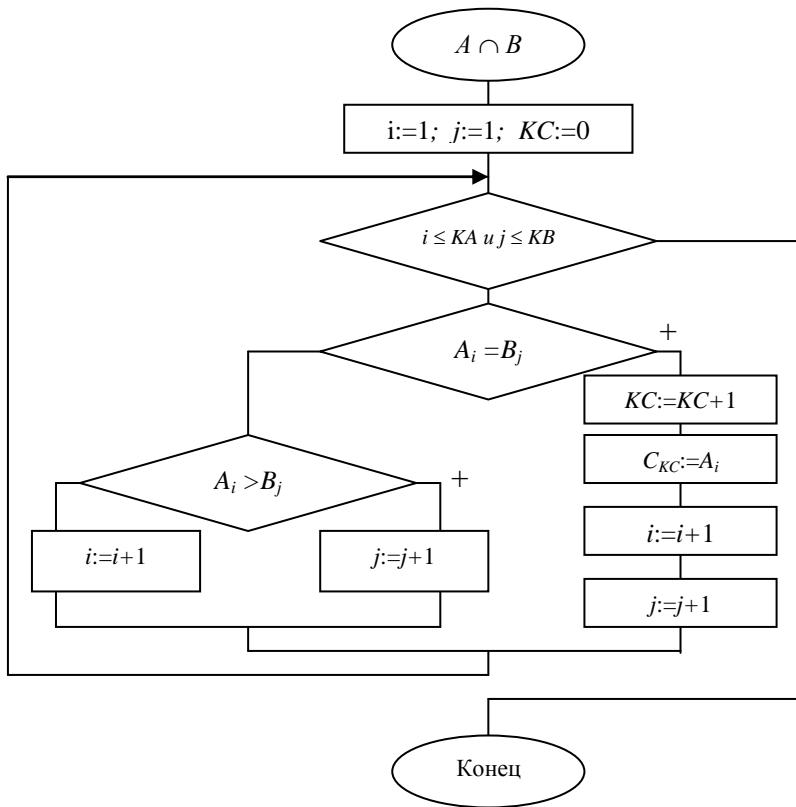


Рис.1.12. Блок-схема алгоритма вычисления пересечения A и B

*Алгоритм 1.12 (рис.1.13) вычисления разности A и B ($A - B$).
Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.
Выход: C — массив, хранящий разность множеств A и B , $KC = |C|$.*

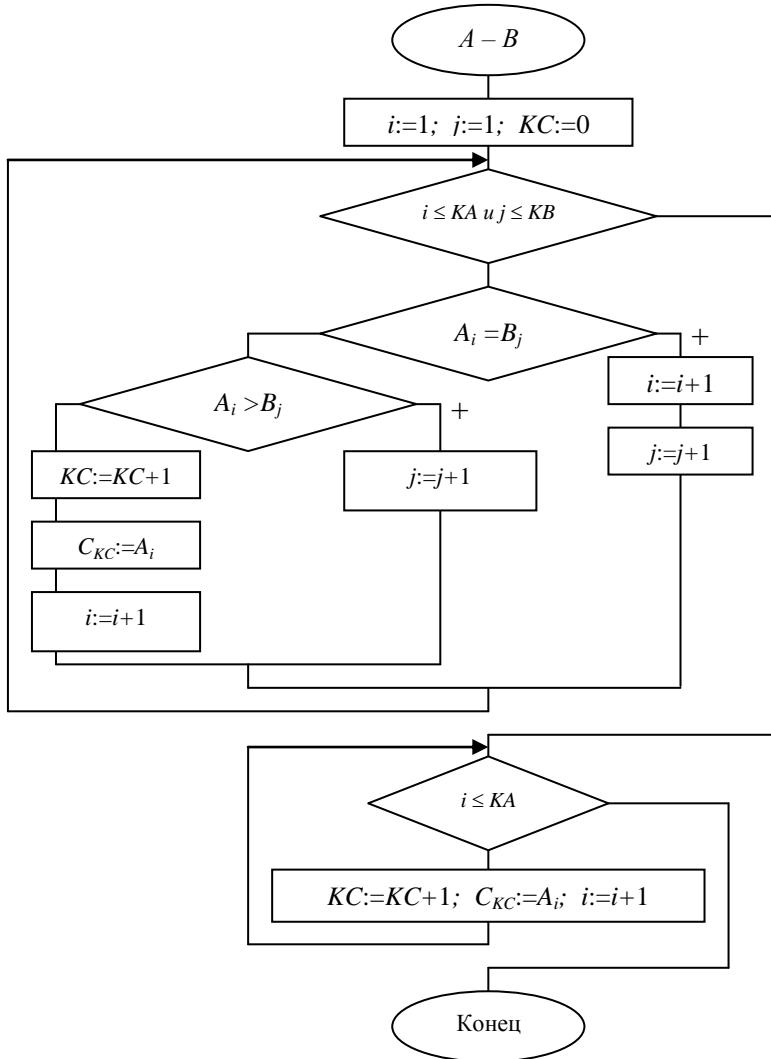


Рис.1.13. Блок-схема алгоритма вычисления разности A и B

Алгоритм 1.13 (рис.1.14) вычисления симметрической разности A и B ($A \Delta B$).

Вход: A — массив, хранящий элементы множества A , $KA = |A|$;
 B — массив, хранящий элементы множества B , $KB = |B|$.

Выход: C — массив, хранящий симметрическую разность A и B ,
 $KC = |C|$.

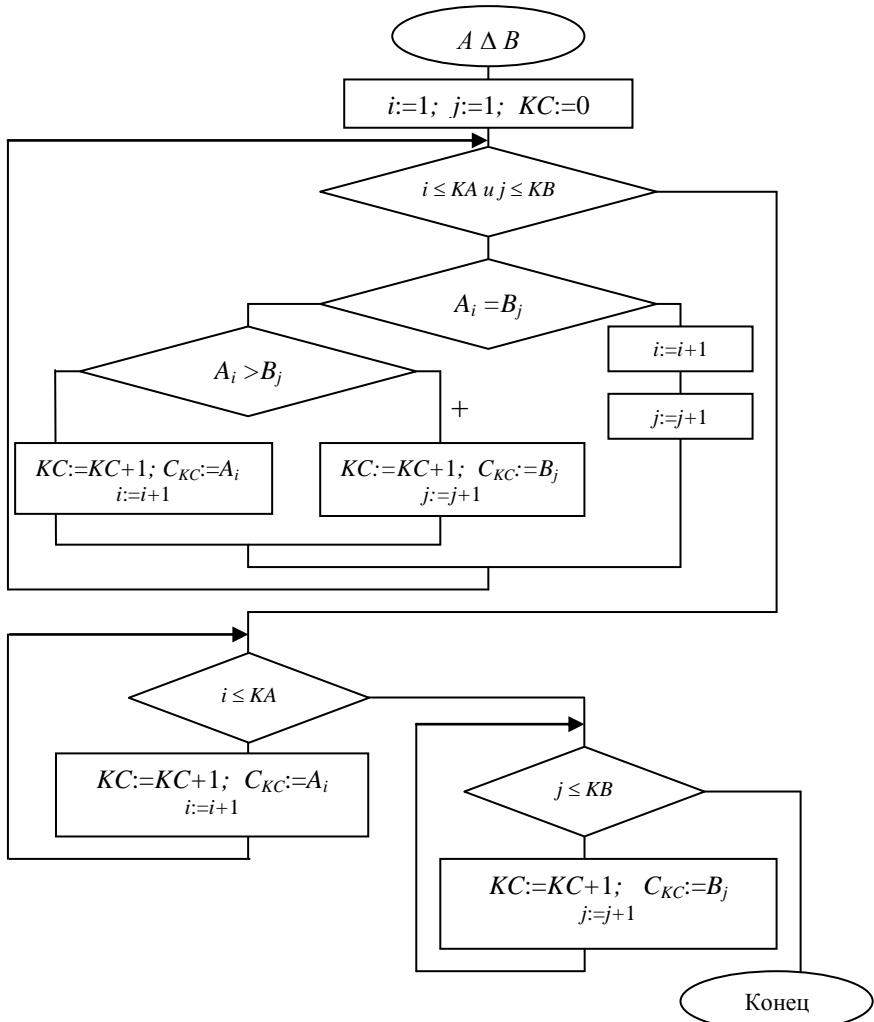


Рис.1.14. Блок-схема алгоритма вычисления симметрической разности A и B

Алгоритм 1.14 (рис.1.15) вычисления дополнения A (\bar{A}).
 Вход: A — массив, хранящий элементы множества A , $KA = |A|$.
 Выход: C — массив, хранящий дополнение множества A , $KC = |C|$.

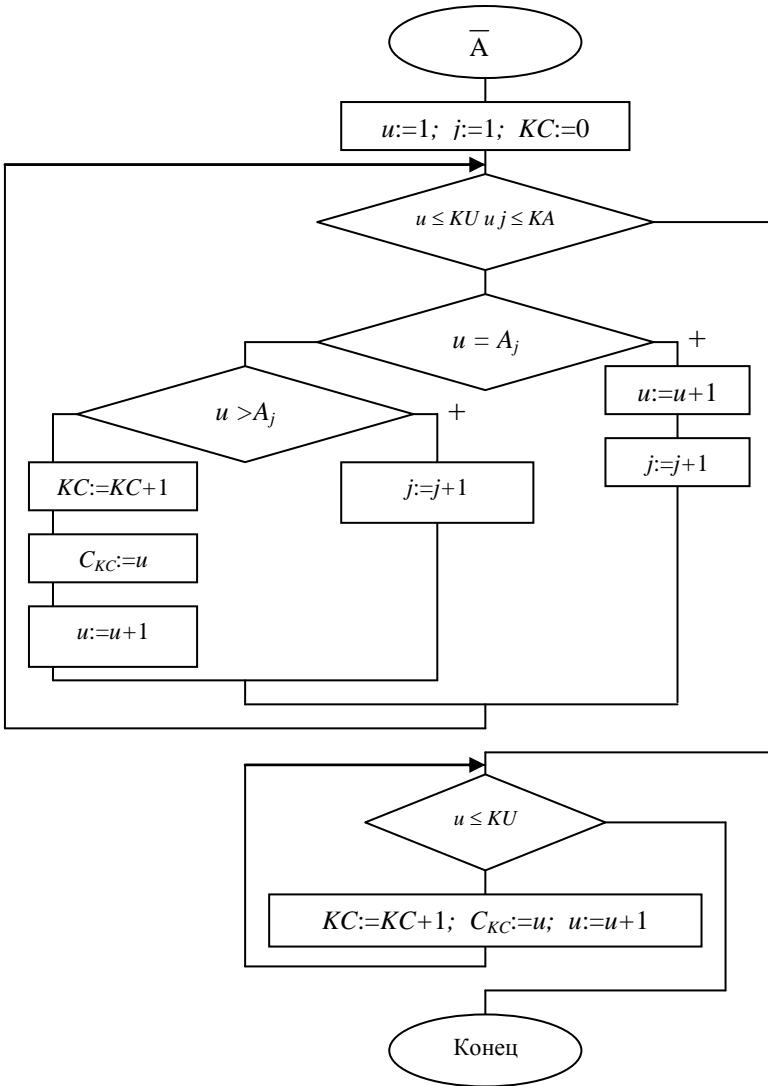


Рис.1.15. Блок-схема алгоритма вычисления дополнения A

3. Элементы универсума нумеруются: $U = \{u_1, \dots, u_n\}$. Элементы множества A хранятся в переменной A типа массив, элементы которого типа *boolean*. Если $u_i \in A$, то $A_i = true$, иначе $A_i = false$. Количество элементов в массиве A равно мощности универсума.

Алгоритм 1.15 (рис.1.16) вычисления равенства A и B ($A = B$).

Вход: A — массив, хранящий элементы множества A ;

B — массив, хранящий элементы множества B .

Выход: $F = true$, если $A = B$, иначе $F = false$.

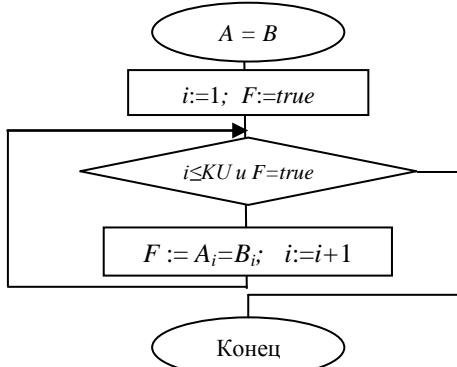


Рис.1.16. Блок-схема алгоритма вычисления равенства A и B

Алгоритм 1.16 (рис.1.17) вычисления включения A в B ($A \subseteq B$).

Вход: A — массив, хранящий элементы множества A ;

B — массив, хранящий элементы множества B .

Выход: $F = true$, если $A \subseteq B$, иначе $F = false$.

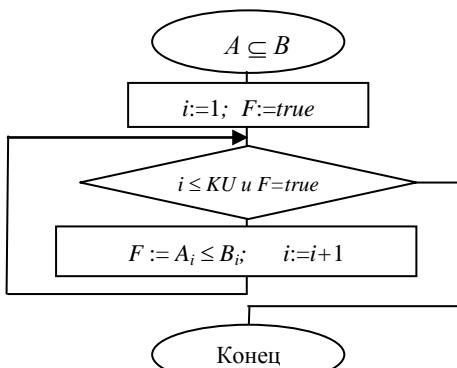


Рис.1.17. Блок-схема алгоритма вычисления включения A в B

Алгоритм I.17 (рис.1.18) вычисления объединения A и B ($A \cup B$).

Вход: A — массив, хранящий элементы множества A;
 B — массив, хранящий элементы множества B.

Выход: C — массив, хранящий объединение множеств A и B.

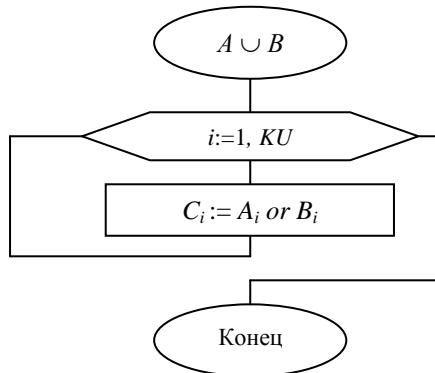


Рис.1.18. Блок-схема алгоритма вычисления объединения A и B

Алгоритм I.18 (рис.1.19) вычисления пересечения A и B ($A \cap B$).

Вход: A — массив, хранящий элементы множества A;
 B — массив, хранящий элементы множества B.

Выход: C — массив, хранящий пересечение множеств A и B.

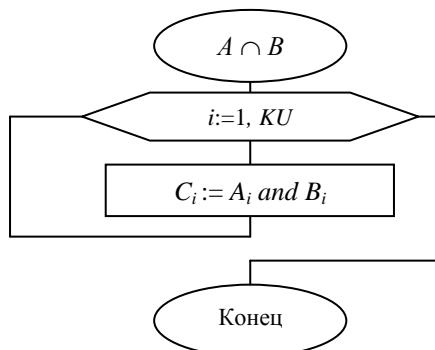


Рис.1.19. Блок-схема алгоритма вычисления пересечения A и B

Алгоритм 1.19 (рис.1.20) вычисления разности A и B ($A - B$).

Вход: A — массив, хранящий элементы множества A ;

B — массив, хранящий элементы множества B .

Выход: C — массив, хранящий разность множеств A и B .

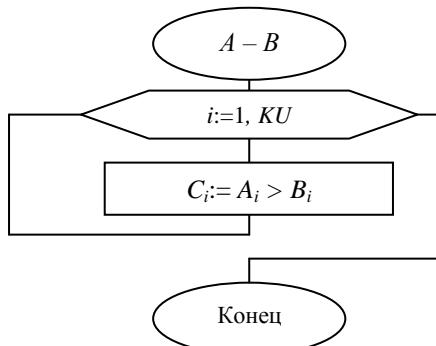


Рис.1.20. Блок-схема алгоритма вычисления разности A и B

Алгоритм 1.20 (рис1.21) вычисления симметрической разности A и B ($A \Delta B$).

Вход: A — массив, хранящий элементы множества A ;

B — массив, хранящий элементы множества B .

Выход: C — массив, хранящий симметрическую разность множеств A и B .

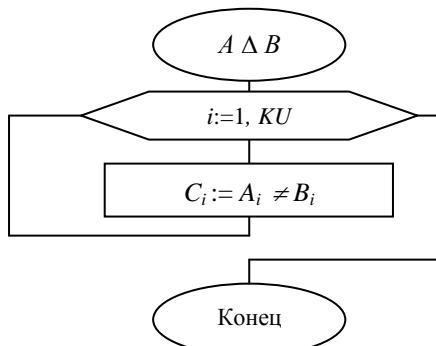


Рис.1.21. Блок-схема алгоритма вычисления симметрической разности A и B

Алгоритм 1.21 (рис.1.22) вычисления дополнения A (\bar{A}).
 Вход: A — массив, хранящий элементы множества A ;
 Выход: C — массив, хранящий дополнение множества A

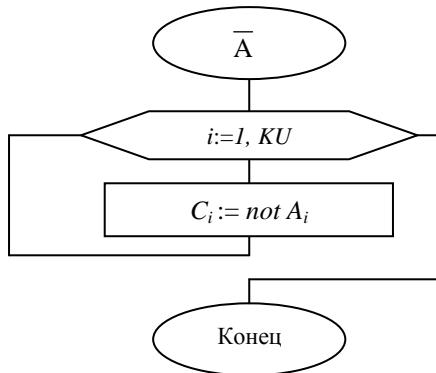


Рис.1.22. Блок-схема алгоритма вычисления дополнения A

4. Элементы универсума нумеруются: $U = \{u_1, \dots, u_n\}$. Множество A представляется кодом C , в котором: $C_i = 1$, если $u_i \in A$ и $C_i = 0$, если $u_i \notin A$, где C_i — i -й разряд кода C . В зависимости от мощности универсума код C может храниться в простой переменной или в массиве.

Алгоритмы реализации операций при таком способе представления множества в памяти ЭВМ можно получить путем замены логических операций на побитовые в предшествующих алгоритмах и корректировкой конечного значения параметра циклов.

5. Для хранения множества можно использовать множественный тип, если он имеется в языке программирования, и выполнять предусмотренные типом операции над множествами.

Практическое занятие 1.1

Операции над множествами

Цель занятия: изучить и научиться использовать операции над множествами, изучить различные способы представления множеств в памяти ЭВМ, научиться программно реализовывать операции над множествами и вычислять значения теоретико-множественных выражений.

Задания

1. Вычислить значение выражения (см. “Варианты заданий”, п. а).

Решение изобразить с помощью кругов Эйлера.

2. Записать теоретико-множественное выражение, значение которого при заданных множествах A , B и C равно множеству D (см. “Варианты заданий”, п. б).

3. Программно реализовать операции над множествами, используя следующие способы представления множества в памяти ЭВМ:

- а) элементы множества A хранятся в массиве A . Элементы массива A неупорядочены;
- б) элементы множества A хранятся в массиве A . Элементы массива A упорядочены по возрастанию;
- в) элементы множества A хранятся в массиве A , элементы которого типа *boolean*. Если $i \in A$, то $A_i = true$, иначе $A_i = false$.

4. Написать программы для вычисления значений выражений (см. “Задания”, п.1 и п.2).

5. Используя программы (см. “Задания”, п.4), вычислить значения выражений (см. “Задания”, п.1 и п.2).

Варианты заданий

Вариант 1.

- $$D = (B - A) \cup (A - (B \cup C)) \cup (C - A)$$
$$A = \{1,2,3,4,5,6,7\} \quad B = \{2,5,6,9,10\} \quad C = \{4,7,8,11,12\}$$
- $$A = \{1,2,8\} \quad B = \{6,7\} \quad C = \{2,3,4,5,7\} \quad D = \{3,4,5\}$$

Вариант 2.

- $$D = B \cap (A \Delta B) \cup (C - A)$$
$$A = \{2,5,6,7,9\} \quad B = \{1,4,5,9\} \quad C = \{3,7,8,10\}$$
- $$A = \{1,2,3,8\} \quad B = \{3,6,7\} \quad C = \{2,3,4,5,7\}$$
$$D = \{1,3,4,5,6,8\}$$

Вариант 3.

- a) $D = (A - (B \Delta C)) \cup ((B \Delta C) - A)$
 $A = \{1,2,4,5,8\} \quad B = \{2,3,5,6,9\} \quad C = \{4,5,6,7,9\}$
- b) $A = \{2,3,4,5,6\} \quad B = \{1,2,4,9\} \quad C = \{4,5,7,8\}$
 $D = \{3,6\}$

Вариант 4.

- a) $D = (A - (B \Delta C)) \cup ((B \cup C) - A)$
 $A = \{1,2,3,4,6,7\} \quad B = \{1,3,6,7\} \quad C = \{3,4,5,6,8\}$
- b) $A = \{1,2,3,8\} \quad B = \{3,6,7\} \quad C = \{2,3,4,5,7\}$
 $D = \{1,3,6,8\}$

Вариант 5.

- a) $D = (A \Delta C) \cap (B \cup C)$
 $A = \{1,2,3,4,8\} \quad B = \{2,3,8\} \quad C = \{3,4,5,6,7\}$
- b) $A = \{1,2,3,4,6,7\} \quad B = \{1,3,6,7\} \quad C = \{3,4,5,6,8\}$
 $D = \{1,5,7,8\}$

Вариант 6.

- a) $D = (A \cup B) - (B \Delta C) \cup (C - A)$
 $A = \{2,3,4,5,6\} \quad B = \{1,2,4,9\} \quad C = \{4,5,7,8\}$
- b) $A = \{1,2,4,5,8\} \quad B = \{2,3,5,6,9\} \quad C = \{4,5,6,7,9\}$
 $D = \{2,4,6,9\}$

Вариант 7.

- a) $D = (A \cap B) \cup (A - C) \cup (B - C)$
 $A = \{1,2,3,8\} \quad B = \{3,5,6,7\} \quad C = \{2,3,4,7\}$
- b) $A = \{1,2,3,4,5,6,7\} \quad B = \{2,5,6,9,10\} \quad C = \{4,7,8,11,12\}$
 $D = \{2,5,6,7,4\}$

Вариант 8.

- a) $D = ((A \cup B) - C) \cup (A \cap B)$
 $A = \{1,2,3,8\} \quad B = \{3,6,7\} \quad C = \{2,3,4,5,7\}$
- b) $A = \{1,2,3,4,5,6,7\} \quad B = \{2,5,6,9,10\} \quad C = \{4,7,8,11,12\}$
 $D = \{1,3,8,9,10,11,12\}$

Вариант 9.

- a) $D = ((A \Delta B) - C) \cup (C - A)$
 $A = \{1,2,3,4,5,6,7\} \quad B = \{2,5,6,9,10\} \quad C = \{4,7,8,11,12\}$
- b) $A = \{2,5,6,7,9\} \quad B = \{1,4,5,9\} \quad C = \{3,7,8,10\}$
 $D = \{1,3,4,8,10\}$

Вариант 10.

- a) $D = A \cap C - B \cup B \cap (A \Delta C)$
 $A = \{1,2,4,5,8\} \quad B = \{2,3,5,6,9\} \quad C = \{4,5,6,7,9\}$
- b) $A = \{1,2,3,4,6,7\} \quad B = \{1,3,6,7\} \quad C = \{3,4,5,6,8\}$
 $D = \{2,5,8\}$

Вариант 11.

- a) $D = (B - C) \cup (C - A)$
 $A = \{1,2,3,4,6,7\} \quad B = \{1,3,6,7\} \quad C = \{3,4,5,6,8\}$
- b) $A = \{1,2,3,8\} \quad B = \{3,5,6,7\} \quad C = \{2,3,4,7\}$
 $D = \{1,3,5,6,8\}$

Вариант 12.

- a) $D = ((A - C) \Delta B) \cup (C - A)$
 $A = \{1,2,3,4,8\} \quad B = \{2,3,8\} \quad C = \{3,4,5,6,7\}$
- b) $A = \{2,3,4,5,6\} \quad B = \{1,2,4,9\} \quad C = \{4,5,7,8\}$
 $D = \{3,4,6,7,8\}$

Вариант 13.

- a) $D = A - (C \Delta B) - (B \cup C)$
 $A = \{2,3,4,5,6\} \quad B = \{1,2,4,9\} \quad C = \{4,5,7,8\}$
- b) $A = \{1,2,4,5,8\} \quad B = \{2,3,5,6,9\} \quad C = \{4,5,6,7,9\}$
 $D = \{1,3,5,7,8\}$

Вариант 14.

- a) $D = C - (A \Delta B) \cup (A \Delta B - C)$
 $A = \{1,2,3,8\} \quad B = \{3,6,7\} \quad C = \{2,3,4,5,7\}$
- b) $A = \{1,2,3,4,8\} \quad B = \{2,3,8\} \quad C = \{3,4,5,6,7\}$
 $D = \{2,5,6,7,8\}$

Вариант 15.

- a) $D = C - (A \cap C \cup B \cap C)$
 $A = \{1,2,8\} \quad B = \{6,7\} \quad C = \{2,3,4,5,7\}$
- b) $A = \{1,2,3,4,8\} \quad B = \{2,3,8\} \quad C = \{3,4,5,6,7\}$
 $D = \{3,5,6,7\}$

Практическое занятие 1.2

Нормальные формы Кантора

Цель занятия: изучить способы получения различных нормальных форм Кантора множества, заданного произвольным теоретико-множественным выражением.

Задания

1. Представить множество, заданное исходным выражением (см. табл. 1.4), в нормальной форме Кантора.
2. Получить совершенную нормальную форму Кантора множества, заданного исходным выражением.
3. Получить сокращенную нормальную форму Кантора множества, заданного исходным выражением.
4. Получить тупиковые нормальные формы Кантора множества, заданного исходным выражением. Выбрать минимальную нормальную форму Кантора.

Таблица 1.4

Варианты заданий

Вариант	Исходное выражение
1	$(\overline{B\Delta(A \cup D) \cup C}) - B\Delta\overline{A}$
2	$D \cap (D - C) \cup A - B - C\Delta(B - A)$
3	$(\overline{A\Delta\overline{C}} - B\Delta(C - A)\Delta B) \cap D$
4	$A \cup C - B \cap (D - C) \Delta (D - A)$
5	$\overline{D \cap (D - A) \cup C - B\Delta(C - A)}$
6	$C\Delta(D - A) \cup B - C \cap (B - A) - D$
7	$A - (C\Delta A) \cup B - D\Delta(C - B)$
8	$(C - A)\Delta\overline{A} \cap (B - D) - B \cap \overline{C}$
9	$B \cup A - C \cap (D - A) \Delta (D \cup A)$
10	$((C \cup B) - D\Delta(C - B) \Delta A) \cap A$
11	$(C\Delta A) \cap A \cup D\Delta(C \cup B) - B$
12	$\overline{C\Delta A} - (A\Delta B) \cup D\Delta(B - C)$
13	$(A\Delta(B \cup C) \cup D - B\Delta A) \cap C$
14	$(D \cup C - B\Delta(B \cup A) \Delta C) \cap A$
15	$A - (B\Delta C) \cup D\Delta(B - A) \Delta C$
16	$(B\Delta(A - C) \cup D) - A \cap (B - C)$

Практическое занятие 1.3

Теоретико-множественные тождества

Цель занятия: изучить методы доказательства теоретико-множественных тождеств.

Задания

1. На рис.1.23 изображены круги Эйлера, соответствующие множествам A , B и C , с пронумерованными элементарными областями (не содержащими внутри себя других областей). Заштриховать элементарные области в соответствии с вариантом задания (см. табл.1.5).

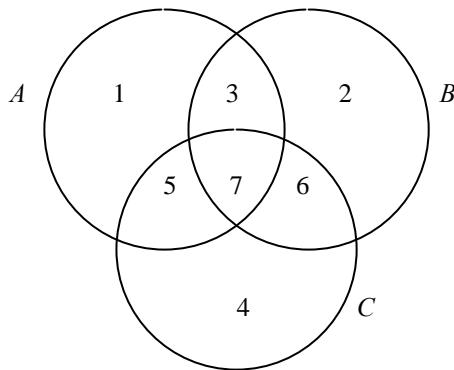


Рис.1.23. Круги Эйлера, соответствующие множествам A , B и C с пронумерованными элементарными областями

2. Написать выражение 1 над множествами A , B и C , определяющее заштрихованную область, используя операции пересечения, объединения и дополнения.
3. Используя свойства операций над множествами, преобразовать выражение 1 в выражение 2, не содержащее операции дополнения множества.
4. Используя свойства операций над множествами, преобразовать выражение 2 в выражение 3, не содержащее операции объединения множеств.

5. Используя свойства операций над множествами, преобразовать выражение 3 в выражение 4, не содержащее операции пересечения множеств.

6. Доказать тождественность выражений 2 и 3 методом характеристических функций.

7. Доказать тождественность выражений 2 и 4 методом логических функций. Для автоматизации доказательства написать программу, которая получает и сравнивает таблицы истинности логических функций.

8. Доказать тождественность выражений 3 и 4 теоретико-множественным методом. Для автоматизации доказательства написать программу, в которой вычисляются и сравниваются значения выражений 3 и 4 при $A = \{1,3,5,7\}$, $B = \{2,3,6,7\}$ и $C = \{4,5,6,7\}$.

Таблица 1.5

Варианты заданий

Вариант	Номера областей
1	1, 2, 3
2	1, 2, 4
3	1, 2, 7
4	1, 3, 4
5	1, 3, 7
6	1, 4, 6
7	1, 5, 6
8	1, 5, 7
9	1, 6, 7
10	2, 3, 4
11	2, 3, 7
12	2, 4, 6
13	2, 5, 7
14	2, 6, 7
15	3, 4, 5
16	3, 4, 6
17	3, 5, 6
18	4, 5, 6
19	4, 5, 7
20	4, 6, 7

Практическое занятие 1.4

Теоретико-множественные уравнения

Цель занятия: научиться решать теоретико-множественные уравнения с применением ЭВМ.

Задания

1. Преобразовать исходное уравнение (см. “Варианты заданий”) в уравнение с пустой правой частью.
2. Преобразовать левую часть уравнения к виду $\overline{X} \cap \varphi^\emptyset \cup X \cap \varphi^U$, используя разложение Шеннона по неизвестному множеству X .
3. Написать программу, вычисляющую значения множеств φ^\emptyset и $\overline{\varphi^U}$ при заданных исходных множествах.
4. Вычислить значения множеств φ^\emptyset и $\overline{\varphi^U}$ и сделать вывод о существовании решения уравнения. Если решения уравнения не существует, то выполнить п.п. 1 — 4 для следующего (предыдущего) варианта.
5. Определить мощность общего решения, найти некоторые (или все) частные решения, в том числе частные решения наименьшей и наибольшей мощности.
6. Написать программу для проверки найденных решений.

Варианты заданий

Вариант 1.

$$A \Delta B \cap \overline{X \cap C} = \overline{A - X} \Delta B \cap X$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{3, 4, 5, 6, 8, 10\}$$

$$B = \{1, 2, 7, 8, 9, 10\}$$

$$C = \{1, 2, 3, 4, 5, 10\}$$

$$X — ?$$

Вариант 2.

$$(A - X) \Delta B \cap X = \overline{A - \overline{X}} \Delta (C - X)$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{3, 7, 9, 10\}$$

$$B = \{1, 3, 8, 9, 10\}$$

$$C = \{2, 4, 5, 6, 7\}$$

$$X — ?$$

Вариант 3.

$$A - (B - (C - X)) = A - (B \cap (C - X))$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 5, 7\}$$

$$B = \{2, 4, 6, 10\}$$

$$C = \{1, 3, 5, 6, 8, 10\}$$

$$X — ?$$

Вариант 4.

$$A - (B - (C - X)) = B - (C - (A - X))$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 5, 7\}$$

$$B = \{2, 4, 6, 10\}$$

$$C = \{1, 3, 5, 6, 8, 10\}$$

$$X — ?$$

Вариант 5.

$$A \cap X \cup (X - B) = X \cap \overline{B} - C$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{2, 3, 4, 7, 10\}$$

$$B = \{1, 2, 6, 9\}$$

$$C = \{3, 5, 7, 9\}$$

$$X — ?$$

Вариант 6.

$$A - (X \cap B) \cup (C \cap \overline{X}) = (X - A) \cap (X - B) - C$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{4, 8, 9\}$$

$$B = \{1, 2, 4, 6, 8, 9\}$$

$$C = \{4, 5, 7, 9\}$$

$$X — ?$$

Вариант 7.

$$(X \cup A) \cap (X - B) \cup C = A \cap \overline{X} \Delta (C - X)$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{2, 9, 10\}$$

$$B = \{1, 2, 8, 9, 10\}$$

$$C = \{1, 3, 6, 7\}$$

$$X — ?$$

Вариант 8.

$$\overline{X \cap A} - X \cap \overline{B} = \overline{C \cap X} \cap A \cup B$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{3, 4, 5, 6, 10\}$$

$$B = \{1, 2, 3, 7, 9\}$$

$$C = \{3, 4, 5, 8, 10\}$$

$$X — ?$$

Вариант 9.

$$(A \Delta X) \cap (B \cup X) - C = \overline{A \cup X} \cup (C \Delta X)$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 3, 5, 6, 8\}$$

$$B = \{2, 4, 6, 9\}$$

$$C = \{2, 6, 7, 10\}$$

$$X — ?$$

Вариант 10.

$$A \cap (B \cup X) \Delta C = A - (B \cap \overline{X}) \Delta C \cap X$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{4, 5, 7, 8, 9, 10\}$$

$$B = \{2, 3, 4, 5, 6, 10\}$$

$$C = \{4, 5, 7, 8, 10\}$$

$$X — ?$$

Вариант 11.

$$B \cap (\overline{X} \cup \overline{C}) \Delta A = (\overline{A} \cup X) \Delta B \cap X$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 3, 6, 7, 8, 10\}$$

$$B = \{1, 3, 4, 5, 9\}$$

$$C = \{1, 3, 6, 7, 10\}$$

$$X — ?$$

Вариант 12.

$$A - B \cap \overline{C \cap \overline{X}} = A \cap \overline{B \cap (C - X)}$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 4, 10\}$$

$$B = \{3, 5, 6, 7\}$$

$$C = \{3, 6, 7, 10\}$$

$$X — ?$$

Вариант 13.

$$A \cap \overline{B - X} \Delta X \cap C = A \cap \overline{B - (C - X)}$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{2, 3, 9, 10\}$$

$$B = \{1, 5, 8\}$$

$$C = \{4, 5, 7\}$$

$$X — ?$$

Вариант 14.

$$(A \cap \overline{X}) \Delta B \cap X \cup C = C \cap \overline{X} \Delta (A - X)$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 4, 8\}$$

$$B = \{1, 5, 7, 10\}$$

$$C = \{3, 6, 7, 10\}$$

$$X — ?$$

Вариант 15.

$$(A \cap X \cup B) \cap (C - X) = \overline{X} \cup (A - X) \cap C$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 2, 5, 8, 9\}$$

$$B = \{1, 2, 6, 7, 9, 10\}$$

$$C = \{2, 4, 6, 9, 10\}$$

$$X — ?$$

Вариант 16.

$$B \cap \overline{X \cap C} \Delta A = B \cap X \Delta (\overline{A} \cup X)$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{1, 3, 6, 7, 8, 10\}$$

$$B = \{1, 3, 4, 5, 9\}$$

$$C = \{1, 3, 6, 7, 10\}$$

$$X — ?$$

Вариант 17.

$$(C - X) \Delta \overline{A - \overline{X}} = (A - X) \Delta B \cap X$$

$$U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

$$A = \{3, 7, 9, 10\}$$

$$B = \{1, 3, 8, 9, 10\}$$

$$C = \{2, 4, 5, 6, 7\}$$

$$X — ?$$

Контрольные вопросы и задания

1. Дать определение понятиям:
 - множество, элемент множества;
 - конечное множество, мощность множества;
 - бесконечное, счетное и несчетное множество;
 - пустое множество, универсум;
 - подмножество, собственное подмножество;
 - булеван, операции над множествами.
2. Привести примеры задания множества различными способами.
3. Проиллюстрировать операции над множествами и свойства операций диаграммами Эйлера.
4. Вычислить значение теоретико-множественного выражения.
5. Определить количество подмножеств заданного множества.
6. Дать определение понятиям:
 - первичный терм;
 - элементарное пересечение;
 - нормальная форма кантора;
 - конституента;
 - совершенная НФК;
 - простая импликанта;
 - сокращенная НФК;
 - тупиковая НФК;
 - минимальная НФК.
7. Как получить сокращенную НФК?
8. Что такое импликантная матрица Квайна? Для чего она используется?
9. Доказать теоретико-множественное тождество различными методами.
10. Дать определение теоретико-множественному уравнению.
11. Что такое частное и общее решение теоретико-множественного уравнения?
12. Что является условием существования решения теоретико-множественного уравнения?
13. Как определить мощность общего решения теоретико-множественного уравнения?
14. Сравнить способы представления множества в памяти ЭВМ (объем памяти, время выполнения операций).
15. Программно реализовать заданную операцию над множествами при заданном способе представления множества в памяти ЭВМ.
16. Изменить алгоритмы 1.19 и 1.20, используя логические операции вместо операций сравнения.

2. КОМБИНАТОРНЫЕ ОБЪЕКТЫ

2.1. Введение

Под *комбинаторным объектом* будем понимать набор, составленный из элементов множества и обладающий свойством, определяющим заданный комбинаторный объект.

Основными простейшими комбинаторными объектами являются подмножество, перестановка, размещение, сочетание, разбиение. При изучении комбинаторного объекта необходимо:

- подсчитать количество всех различных комбинаторных объектов заданного типа (обладающих заданным свойством), которые можно составить из элементов заданного множества;

- определить способ (алгоритм) получения всех различных комбинаторных объектов заданного типа, которые можно составить из элементов заданного множества.

При подсчете количества всех различных комбинаторных объектов заданного типа часто применяется *правило произведения*, которое заключается в следующем.

Пусть требуется выполнить одно за другим K действий. Если первое действие можно выполнить N_1 способами, второе действие — N_2 способами, третье действие — N_3 способами и так до K -го действия, которое можно выполнить N_k способами, то все K действий могут быть выполнены $N_1 \cdot N_2 \cdot N_3 \cdots N_k$ способами.

2.2. Метод поиска с возвращением

Пусть задача имеет множество решений, каждое из которых можно представить последовательностью R конечной, но, в общем случае, неопределенной длины, состоящей из элементов, принадлежащих некоторому множеству A , и требуется найти все решения задачи. Их можно найти, используя метод поиска с возвращением, который заключается в следующем.

Для получения одного решения будем последовательно, элемент за элементом, формировать последовательность R . Сначала определим подмножество элементов множества A , которые можно поставить на 1-е место и один из них поставим на 1-е место. Затем, по аналогии, будем определять и ставить элементы на 2-е, 3-е и так далее место, продвигаясь по последовательности R вперед, т.е. делать шаги вперед. Так, двигаясь вперед, на некотором i -м шаге мы либо получим одно из ре-

шений (случай 1), либо обнаружим, что во множестве A нет ни одного элемента, который можно поставить на очередное место, не нарушая условие задачи (случай 2). Случай 2 означает, что ни одно решение задачи не содержит в качестве первых $i - 1$ элементов элементы, поставленные в последовательность R . В этой ситуации нужно сделать шаг назад, к $i - 1$ -му месту и попытаться поставить другой допустимый элемент на $i - 1$ -е место. В случае 1, когда решение получено, для продолжения поиска других решений необходимо попытаться поставить другой допустимый элемент на i -е место, а если такого элемента нет, то нужно сделать шаг назад. Если нужно выполнить шаг назад тогда, когда находимся на 1-м месте в последовательности R , то поиск решений прекращается, все решения найдены.

Метод поиска с возвращением можно реализовать на ЭВМ, используя рекурсивный или итеративный алгоритмы, структуры которых представлены на рис.2.1 и 2.2 соответственно.

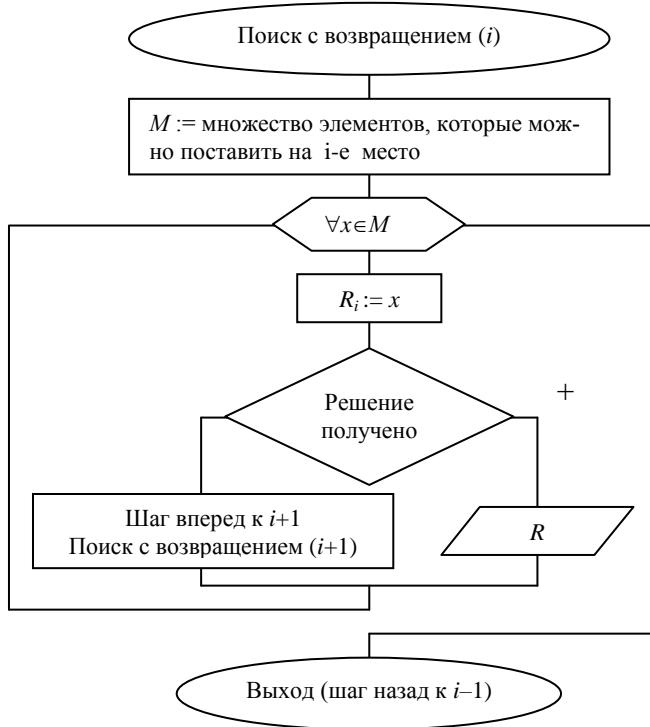


Рис.2.1. Рекурсивный алгоритм поиска с возвращением

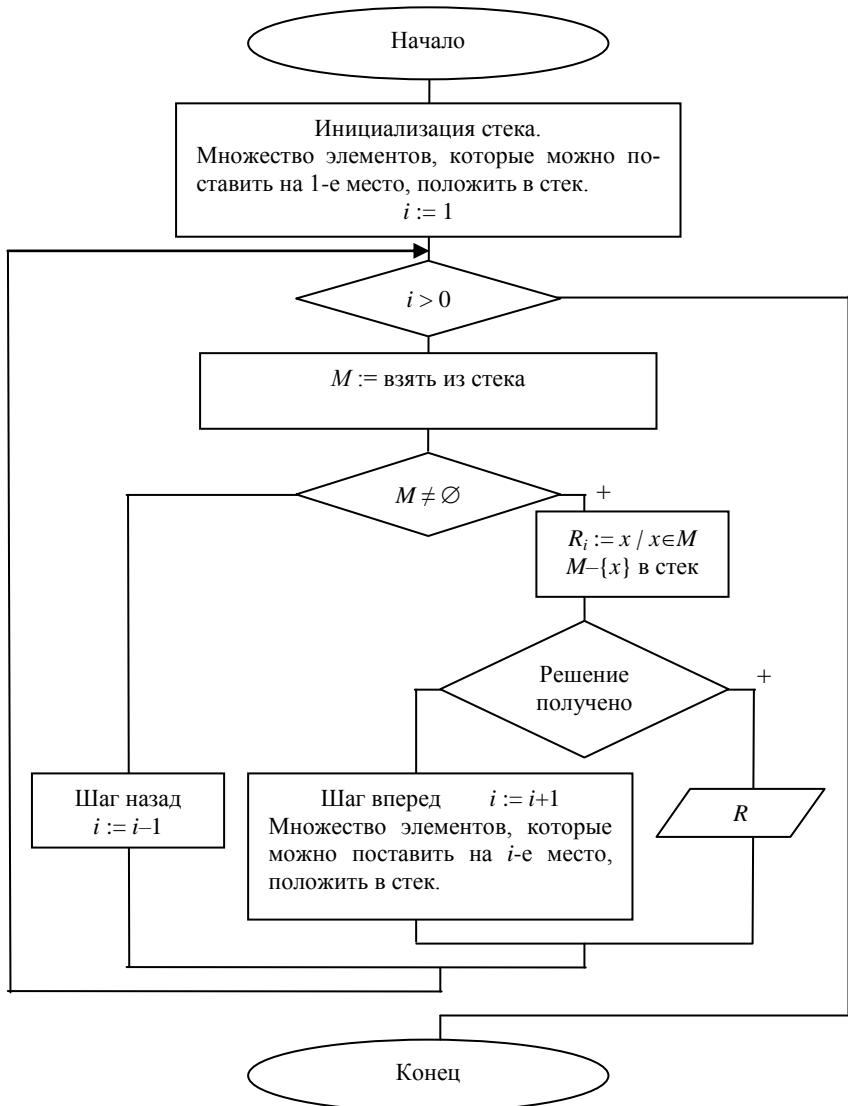


Рис.2.2. Итеративный алгоритм поиска с возвращением

Рекурсивная подпрограмма формирует i -й элемент последовательности R , поэтому при первом обращении к ней нужно передать параметр, равный 1.

Метод поиска с возвращением можно также использовать, если требуется определить, существует ли решение задачи, подсчитать количество возможных решений или найти одно из возможных решений.

2.3. Подмножества

Множество A называется *подмножеством* множества B , если истинно включение A в B ($A \subseteq B$ = истина).

Теорема 2.1. Количество всех различных подмножеств n -элементного множества равно 2^n .

Доказательство. Поставим в соответствие каждому подмножеству A множества B n -разрядный двоичный вектор, i -й разряд которого равен единице только тогда, когда i -й элемент множества B принадлежит подмножеству A . Для того, чтобы сформировать n -разрядный двоичный вектор, нужно выполнить одно за другим n действий: заполнить 1-й разряд вектора, 2-й разряд и так до n -го разряда. Каждое действие можно выполнить двумя способами (разряд двоичного вектора можно заполнить только нулем или единицей). По правилу произведения все n действий могут быть выполнены $2 \cdot 2 \cdot \dots \cdot 2 = 2^n$ способами, т.е. может быть получено 2^n различных n -разрядных двоичных векторов, следовательно, количество всех различных подмножеств n -элементного множества равно 2^n .

Для получения (порождения) всех подмножеств заданного множества B достаточно сформировать все двоичные векторы, разрядность которых равна $n=|B|$. Можно сформировать все двоичные n -разрядные вектора, используя метод поиска с возвращением, выполняя действия, описанные в доказательстве теоремы 2.1. Формирование i -го разряда опишем рекурсивным алгоритмом 2.1, блок-схема которого представлена на рис.2.3. В цикле выполняем один из способов заполнения i -го разряда. Если заполнен последний разряд, то вектор сформирован и выводим его, иначе заполняем следующий разряд по алгоритму 2.1.

Алгоритм 2.1 порождения n -разрядных двоичных векторов.

Вход: i — формируемый разряд вектора D .

Выход: последовательность всех n -разрядных двоичных векторов.

Глобальные параметры: D — формируемый вектор;

n — разрядность вектора.

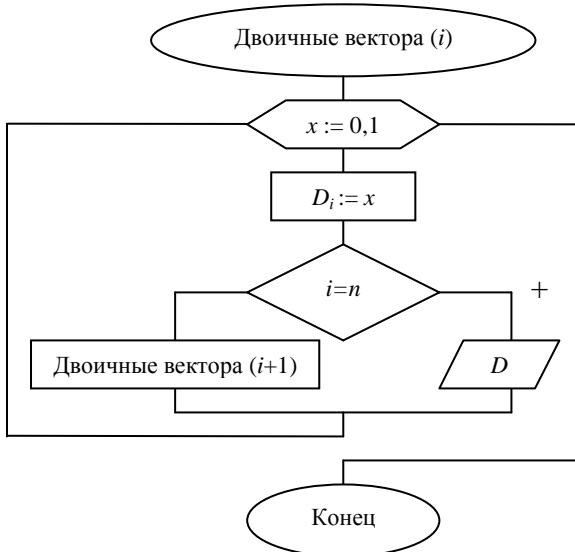


Рис.2.3. Рекурсивный алгоритм порождения n -разрядных двоичных векторов

Пусть задано множество $\{1,2,3\}$ и требуется получить все его подмножества. Схема получения всех двоичных трехразрядных векторов по алгоритму 2.1 и соответствующих им подмножеств заданного множества представлена на рис.2.4. Самый правый (первый) разряд вектора соответствует элементу 1, средний (второй) — элементу 2, самый левый (третий) — элементу 3. В исходной ситуации все разряды двоичного вектора неопределены. В схеме к стрелкам приписаны номера действий. После последовательного выполнения действий 1, 2 и 3 заполняются нулями третий, второй и первый разряды и вектор $(0,0,0)$, соответствующий пустому подмножеству, сформирован. После выполнения действия 4 заполняется единицей первый разряд и вектор $(0,0,1)$, соответствующий подмножеству $\{1\}$, сформирован. Затем выполняются два шага назад и действие 5, заполняющее единицей второй разряд вектора. Далее, после выполнения действия 6, заполняется нулем первый разряд и вектор $(0,1,0)$, соответствующий подмножеству $\{2\}$, сформирован. Аналогичным образом формируются все остальные двоичные вектора. Действия 1 и 8 заполняют третий разряд вектора, действия 2, 5, 9, 12 — второй разряд, действия 3, 4, 6, 7, 10, 11, 13, 14 — первый разряд.

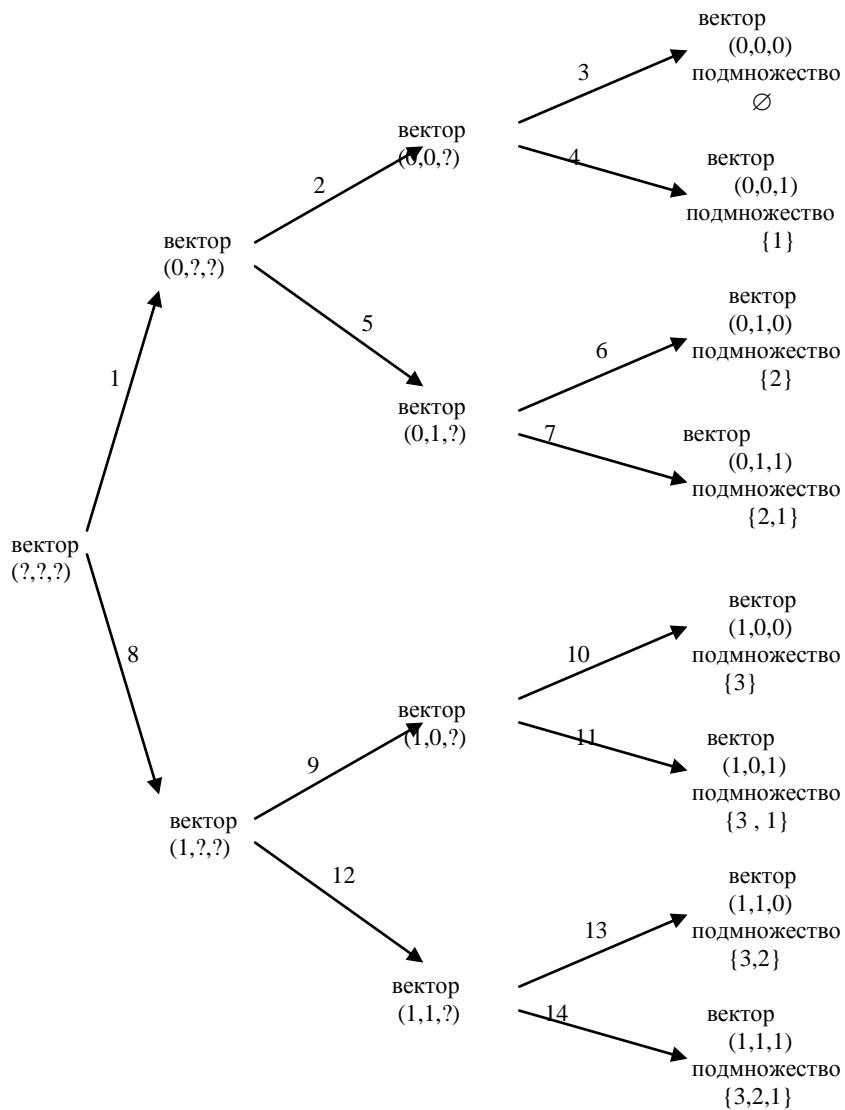


Рис.2.4. Схема порождения двоичных векторов по алгоритму 2.1

2.4. Перестановки

Перестановкой называется расположенные в определенном порядке элементы множества M .

Теорема 2.2. Количество всех различных перестановок n -элементного множества M (количество способов упорядочивания множества) определяется формулой $P_n = n!$.

Доказательство. Перестановку можно представить последовательностью из n мест. Для того чтобы получить одну перестановку, нужно выполнить одно за другим n действий: заполнить 1-е место в последовательности, 2-е место и так до n -го места. Для выполнения 1-го действия (заполнения 1-го места) можно взять любой элемент из множества M и поставить его на 1-е место, т.е. его можно выполнить n -способами, и после этого в множестве M останется $n - 1$ элемент. Для выполнения 2-го действия (заполнения 2-го места) можно взять любой элемент из оставшихся в множестве M и поставить его на 2-е место, т.е. его можно выполнить $n - 1$ -способами, и после этого в множестве M останется $n - 2$ элемента и т.д. По правилу произведения все n действий могут быть выполнены $n \cdot (n - 1) \cdot (n - 2) \cdots 2 \cdot 1 = n!$ способами, следовательно, количество всех различных перестановок n -элементного множества равно $n!$.

Получить все перестановки n -элементного множества M можно, используя метод поиска с возвращением, выполняя действия, описанные в доказательстве теоремы 2.2. Формирование i -го элемента перестановки опишем рекурсивным алгоритмом 2.2, блок-схема которого представлена на рис.2.5. В цикле перебираются элементы множества M , которые можно поставить на i -е место. Если заполнено последнее место, то перестановка сформирована и выводим ее, иначе заполняем следующее $i + 1$ -е место по алгоритму 2.2 элементами множества M , за исключением элемента, поставленного на i -е место.

Алгоритм 2.2 порождения перестановок n -элементного множества.

Вход: M — множество, элементы которого можно поставить на i -е место;

i — заполняемое место в последовательности P .

Выход: последовательность всех перестановок n -элементного множества.

Глобальные параметры: P — формируемая перестановка;

n — мощность множества.

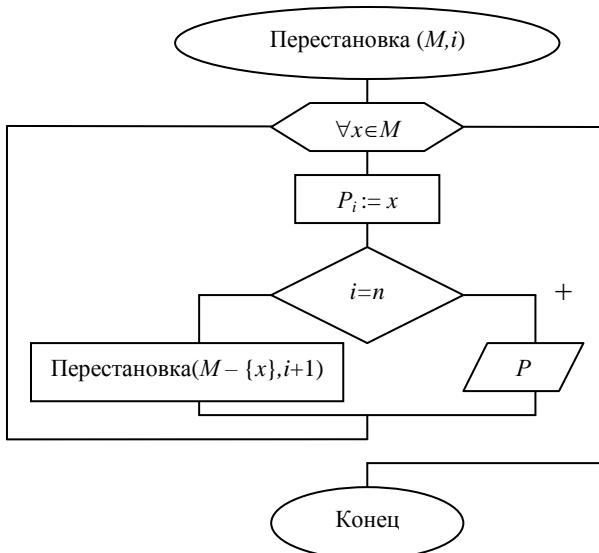


Рис.2.5. Рекурсивный алгоритм порождения перестановок n -элементного множества

Пусть задано множество $\{1,2,3\}$ и требуется получить все его перестановки. Схема получения всех перестановок по алгоритму 2.2 представлена на рис.2.6. В этой схеме представлены формируемые перестановки и множества, элементы которых можно поставить на очередное место в перестановке. В исходной ситуации все места перестановки неопределены и любой элемент множества $\{1,2,3\}$ можно поставить на первое место. В схеме к стрелкам приписаны номера действий. В результате выполнения действия 1 на первое место перестановки устанавливается элемент 1, а на второе место после этого можно поставить любой элемент множества $\{2,3\}$. После выполнения действия 2 на второе место перестановки устанавливается элемент 2, а на третье место после этого можно поставить только элемент множества $\{3\}$ (действие 3) и первая перестановка $(1,2,3)$ сформирована. Затем выполняются два шага назад и действие 4, ставящее элемент 3 из множества $\{2,3\}$ на второе место. Далее действие 5 ставит оставшийся элемент 2 на третье место и вторая перестановка $(1,3,2)$ сформирована. Аналогичным образом формируются все остальные перестановки. Действия 1, 6 и 11 заполняют первое место перестановки, действия 2, 4, 7, 9, 12 и 14 — второе место, действия 3, 8, 10, 13 и 15 — третье место.

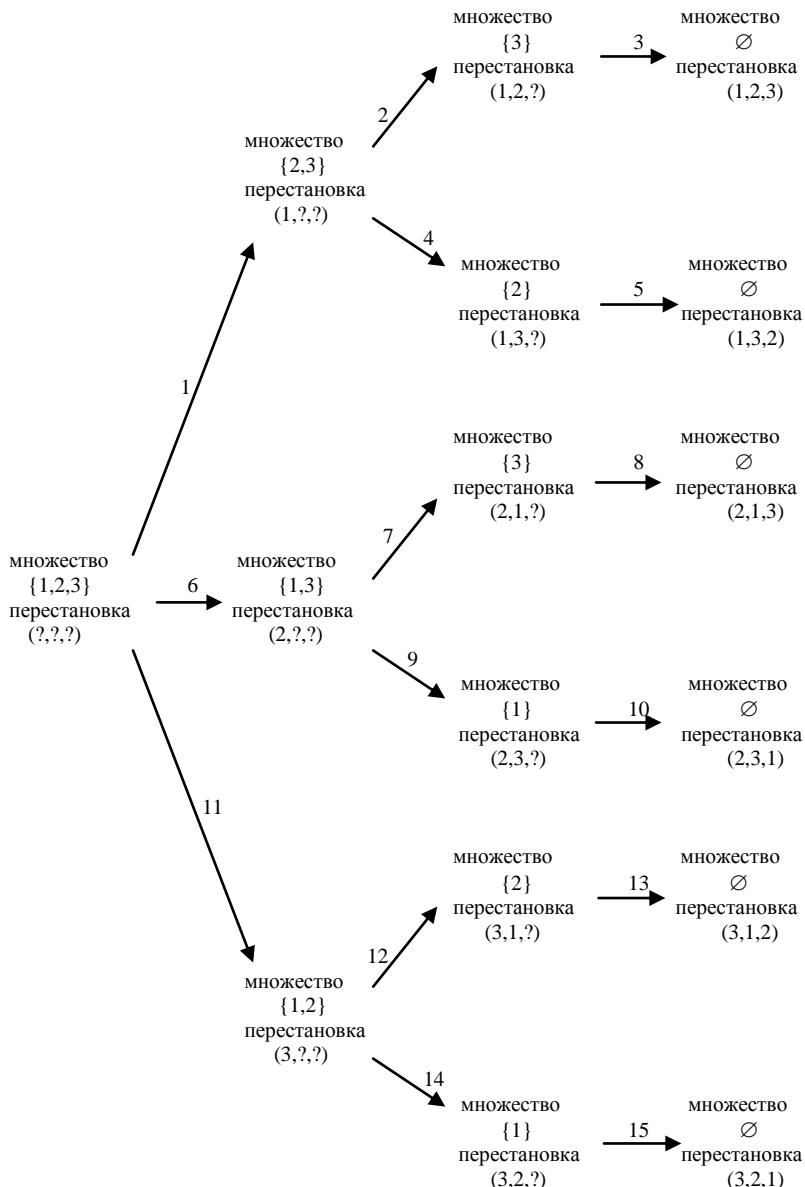


Рис.2.6. Схема получения всех перестановок по алгоритму 2.2

2.5. Размещения

Размещением n -элементного множества по k местам называется расположенные в определенном порядке элементы k -элементного подмножества n -элементного множества. Количество мест не превышает количества элементов в множестве.

Теорема 2.3. Количество всех различных размещений n -элементного множества M по k местам определяется формулой

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

Доказательство. Размещение можно представить последовательностью из k мест. Для того чтобы получить одно размещение, нужно выполнить одно за другим k действий: заполнить 1-е место в последовательности, 2-е место и так до k -го места. Для выполнения 1-го действия (заполнения 1-го места) можно взять любой элемент из множества M и поставить его на 1-е место, т.е. его можно выполнить n -способами, и после этого в множестве M останется $n - 1$ элемент. Для выполнения 2-го действия (заполнения 2-го места) можно взять любой элемент из оставшихся в множестве M и поставить его на 2-е место, т.е. его можно выполнить $n - 1$ -способами, и после этого в множестве M останется $n - 2$ элемента и так далее до k -го места. По правилу произведения все k действий могут быть выполнены

$$n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n-k+1) = \frac{n!}{(n-k)!} \text{ способами, следова-}$$

тельно, количество всех различных размещений n -элементного множества M по k местам равно $\frac{n!}{(n-k)!}$.

Получить все размещения n -элементного множества M по k местам можно, используя метод поиска с возвращением, выполняя действия, описанные в доказательстве теоремы 2.3. Формирование i -го элемента размещения опишем рекурсивным алгоритмом 2.3, блок-схема которого представлена на рис.2.7. В цикле перебираются элементы множества M , которые можно поставить на i -е место. Если заполнено k -е место, то размещение сформировано и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму 2.3 элементами множества M , за исключением элемента, поставленного на i -е место.

Алгоритм 2.3 порождения размещений n -элементного множества по k местам.

Вход: M — множество, элементы которого можно поставить на i -е место;

i — заполняемое место в последовательности P .

Выход: последовательность всех размещений n -элементного множества по k местам.

Глобальные параметры: A — формируемое размещение;
 k — количество мест.

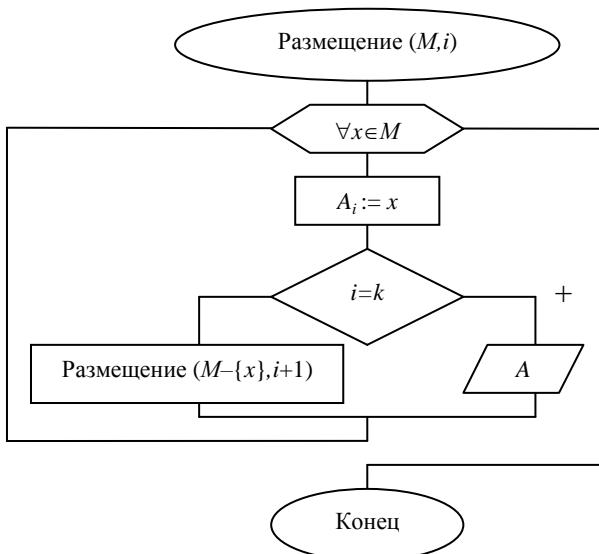


Рис.2.7. Рекурсивный алгоритм порождения размещений n -элементного множества по k местам

Пусть задано множество $\{1,2,3,4\}$ и требуется получить все его размещения по двум местам. Схема получения всех размещений по алгоритму 2.3 представлена на рис.2.8. В исходной ситуации все места размещения неопределены. В схеме к стрелкам приписаны номера действий, которые выполняются последовательно в порядке нумерации. Действия 1, 5, 9 и 13 заполняют первое место размещения, действия 2, 3, 4, 6, 7, 8, 10, 11, 12, 14, 15 и 16 — второе (последнее) место размещения.

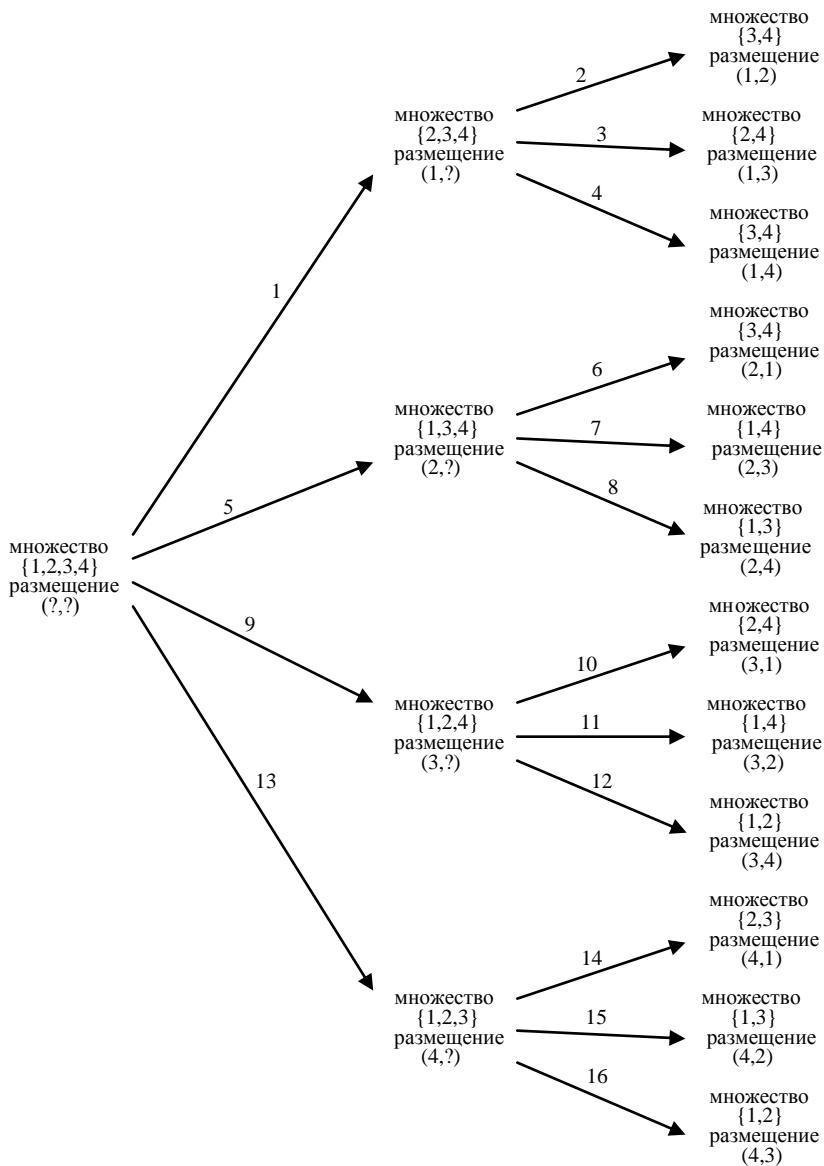


Рис.2.8. Схема получения всех размещений по алгоритму 2.3

2.6. Размещения с повторениями

Размещением с повторениями n -элементного множества по k местам называется k -элементная последовательность, состоящая из элементов n -элементного множества M , причем элементы в последовательности могут повторяться. Количество мест может превышать количество элементов в множестве.

Теорема 2.4. Количество всех различных размещений с повторениями n -элементного множества M по k местам равно n^k .

Доказательство. Для того чтобы получить одно размещение с повторениями, нужно выполнить одно за другим k действий: заполнить 1-е место в последовательности, 2-е место и так до k -го места. Для выполнения каждого действия можно взять любой элемент из множества M и поставить его на соответствующее место, т.е. каждое из k действий можно выполнить n способами. По правилу произведения все k действий могут быть выполнены $n \cdot n \cdot \dots \cdot n = n^k$ способами, следовательно, количество всех различных размещений с повторениями n -элементного множества по k местам равно n^k .

Получить все размещения с повторениями n -элементного множества M по k местам можно, используя метод поиска с возвращением, выполняя действия, описанные в доказательстве теоремы 2.4. Формирование i -го элемента размещения с повторениями опишем рекурсивным алгоритмом 2.4, блок-схема которого представлена на рис.2.9. В цикле перебираются элементы множества M (каждый элемент можно поставить на i -е место). Если заполнено k -е место, то размещение с повторениями сформировано и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму 2.4 элементами множества M .

Алгоритм 2.4 порождения размещений с повторениями n -элементного множества по k местам.

Вход: i — заполняемое место в последовательности R .

Выход: последовательность всех размещений с повторениями n -элементного множества M по k местам.

Глобальные параметры: M — n -элементное множество;

R — формируемое размещение с повторениями;

k — количество мест.

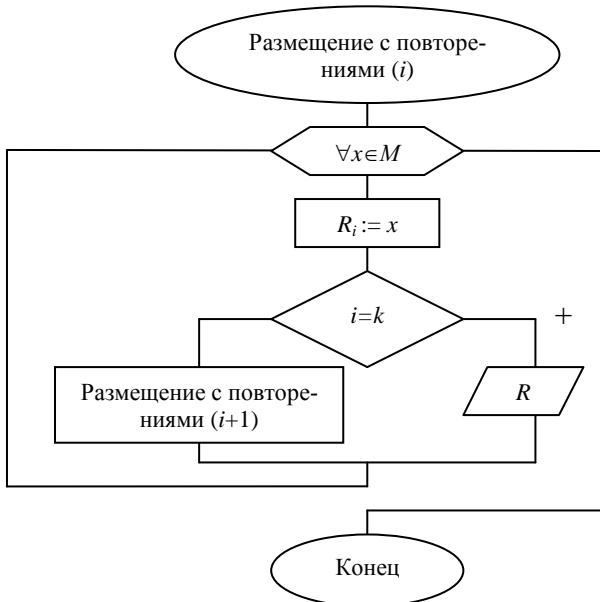


Рис.2.9. Рекурсивный алгоритм порождения размещений с повторениями n -элементного множества по k местам

Пусть задано множество $\{1,2,3\}$ и требуется получить все его размещения с повторениями по двум местам. Схема получения всех размещений по алгоритму 2.4 представлена на рис.2.10. В этой схеме представлены формируемые размещения. В каждой ситуации на очередное место в размещении можно поставить любой элемент из заданного множества $\{1,2,3\}$. В исходной ситуации все места размещения неопределены и любой элемент множества $\{1,2,3\}$ можно поставить на первое место. В схеме к стрелкам приписаны номера действий. В результате выполнения действия 1 на первое место размещения устанавливается элемент 1. После выполнения действия 2 на второе место размещения также устанавливается элемент 1 и первое размещение $(1,1)$ сформировано. Затем выполняется шаг назад и действие 3 ставит элемент 2 на второе место и второе размещение $(1,2)$ сформировано. Далее выполняется шаг назад и действие 4 заканчивает формирование очередного размещения $(1,3)$. После этого выполняется два шага назад. Аналогичным образом формируются все остальные размещения. Действия 1, 5 и 9 заполняют первое место размещения, действия 2, 3, 4, 6, 7, 8, 10, 11 и 12 — второе место.

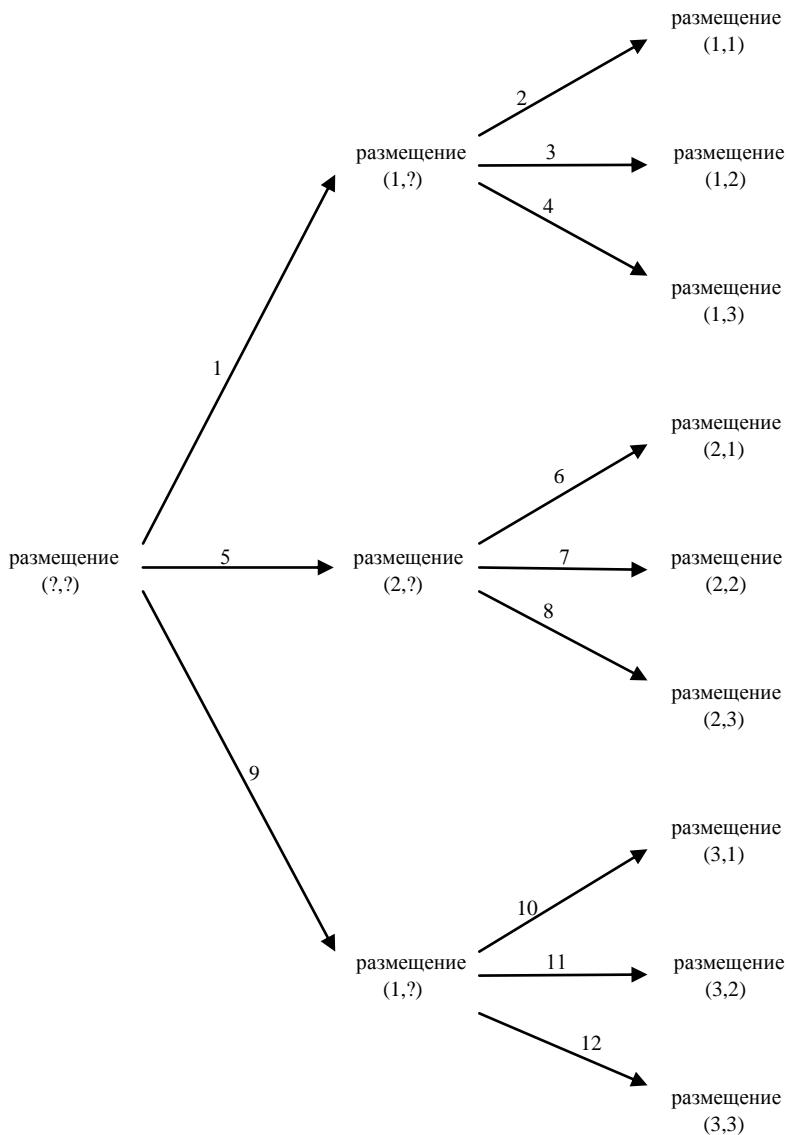


Рис.2.10. Схема получения всех размещений с повторениями по алгоритму 2.4

2.7. Сочетания

Сочетанием из n элементов по k называется k -элементное подмножество n -элементного множества.

Теорема 2.5. Количество всех различных сочетаний из n элементов по k определяется формулой $C_n^k = \frac{n!}{(n-k)!k!}$.

Доказательство. Можно получить все A_n^k размещений, упорядочив всеми возможными способами каждое из C_n^k сочетаний. Количество способов упорядочивания одного сочетания равно $k!$, следовательно, $A_n^k = C_n^k \cdot k!$. Отсюда $C_n^k = A_n^k / k! = \frac{n!}{(n-k)!k!}$.

Пусть элементами множества M являются натуральные числа от 1 до n . Для сокращения записи такое множество будем обозначать $\{1..n\}$.

Рассмотрим алгоритм порождения сочетаний в возрастающем лексикографическом порядке, т.е. в порядке, когда в каждом сочетании элементы расположены в порядке возрастания слева направо, а каждое следующее сочетание больше предыдущего (при сравнении одно сочетание рассматриваем как число). Например, сочетания из шести элементов по три ($C_6^3 = 20$), в возрастающем лексикографическом порядке запишутся следующим образом:

123	135	234	256
124	136	235	345
125	145	236	346
126	146	245	356
134	156	246	456

Сочетания в лексикографическом порядке можно порождать, используя метод поиска с возвращением. Начиная с 1-го места будем последовательно формировать элементы сочетания C . На первое место в сочетании можно ставить элементы из множества $\{1..n - k + 1\}$. Формирование i -го элемента сочетания опишем рекурсивным алгоритмом 2.5, блок-схема которого представлена на рис.2.11. В цикле перебираются элементы множества $\{b..n - k + i\}$, которые можно поставить на i -е место. Если на i -е место поставлен элемент x , то минимальный элемент, который можно поставить на $i + 1$ -е место, равен $x + 1$. Если заполнено k -е место, то сочетание сформировано и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму 2.5.

Алгоритм 2.5 порождения сочетаний из n элементов по k .

Вход: i — заполняемое место в сочетании C ;

b — минимальный элемент, который можно поставить на i -е место.

Выход: последовательность всех сочетаний из n элементов по k в возрастающем лексикографическом порядке.

Глобальные параметры: C — формируемое сочетание;

k — количество элементов в сочетании.

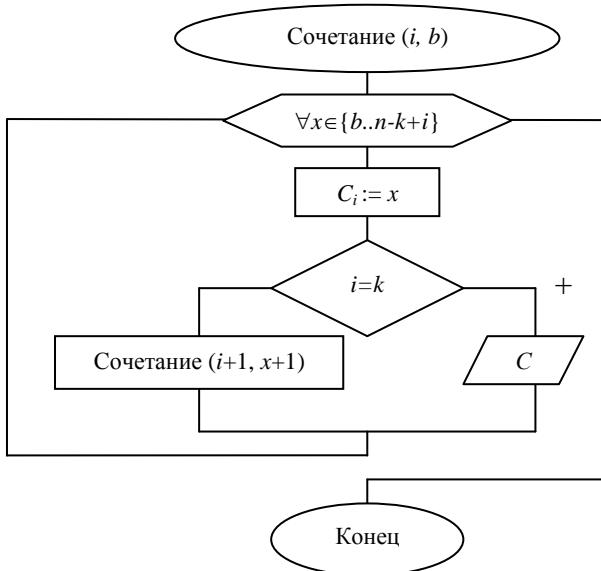


Рис.2.11. Рекурсивный алгоритм порождения сочетаний из n элементов по k

Пусть задано множество $\{1,2,3,4,5\}$ и требуется получить все его сочетания по 3. Схема получения всех сочетаний по алгоритму 2.5 представлена на рис.2.12. В этой схеме представлены формируемые сочетания и множества, элементы которых можно поставить на очередное место в сочетании. В исходной ситуации все места сочетания неопределены и на первое место можно поставить любой элемент множества $\{1,2,3\}$. В схеме к стрелкам приписаны номера действий. В результате

выполнения действия 1 на первое место сочетания устанавливается элемент 1, а на второе место после этого можно поставить любой элемент множества {2,3,4}. После выполнения действия 2 на второе место сочетания устанавливается элемент 2, а на третье место после этого можно поставить любой элемент множества {3,4,5} (действие 3) и первое сочетание {1,2,3} сформировано. Затем выполняется шаг назад и действие 4, ставящее элемент 4 из множества {3,4,5} на третье место и второе сочетание {1,2,4} сформировано. Аналогичным образом формируются все остальные сочетания. Действия 1, 11 и 17 заполняют первое место сочетания, действия 2, 6, 9, 12, 15 и 18 — второе место, действия 3, 4, 5, 7, 8, 10, 13, 14, 16 и 19 — третье место.

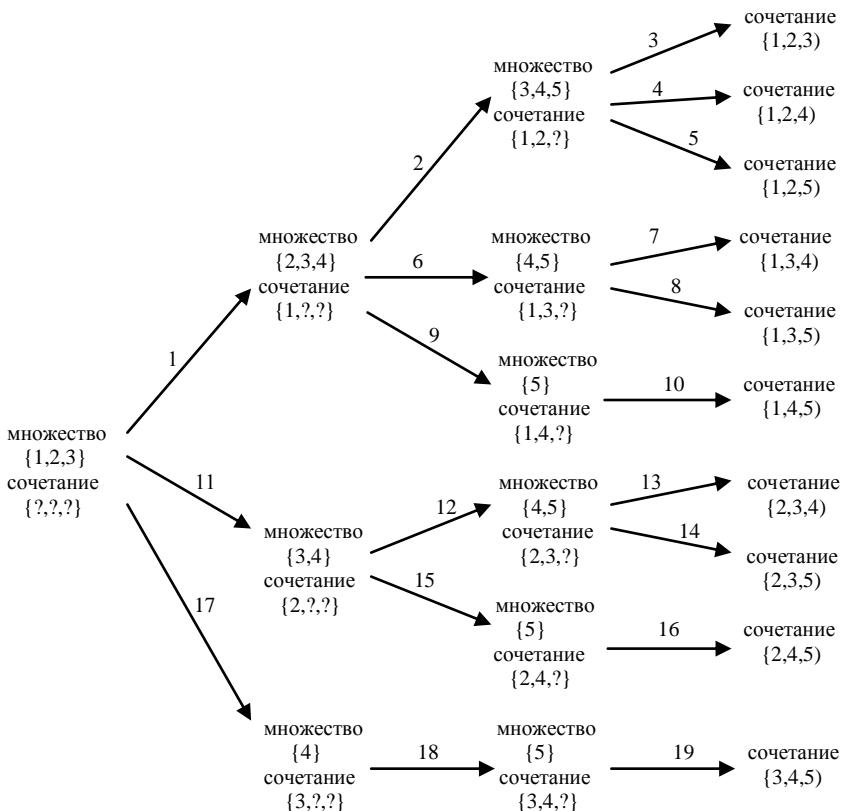


Рис.2.12. Схема получения всех сочетаний по алгоритму 2.5

2.8. Перестановки с повторениями

Множество, которое может содержать одинаковые элементы, называется *мультимножеством*. Например $M = \{1,1,2,3,1,2,3\}$ — мультимножество, которое содержит элементы 1, 2 и 3, причем элемент 1 входит в мультимножество 3 раза, а элементы 2 и 3 — по 2 раза. Мультимножества равны, если они состоят из одних и тех же элементов, поэтому $\{1,1,2,3,1,2,3\} = \{1,1,2,2,3,3\}$, т.е. порядок расположения элементов не важен. Мультимножество, состоящее из конечного числа элементов, называется *конечным*. Количество элементов в мультимножестве S называется *мощностью мультимножества* и обозначается $|S|$. Мощность мультимножества $M = \{1,1,2,3,1,2,3\}$ равна семи, $|M| = 7$. Количество n_i вхождений элемента s_i в мультимножество S называется *кратностью элемента*. Конечное мультимножество S будем записывать в следующем виде:

$$S = \{ \underbrace{s_1, \dots, s_1}_{n_1\text{-раз}}, \underbrace{s_2, \dots, s_2}_{n_2\text{-раз}}, \dots, \underbrace{s_k, \dots, s_k}_{n_k\text{-раз}} \} = \{n_1 * s_1, n_2 * s_2, \dots, n_k * s_k\}.$$

Здесь все s_i различны и n_i — кратность элемента s_i . Тогда мощность S равна $|S| = \sum_{i=1}^k n_i$. Учитывая введенное обозначение, мультимножество $M = \{1,1,2,3,1,2,3\}$ можно записать так: $\{3*1,2*2,2*3\}$.

Перестановкой с повторениями называется расположенные в определенном порядке элементы мультимножества.

Теорема 2.6. Число перестановок с повторениями для мультимножества $S = \{n_1 * s_1, n_2 * s_2, \dots, n_k * s_k\}$ выражается формулой

$$P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1! n_2! \dots n_k!}, \quad \text{где } n = |S| = \sum_{i=1}^k n_i.$$

Доказательство. Рассмотрим одну перестановку мультимножества и заменим в ней все одинаковые элементы разными. Тогда, число различных перестановок, которые можно составить из рассматриваемой, равно $n_1! n_2! \dots n_k!$. Проделав это для каждой перестановки, получим $n!$ перестановок. Следовательно, $P_n(n_1, n_2, \dots, n_k) \cdot n_1! n_2! \dots n_k! = n!$. Отсюда

$$P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1! n_2! \dots n_k!}$$

Алгоритм 2.6 (рис.2.13) порождения всех перестановок с повторениями практически не отличается от алгоритма 2.2 порождения всех перестановок множества, за исключением того, что в алгоритме 2.6 обрабатывается мульти множество.

Определим мульти множество C , являющееся результатом разности мульти множеств A и B ($C = A - B$). Если элемент универсума не принадлежит мульти множеству, то будем считать, что его кратность равна нулю. Элемент $x \in C$, если кратность k_A элемента x в мульти множестве A больше кратности k_B элемента x в мульти множестве B и кратность k_C элемента x в мульти множестве C определяется по формуле $k_C = k_A - k_B$.

Алгоритм 2.6 порождения перестановок с повторениями.

Вход: M — мульти множество, элементы которого можно поставить на i -е место;

i — заполняемое место в последовательности P .

Выход: последовательность всех перестановок с повторениями.

Глобальные параметры: P — формируемая перестановка с повторениями;

n — мощность мульти множества.

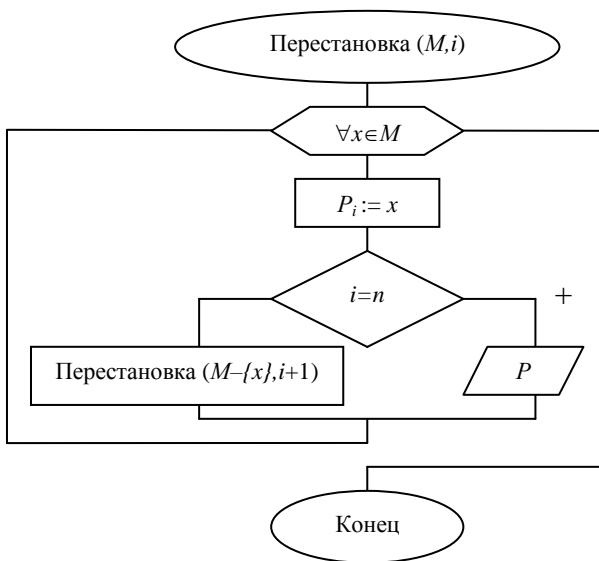


Рис.2.13. Рекурсивный алгоритм порождения перестановок с повторениями

Пусть задано мульти множество $\{1,1,2,2\} = \{2*1,2*2\}$ и требуется получить все перестановки с повторениями. Это мульти множество содержит элементы 1 и 2 кратности 2. Схема получения всех перестановок по алгоритму 2.6 представлена на рис.2.14. В этой схеме формируемые перестановки заключены в круглые скобки, а мульти множества, элементы которых можно поставить на очередное место в перестановке – в фигурные. В исходной ситуации все места перестановки неопределены и на первое место можно поставить любой элемент мульти множества: 1 или 2. В схеме к стрелкам приписаны номера действий. В результате выполнения действия 1 на первое место перестановки устанавливается элемент 1, а на второе место после этого можно поставить любой элемент мульти множества $\{1,2,2\}$. После выполнения действия 2 на второе место перестановки устанавливается элемент 1, а на третье место после этого можно поставить любой элемент мульти множества $\{2,2\}$. После выполнения действий 3 и 4 первая перестановка $\{1,1,2,2\}$ сформирована. Затем выполняется 3 шага назад и действие 5, ставящее второй элемент из мульти множества $\{1,2,2\}$ на третье место, после чего на четвертое место можно поставить любой элемент из мульти множества $\{1,2\}$. Далее, после последовательного выполнения действий 6 и 7 вторая перестановка $\{1,1,2,1\}$ сформирована. Аналогичным образом формируются все остальные перестановки.

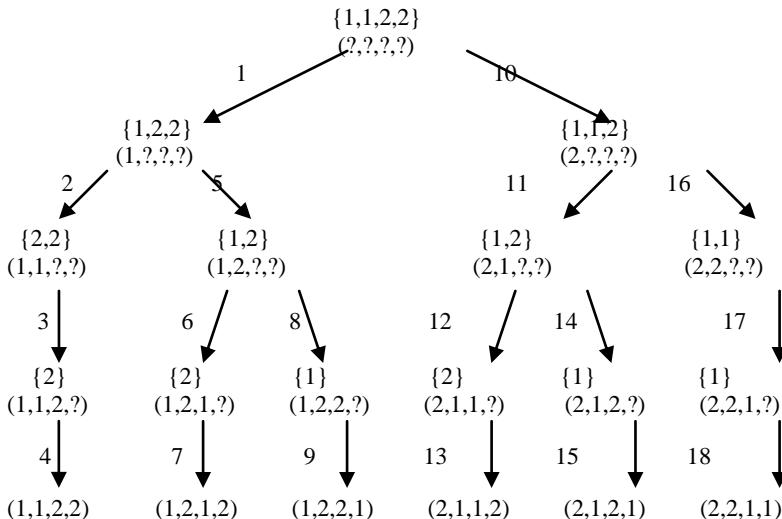


Рис.2.14. Схема получения всех перестановок с повторениями по алгоритму 2.6

2.9. Сочетания с повторениями

Сочетанием с повторениями из n элементов по k называется мульти множество мощности k , составленное из элементов n -элементного множества.

Например, из элементов множества $M = \{1,2,3\}$ можно составить такие сочетания по два с повторениями: $\{1,1\}$, $\{1,2\}$, $\{1,3\}$, $\{2,2\}$, $\{2,3\}$, $\{3,3\}$.

Теорема 2.7. Количество различных сочетаний из n элементов по k с повторениями равно

$$F_n^k = C_{n+k-1}^{n-1}.$$

Доказательство. Каждое сочетание полностью определяется, если указать, сколько раз входит каждый элемент множества в сочетание. Поставим в соответствие каждому сочетанию последовательность нулей и единиц, составленную по следующему правилу: ставим подряд столько единиц, сколько раз входит первый элемент множества в сочетание, далее ставим нуль, и после него пишем столько единиц, сколько раз входит второй элемент множества в это сочетание и т.д. Например, написанным выше сочетаниям из трех элементов по два будут соответствовать такие последовательности:

$$1100, 1010, 1001, 0110, 0101, 0011.$$

Таким образом, каждому сочетанию с повторениями из n по k соответствует последовательность из k единиц и $n - 1$ нулей. Количество таких последовательностей равно числу способов, которыми можно выбрать $n - 1$ мест для нулей из $n + k - 1$ общего числа мест (C_{n+k-1}^{n-1}), или, то же самое, — числу способов выбора k мест для единиц из $n + k - 1$ мест (C_{n+k-1}^k).

Рассмотрим алгоритм порождения сочетаний с повторениями в неубывающем лексикографическом порядке. Сочетания с повторениями из трех элементов по три ($F_3^3 = 10$), в неубывающем лексикографическом порядке запишутся следующим образом:

111	133
112	222
113	223
122	233
123	333

Сочетания с повторениями в лексикографическом порядке можно порождать, используя метод поиска с возвращением. Начиная с 1-го места будем последовательно формировать элементы сочетания F . На

первое место в сочетании можно ставить элементы из множества $\{1..n\}$. Формирование i -го элемента сочетания опишем рекурсивным алгоритмом 2.7, блок-схема которого представлена на рис.2.15. В цикле перебираются элементы множества $\{b..n\}$, которые можно поставить на i -е место. Если на i -е место поставлен элемент x , то минимальный элемент, который можно поставить на $i + 1$ -е место, тоже равен x . Если заполнено k -е место, то сочетание сформировано и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму 2.7.

Алгоритм 2.7 порождения сочетаний с повторениями из n элементов по k .

Вход: i — заполняемое место в сочетании F ;

b — минимальный элемент, который можно поставить на i -е место.

Выход: последовательность всех сочетаний с повторениями из n элементов по k в неубывающем лексикографическом порядке.

Глобальные параметры: F — формируемое сочетание;

n — мощность множества;

k — количество элементов в сочетании.

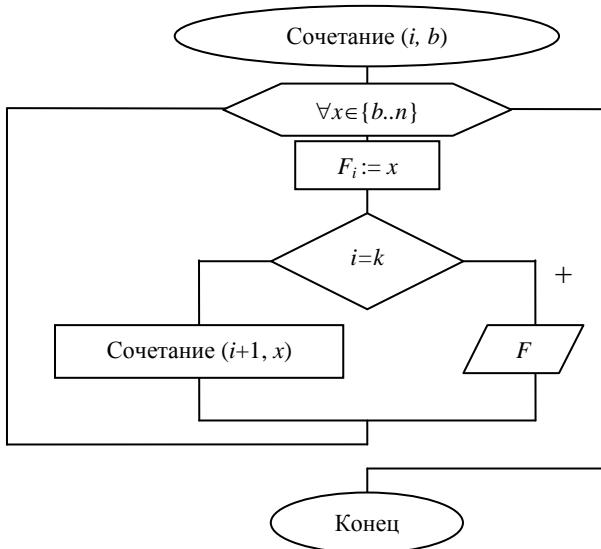


Рис.2.15. Рекурсивный алгоритм порождения сочетаний с повторениями из n элементов по k

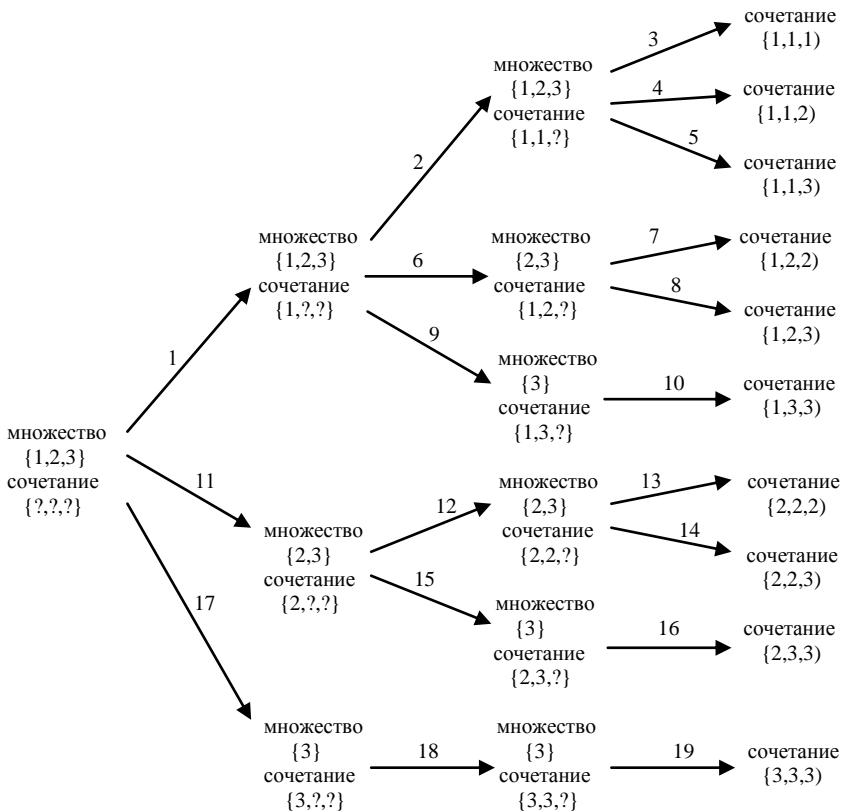


Рис.2.16. Схема получения всех сочетаний с повторениями по алгоритму 2.7

Пусть задано множество $\{1,2,3\}$ и требуется получить все сочетания с повторениями из элементов этого множества по 3. Схема получения всех сочетаний по алгоритму 2.7 представлена на рис.2.16. В этой схеме представлены формируемые сочетания и множества, элементы которых можно поставить на очередное место в сочетании. В исходной ситуации все места сочетания не определены и на первое место можно поставить любой элемент множества $\{1,2,3\}$. В схеме к стрелкам приписаны номера действий. В результате выполнения первого действия на первое место сочетания ставится элемент 1 и на второе место после этого можно поставить любой элемент того же множества $\{1,2,3\}$. После выполнения действия 2 на второе место сочетания устанавливается элемент 1 и на

третье место после этого опять можно поставить любой элемент множества {1,2,3}. После выполнения действия 3 первое сочетание {1,1,1} сформировано. Затем выполняется шаг назад и действие 4, ставящее элемент 2 из множества {1,2,3} на третье место и второе сочетание {1,1,2} сформировано и т.д. После выполнения шестого действия на второе место устанавливается элемент 2 и, после этого, на третье место можно поставить элемент из множества {2,3}, и следующее сочетание, после выполнения седьмого действия, будет {1,2,2}. Аналогичным образом формируются все остальные сочетания.

2.10. Упорядоченные разбиения множества

Упорядоченным разбиением множества M называется последовательность из непустых, попарно непересекающихся подмножеств множества M, объединение которых дает множество M.

2.10.1. Упорядоченные разбиения множества M на k подмножеств с мощностями (n₁, n₂, ..., n_k)

Рассмотрим упорядоченные разбиения множества M на k подмножеств с мощностями (n₁, n₂, ..., n_k). Одно такое разбиение представляет собой последовательность (M₁, M₂, ..., M_k) подмножеств множества M, таких, что |M₁| = n₁, |M₂| = n₂, ..., |M_k| = n_k. На первом месте в разбиении может располагаться любое подмножество мощности n₁, на втором — любое подмножество мощности n₂ и т.д.

Теорема 2.8. Количество всех различных упорядоченных разбиений n-элементного множества M на k подмножеств с мощностями (n₁, n₂, ..., n_k) определяется формулой $P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1! \cdot n_2! \cdots n_k!}$.

Доказательство 1. При формировании последовательности (M₁, M₂, ..., M_k) на первое место подмножество M₁ мощности n₁ можно выбрать из n элементов множества M C_n^{n₁} способами, на второе место подмножество M₂ мощности n₂ можно выбрать из оставшихся n - n₁ элементов множества M C_{n-n_1}^{n₂} способами и так далее, на последнее место подмножество M_k мощности n_k можно выбрать из оставшихся n - n₁ - n₂ - ... - n_{k-1} элементов множества M C_{n-n_1-n_2-\dots-n_{k-1}}^{n_k} способами. По правилу произведения получаем, что общее количество упорядоченных разбиений равно

$$P_n(n_1, n_2, \dots, n_k) = C_n^{n_1} \cdot C_{n-n_1}^{n_2} \cdot C_{n-n_1-n_2}^{n_3} \cdots C_{n-n_1-n_2-\dots-n_{k-1}}^{n_k} = \frac{n!}{n_1! n_2! \cdots n_k!},$$

что совпадает с числом перестановок с повторениями.

Доказательство 2. Упорядоченное разбиение можно задать n -разрядным вектором P , в котором P_i представляет собой номер подмножества, которому принадлежит элемент i множества M . Вектор P представляет собой перестановку с повторениями мульти множества $\{n_1*1, n_2*2, \dots, n_k*k\}$. Следовательно, количество всех различных упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями n_1, n_2, \dots, n_k равно количеству перестановок с повторениями мульти множества $\{n_1*1, n_2*2, \dots, n_k*k\}$ и определяется формулой

$$P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1!n_2!\cdots n_k!}.$$

Алгоритм порождения упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями (n_1, n_2, \dots, n_k) представляет собой алгоритм 2.6 порождения перестановок с повторениями мульти множества $\{n_1*1, n_2*2, \dots, n_k*k\}$, в котором перестановки с повторениями интерпретируются как разбиения. В табл. 2.1 представлены упорядоченные разбиения множества $M = \{1,2,3,4,5\}$ на три подмножества: первые два — двухэлементные и третье — одноэлементное.

Таблица 2.1
Упорядоченные разбиения множества $M = \{1,2,3,4,5\}$

№ п/п	Перестановка с повторениями	Упорядоченное разбиение
1	2	3
1	(1,1,2,2,3)	({1,2},{3,4},{5})
2	(1,1,2,3,2)	({1,2},{3,5},{4})
3	(1,1,3,2,2)	({1,2},{4,5},{3})
4	(1,2,1,2,3)	({1,3},{2,4},{5})
5	(1,2,1,3,2)	({1,3},{2,5},{4})
6	(1,2,2,1,3)	({1,4},{2,3},{5})
7	(1,2,2,3,1)	({1,5},{2,3},{4})
8	(1,2,3,1,2)	({1,4},{2,5},{3})
9	(1,2,3,2,1)	({1,5},{2,4},{3})
10	(1,3,1,2,2)	({1,3},{4,5},{2})
11	(1,3,2,1,2)	({1,4},{3,5},{2})
12	(1,3,2,2,1)	({1,5},{3,4},{2})
13	(2,1,1,2,3)	({2,3},{1,4},{5})
14	(2,1,1,3,2)	({2,3},{1,5},{4})
15	(2,1,2,1,3)	({2,4},{1,3},{5})
16	(2,1,2,3,1)	({2,5},{1,3},{4})
17	(2,1,3,1,2)	({2,4},{1,5},{3})

Окончание табл.2.1

1	2	3
18	(2,1,3,2,1)	({2,5},{1,4},{3})
19	(2,2,1,1,3)	({3,4},{1,2},{5})
20	(2,2,1,3,1)	({3,5},{1,2},{4})
21	(2,2,3,1,1)	({4,5},{1,2},{3})
22	(2,3,1,1,2)	({3,4},{1,5},{2})
23	(2,3,1,2,1)	({3,5},{1,4},{5})
24	(2,3,2,1,1)	({4,5},{1,3},{2})
25	(3,1,1,2,2)	({2,3},{4,5},{1})
26	(3,1,2,1,2)	({2,4},{3,5},{1})
27	(3,1,2,2,1)	({2,5},{3,4},{1})
28	(3,2,1,1,2)	({3,4},{2,5},{1})
29	(3,2,1,2,1)	({3,5},{2,4},{1})
30	(3,2,2,1,1)	({4,5},{2,3},{1})

2.10.2. Упорядоченные разбиения множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$

Рассмотрим упорядоченные разбиения множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$, где $\{n_1, n_2, \dots, n_k\}$ может быть мульти множеством. Одно такое разбиение представляет собой последовательность (M_1, M_2, \dots, M_k) подмножеств множества M , таких, что $|M_1| = n_1$, $|M_2| = n_2, \dots, |M_k| = n_k$. На первое место в разбиении можно поставить подмножество любой мощности $n_i \in \{n_1, n_2, \dots, n_k\}$, после этого на второе место — подмножество любой мощности $n_j \in \{n_1, n_2, \dots, n_k\} - \{n_i\}$ и т.д. Среди подмножеств в разбиении (M_1, M_2, \dots, M_k) можно выделить l_1 подмножеств мощности 1, l_2 подмножеств мощности 2 и т.д., l_n подмножеств мощности n . Например, в разбиении $(\{1,2\}, \{3,4\}, \{5\})$ $l_1 = 1$, $l_2 = 2$, $l_3 = l_4 = l_5 = 0$. Очевидно, что $1 \cdot l_1 + 2 \cdot l_2 + \dots + n \cdot l_n = n$.

Теорема 2.9. Количество всех различных упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$ определяется формулой

$$\begin{aligned} P_n\{n_1, n_2, \dots, n_k\} &= \frac{n!}{n_1! n_2! \dots n_k!} \cdot \frac{k!}{l_1! l_2! \dots l_n!} = \\ &= \frac{n!}{(1!)^{l_1} \cdot (2!)^{l_2} \cdot \dots \cdot (n!)^{l_n}} \cdot \frac{k!}{l_1! l_2! \dots l_n!} \end{aligned}$$

Доказательство. Рассмотрим упорядоченное разбиение множества M на k подмножеств с мощностями (n_1, n_2, \dots, n_k) . Если мощности подмножеств попарно различны, то из такого разбиения можно получить $k!$

упорядоченных разбиений множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$, изменяя всеми способами порядок следования подмножеств. Но так как упорядоченное разбиение множества M на k подмножеств с мощностями (n_1, n_2, \dots, n_k) может содержать подмножества с одинаковыми мощностями, то из такого разбиения можно получить

$$\frac{k!}{l_1!l_2!\dots l_n!}$$

упорядоченных разбиений множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$. Количество всех различных упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями (n_1, n_2, \dots, n_k) определяется формулой $P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1!n_2!\dots n_k!}$, следо-

$$\begin{aligned} \text{вательно, } P_n\{n_1, n_2, \dots, n_k\} &= P_n(n_1, n_2, \dots, n_k) \cdot \frac{k!}{l_1!l_2!\dots l_n!} = \\ &= \frac{n!}{n_1!n_2!\dots n_k!} \cdot \frac{k!}{l_1!l_2!\dots l_n!}. \end{aligned}$$

$$\text{Рассмотрим формулу } P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1!n_2!\dots n_k!}.$$

В знаменателе сначала расположим все сомножители со значением $1!$, их будет l_1 штук, затем — все сомножители со значением $2!$, их будет l_2 штук, и так далее, и в конце — все сомножители со значением $n!$, их будет l_n штук. Теперь эту формулу можно записать в виде

$$P_n(n_1, n_2, \dots, n_k) = \frac{n!}{(1!)^{l_1} \cdot (2!)^{l_2} \cdot \dots \cdot (n!)^{l_n}},$$

а формулу, определяющую количество всех различных упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$ — в виде

$$P_n\{n_1, n_2, \dots, n_k\} = \frac{n!}{(1!)^{l_1} \cdot (2!)^{l_2} \cdot \dots \cdot (n!)^{l_n}} \cdot \frac{k!}{l_1!l_2!\dots l_n!}.$$

Для порождения упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$ можно порождать перестановки с повторениями мульти множества $\{n_1, n_2, \dots, n_k\}$ и каждую перестановку использовать для порождения упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями (n_1, n_2, \dots, n_k) , в результате чего будут получены все упорядоченные разбиения n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.

2.10.3. Все упорядоченные разбиения множества M на k подмножеств

Рассмотрим все упорядоченные разбиения множества M на k подмножеств. Одно такое разбиение представляет собой последовательность (M_1, M_2, \dots, M_k) подмножеств множества M , таких, что $|M_1| + |M_2| + \dots + |M_k| = |M|$.

При подсчете количества упорядоченных разбиений множества M на k подмножеств будем использовать понятие композиции числа n из k частей. *Композицией натурального числа n из k частей с ограничением $n_i > 0$* называется k -элементная последовательность (n_1, n_2, \dots, n_k) натуральных чисел, такая, что $n_1 + n_2 + \dots + n_k = n$.

Теорема 2.10. Количество композиций числа n из k частей с ограничением $n_i > 0$ равно C_{n-1}^{k-1} .

Доказательство. Разделим отрезок длины n на n отрезков единичной длины с помощью $(n - 1)$ точки. Тогда композиции n взаимно однозначно соответствуют пометка некоторых $k - 1$ точек разделения. Элементами композиции в этом случае будут расстояния между соседними точками разделения. Например, композиция $(2, 2, 1)$ числа 5 представлена на рис.2.17.

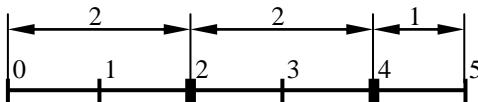


Рис.2.17. Представление композиции $(2,2,1)$ числа 5

Каждая композиция числа n из k частей с ограничением $n_i > 0$ соответствует способу выбора $(k - 1)$ -элементного подмножества точек из $n - 1$ точек. То есть число таких композиций равно C_{n-1}^{k-1} .

Теорема 2.11. Количество всех различных упорядоченных разбиений n -элементного множества M на k подмножеств определяется формулой

$$P_n(k) = \sum_{i=1}^m \frac{n!}{n_{1i}! n_{2i}! \dots n_{ki}!},$$

где $m = C_{n-1}^{k-1}$, $(n_{1i}, n_{2i}, \dots, n_{ki})$ — i -я композиция числа n из k частей с ограничением $n_i > 0$.

Доказательство. Элементы композиции (n_1, n_2, \dots, n_k) числа n из k частей будем считать мощностями подмножеств в упорядоченном разбиении

ния (M_1, M_2, \dots, M_k) множества M на k частей. Количество всех различных упорядоченных разбиений n -элементного множества M на k подмножеств с мощностями (n_1, n_2, \dots, n_k) определяется формулой

$$P_n(n_1, n_2, \dots, n_k) = \frac{n!}{n_1! n_2! \dots n_k!},$$

а количество композиций числа n из k частей определяется формулой $m = C_{n-1}^{k-1}$, следовательно, количество всех различных упорядоченных разбиений n -элементного множества M на k подмножеств определяется формулой $P_n(k) = \sum_{i=1}^m \frac{n!}{n_{1i}! n_{2i}! \dots n_{ki}!}$,

где $m = C_{n-1}^{k-1}$, $(n_{1i}, n_{2i}, \dots, n_{ki})$ — i -я композиция числа n из k частей с ограничением $n_i > 0$.

Для порождения упорядоченных разбиений n -элементного множества M на k подмножеств можно получить все композиции числа n из k частей с ограничением $n_i > 0$ и к каждой композиции (n_1, n_2, \dots, n_k) применить алгоритм 2.6 порождения перестановок с повторениями мульти множества $\{n_1 * 1, n_2 * 2, \dots, n_k * k\}$, в котором перестановки с повторениями интерпретировать как разбиения.

Для порождения композиций числа n из k частей с ограничением $n_i > 0$ можно использовать алгоритм 2.5 порождения сочетаний и каждое сочетание преобразовать в соответствующую ему композицию или непосредственно, используя метод поиска с возвращением. Начиная с 1-го места будем последовательно формировать элементы композиции C . Формирование i -го элемента композиции опишем рекурсивным алгоритмом 2.8, блок-схема которого представлена на рис.2.18. В цикле перебираются элементы множества, которые можно поставить на i -е место. Минимальный элемент, который можно поставить на i -е место при $i < k$ равен 1, а максимальный — такой, что сумма сформированных i элементов плюс количество оставшихся элементов композиции не превышает числа n , т.е. $n - S - k + i$, где S — сумма сформированных $i - 1$ элементов. На последнее, k -е место композиции можно поставить только $n - (S + x)$, где x — элемент, поставленный на $k - 1$ -е место, $S + x$ — сумма $k - 1$ элементов композиции. Если заполнено $k - 1$ -е место, то однозначно заполняем последнее место, как описано выше, и выводим композицию, иначе заполняем следующее $i + 1$ -е место по алгоритму 2.8.

Алгоритм 2.8 порождения композиций числа n из k частей с ограничением $n_i > 0$

Вход: i — заполняемое место в композиции;

S — сумма первых $i - 1$ элементов композиции.

Выход: последовательность всех композиций числа n из k частей с ограничением $n_i > 0$.

Глобальные параметры: C — формируемая композиция;

n — исходное число;

k — количество элементов в композиции.

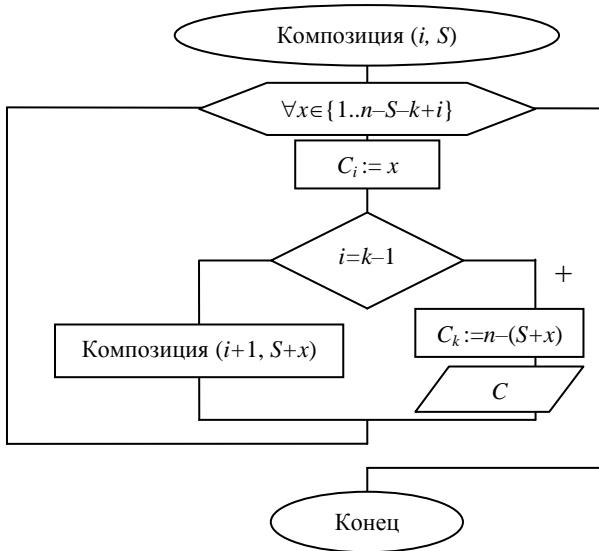


Рис.2.18. Рекурсивный алгоритм порождения композиций числа n из k частей с ограничением $n_i > 0$

2.10.4. Все упорядоченные разбиения множества M

Рассмотрим все упорядоченные разбиения множества M .

Теорема 2.12. Количество всех различных упорядоченных разбиений n -элементного множества M определяется формулой

$$R_n = \sum_{k=1}^n \sum_{i=1}^m \frac{n!}{n_{1i}! n_{2i}! \dots n_{ki}!},$$

где $m = C_{n-1}^{k-1}$, $(n_{1i}, n_{2i}, \dots, n_{ki})$ — i -я композиция числа n из k частей с ограничением $n_i > 0$.

Доказательство. n -элементное множество можно разбить на 1 подмножество, на 2 подмножества и так далее, на n подмножеств, т.е. количество k непустых подмножеств, на которые можно разбить n -элементное множество, может быть от 1 до n , отсюда количество всех различных упорядоченных разбиений n -элементного множества M определяется формулой

$$R_n = \sum_{k=1}^n \sum_{i=1}^m \frac{n!}{n_{1i}! n_{2i}! \dots n_{ki}!},$$

где $m = C_{n-1}^{k-1}$, $(n_{1i}, n_{2i}, \dots, n_{ki})$ — i -я композиция числа n из k частей с ограничением $n_i > 0$.

Для порождения всех упорядоченных разбиений n -элементного множества M достаточно, перебирая все k в диапазоне от 1 до n , получать все композиции числа n из k частей с ограничением $n_i > 0$, используя алгоритм 2.8, и к каждой композиции (n_1, n_2, \dots, n_k) применить алгоритм 2.6 порождения перестановок с повторениями мульти множества $\{n_1*1, n_2*2, \dots, n_k*k\}$, в котором перестановки с повторениями интерпретировать как разбиения.

2.11. Разбиения множества

Разбиением множества M называется множество из непустых, попарно непересекающихся подмножеств множества M , объединение которых дает множество M .

2.11.1. Разбиения множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$

Рассмотрим разбиения n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$, где $\{n_1, n_2, \dots, n_k\}$ — мульти множество. Одно такое разбиение представляет собой множество $\{M_1, M_2, \dots, M_k\}$ подмножеств множества M , таких, что $|M_1| = n_1$, $|M_2| = n_2, \dots, |M_k| = n_k$ и $n_1 + n_2 + \dots + n_k = n$. Среди подмножеств в разбиении $\{M_1, M_2, \dots, M_k\}$ можно выделить l_1 подмножеств мощности 1, l_2 подмножеств мощности 2 и так далее, l_n подмножеств мощности n , при этом $1 \cdot l_1 + 2 \cdot l_2 + \dots + n \cdot l_n = n$.

Теорема 2.13. Количество всех различных разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$ определяется формулой $\Phi_n(n_1, n_2, \dots, n_k) = \frac{n!}{(1!)^{l_1} \cdot (2!)^{l_2} \cdot \dots \cdot (n!)^{l_n} \cdot l_1! l_2! \dots l_n!}$.

Доказательство. Каждое разбиение n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$ можно упорядочить $k!$ способами и получить все упорядоченные разбиения n -элементного мно-

жества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$, число которых определяется формулой

$$P_n\{n_1, n_2, \dots, n_k\} = \frac{n!}{(1!)^{l_1} \cdot (2!)^{l_2} \cdot \dots \cdot (n!)^{l_n}} \cdot \frac{k!}{l_1! l_2! \dots l_n!},$$

следовательно, количество всех различных разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$ определяется формулой

$$\Phi_n(n_1, n_2, \dots, n_k) = \frac{P_n\{n_1, n_2, \dots, n_k\}}{k!} = \frac{n!}{(1!)^{l_1} \cdot (2!)^{l_2} \cdot \dots \cdot (n!)^{l_n} \cdot l_1! l_2! \dots l_n!}.$$

Рассмотрим алгоритм порождения разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$. Обозначим R^i – некоторое разбиение i -элементного множества $\{1..i\}$, а R^n – любое разбиение n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.

Будем последовательно получать разбиения $R^1, R^2, \dots, R^i, \dots, R^n$, не противоречащие разбиениям R^n , т.е. R^i такое, что из него путем добавления элементов в его подмножества или добавления подмножеств можно получить разбиение R^n .

Если сформировано разбиение R^{i-1} множества $\{1..i-1\}$ на p подмножеств, то из него можно получить разбиения R^i , добавляя элемент i сначала в первое, затем во второе и так далее, в p -е подмножество разбиения R^{i-1} или добавляя новое, $p+1$ -е одноэлементное подмножество $\{i\}$. Но не из каждого, сформированного таким образом, разбиения R^i можно будет получить разбиения R^n .

Для определения возможности получения из разбиения R^i разбиений R^n введем характеристику $H(R^i)$ разбиения R^i . Эта характеристика представляет собой k -элементную, упорядоченную по неубыванию, последовательность мощностей подмножеств разбиения R^i . Если количество подмножеств в разбиении R^i меньше k , то характеристика $H(R^i)$ будет содержать нулевые элементы. Разбиение R^i можно задать i -разрядным вектором S , в котором S_j – номер подмножества, которому принадлежит элемент j . Пример разбиений и их характеристик приведен в табл. 2.2. Введенную характеристику разбиения используем следующим образом. Если $H(R^i) \leq H(R^n)$ ($H(R^i) \leq H(R^n)$ истинно, если каждый элемент $H(R^i)$ не больше соответствующего элемента из $H(R^n)$), то из разбиения R^i можно получить разбиение R^n . Например, если $H(R^i) = (1, 2, 2)$, то, анализируя характеристики (см. табл. 2.2), видим, что из 4-го разбиения $\{\{1, 2, 3\}\}$, 9-го разбиения

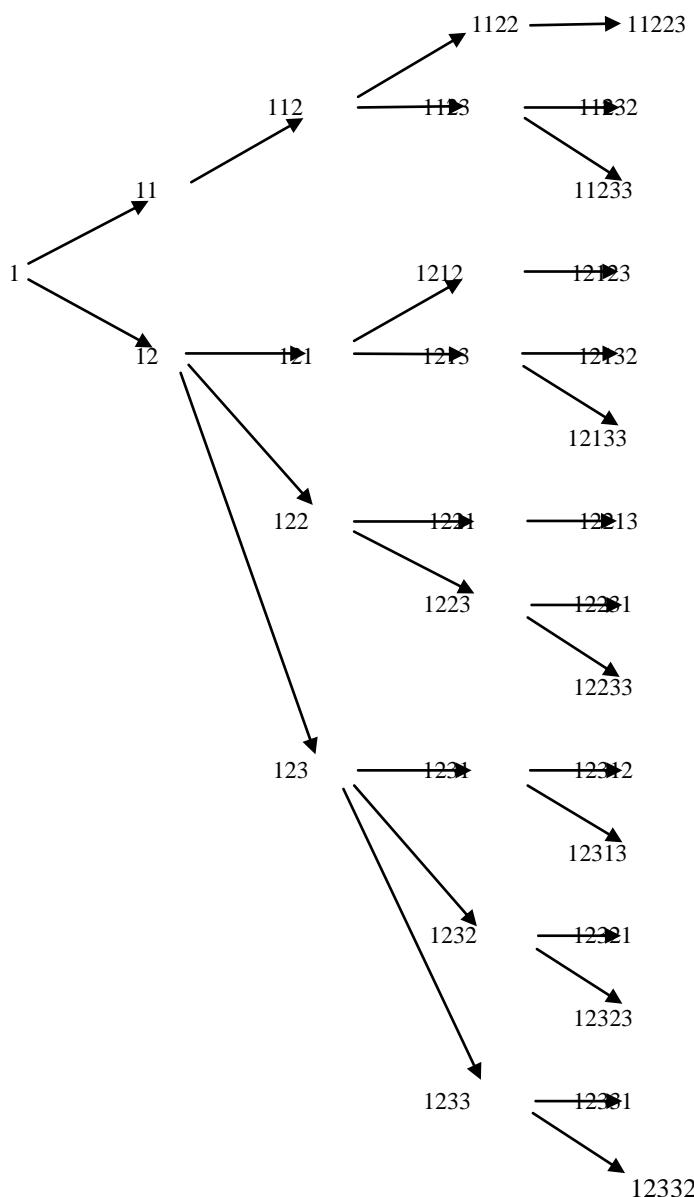
$\{\{1,2,4\},\{3\}\}$ и 12-го разбиения $\{\{1,3,4\},\{2\}\}$ нельзя получить разбиения 5-элементного множества на 3 подмножества с мощностями $\{1,2,2\}$, а из всех остальных — можно.

Таблица 2.2
Характеристики разбиений

№ п/п	Разбиение R	Вектор S	Характеристика $H(R)$
1	$\{1\}$	1	001
2	$\{1,2\}$	11	002
3	$\{1\},\{2\}$	12	011
4	$\{1,2,3\}$	111	003
5	$\{1,2\},\{3\}$	112	012
6	$\{1,3\},\{2\}$	121	012
7	$\{1\},\{2,3\}$	122	012
8	$\{1\},\{2\},\{3\}$	123	111
9	$\{1,2,4\},\{3\}$	1121	013
10	$\{1,2\},\{3,4\}$	1122	022
11	$\{1,2\},\{3\},\{4\}$	1123	112
12	$\{1,3,4\},\{2\}$	1211	013
13	$\{1,3\},\{2,4\}$	1212	022
14	$\{1,3\},\{2\},\{4\}$	1213	112

В алгоритме порождения разбиений разбиения представлены вектором S . Формирование i -го разряда вектора S соответствует определению подмножества разбиения, в которое добавляется элемент i . Если разбиение R^{i-1} содержит p подмножеств, то элемент i можно включить в подмножество с номером x от 1 до $p + 1$, если $p + 1 \leq k$, или до k , в противном случае, что соответствует записи в i -й разряд значения x , но при этом должно соблюдаться условие $H(R^i) \leq H(R^u)$. Если $i = n$, то разбиение получено, иначе по этому же алгоритму формируем $i + 1$ -й разряд вектора S (разбиение R^{i+1}). При рекурсивном обращении передаем количество $\max(x, p)$ подмножеств в сформированном разбиении R^i . При первом обращении передаем $I = 1$ для формирования 1-го разряда вектора S и $p = 0$, означающее пустоту начального разбиения.

Схема формирования векторов S , соответствующих разбиениям множества $M=\{1,2,3,4,5\}$ на три подмножества с мощностями $\{1,2,2\}$, показана на рис.2.19, а блок-схема алгоритма 2.9 порождения разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1,n_2,\dots,n_k\}$ — на рис.2.20.

Рис.2.19. Схема формирования векторов S

Алгоритм 2.9 порождения разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.

Вход: i — заполняемое место в векторе S ;

p — количество подмножеств в разбиении R^{i-1} .

Выход: последовательность всех векторов S , задающих разбиения.

Глобальные параметры: S — формируемый вектор;

n — мощность множества;

k — количество подмножеств в разбиении.

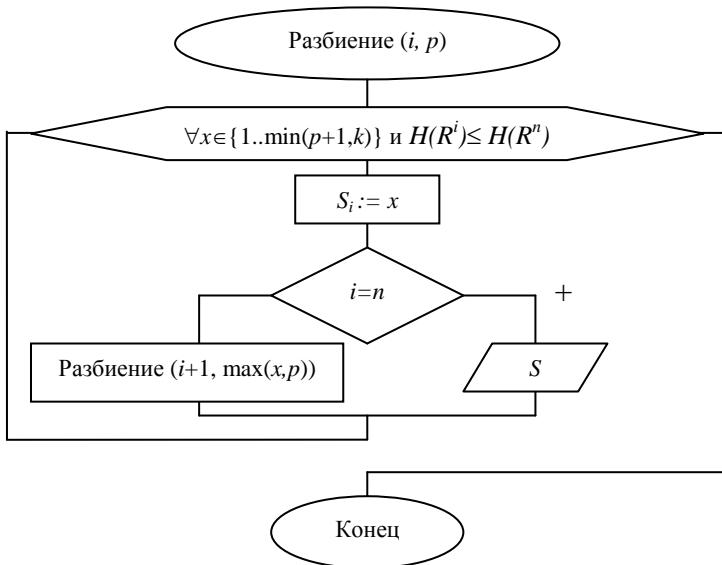


Рис.2.20. Рекурсивный алгоритм порождения разбиений n -элементного множества M на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.

2.11.2. Все разбиения множества M на k подмножеств

Рассмотрим все разбиения n -элементного множества M на k подмножеств.

Теорема 2.14. Количество всех различных разбиений n -элементного множества M на k подмножеств определяется формулой

$$\Phi_n(k) = \frac{1}{k!} \sum_{i=1}^m \frac{n!}{n_{1i}! n_{2i}! \dots n_{ki}!},$$

где $m = C_{n-1}^{k-1}$, $(n_{1i}, n_{2i}, \dots, n_{ki})$ — i -я композиция числа n из k частей с ограничением $n_i > 0$.

Доказательство. Каждое разбиение n -элементного множества M на k подмножеств можно упорядочить $k!$ способами и получить все упорядоченные разбиения n -элементного множества M на k подмножеств, число которых определяется формулой $P_n(k) = \sum_{i=1}^m \frac{n!}{n_{1i}!n_{2i}!\dots\cdot n_{ki}!}$,

где $m = C_{n-1}^{k-1}$, $(n_{1i}, n_{2i}, \dots, n_{ki})$ — i -я композиция числа n из k частей с ограничением $n_i > 0$. Следовательно, количество всех различных разбиений n -элементного множества M на k подмножеств определяется формулой

$$\Phi_n(k) = \frac{P_n(k)}{k!} = \frac{1}{k!} \sum_{i=1}^m \frac{n!}{n_{1i}!n_{2i}!\dots\cdot n_{ki}!},$$

где $m = C_{n-1}^{k-1}$, $(n_{1i}, n_{2i}, \dots, n_{ki})$ — i -я композиция числа n из k частей с ограничением $n_i > 0$.

Для порождения всех разбиений n -элементного множества на k подмножеств можно получить все такие мульти множества $\{n_1, n_2, \dots, n_k\}$, что $n_1 + n_2 + \dots + n_k = n$, и, считая, что n_1, n_2, \dots, n_k — мощности подмножеств в разбиении, применить к каждому из них алгоритм 2.9 порождения разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.

Мульти множество $\{n_1, n_2, \dots, n_k\}$, состоящее из натуральных чисел, такое, что $n_1 + n_2 + \dots + n_k = n$, называется *разбиением* числа n на k частей. Последовательность всех разбиений числа n на k частей можно записать в неубывающем лексикографическом порядке, например, все разбиения числа 8 на 4 части записутся следующим образом:

1115
1124
1133
1223
2222

Для порождения разбиений числа n из k можно использовать метод поиска с возвращением. Начиная с 1-го места, будем последовательно формировать элементы разбиения R . Формирование i -го элемента разбиения опишем рекурсивным алгоритмом 2.10, блок-схема которого представлена на рис.2.21. В цикле перебираются элементы множества, которые можно поставить на i -е место. Минимальный элемент, который можно поставить на первое место, равен 1. Минимальный элемент, ко-

торый можно поставить на очередное i -е место при $1 < i < k$, равен элементу, поставленному на $i - 1$ -е место, а максимальный — такой, что сумма сформированных $i - 1$ элементов плюс сумма минимальных значений, которые можно поставить на оставшиеся места разбиения, не превышает числа n , т.е. $[(n - S) / (k - i + 1)]$, где S — сумма сформированных $i - 1$ элементов, $k - i + 1$ — количество оставшихся мест. На последнее, k -е место разбиения можно поставить только $n - (S + x)$, где x — элемент, поставленный на $k - 1$ -е место, $S + x$ — сумма $k - 1$ элементов разбиения. Если заполнено $k - 1$ -е место, то однозначно заполняем последнее место, как описано выше, и выводим разбиение, иначе заполняем следующее $i + 1$ -е место по алгоритму 2.10.

Алгоритм 2.10 порождения разбиений числа n из k частей.

Вход: i — заполняемое место в разбиении;

b — минимальное число, которое можно поставить на i -е место;

S — сумма первых $i - 1$ элементов разбиения.

Выход: последовательность всех разбиений числа n из k частей.

Глобальные параметры: R — формируемое разбиение;

n — исходное число;

k — количество элементов в разбиении.

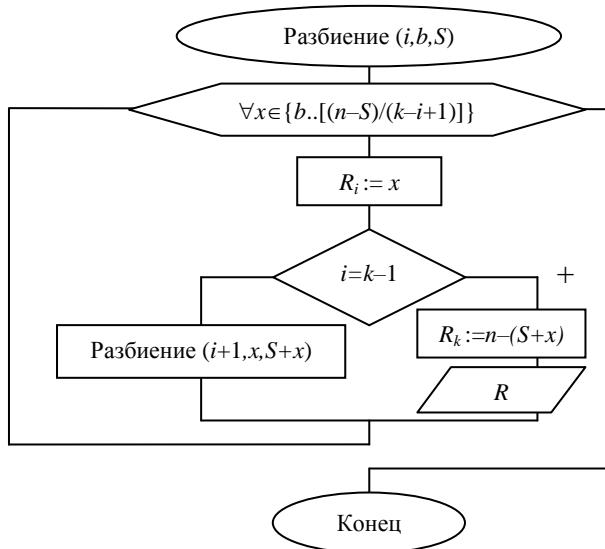


Рис.2.21. Рекурсивный алгоритм порождения разбиений числа n из k частей

Схема получения всех разбиений числа 8 на 4 части по алгоритму 2.10 представлена на рис.2.22. В исходной ситуации все места разбиения неопределены и на первое место можно поставить числа от 1 до $[(n - S) / (k - i + 1)] = [(8 - 0) / (4 - 1 + 1)] = 2$. В схеме к стрелкам приписаны номера действий. В результате выполнения действия 1 на первое место разбиения устанавливается число 1, на второе место после этого можно ставить числа от 1 до $[(n - S) / (k - i + 1)] = [(8 - 1) / (4 - 2 + 1)] = 2$. После выполнения действия 2 на второе место разбиения устанавливается число 1, а на третье место после этого можно поставить числа от 1 до $[(n - S) / (k - i + 1)] = [(8 - 2) / (4 - 3 + 1)] = 3$. После выполнения действия 3 на третье место разбиения устанавливается число 1 и действием 4 остается установить $n - (S + x) = 8 - (2 + 1) = 5$ на четвертое место ($x = 1$ – число на третьем месте). Первое разбиение {1,1,1,5} получено. Затем выполняется 2 шага назад и действие 5, ставящее число 2 на третье место разбиения, после чего на четвертое место остается установить число $n - (S + x) = 8 - (2 + 2) = 4$ действием 6. Второе разбиение {1,1,2,4} получено. Аналогичным образом формируются все остальные разбиения.

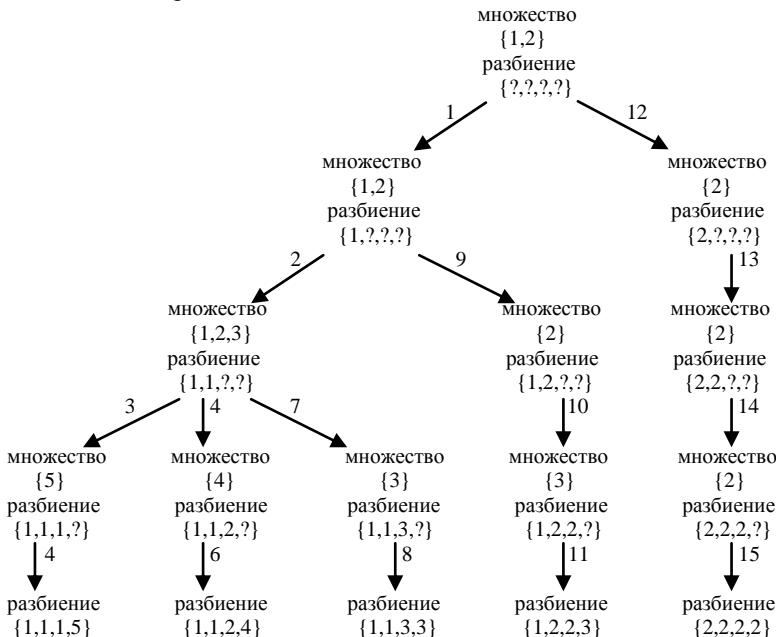


Рис.2.22. Схема получения всех разбиений числа 8 на 4 части

Рассмотрим еще один алгоритм порождения всех разбиений n -элементного множества на k подмножеств, построенный на основе метода поиска с возвращением, не использующий другие алгоритмы. Так же, как и в алгоритме порождения разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$, будем последовательно получать разбиения $R^1, R^2, \dots, R^i, \dots, R^n$, не противоречащие разбиениям R^n , где R^n — разбиение n -элементного множества на k непустых подмножеств без ограничения на мощности подмножеств, т.е. R^i такое, что из него, путем добавления элементов в его подмножества или добавления подмножеств, можно получить разбиение R^n .

Если сформировано разбиение R^{i-1} множества $\{1..i-1\}$ на p подмножеств, то из него можно получить разбиения R^i , добавляя элемент i сначала в первое, затем во второе и так далее, в p -е подмножество разбиения R^{i-1} или добавляя новое, $p+1$ -е одноэлементное подмножество $\{i\}$. Но не из каждого, сформированного таким образом, разбиения R^i можно будет получить разбиения R^n .

Во-первых, в разбиении R^n должно быть ровно k подмножеств, поэтому, если разбиение R^{i-1} содержит $p = k$ подмножеств, то новое, $p+1$ -е подмножество в разбиении R^i создавать нельзя. Далее, если разбиение R^{i-1} содержит p подмножеств, то будем говорить, что $i-1$ элемент n -элементного множества распределен по p подмножествам разбиения R^{i-1} и остаются нераспределенными $n-(i-1)$ элементов, при этом остаются несформированными $k-p$ подмножеств. Если количество нераспределенных элементов $n-(i-1)$ больше количества $k-p$ подмножеств, которые еще необходимо сформировать, то элемент i можно добавить в любое подмножество разбиения R^{i-1} , иначе, для получения разбиения R^i , в разбиение R^{i-1} можно только добавить новое, $p+1$ -е, подмножество $\{i\}$.

Для хранения разбиения используется вектор S , в котором S_i — номер подмножества, которому принадлежит элемент i . Формирование i -го разряда вектора S соответствует определению подмножества разбиения, в которое добавляется элемент i . Если разбиение R^{i-1} содержит p подмножеств, то элемент i можно включить в подмножество, минимальный номер которого равен 1, если $n-(i-1) > k-p$, или $p+1$ — в противном случае, а максимальный номер — $p+1$, если $p+1 \leq k$, или k — в противном случае. Если $i=n$, то разбиение получено, иначе по этому же алгоритму формируем $i+1$ -й разряд вектора S (разбиение R^{i+1}). При рекурсивном обращении передаем количество $\max(x,p)$ подмножеств в сформированном разбиении R^i . При первом обращении передаем $i=1$ для формирования 1-го разряда вектора S и $p=0$, означающее пустоту начального разбиения.

Блок-схема алгоритма 2.11 порождения всех разбиений n -элементного множества M на k подмножеств представлена на рис.2.23, а схемы формирования разбиений множества $M=\{1,2,3,4\}$ на три подмножества и соответствующих им векторов S показаны на рис.2.24.

Алгоритм 2.11 порождения всех разбиений n -элементного множества M на k подмножеств.

Вход: i — заполняемое место в векторе S ;

p — количество подмножеств в разбиении R^{i-1} .

Выход: последовательность всех векторов S , задающих разбиения.

Глобальные параметры: S — формируемый вектор;

n — мощность множества;

k — количество подмножеств в разбиении.

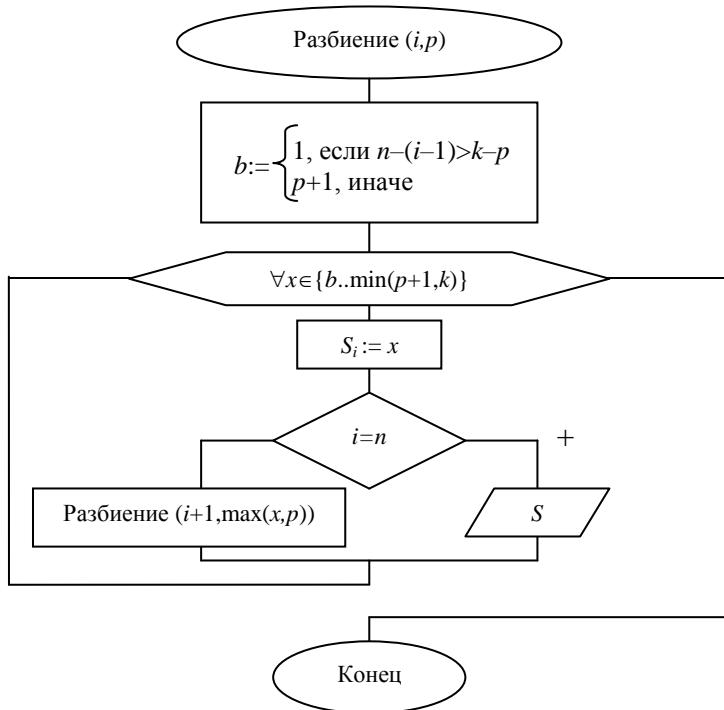


Рис.2.23. Рекурсивный алгоритм порождения всех разбиений n -элементного множества M на k подмножеств

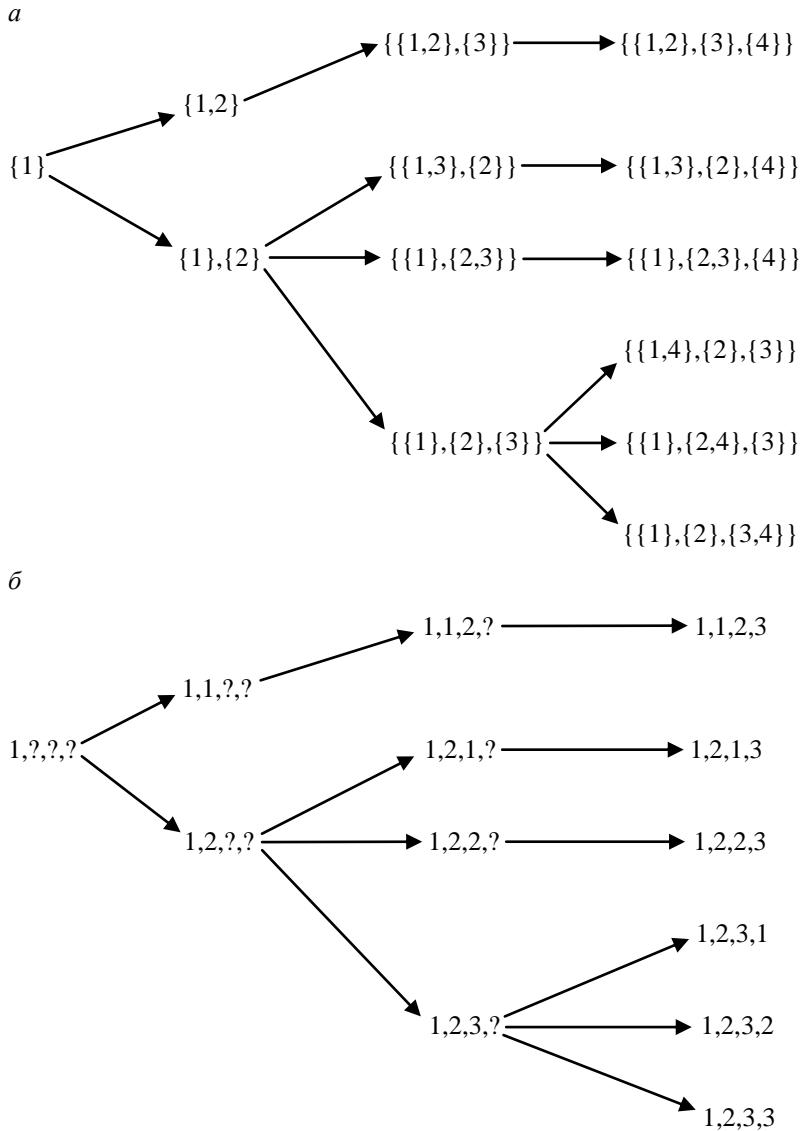


Рис.2.24. Схемы формирования разбиений и соответствующих им векторов S :
 а — схема формирования разбиений;
 б — схема формирования векторов S

2.11.3. Все разбиения множества M

Рассмотрим все разбиения множества M .

Теорема 2.15. Количество всех различных разбиений n -элементного множества M определяется формулой

$$\Phi_n = \sum_{k=1}^n \Phi_n(k)$$

Доказательство. n -элементное множество можно разбить на одно подмножество, на два подмножества и так далее, на n подмножеств, т.е. количество k непустых подмножеств, на которые можно разбить n -элементное множество, может быть от 1 до n , отсюда количество всех различных разбиений n -элементного множества M определяется формулой $\Phi_n = \sum_{k=1}^n \Phi_n(k)$.

Исходя из доказательства теоремы 2.15 можно порождать все разбиения n -элементного множества, многократно используя алгоритм порождения разбиений n -элементного множества на k подмножеств.

Рассмотрим алгоритм порождения всех разбиений n -элементного множества M , не использующий другие алгоритмы. Пусть сформировано разбиение R^{i-1} $i - 1$ -элементного множества $\{1..i - 1\}$ на p подмножеств $\{M_1, M_2, \dots, M_p\}$. Из этого разбиения можно получить разбиение R^i i -элементного множества $\{1..i\}$ путем добавления элемента i в каждое подмножество M_j и создания нового $M_{p+1} = \{i\}$ подмножества, содержащего только один элемент i . Начнем формирование разбиений n -элементного множества с создания разбиения одноэлементного множества $\{1\}$, состоящего из этого же множества. Далее последовательно формируем разбиения двух, трех, ..., n -элементного множества. Схема формирования всех разбиений множества $M = \{1, 2, 3\}$ показана на рис.2.25.

Задать разбиение n -элементного множества можно n -элементным вектором S , в котором S_i — номер подмножества, которому принадлежит элемент i . Формирование i -го разряда вектора S соответствует определению подмножества разбиения, в которое добавляется элемент i . После добавления элемента n в одно из подмножеств вектор S сформирован и разбиение получено. Схема формирования векторов S , соответствующих разбиениям множества $M = \{1, 2, 3\}$, показана на рис.2.26. Рекурсивный алгоритм 2.11 порождения всех разбиений n -элементного множества представлен на рис.2.27. При первом обращении $i = 2$, $p = 1$ и $S_i = 1$.

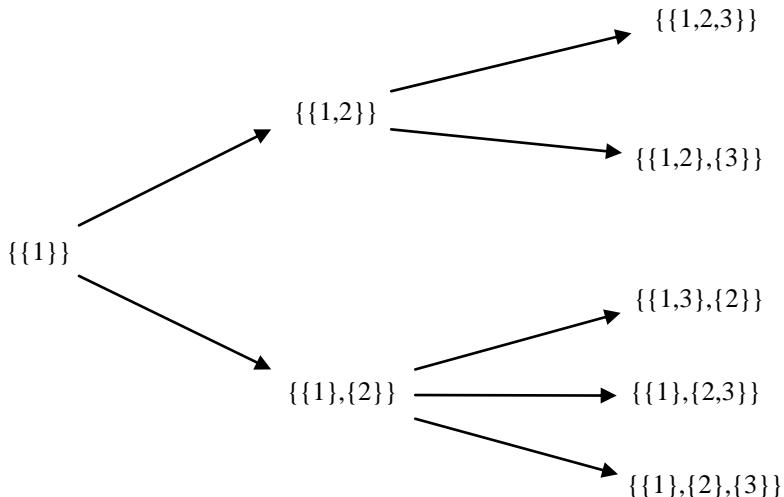


Рис.2.25. Схема формирования разбиений множества $M=\{1,2,3\}$

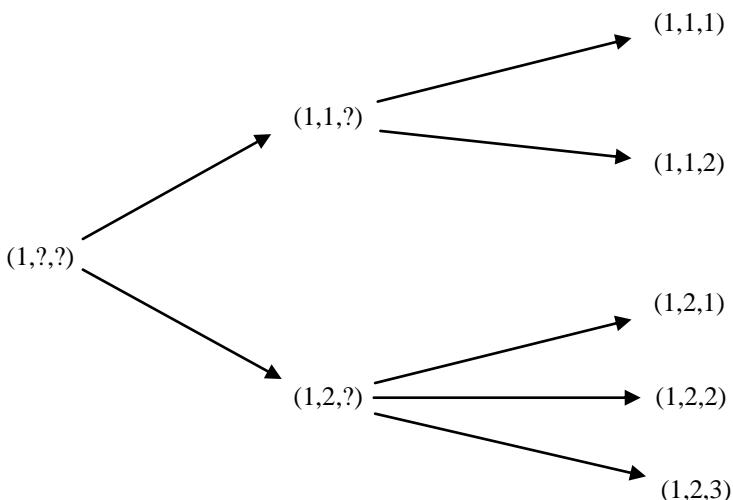


Рис.2.26. Схема формирования векторов S ,
соответствующих разбиениям множества $M = \{1,2,3\}$

Алгоритм 2.11 порождения разбиений n -элементного множества.

Вход: i — заполняемое место в векторе S ;

p — количество подмножеств в разбиении.

Выход: последовательность всех векторов S , задающих разбиения.

Глобальные параметры: S — формируемый вектор;

n — мощность множества.

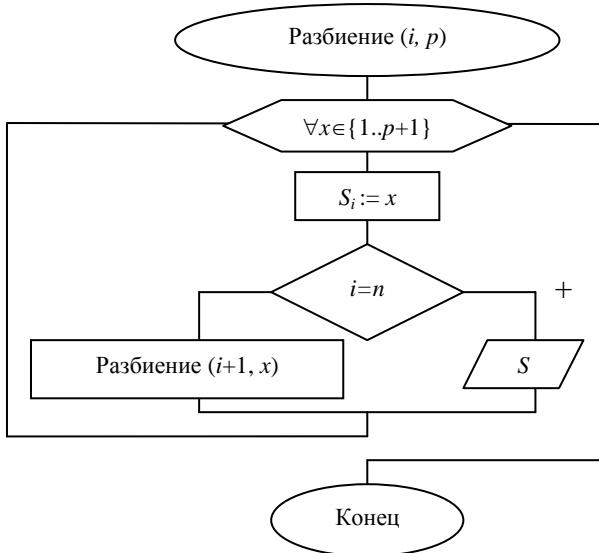


Рис.2.27. Рекурсивный алгоритм порождения разбиений n -элементного множества

2.12. Использование алгоритмов порождения комбинаторных объектов при проектировании полнопереборных алгоритмов решения задач выбора

Задачи, для которых существует конечное множество M объектов, содержащее решение задачи, относятся к классу задач выбора. Элементы множества M называются *траекториями* задачи. Каждой траектории можно поставить в соответствие некоторую, как правило, числовую характеристику, называемую *функционалом*, позволяющую распознать траекторию, являющуюся решением задачи.

В общем случае, в постановке задачи выбора может потребоваться:

- 1) определить, существует ли решение;
- 2) найти все решения задачи;
- 3) определить значение функционала “лучшего” решения;
- 4) найти одно или все “лучшие” решения.

Для того, чтобы решить задачу выбора, можно организовать полный перебор всех траекторий и выбрать из них траекторию (траектории), удовлетворяющую условию решения задачи. Такие алгоритмы называются *полнопереборными*. Одним из методов организации перебора всех траекторий является метод поиска с возвращением, который мы использовали для порождения комбинаторных объектов. Алгоритмы порождения комбинаторных объектов также могут быть использованы для перебора всех траекторий. В этом случае проектирование алгоритма решения задачи выбора заключается в следующем:

- 1) определение класса комбинаторных объектов, содержащих решение задачи;
- 2) определение способа вычисления функционала;
- 3) определение способа распознавания решения по значению функционала.

Алгоритм решения задачи выбора представляет собой порождение комбинаторных объектов, содержащих решение задачи, вычисление функционала для каждого объекта и определение принадлежности объекта множеству решений. Для того, чтобы преобразовать алгоритм порождения комбинаторных объектов в алгоритм решения задачи выбора, достаточно заменить блок вывода сформированного объекта на блок, в котором вычисляется значение функционала для полученного объекта и определяется принадлежность объекта множеству решений.

Рассмотрим примеры задач выбора и способы их решения.

Задача 2.1. Данна последовательность из n целых чисел ($n > 1$) и целое число m . Определить, можно ли между каждой парой чисел в последовательности расставить знаки “+” или “−” так, чтобы получилось выражение, значение которого равно m .

Решение. Всего необходимо поставить $n - 1$ знаков. Каждому варианту расстановки знаков можно поставить в соответствие двоичный вектор, в котором значение 1(0) в i -м разряде соответствует знаку “+” (“−”) между i -м и $i + 1$ -м числом в последовательности. Траекториями задачи будут являться все двоичные $n - 1$ -разрядные вектора.

Функционалом будет являться значение выражения, полученного из

заданной последовательности чисел путем расстановки знаков, соответствующих двоичному вектору. Пусть P — исходная последовательность чисел, а D — двоичный вектор. Тогда значение выражения S можно вычислить по следующему алгоритму:

1. $S := P_1$
2. Для всех i от 1 до $n - 1$ выполнить $S := S + (2*D_i - 1)*P_{i+1}$

или, более наглядно:

1. $S := P_1$
2. Для всех i от 1 до $n - 1$ выполнить
если $D_i = 1$ то $S := S + P_{i+1}$ иначе $S := S - P_{i+1}$

Если значение выражения S будет равно m , то двоичный вектор соответствует требуемой расстановки знаков, ответ будет положительным и на этом заканчивается решение задачи. Если же ни один вектор не определил требуемую расстановку знаков, то ответ будет отрицательным.

Задача 2.2. На предприятии работают n рабочих. В выездной бригаде должны находиться рабочие, которые все вместе владеют заданными специальностями. Множество специальностей, которыми владеет каждый рабочий — известно. Необходимо сформировать бригаду из минимального числа рабочих.

Решение. Бригада из минимального числа рабочих представляет собой подмножество множества всех рабочих, поэтому траекториями данной задачи являются подмножества выбранных рабочих. Учитывая то, что требуется сформировать бригаду из минимального числа рабочих, целесообразно порождать подмножества в порядке увеличения мощности. Это можно сделать, применяя алгоритм порождения сочетаний. Сначала будем порождать сочетания из n по 1 (может быть, один из рабочих владеет всеми необходимыми специальностями), затем — сочетания из n по 2 и так далее до порождения сочетаний из n по n .

Функционалом будет являться множество MS специальностей, которыми владеет множество из k выбранных рабочих. Обозначим M — множество заданных специальностей, S_i — множество специальностей, которыми владеет i -й рабочий, C — сочетание из n по k , записанное в возрастающем порядке. Алгоритм вычисления функционала может быть следующим:

1. $MS := \emptyset$
2. Для всех i от 1 до k выполнить $MS := MS \cup S_{Ci}$

Если при обработке очередного сочетания окажется, что M является

подмножеством MS , то бригада из минимального числа рабочих соответствует этому сочетанию, решение найдено. Если же $M \subseteq MS$ будет ложным для всех порожденных подмножеств, то это означает, что сформировать бригаду из имеющихся рабочих, владеющих совместно всеми заданными специальностями, невозможно, задача не имеет решения.

Задача 2.3. Коммивояжер (агент по сбыту), отправляясь из своего города, должен ровно по одному разу посетить $n - 1$ заданных городов и вернуться назад, затратив при этом минимальную сумму на поездку. Какой маршрут ему выбрать, если затраты на проезд между городами заданы.

Решение. Обозначим города $0, 1, \dots, n-1, 0$ — исходный город, C_{ij} — стоимость проезда от города i до города j . Маршрут коммивояжера представляет собой последовательность $(0, i, \dots, j, 0)$, состоящую из $n + 1$ элементов, причем первый и последний — 0 — фиксированный элемент, а остальные $n - 1$ — попарно неравные элементы. Эта часть последовательности, от второго элемента до предпоследнего, однозначно определяет маршрут коммивояжера и представляет собой перестановку множества $\{1..n - 1\}$, поэтому траекториями задачи будут все возможные перестановки $n - 1$ -элементного множества.

Функционалом будет являться стоимость маршрута, определяемого перестановкой P $n - 1$ -элементного множества. Алгоритм вычисления функционала может быть следующим:

$$1. \quad S := C_{0, P_1} + C_{P_{n-1}, 0}$$

$$2. \quad \text{Для всех } i \text{ от } 2 \text{ до } n-1 \text{ выполнить } S := S + C_{P_{i-1}, P_i}$$

Обозначим стоимость маршрута, принятого за оптимальный — $MinS$, вначале это некоторое заведомо большое число. Если при обработке очередной перестановки получим $S < MinS$, то эту перестановку нужно запомнить как лучшую и $MinS := S$. Решение будет получено только после обработки всех $(n - 1)!$ перестановок.

2.13. О неэффективности полнопереборных алгоритмов. Пример

Полнопереборные алгоритмы решения задач выбора с использованием алгоритмов порождения комбинаторных объектов являются универсальными, достаточно простыми для проектирования и программирования, но время их выполнения может быть очень велико даже при небольших размерностях задачи (количество исходных данных), причем настолько велико (может составлять годы или даже века), что о возмож-

ности их практического применения не может быть и речи. Поэтому, при проектировании таких алгоритмов, обязательно нужно теоретически оценивать время их работы на конкретных данных.

Для решения некоторых задач выбора можно построить уникальные, “быстрые” алгоритмы, позволяющие быстро получить решение даже при больших размерностях задачи. Покажем это на примере.

Рассмотрим задачу выбора, полнопереборный алгоритм ее решения и более эффективный алгоритм.

Задача 2.4. Имеется n деталей. Каждая деталь должна быть обработана сначала на станке A , затем на станке B . Время обработки a_i каждой детали на станке A задано в последовательности (a_1, a_2, \dots, a_n) , а на станке B — в последовательности (b_1, b_2, \dots, b_n) . Требуется определить такую последовательность обработки деталей, при которой время обработки всех деталей минимально.

Решение. Определим алгоритм вычисления времени обработки всех деталей при заданной последовательности обработки деталей.

Пусть ak_{i-1} — время окончания обработки $i - 1$ -й детали на станке A . Следующую i -ю деталь можно начать обрабатывать на станке A в это же время и время окончания обработки i -й детали на станке A будет $ak_i = ak_{i-1} + a_i$.

Пусть bk_{i-1} — время окончания обработки $i - 1$ -й детали на станке B . Если это время больше времени ak_i (рис.2.28,*а*) окончания обработки i -й детали на станке A , то в это же время bk_{i-1} можно начать обрабатывать i -ю деталь на станке B и время окончания обработки i -й детали на станке B будет $bk_i = bk_{i-1} + b_i$. Если же время bk_{i-1} окончания $i - 1$ -й детали на станке B меньше ak_i (рис.2.28,*б*), то начать обработку i -й детали на станке B можно только в ak_i , а закончить — в $bk_i = ak_{i-1} + b_i$.



Рис.2.28. Определение времени окончания обработки i -й детали на станках A и B

Исходя из этого получаем простой алгоритм вычисления времени обработки последовательности деталей, представленный блок-схемой на рис.2.29, которое совпадает с окончанием обработки последней детали на станке В.

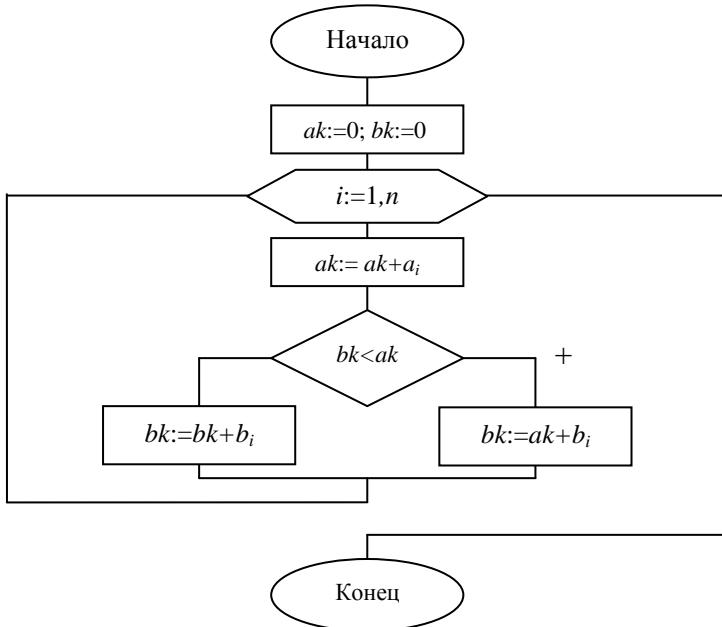


Рис.2.29. Алгоритм вычисления времени обработки последовательности деталей

Естественно, это время зависит от порядка обработки деталей. Например, если рассматривать данные, приведенные в табл. 2.3, то время обработки последовательности деталей (1,2,3,4,5,6,7) будет 117. Время окончания обработки каждой детали на станках А и В для этой последовательности представлено в табл. 2.4.

Таблица 2.3
Время обработки деталей на станках А и В

Номер детали	1	2	3	4	5	6	7
Время обработки на станке А	20	16	8	6	14	14	20
Время обработки на станке В	21	18	12	10	10	8	18

Таблица 2.4
Время окончания обработки деталей на станках А и В

Шаг обработки	Номер детали	Станок А	Станок В
1	1	20	41
2	2	36	59
3	3	44	71
4	4	50	81
5	5	64	91
6	6	78	99
7	7	98	117

Если обрабатывать детали в последовательности (2,4,3,7,6,5,1), то время обработки всех деталей будет 119. Время окончания обработки каждой детали на станках А и В для этой последовательности представлено в табл. 2.5.

Таблица 2.5
Время окончания обработки деталей на станках А и В

Шаг обработки	Номер детали	Станок А	Станок В
1	2	16	34
2	4	22	44
3	3	30	56
4	7	50	74
5	6	64	82
6	5	78	92
7	1	98	119

Для определения последовательности обработки деталей, при которой время обработки всех деталей минимально, можно использовать полнопереборный алгоритм решения задачи выбора. Траекториями задачи будут все перестановки деталей, в данном случае $7!=5040$, функционал траектории – время обработки всех деталей в последовательности, соответствующей перестановки (траектории). Для решения задачи необходимо перебрать все траектории (5040), для каждой траектории вычислить функционал (алгоритм на рис.2.29) и выбрать траекторию с минимальным значением функционала. Для определения последовательности обработки 16-ти деталей на ЭВМ потребуется несколько лет.

Можно предложить другой, более эффективный, алгоритм решения этой задачи:

1. $i1:=1$ — номер наименьшего (по номеру) свободного места в решении P ;
- $i2:=n$ — номер наибольшего (по номеру) свободного места в решении P .

2. Выполнить п.3 n раз, n — количество деталей.
3. Найти минимум среди $(a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n)$,
где a_i — время обработки i -й детали на станке A ;
 b_i — время обработки i -й детали на станке B .

Если это a_i , то $P_{i1} := i$ (поставить i -ю деталь на $i1$ -е место в решении) и $i1 := i1 + 1$, если же это b_i , то $P_{i2} := i$ и $i2 := i2 + 1$. Элементы a_i и b_i далее не рассматривать.

Результат определения последовательности обработки 16-ти деталей этим алгоритмом на ЭВМ будет получен практически мгновенно.

По этому алгоритму для данных (табл.2.3) получим последовательность (4,3,2,1,7,5,6). Время окончания обработки каждой детали на станках A и B для этой последовательности представлено в табл. 2.6.

Таблица 2.6
Время окончания обработки деталей на станках A и B

Шаг обработки	Номер детали	Станок A	Станок B
1	4	6	16
2	3	14	28
3	2	30	48
4	1	50	71
5	7	64	89
6	5	78	99
7	6	98	107

Практическое занятие 2.1

Алгоритмы порождения комбинаторных объектов

Цель занятия: изучить основные комбинаторные объекты, алгоритмы их порождения, программно реализовать и оценить времененную сложность алгоритмов.

Задания

1. Реализовать алгоритм порождения подмножеств.
2. Построить график зависимости количества всех подмножеств от мощности множества.
3. Построить графики зависимости времени выполнения алгоритмов п.1 на вашей ЭВМ от мощности множества.
4. Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на вашей ЭВМ.
5. Определить максимальную мощность множества, для которого можно получить все подмножества не более чем за час, сутки, месяц, год на ЭВМ, в 10 и в 100 раз быстрее вашей.
6. Реализовать алгоритм порождения сочетаний.
7. Построить графики зависимости количества всех сочетаний из n по k от k при $n = (5, 7, 9)$.
8. Реализовать алгоритм порождения перестановок.
9. Построить график зависимости количества всех перестановок от мощности множества.
10. Построить графики зависимости времени выполнения алгоритма п.8 на вашей ЭВМ от мощности множества.
11. Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на вашей ЭВМ.
12. Определить максимальную мощность множества, для которого можно получить все перестановки не более чем за час, сутки, месяц, год на ЭВМ, в 10 и в 100 раз быстрее вашей.
13. Реализовать алгоритм порождения размещений.
14. Построить графики зависимости количества всех размещений из n по k от k при $n = (5, 7, 9)$.

Практическое занятие 2.2

Разбиения множеств

Цель занятия: изучить различные разбиения множеств, алгоритмы их порождения, программно реализовать и оценить временную сложность алгоритмов.

Задания

1. Реализовать алгоритм порождения всех упорядоченных разбиений n -элементного множества на k подмножеств.
2. Написать программу, формирующую таблицу T (табл. 2.7), в которой строки соответствуют количеству n элементов в множестве, а столбцы — количеству k подмножеств в разбиении. Клетка таблицы $T_{n,k}$ при $k \leq n$ должна содержать количество всех упорядоченных разбиений n -элементного множества на k подмножеств, а при $k > n$ клетка таблицы $T_{n,k}$ не заполняется.

Таблица 2.7

**Количество упорядоченных разбиений
 n -элементного множества на k подмножеств**

Мощность множества	Количество k подмножеств в разбиении				
	1	2	3	N
1					
2					
3					
:					
:					
n					

3. Реализовать алгоритм порождения всех упорядоченных разбиений n -элементного множества.
4. Написать программу, вычисляющую количество всех упорядоченных разбиений n -элементного множества. Результат представить в виде таблицы (табл. 2.8).
5. Реализовать алгоритм порождения всех разбиений n -элементного множества на k подмножеств.
6. Написать программу, формирующую таблицу T (табл. 2.9), в которой строки соответствуют количеству n элементов в множестве, а столбцы — количеству k подмножеств в разбиении. Клетка таблицы

$T_{n,k}$ при $k \leq n$ должна содержать количество всех разбиений n -элементного множества на k подмножеств, а при $k > n$ клетка таблицы $T_{n,k}$ не заполняется.

Таблица 2.8

Количество упорядоченных разбиений

Мощность множества	Количество упорядоченных разбиений
1	
2	
3	
:	
:	
n	

Таблица 2.9

Количество разбиений n -элементного множества на k подмножеств

Мощность множества	Количество k подмножеств в разбиении				
	1	2	3	N
1					
2					
3					
:					
:					
n					

7. Реализовать алгоритм порождения всех разбиений n -элементного множества.

8. Написать программу, вычисляющую количество всех разбиений n -элементного множества. Результат представить в виде таблицы (табл. 2.10).

Таблица 2.10

Количество разбиений

Мощность множества	Количество разбиений
1	
2	
3	
:	
:	
n	

Практическое занятие 2.3

Задачи выбора

Цель занятия: приобрести практические навыки в использовании алгоритмов порождения комбинаторных объектов при проектировании алгоритмов решения задач выбора.

Задания

1. Ознакомиться с задачей (см. варианты заданий).
2. Определить класс комбинаторных объектов, содержащих решение задачи (траекторию задачи).
3. Определить, что в задаче является функционалом и способ его вычисления.
4. Определить способ распознавания решения по значению функционала.
5. Реализовать алгоритм решения задачи.
6. Подготовить тестовые данные и решить задачу.

Варианты заданий

1. Имеется конечное множество исполнителей $\{x_1, \dots, x_n\}$, каждый из которых может выполнять некоторые из работ $\{y_1, \dots, y_n\}$. Для каждого исполнителя задано множество работ, которые он может выполнять. Нужно распределить исполнителей по работам, т.е. назначить по одному исполнителю на каждую работу, так, чтобы выполнить все работы. Найти все возможные распределения исполнителей по работам.
2. Числа из заданного девятивалентного множества разместить в “числовом колесе” (рис.2.30) так, чтобы одно число было в центре колеса, остальные — у концов каждого диаметра, и чтобы суммы чисел каждого ряда были бы одинаковыми.

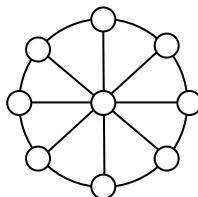


Рис.2.30. Числовое колесо

3. Определить, существуют ли решения в заданном k -элементном множестве X целых чисел следующего уравнения:

$$a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_n \cdot x_n = b, \quad n, b \text{ и } a_1, a_2, \dots, a_n \text{ — заданы, } x_i \in X.$$

4. На прямой расположены n равноотстоящих друг от друга узлов. Можно ли в узлах разместить n предметов из n -элементного множества так, чтобы центр тяжести находился в одном из узлов. Вес каждого предмета задан.

5. Задана точка с натуральными координатами (x, y) на целочисленной решетке. Найти “монотонную” траекторию минимальной стоимости, ведущую из начала координат в заданную точку. На каждом шаге “монотонной” траектории можно двигаться только вправо или вверх к соседней точке. Стоимость возможных шагов из каждой точки решетки задана.

6. Есть n станков. Каждый из станков делает различные операции (какие заданы). При обработке детали необходимо выполнить k заданных операций. Найти минимальное множество станков, требуемых для обработки детали.

7. Заданы n^2 попарно различных целых чисел. Разместить их, если можно, в матрице $n \times n$ так, чтобы суммы строк, столбцов и диагоналей были одинаковыми. Определить количество таких размещений.

8. Найти все решения уравнения $x_1 + x_2 + \dots + x_n = b$ в натуральных числах, n и b — заданы.

9. Имеется n исполнителей, каждый из которых может выполнять некоторые из n работ. Для каждого исполнителя задано множество работ, которые он может выполнять, и стоимость выполнения каждой работы. Нужно распределить исполнителей по работам, т.е. назначить по одному исполнителю на каждую работу, так, чтобы выполнить все работы, минимизируя затраты.

10. Задана точка в узле целочисленной решетке размером $n \times n$. Расположить всеми возможными способами k точек в узлах решетки так, чтобы расстояние от каждой до заданной было одинаковым.

11. Необходимо перевести n тонн груза. Имеется k самолетов грузоподъемностью s_1, s_2, \dots, s_k тонн, $s_1 + s_2 + \dots + s_k > n$. Стоимость рейса c_1, c_2, \dots, c_k руб. не зависит от массы груза, перевозимого самолетом. Найти оптимальное множество самолетов для перевозки груза.

12. Задано множество предметов. Для каждого предмета известен его вес. Можно ли разделить предметы на две равные по весу части.

13. В матрице $n \times m$ целых чисел встречаются нули. Можно ли перестановкой строк добиться того, чтобы нули в каждом столбце занимали только самые нижние позиции? Получить все такие матрицы.

14. Имеется n различных предметов. Известны масса каждого предмета и его стоимость. Определить, какие предметы надо положить в рюкзак, чтобы масса не превышала заданной, а общая стоимость была максимальна.

15. Каждая из n деталей последовательно обрабатывается на m станках (сначала на 1-м, затем на 2-м и т.д.). Для каждой детали известно время обработки на каждом станке. Определить, в какой последовательности необходимо обрабатывать детали, чтобы время обработки всех деталей было минимальным.

16. Найти все решения в $(0,1)$ -числах следующего неравенства:

$$a_1 \cdot x_1 + a_2 \cdot x_2 + \dots + a_n \cdot x_n \neq a_{n+1} \cdot x_{n+1} + a_{n+2} \cdot x_{n+2} + \dots + a_m \cdot x_m,$$

n, m и a_1, a_2, \dots, a_m — заданы.

17. Задана целочисленная матрица размером $n \times m$. Выбрать минимальное количество строк матрицы, таких, что сумма элементов каждого столбца была бы больше заданного числа.

18. Имеется n станков и n деталей. Каждую деталь можно обработать на любом станке, но время обработки детали на одном станке может отличаться от времени ее обработки на другом. Нужно разместить детали по станкам так, чтобы суммарное время работы было наименьшим.

19. Заданы множества A и B , причем $B = \{x / x \subseteq A\}$. Подмножество множества B называется покрытием множества A , если объединение всех его элементов равно множеству A . Найти все минимальные по мощности покрытия множества A , составленные из элементов множества B .

20. Необходимо перевести n тонн груза. Имеются k самолетов грузоподъемностью s_1, s_2, \dots, s_k тонн, $\sum_{i=1}^k s_i > n$. Стоимость перевозки самолетами тонны груза составляет c_1, c_2, \dots, c_k руб. Найти оптимальное множество самолетов для перевозки груза.

Контрольные вопросы и задания

1. Что называется комбинаторным объектом?
2. В чем заключается правило произведения? Для чего, когда и как его применять?
3. В чем заключается метод поиска с возвращением?
4. Изобразите общий рекурсивный алгоритм поиска с возвращением в виде блок-схемы.
5. Изобразите общий итеративный алгоритм поиска с возвращением в виде блок-схемы.
6. Докажите теорему о количестве подмножеств n -элементного множества.
7. Опишите алгоритм порождения подмножеств. Можете ли вы предложить другие алгоритмы порождения подмножеств?
8. Дайте определение перестановки.
9. Докажите теорему о количестве перестановок n -элементного множества.
10. Опишите алгоритм порождения перестановок. Предложите итеративный алгоритм порождения перестановок, используя метод поиска с возвращением.
11. Дайте определение размещению.
12. Докажите теорему о количестве размещений n -элементного множества по k местам.
13. Опишите алгоритм порождения размещений. Предложите итеративный алгоритм порождения размещений, используя метод поиска с возвращением.
14. Дайте определение размещению с повторениями.
15. Докажите теорему о количестве размещений с повторениями n -элементного множества по k местам.
16. Опишите алгоритм порождения размещений с повторениями. Предложите итеративный алгоритм порождения размещений с повторениями, используя метод поиска с возвращением.
17. Дайте определение сочетанию.
18. Докажите теорему о количестве сочетаний из n по k .
19. Опишите алгоритм порождения сочетаний. Предложите итеративный алгоритм порождения сочетаний, используя метод поиска с возвращением.
20. Предложите алгоритм порождения подмножеств в порядке увеличения (уменьшения) мощности.
21. Дайте определение мульти множеству, мощности мульти множества, кратности элемента.

22. Предложите способы хранения мультимножества в памяти ЭВМ.
23. Как реализовать операцию разности мультимножеств?
24. Дайте определение перестановки с повторениями.
25. Докажите теорему о количестве перестановок с повторениями.
26. Опишите алгоритм порождения перестановок с повторениями.
27. Дайте определение сочетания с повторениями.
28. Докажите теорему о количестве сочетаний с повторениями из n элементов по k .
29. Опишите алгоритм порождения сочетаний с повторениями.
30. Предложите алгоритм порождения сочетаний с повторениями исходя из доказательства теоремы о количестве сочетаний с повторениями из n элементов по k .
31. Дайте определение упорядоченному разбиению.
32. Докажите теорему о количестве упорядоченных разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.
33. Опишите алгоритм порождения упорядоченных разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.
34. Докажите теорему о количестве упорядоченных разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.
35. Опишите алгоритм порождения упорядоченных разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.
36. Дайте определение композиции натурального числа n из k частей с ограничением $n_i > 0$.
37. Докажите теорему о количестве композиций натурального числа n из k частей с ограничением $n_i > 0$.
38. Как можно порождать композиции натурального числа n из k частей с ограничением $n_i > 0$? Приведите алгоритмы порождения.
39. Докажите теорему о количестве всех упорядоченных разбиений n -элементного множества на k подмножеств.
40. Опишите алгоритм порождения всех упорядоченных разбиений n -элементного множества на k подмножеств.
41. Докажите теорему о количестве всех упорядоченных разбиений n -элементного множества.
42. Опишите алгоритм порождения всех упорядоченных разбиений n -элементного множества.
43. Дайте определение разбиению множества. Чем оно отличается от упорядоченного разбиения?
44. Докажите теорему о количестве разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.
45. Опишите алгоритм порождения разбиений n -элементного множества на k подмножеств с мощностями $\{n_1, n_2, \dots, n_k\}$.

46. Докажите теорему о количестве разбиений n -элементного множества на k подмножеств.

47. Используя литературу, определите, как можно вычислить количество разбиений n -элементного множества на k подмножеств, как называется это число?

48. Дайте определение разбиению натурального числа n на k частей.

49. Опишите алгоритм порождения разбиений натурального числа n на k частей.

50. Опишите алгоритм порождения всех разбиений n -элементного множества на k подмножеств.

51. Докажите теорему о количестве всех разбиений n -элементного множества.

52. Используя литературу, определите, как можно вычислить количество всех разбиений n -элементного множества, как называется это число?

53. Опишите алгоритм порождения всех разбиений n -элементного множества.

54. Какие задачи относятся к задачам выбора? Что характерно для задач выбора?

55. Что называют траекториями задачи выбора?

56. Что называют функционалом траектории, зачем он нужен?

57. В чем заключается проектирование алгоритма решения задачи выбора с использованием алгоритмов порождения комбинаторных объектов?

58. Как преобразовать алгоритм порождения комбинаторных объектов в алгоритм решения задачи выбора?

3. ОТНОШЕНИЯ

3.1. Основные понятия

Двухэлементное множество $\{a,b\}$, в котором элементы расположены в определенном порядке (порядок важен) называется *упорядоченной парой* (a,b) . Упорядоченные пары (a,b) и (b,a) неравны (в отличие от множеств $\{a,b\}$ и $\{b,a\}$). Упорядоченные пары равны, если равны их первые элементы и равны их вторые элементы. Допускается равенство первого и второго элементов пары (a,a) .

Пусть A и B — два множества. *Прямым (декартовым) произведением* двух множеств A и B называется множество упорядоченных пар, в котором первый элемент каждой пары принадлежит A , а второй принадлежит B :

$$A \times B = \{(x,y) / x \in A \text{ и } y \in B\}, |A \times B| = |A| \cdot |B|.$$

Любое подмножество $P \subseteq A \times B$ называется *бинарным соответствием из A в B*. При этом множество A называется *областью отправления*, множество B — *областью прибытия* соответствия P . Множество $\{x / (x,y) \in P \text{ и } x \in A \text{ и } y \in B\}$ называется *областью определения*, а множество $\{y / (x,y) \in P \text{ и } x \in A \text{ и } y \in B\}$ — *областью значений*. Множество $\{y / (a,y) \in P \text{ и } y \in B\}$ называется *образом элемента a* при соответствии P , а множество $\{x / (x,b) \in P \text{ и } x \in A\}$ — *прообразом элемента b* при соответствии P .

Существует много способов задания соответствия. На рис.3.1 показано графовое представление соответствия $P = \{(1,a), (3,a), (3,b), (3,c), (4,c), (4,e)\}$ из множества $A = \{1,2,3,4\}$ в множество $B = \{a,b,c,d,e\}$.

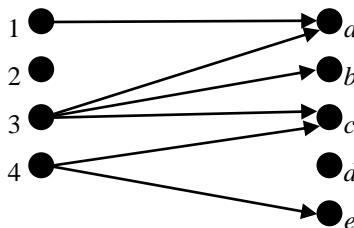


Рис.3.1. Графовое представление соответствия

Областью определения этого соответствия является множество $\{1,3,4\}$, областью значений — множество $\{a,b,c,e\}$. Образ элемента 1 — множество $\{a\}$, образ элемента 3 — множество $\{a,b,c\}$, образ элемента

4 — множество $\{c,e\}$. Прообраз элемента a — множество $\{1,3\}$, прообраз элемента b — множество $\{3\}$, прообраз элемента c — множество $\{3,4\}$, прообраз элемента e — множество $\{4\}$.

Соответствие называется *функциональным* или *функцией*, если для каждого элемента из области определения существует одноэлементный образ (*однозначное соответствие*).

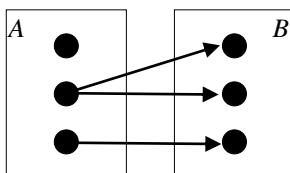
Функция, у которой область определения равна области отправления, называется *полностью определенной* или *отображением*.

Если область значений равна области прибытия, то функция называется *сюръективной* или *сюръекцией*.

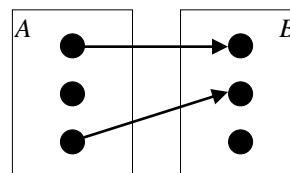
Функция называется *инъекцией*, если различные элементы из области определения имеют различные образы или, по другому, если прообразом любого элемента из области значений является одноэлементное множество.

Отображение, которое и сюръективное и инъективное, называется *биективным* или *взаимно-однозначным отображением*.

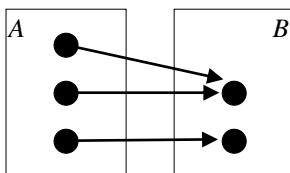
На рис.3.2 приведены различные виды соответствий из A в B . Множества A и B представлены прямоугольниками, элементы множеств — кружочками, элементы соответствий — стрелочками, направленными от элементов множества A к элементам множества B .



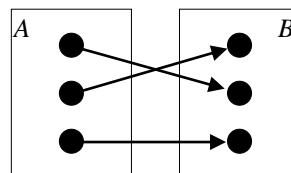
Соответствие, но не функция



Инъекция, но не сюръекция



Сюръекция, но не инъекция



Биекция

Рис.3.2. Различные виды соответствий

Степенью множества A называется его прямое произведение самого по себе : $A^1 = A$, $A^2 = A \times A$, $A^n = A \times A^{n-1}$.

Бинарным отношением R называется подмножество $R \subseteq A^2$ и является отношением на множестве A. Бинарное отношение является частным случаем бинарного соответствия, поэтому оно может быть не функцией, функцией, отображением, сюръекцией, инъекцией, биекцией.

Бинарное отношение называется *пустым*, если $R = \emptyset$, *тождественным* $I = \{(x,x) / x \in A\}$, *универсальным* $U = \{(x,y) / x \in A \text{ и } y \in A\} = A^2$.

Далее будем рассматривать только бинарные отношения и слово “бинарные” будем опускать.

Обобщенным понятием отношения является *n*-арное отношение R — подмножество прямого произведения n множеств: $R \subseteq A_1 \times A_2 \times \dots \times A_n = \{(a_1, a_2, \dots, a_n) / a_1 \in A_1, a_2 \in A_2, \dots, a_n \in A_n\}$.

3.2. Способы задания отношений

1. *Перечислением всех упорядоченных пар, принадлежащих отношению.*

$$A = \{(1,2), (1,3), (3,1)\}, \quad B = \{(3,1), (2,1), (1,3)\}, \quad C = \{(1,3), (1,2), (3,1)\}.$$

Из определения равенства множеств вытекает, что $A = C$ и $A \neq B$.

2. Заданием *характеристического свойства*, выделяющего упорядоченные пары данного отношения среди упорядоченных пар других отношений.

$$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x \leq 10 \text{ и } y < 10 \text{ и } x \text{ — четно и } y \text{ — нечетно}\}.$$

3. Описанием *порождающей процедуры* с указанием множества (или множеств), которое пробегает параметр (или параметры) этой процедуры.

$$A = \{(x,y^2) / x \in N, y \in N\}.$$

4. Пару (x,y) можно изобразить точкой на координатной плоскости с абсциссой x и ординатой y, а бинарное отношение A — множеством точек, называемым *графиком* отношения A. График отношения $A = \{(1,2), (1,3), (3,1)\}$ представлен на рис.3.3

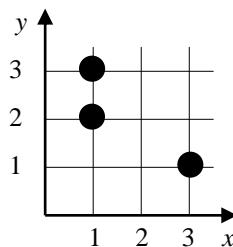


Рис.3.3. График отношения

5. Бинарное отношение A можно представить в графической форме в виде *графа*, в котором *вершины* (кружочки или точки) соответствуют элементам множества и если $(x,y) \in A$, то вершины, соответствующие x и y , соединяются *дугой* (стрелочкой) от x к y . Если же $(x,y) \in A$ и $(y,x) \in A$, то вершины, соответствующие x и y соединяются *ребром* (линией, не имеющей направления). Граф отношения $A = \{(1,2),(1,3),(3,1)\}$ представлен на рис.3.4

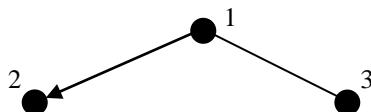


Рис.3.4. Граф отношения

6. Бинарное отношение A можно задать *характеристической функцией* $A(x,y)$, которая принимает истинное значение, если $(x,y) \in A$ и ложное — в противном случае. Характеристическую функцию отношения на конечном множестве мощностью N можно представить *матрицей* размером $N \times N$. Матрица характеристической функции отношения $A = \{(1,2),(1,3),(3,1)\}$ представлена на рис.3.5

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Рис.3.5. Матрица характеристической функции отношения

3.3. Операции над отношениями

1. *Включение* A в B ($A \subseteq B$ или $B \supseteq A$) истинно, если каждая упорядоченная пара (x,y) отношения A принадлежит отношению B .

2. *Равенство* A и B ($A = B$) истинно, если отношения A и B состоят из одних и тех же упорядоченных пар, или если $A \subseteq B$ и $B \subseteq A$.

3. *Строгое включение* A в B ($A \subset B$ или $B \supset A$) истинно, если $A \subseteq B$ и $A \neq B$.

4. *Объединение* A и B ($A \cup B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар, которые принадлежат A или B , т.е.

$$A \cup B = \{(x,y) / (x,y) \in A \text{ или } (x,y) \in B\}.$$

5. *Пересечение A и B* ($A \cap B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар, которые принадлежат каждому из отношений A и B, т.е.

$$A \cap B = \{(x,y) / (x,y) \in A \text{ и } (x,y) \in B\}.$$

6. *Разность A и B* ($A - B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар отношения A, которые не принадлежат отношению B, т.е.

$$A - B = \{(x,y) / (x,y) \in A \text{ и } (x,y) \notin B\}.$$

7. *Симметрическая разность A и B* ($A \Delta B$) есть отношение, состоящее из всех тех и только тех упорядоченных пар отношения A, которые не принадлежат отношению B и только тех упорядоченных пар отношения B, которые не принадлежат отношению A, т.е.

$$A \Delta B = \{(x,y) / (x,y) \in A \text{ и } (x,y) \notin B \text{ или } (x,y) \in B \text{ и } (x,y) \notin A\}.$$

8. *Дополнение A* до универсального отношения U (\overline{A}) есть отношение, состоящее из всех тех и только тех упорядоченных пар универсального отношения U, которые не принадлежат отношению A, т.е.

$$\overline{A} = \{(x,y) / (x,y) \notin A\}.$$

Операции 1 — 8 над отношениями аналогичны операциям над множествами.

9. *Обращением A* (A^{-1}) называется отношение, которое состоит из всех тех и только тех упорядоченных пар (x,y) , для которых (y,x) принадлежит отношению A, т.е.

$$A^{-1} = \{(x,y) / (y,x) \in A\}.$$

10. *Композицией* (суперпозицией или умножением) A и B ($A \circ B$) называется отношение, состоящее из всех тех и только тех упорядоченных пар (x,y) , для которых найдётся хотя бы один элемент z такой, что $(x,z) \in A$ и $(z,y) \in B$, т.е.

$$A \circ B = \{(x,y) | \text{существует } z \text{ такой, что } (x,z) \in A \text{ и } (z,y) \in B\}.$$

11. *Степенью* отношения A называется его композиция с самим собой: $A^0 = I$, $A^1 = A$, $A^2 = A \circ A$, $A^n = A \circ A^{n-1}$.

Пара (x,y) принадлежит отношению A^n , если в графе отношения A есть цепочка из n дуг (ребер), соединяющая вершину x с вершиной y .

Отношения часто встречаются в математике и в повседневной жизни. К математическим отношениям на множестве целых чисел можно отнести отношение МЕНЬШЕ, представляющее собой множество таких упорядоченных пар, в которых первый элемент меньше второго. Харак-

теристическим свойством это отношение можно задать так:

$$\text{МЕНЬШЕ} = \{(x,y) \mid x < y\}.$$

По аналогии определим отношения БОЛЬШЕ, РАВНО, НЕРАВНО, МЕНЬШЕ ИЛИ РАВНО, БОЛЬШЕ ИЛИ РАВНО:

$$\text{БОЛЬШЕ} = \{(x,y) \mid x > y\}$$

$$\text{РАВНО} = \{(x,y) \mid x = y\}$$

$$\text{НЕРАВНО} = \{(x,y) \mid x \neq y\}$$

$$\text{МЕНЬШЕ ИЛИ РАВНО} = \{(x,y) \mid x \leq y\}$$

$$\text{БОЛЬШЕ ИЛИ РАВНО} = \{(x,y) \mid x \geq y\}$$

В результате выполнения операций над этими отношениями будем получать новые отношения.

$$\text{МЕНЬШЕ} \cup \text{РАВНО} = \text{МЕНЬШЕ ИЛИ РАВНО}$$

$$\text{МЕНЬШЕ ИЛИ РАВНО} \cap \text{БОЛЬШЕ ИЛИ РАВНО} = \text{РАВНО}$$

$$\text{МЕНЬШЕ ИЛИ РАВНО} \Delta \text{БОЛЬШЕ ИЛИ РАВНО} = \text{НЕРАВНО}$$

$$\text{МЕНЬШЕ ИЛИ РАВНО} - \text{РАВНО} = \text{МЕНЬШЕ}$$

$$\underline{\text{МЕНЬШЕ}}^{-1} = \text{БОЛЬШЕ}$$

$$\underline{\text{МЕНЬШЕ}} = \text{БОЛЬШЕ ИЛИ РАВНО}$$

$$\underline{\text{РАВНО}}^{-1} = \text{РАВНО}$$

$$\underline{\text{РАВНО}} = \text{НЕРАВНО}$$

$$\text{МЕНЬШЕ} \circ \text{МЕНЬШЕ} = \text{МЕНЬШЕ}^2$$

Отношение МЕНЬШЕ² можно назвать МЕНЬШЕ КАК МИНИМУМ НА ДВА (почему?).

Теперь рассмотрим некоторые родственные отношения на множестве людей. Отношение ЖЕНА представляет собой множество таких упорядоченных пар, в которых первый человек является женой второго человека. Характеристическим свойством это отношение можно задать так: ЖЕНА = $\{(x,y) \mid x — жена y\}$. По аналогии определим другие родственные отношения. Далее представим результаты выполнения операций над такими отношениями.

$$\text{ЖЕНА}^{-1} = \text{МУЖ}$$

$$(\text{МАТЬ} \cup \text{ОТЕЦ})^{-1} = \text{РЕБЕНОК}$$

$$\text{МАТЬ} \circ (\text{МАТЬ} \cup \text{ОТЕЦ}) = \text{БАБУШКА}$$

$$\text{ОТЕЦ} \circ (\text{МАТЬ} \cup \text{ОТЕЦ}) = \text{ДЕД}$$

$$\text{СЫН} \circ (\text{СЫН} \cup \text{ДОЧЬ}) = \text{ВНУК}$$

$$\text{МАТЬ} \circ \text{ЖЕНА} = \text{ТЕЩА}$$

$$\text{РЕБЕНОК} - \text{СЫН} = \text{ДОЧЬ}$$

$$\text{МАТЬ} \cup \text{ОТЕЦ} = \text{РОДИТЕЛЬ}$$

$$\text{ЖЕНА} \circ \text{ОТЕЦ} = \text{МАТЬ ИЛИ МАЧЕХА}$$

$$\text{СЫН} \circ (\text{МАТЬ} \cup \text{ОТЕЦ}) = \text{БРАТ}$$

3.4. Алгоритмы реализации операций над отношениями

Бинарное отношение R на множестве A будем задавать *характеристической функцией* $R(x,y)$, представленной *матрицей* R размером $N \times N$, где $N = |A|$ и $R_{x,y}$ принимает истинное значение, если $(x,y) \in R$ и ложное — в противном случае. Для хранения матрицы будем использовать двумерный массив R с элементами типа *boolean*.

Алгоритм 3.1 (рис.3.6) вычисления равенства отношений A и B ($A = B$).

Вход: A — двумерный массив, хранящий матрицу отношения A ;

B — двумерный массив, хранящий матрицу отношения B ;

N — мощность множества.

Выход: $F = true$, если $A = B$, иначе $F = false$.

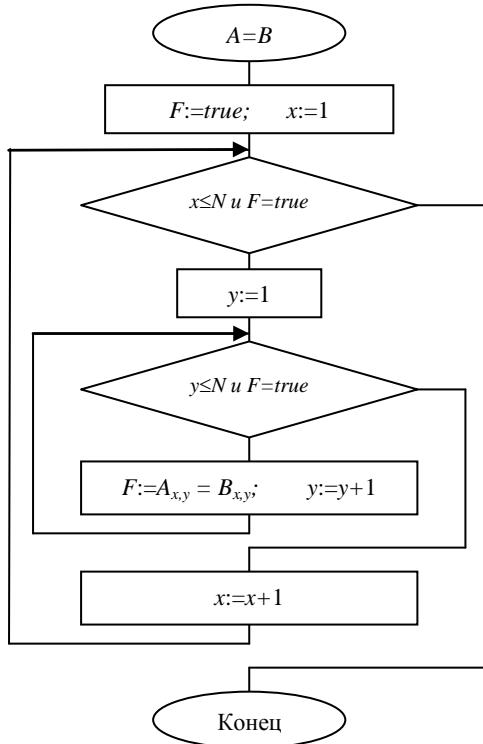


Рис.3.6. Блок-схема алгоритма вычисления равенства отношений

Алгоритм 3.2 (рис.3.7) вычисления включения отношения A в B ($A \subseteq B$).

Вход: A — двумерный массив, хранящий матрицу отношения A ;
 B — двумерный массив, хранящий матрицу отношения B ;
 N — мощность множества.

Выход: $F = true$, если $A \subseteq B$, иначе $F = false$.

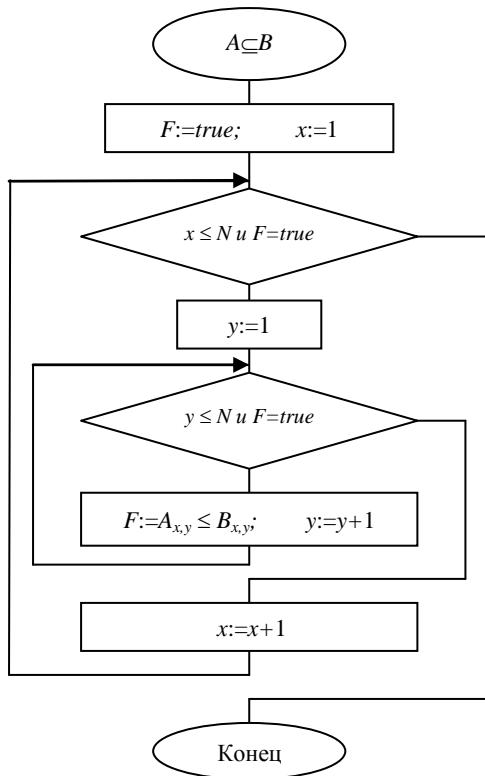


Рис.3.7. Блок-схема алгоритма вычисления включения отношения A в B

Алгоритм 3.3 (рис.3.8) вычисления объединения отношений A и B ($A \cup B$).

Вход: A — двумерный массив, хранящий матрицу отношения A ;

B — двумерный массив, хранящий матрицу отношения B ;

N — мощность множества.

Выход: C — двумерный массив, хранящий матрицу объединения A и B .

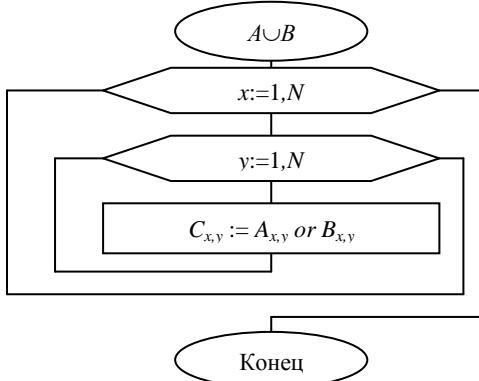


Рис.3.8. Блок-схема алгоритма вычисления объединения отношений

Алгоритм 3.4 (рис.3.9) вычисления пересечения отношений A и B ($A \cap B$).

Вход: A — двумерный массив, хранящий матрицу отношения A ;

B — двумерный массив, хранящий матрицу отношения B ;

N — мощность множества.

Выход: C — двумерный массив, хранящий матрицу пересечения A и B .

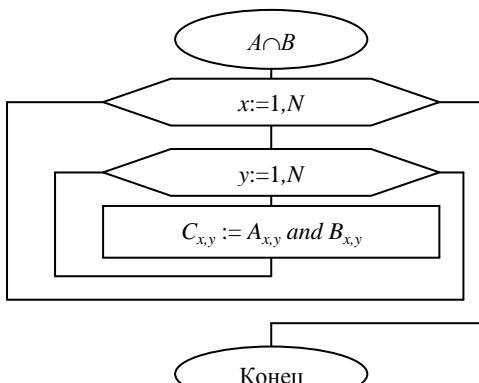


Рис.3.9. Блок-схема алгоритма вычисления пересечения отношений

Алгоритм 3.5 (рис.3.10) вычисления разности отношений A и B ($A - B$).

Вход: A — двумерный массив, хранящий матрицу отношения A ;

B — двумерный массив, хранящий матрицу отношения B ;

N — мощность множества.

Выход: C — двумерный массив, хранящий матрицу разности A и B .

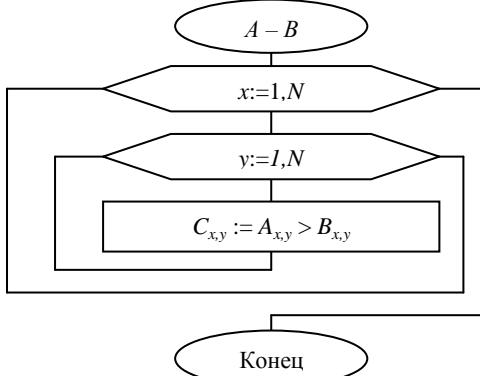


Рис.3.10. Блок-схема алгоритма вычисления разности отношений

Алгоритм 3.6 (рис.3.11) вычисления симметрической разности отношений A и B ($A \Delta B$).

Вход: A — двумерный массив, хранящий матрицу отношения A ;

B — двумерный массив, хранящий матрицу отношения B ;

N — мощность множества.

Выход: C — двумерный массив, хранящий матрицу симметрической разности A и B .

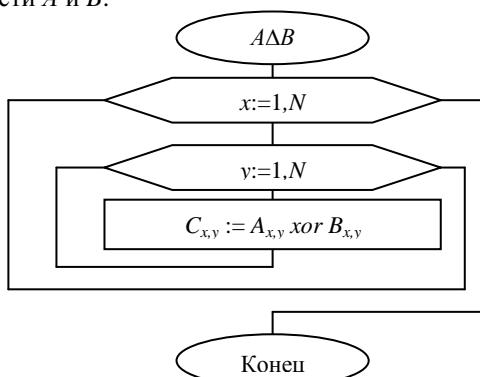


Рис.3.11. Блок-схема алгоритма вычисления симметрической разности отношений

Алгоритм 3.7 (рис.3.12) вычисления дополнения отношения A (\bar{A}).
 Вход: A — двумерный массив, хранящий матрицу отношения A ;
 N — мощность множества.
 Выход: C — двумерный массив, хранящий матрицу дополнения A .

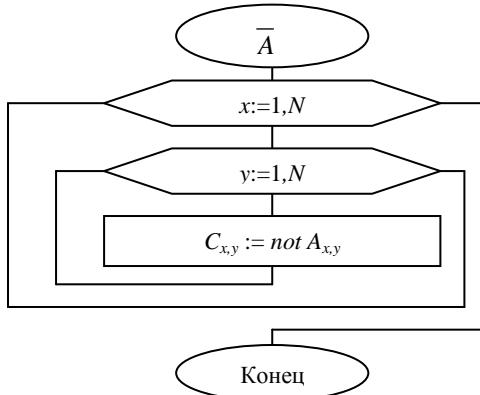


Рис.3.12. Блок-схема алгоритма вычисления дополнения отношения

Алгоритм 3.8 (рис.3.13) вычисления обращения отношения A (A^{-1}).
 Вход: A — двумерный массив, хранящий матрицу отношения A ;
 N — мощность множества.
 Выход: C — двумерный массив, хранящий матрицу обращения A .

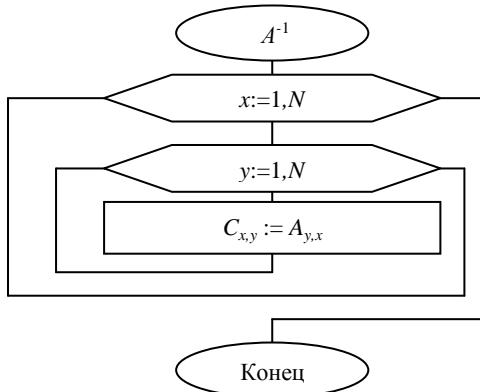


Рис.3.13. Блок-схема алгоритма вычисления обращения отношения

Алгоритм 3.9 (рис.3.14) вычисления композиции отношений A и B (A o B).

Вход: A — двумерный массив, хранящий матрицу отношения A;

B — двумерный массив, хранящий матрицу отношения B;

N — мощность множества.

Выход: C — двумерный массив, хранящий матрицу отношения A о B.

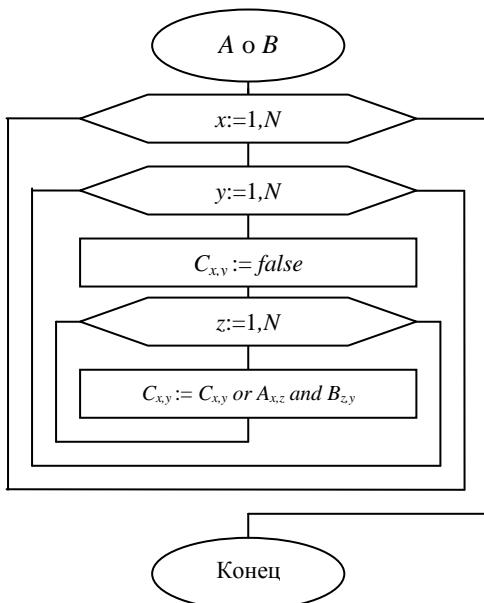


Рис.3.14. Блок-схема алгоритма вычисления композиции отношений

Алгоритм вычисления композиции (произведения) отношений представляет собой обычный алгоритм вычисления произведения матриц, в котором арифметические операции ($+, \cdot$) заменены логическими (*or, and*). Учитывая то, что элементы матрицы имеют булевский тип, самый вложенный цикл с фиксированным числом повторений целесообразно заменить итерационным циклом с предусловием, в котором выражение $C_{x,y} = \text{false}$ и $z \leq N$ является условием входа в цикл.

3.5. Свойства отношений

Пусть R — бинарное отношение на множестве A , I — тождественное отношение, U — универсальное отношение. Вместо $(x,y) \in R$ будем иногда использовать обозначение xRy . Отношения могут обладать следующими основными свойствами:

1. R рефлексивно, если xRx для всех $x \in A$ (граф такого отношения имеет петли при каждой вершине). Рефлексивность отношения R определяется истинностью выражения $I \subseteq R$.

2. R антирефлексивно, если $(x,x) \notin R$ для всех $x \in A$ (граф такого отношения не имеет петель). Антирефлексивность отношения R определяется истинностью выражения $R \cap I = \emptyset$. Нерефлексивное отношение в общем случае не является антирефлексивным. Отношение, граф которого содержит вершины с петлями и без петель — нерефлексивное и не антирефлексивное.

3. R симметрично, если из xRy следует yRx для всех $x,y \in A$ (такое отношение изображается неориентированным графом, возможно с петлями). Симметричность отношения R определяется истинностью выражения $R = R^{-1}$.

4. R антисимметрично, если из xRy следует $(y,x) \notin R$ для всех $x,y \in A$ (такое отношение изображается ориентированным графом, возможно с петлями). Антисимметричность отношения R определяется истинностью выражения $R \cap R^{-1} \subseteq I$. Отношение, граф которого содержит ориентированные дуги и неориентированные рёбра, не является ни симметричным, ни антисимметричным.

5. R транзитивно, если из xRz и zRy следует xRy для всех $x,y,z \in A$. Транзитивность отношения R определяется истинностью выражения $R \circ R \subseteq R$.

6. R антитранзитивно, если из xRz и zRy следует $(x,y) \notin R$ для всех $x,y,z \in A$. Антитранзитивность отношения R определяется истинностью выражения $(R \circ R) \cap R = \emptyset$.

7. R полно, если из $x \neq y$ следует, что xRy или yRx для всех $x,y \in A$ (такое отношение изображается полным графом). Полнота отношения R определяется истинностью выражения $R \cup I \cup R^{-1} = U$.

Следующие производные свойства отношений определяются совокупностью основных свойств.

Отношение R является отношением

- толерантности, если оно рефлексивно, симметрично;
- эквивалентности, если оно рефлексивно, симметрично и транзитивно (обозначается \sim);
- порядка, если оно антисимметрично и транзитивно;

- нестрогого порядка, если оно отношение порядка (антисимметрично и транзитивно) и рефлексивно (обозначается \leq);
- строгого порядка, если оно отношение порядка (антисимметрично и транзитивно) и антирефлексивно (обозначается $<$);
- линейного порядка, если оно отношение порядка (антисимметрично и транзитивно) и полно;
- нестрогого линейного порядка, если оно отношение нестрогого порядка (антисимметрично, транзитивно и рефлексивно) и полно;
- строгого линейного порядка, если оно отношение строгого порядка (антисимметрично, транзитивно и антирефлексивно) и полно.

В табл. 3.1 приведены примеры отношений на множествах целых чисел, прямых и окружностей на плоскости, на множестве людей и их основные и производные свойства.

Таблица 3.1
Отношения и их свойства

Отношение	Рефлексивность	Антирефлексивность	Симметричность	Антисимметричность	Транзитивность	Антитранзитивность	Полнота	Производное свойство
1	2	3	4	5	6	7	8	9
Равно на множестве целых чисел	+		+		+			Эквивалентность
Не равно на множестве целых чисел		+	+				+	
Больше на множестве целых чисел		+		+	+		+	Строгий линейный порядок
Больше или равно на множестве целых чисел	+			+	+		+	Нестрогий линейный порядок
Конгруэнтность на множестве треугольников	+		+		+			Эквивалентность
Включение на подмножествах некоторого множества	+			+	+			Нестрогий порядок

Окончание табл.3.1

1	2	3	4	5	6	7	8	9
Непустое пересечение на подмножествах некоторого множества	+		+					Толерантность
Параллельность прямых на плоскости	+		+		+			Эквивалентность
Перпендикулярность прямых на плоскости		+	+			+		
Касание окружностей на плоскости		+	+					
Концентричность окружностей на плоскости	+		+		+			Эквивалентность
Старше на множестве людей		+		+	+		+	Строгий линейный порядок
Мать на множестве людей		+		+		+		
Родственник на множестве людей	+		+		+			Эквивалентность
Предок на множестве людей		+		+	+			Строгий порядок
Брат на множестве людей	+				+			
Брат на множестве мужчин	+		+		+			Эквивалентность
Двоюродный брат на множестве мужчин		+	+					
Друг на множестве людей	+		+					Толерантность
Семья на множестве людей, живущих в одной квартире	+		+		+		+	Эквивалентность
Семья на множестве людей, живущих в одном городе	+		+		+			Эквивалентность
Любовь на множестве людей	+		+			+		Толерантность

3.6. Замыкание отношений

Замыканием отношения A относительно свойства S , называется отношение C_S , наименьшее по мощности из всех отношений, обладающих свойством S и содержащих A в качестве подмножества. Для того, чтобы получить замыкание C_S отношения A относительно свойства S , в него нужно добавить минимально возможное количество пар, после чего полученное отношение будет обладать свойством S .

Алгоритмы вычисления рефлексивного и симметричного замыкания довольно просты. Чтобы получить рефлексивное замыкание C_{ref} отношения A , достаточно в отношение A добавить все пары вида (x,x) , т.е. $C_{ref} = A \cup I$. Чтобы получить симметричное замыкание C_{sim} отношения A , нужно просмотреть все пары отношения A и если пара $(x,y) \in A$, то пару (y,x) добавить в отношение, т.е. $C_{sim} = A \cup A^{-1}$.

Транзитивное замыкание C_{tran} отношения A вычисляется сложнее. Рассмотрим несколько алгоритмов его вычисления.

Предположим, что отношение A обладает свойством транзитивности, тогда $C_{tran} = A$. Для определения транзитивности C_{tran} вычислим $C2 = C_{tran}^2$ и проверим истинность $C2 \subseteq C_{tran}$. Если $C2 \subseteq C_{tran}$ истинно, то C_{tran} транзитивно и представляет собой транзитивное замыкание отношения A . Если же $C2 \subseteq C_{tran}$ ложно, то C_{tran} не транзитивно и к нему нужно добавить пары, принадлежащие $C2 : C_{tran} := C_{tran} \cup C2$. В результате добавления пар, принадлежащих $C2$, отношение C_{tran} может стать транзитивным или не транзитивным. Для проверки опять вычислим $C2 = C_{tran}^2$ и повторим процесс.

Алгоритм 3.10 (рис.3.15) вычисления транзитивного замыкания отношения A .

Вход: A — двумерный массив, хранящий матрицу отношения A на множестве M .

Выход: C — двумерный массив, хранящий матрицу транзитивного замыкания отношения A .

Существует наглядная интерпретация понятия транзитивного замыкания в терминах представления отношения в виде графа. Пусть дан график, представляющий отношение A . Пара (x,y) принадлежит транзитивному замыканию отношения A тогда и только тогда, когда в графике отношения A есть цепочка дуг, ведущая из вершины x в вершину y . Предположим, в графике отношения A вершина x соединяется с вершиной y цепочкой из k дуг, тогда пара (x,y) принадлежит A^k и транзитивному замыканию C_{tran} отношения A . Если в графике N вершин и вершины x и

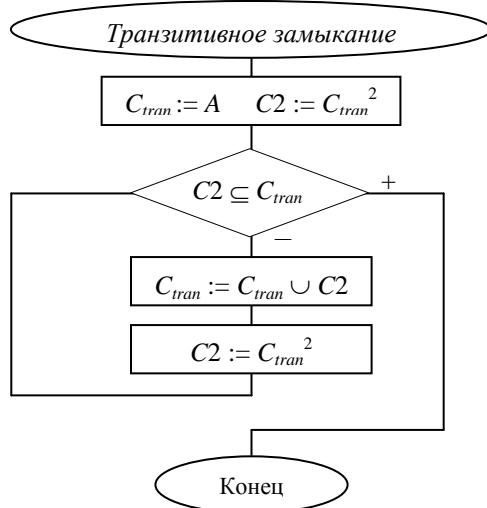


Рис.3.15. Блок-схема алгоритма вычисления транзитивного замыкания

у соединяются цепочкой дуг, то обязательно найдется цепочка, состоящая не более чем из N дуг, соединяющая эти вершины. Поэтому для получения транзитивного замыкания C_{tran} в исходное отношение A нужно добавить пары, принадлежащие A^2, A^3, \dots, A^N , т. е. $C_{tran} = \cup A^i$, $i = 1, N$.

Алгоритм 3.11 (рис.3.16) вычисления транзитивного замыкания отношения A .

Вход: A — двумерный массив, хранящий матрицу отношения A на множестве M ;

N — мощность множества M .

Выход: C — двумерный массив, хранящий матрицу транзитивного замыкания отношения A .

Рассмотрим графический способ получения транзитивного замыкания, основанный на преобразовании графа исходного отношения.

Для того, чтобы построить граф отношения C_{tran} , являющимся транзитивным замыканием отношения A , изготовим копию графа отношения A . Затем будем отыскивать такие тройки вершин x, y и z , для которых в графе есть дуги из x в z и из z в y . Для каждой такой тройки добавим в граф дугу из x в y , если такой дуги еще нет. Когда таким способом уже нельзя добавить новых дуг, каждые две вершины, соединенные

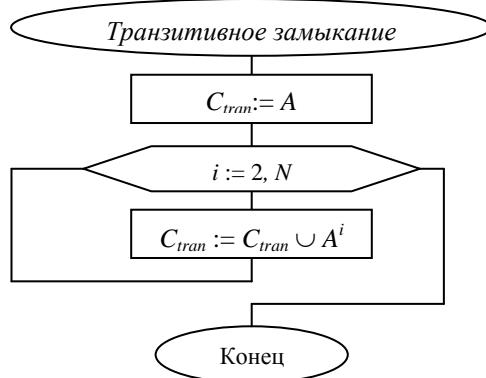
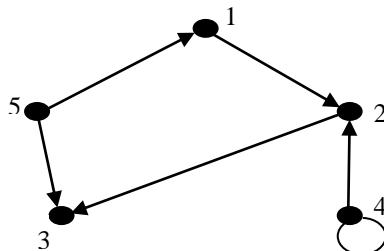


Рис.3.16. Блок-схема алгоритма вычисления транзитивного замыкания

цепочкой дуг, оказываются соединенными одной дугой. Следовательно, построение закончено и граф транзитивного замыкания отношения A получен.

Для того чтобы систематизировать процесс добавления дуг, будем последовательно рассматривать все вершины графа с первой до последней. Для каждого пути длины два, проходящего через рассматриваемую вершину, проведем дугу из начальной вершины этого пути в его конечную вершину. Граф транзитивного замыкания отношения A будет построен после однократного исследования всех вершин графа. Повторное исследование вершин не может добавить новых дуг (доказано Уоршаллом в 1962 году).

Рассмотрим пример. Отношение A задано графиком на рис.3.17.

Рис.3.17. Граф отношения A

Анализируя вершину 1 видим дуги из 5 в 1 и из 1 в 2, образующие путь длины два и проходящий через вершину 1, поэтому добавляем дугу из 5 в 2 (рис.3.18, а). В вершину 2 теперь входят три дуги из 1, 4 и 5, а выходит одна дуга в вершину 3, поэтому добавляем дуги из 1 в 3, из 4 в

3, а дуга из 5 в 3 уже есть (рис.3.18, б). Из третьей вершины дуги не выходят, поэтому ничего не добавляем. Из четвертой вершины ребро идет в нее же и выходит во вторую, в этом случае новые дуги не добавляются. В пятую вершину дуги не входят, поэтому новые дуги не вводим. Таким образом исследованы все вершины графа, добавлены все необходимые дуги и граф (рис.3.18, б) транзитивного замыкания получен.

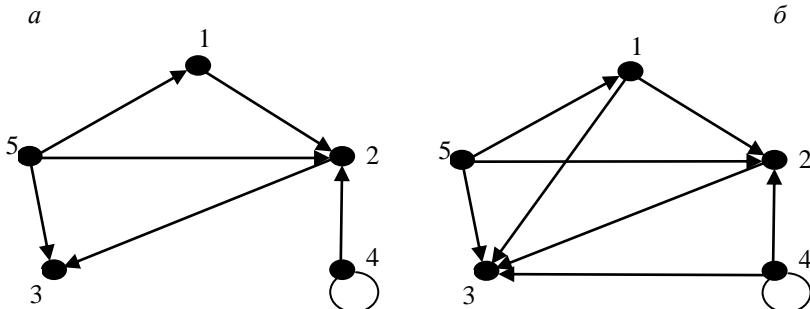


Рис.3.18. Формирование транзитивного замыкания отношения A

Алгоритм 3.12 (рис.3.19) вычисления транзитивного замыкания отношения A основан на рассмотренном выше принципе (алгоритм Уоршалла).

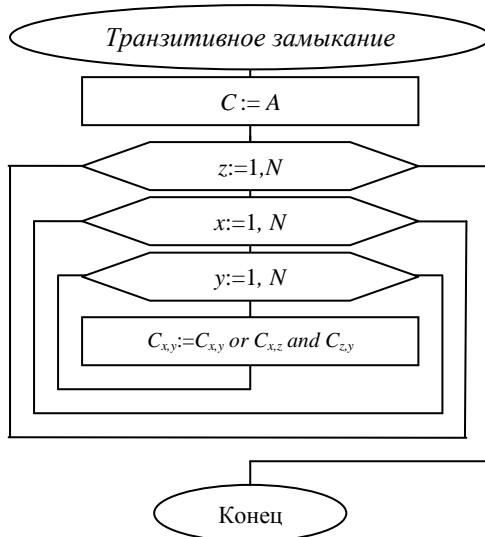


Рис.3.19. Блок-схема алгоритма вычисления транзитивного замыкания

3.7. Классы эквивалентности и фактормножества

Пусть задано отношение эквивалентности R на множестве M и $x \in M$. Подмножество элементов множества M , эквивалентных x , называется *классом эквивалентности* для x : $[x] = \{y / y \in M \text{ и } yRx\}$.

Алгоритм 3.13 построения класса эквивалентности $[x]$.

Вход: M — множество;

$x \in M$;

R — отношение эквивалентности.

Выход: $[x]$ — класс эквивалентности для x .

1. $[x] := \{x\}$;
2. Для всех $y \in M$ выполнить:
если yRx , то включить y в $[x]$;
3. Конец.

Для любого элемента x множества M класс эквивалентности не пуст, так как содержит в себе по крайней мере элемент x ($x \sim x$).

Любые два эквивалентных элемента x и y образуют равные классы эквивалентности, т.е. если $x \sim y$, то $[x] = [y]$.

Неэквивалентные элементы x и y образуют непересекающиеся классы эквивалентности.

Всякое отношение эквивалентности R на множестве M определяет единственное разбиение множества M на классы эквивалентности. Множество всех классов эквивалентности называется *фактормножеством* множества M по эквивалентности R .

Алгоритм 3.14 построения разбиения множества M на классы эквивалентности.

Вход: множество M и отношение эквивалентности R .

Выход: S — разбиение множества M на классы эквивалентности.

1. $A := M; S := \emptyset$;
2. Пока $A \neq \emptyset$ выполнять:
Для $A, x \in A$ и R построить $[x]$ по алгоритму 3.1,
добавить его в S и вычесть из A ;
3. Конец.

Всякое разбиение S множества M определяет единственное отношение эквивалентности R на множестве M . Ниже приведены два алгоритма, определяющие отношение эквивалентности R на множестве M по заданному разбиению S .

Алгоритм 3.15 построения отношения эквивалентности R по разбиению S множества M .

Вход: M — множество;

S — разбиение множества M .

Выход: R — отношение эквивалентности на множестве M , определяемое разбиением S .

1. $R := \emptyset$;
2. Для всех пар $(x,y) \in M^2$ выполнить:
если x и y принадлежат одному и тому же подмножеству разбиения S , то (x,y) включить в R ;
3. Конец.

Алгоритм 3.16 построения отношения эквивалентности R по разбиению S множества M .

Вход: M — множество;

S — разбиение множества M .

Выход: R — отношение эквивалентности на множестве M , определяемое разбиением S .

1. $R := \emptyset$;
2. Для всех подмножеств $S_i \in S$ выполнить:
каждую пару (x,y) , такую, что $x \in S_i$ и $y \in S_i$, включить в R ;
3. Конец.

3.8. Упорядоченные множества

Множество вместе с заданным на нем отношением порядка называют *упорядоченным множеством*. Множество M с заданным на нем отношением порядка \leq будем записывать как пару (M, \leq) .

Элементы x и y упорядоченного множества (M, \leq) называют *сравнимыми* по отношению порядка \leq , если $x \leq y$ или $y \leq x$ (напомним, что запись $x \leq y$ означает, что пара (x, y) принадлежит отношению \leq). В противном случае элементы x и y называются *несравнимыми*. Если отношение \leq является отношением линейного порядка, то все элементы множества M попарно сравнимы и множество M называется *линейно упорядоченным*.

Элемент $a \in M$ называют *наибольшим элементом* множества M , если для всех $x \in M$ верно $x \leq a$.

Элемент $b \in M$ называют *максимальным элементом* множества M , если для всех $x \in M$ верно $x \leq b$ или x и b несравнимы.

Аналогично определяются *наименьший* и *минимальный* элементы множества.

Минимальный и максимальный элемент существует в любом упорядоченном множестве, причем как максимальных, так и минимальных элементов может быть больше одного. Наибольший и наименьший элемент в упорядоченном множестве может и не существовать. Если же наибольший (наименьший) элемент существует, то он единственный и является максимальным (минимальным).

Примером упорядоченного множества может служить множество $\{1, 2, 3, 5, 6, 10, 15, 30\}$ с отношением делимости, т.е. пара (x, y) принадлежит отношению делимости, если x является делителем y (y делится на x без остатка). Матрица характеристической функции отношения делимости на множестве $\{1, 2, 3, 5, 6, 10, 15, 30\}$ представлена на рис. 3.20.

	1	2	3	5	6	10	15	30
1	1	1	1	1	1	1	1	1
2	0	1	0	0	1	1	0	1
3	0	0	1	0	1	0	1	1
5	0	0	0	1	0	1	1	1
6	0	0	0	0	1	0	0	1
10	0	0	0	0	0	1	0	1
15	0	0	0	0	0	0	1	1
30	0	0	0	0	0	0	0	1

Рис.3.20. Матрица характеристической функции отношения делимости

Отношением *< строгого порядка, ассоциированного с отношением порядка \leq* будем называть отношение, состоящее из всех таких пар (x, y) , что $x \leq y$ и $x \neq y$. Чтобы получить матрицу отношения $<$, нужно обнулить все элементы главной диагонали матрицы отношения \leq .

Отношением *\lhd доминирования, ассоциированного с отношением порядка \leq* будем называть отношение, состоящее из всех таких пар (x, y) , что $x < y$ и не существует такого элемента z , что $x < z < y$. Чтобы получить матрицу отношения \lhd , нужно проанализировать каждую еди-

ницу в матрице отношения $<$, и, если единица соответствует паре (x, y) и существует такой элемент z , что $x < z < y$, то заменить ее нулем. Матрица характеристической функции отношения доминирования, ассоциированного с отношением делимости на множестве $\{1, 2, 3, 5, 6, 10, 15, 30\}$, представлена на рис.3.21.

	1	2	3	5	6	10	15	30
1	0	1	1	1	0	0	0	0
2	0	0	0	0	1	1	0	0
3	0	0	0	0	1	0	1	0
5	0	0	0	0	0	1	1	0
6	0	0	0	0	0	0	0	1
10	0	0	0	0	0	0	0	1
15	0	0	0	0	0	0	0	1
30	0	0	0	0	0	0	0	0

Рис.3.21. Матрица характеристической функции отношения доминирования

Отношение доминирования удобно представлять графически в виде *диаграммы Хассе*. На этой диаграмме элементы множества изображаются кружочками. При этом если $x \triangleleft y$, то кружочек, изображающий элемент y , располагается выше кружочка, изображающего элемент x , и соединяются стрелочкой, ведущей от x к y . На рис. 3.22 представлена диаграмма Хассе отношения доминирования, ассоциированного с отношением делимости на множестве $\{1, 2, 3, 5, 6, 10, 15, 30\}$.

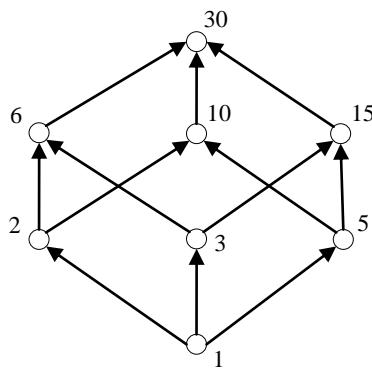


Рис. 3.22. Диаграмма Хассе

На диаграмме Хассе элементы множества располагаются по уровням. Элемент y располагается на нулевом уровне, если не существует такого элемента x , что $x \triangleleft y$. Элементы нулевого уровня соответствуют нулевым столбцам матрицы отношения доминирования. Если из матрицы удалить строки и столбцы, соответствующие элементам нулевого уровня, то элементы, соответствующие нулевым столбцам полученной матрицы, представляют собой элементы второго уровня. Продолжая процесс, получим элементы каждого уровня. Распределение элементов множества по уровням называют *топологической сортировкой*. При программной реализации топологической сортировки не стоит удалять строки и столбцы матрицы и каждый раз анализировать всю матрицу с целью поиска нулевых столбцов. Можно поступить следующим образом:

- 1) найти суммы элементов по всем столбцам матрицы и сохранить их в массиве W ;
- 2) элементы множества, соответствующие нулевым элементам массива W , считать элементами очередного уровня;
- 3) нулевые элементы массива W заменить отрицательным числом;
- 4) если в массиве W есть неотрицательные элементы, то вычесть из него поэлементно каждую строку матрицы, соответствующую элементу полученного в п. 2 уровня и перейти к п. 2, иначе конец алгоритма.

Применим описанный алгоритм к матрице на рис. 3.21.

Суммируя элементы по столбцам матрицы, получим массив $W = (0, 1, 1, 1, 2, 2, 2, 3)$. В этом массиве первый элемент равен нулю, следовательно, соответствующий элемент множества (1) принадлежит нулевому уровню. Заменим его значением -1 и вычтем из массива W строку матрицы, соответствующую элементу 1 множества. Получим $W = (-1, 0, 0, 0, 2, 2, 2, 3)$. В этом массиве второй, третий и четвертый элементы равны нулю, следовательно, соответствующие элементы множества (2, 3 и 5) принадлежат первому уровню. Заменим нули значением -1 и вычтем из массива W строки матрицы, соответствующие элементам 2, 3 и 5 множества. Получим $W = (-1, -1, -1, -1, 0, 0, 0, 3)$. В этом массиве пятый, шестой и седьмой элементы равны нулю, следовательно, соответствующие элементы множества (6, 10 и 15) принадлежат второму уровню. Заменим нули значением -1 и вычтем из массива W строки матрицы, соответствующие элементам 6, 10 и 15 множества. Получим $W = (-1, -1, -1, -1, -1, -1, 0)$. Последний элемент массива равен нулю, следовательно, элемент множества (30) принадлежит третьему уровню. Заменим последний элемент значением -1 . Неотрицательных элементов теперь в массиве нет, конец алгоритма.

Практическое занятие 3.1

Отношения и их свойства

Цель занятия: изучить способы задания отношений, операции над отношениями и свойства отношений, научиться программно реализовывать операции и определять свойства отношений.

Задания

Часть 1. Операции над отношениями

- 1.1. Представить отношения на множестве $\{1,2,3,4,5,6,7,8,9,10\}$ (см. "Варианты заданий", п.а) графиком, графом и матрицей.
- 1.2. Вычислить значение выражения (см. "Варианты заданий", п.б) при заданных отношениях (см. "Варианты заданий", п.а).
- 1.3. Написать программы, формирующие матрицы заданных отношений (см. "Варианты заданий", п.а).
- 1.4. Программно реализовать операции над отношениями.
- 1.5. Написать программу, вычисляющую значение выражения (см. "Варианты заданий", п.б) и вычислить его при заданных отношениях (см. "Варианты заданий", п.а).

Часть 2. Свойства отношений

- 2.1. Определить основные свойства отношений (см. "Варианты заданий", п.а).
- 2.2. Определить, являются ли заданные отношения отношениями толерантности, эквивалентности и порядка.
- 2.3. Написать программу, определяющую свойства отношения, в том числе толерантности, эквивалентности и порядка, и определить свойства отношений (см. "Варианты заданий", п.а).

Варианты заданий

Вариант 1

- a) $A = \{(x,y) / x \in N \wedge y \in N \wedge x < 11 \wedge y < 11 \wedge (x \text{ и } y \text{ четные})\}$
 $B = \{(x,y) / x \in N \wedge y \in N \wedge x < 11 \wedge y < 11 \wedge |x - y| < 5\}$
 $C = \{(x,y) / x \in N \wedge y \in N \wedge x < 11 \wedge y < 11 \wedge y^2 \text{ кратно } x\}$
- б) $D = (A \cup B)^{-1} \circ C \Delta A^2$

Вариант 2

- а) $A = \{(x,y) / x \in N \wedge y \in N \wedge x < 11 \wedge y < 11 \wedge x — \text{четно и } y > x\}$
 $B = \{(x,y) / x \in N \wedge y \in N \wedge x < 11 \wedge y < 11 \wedge (x - y = 1 \text{ или } x + y = 11)\}$

$$C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x,y) \in \{2,5,8,9,10\} \times \{1,3,5,10\}\}$$

6) $D = A^{-1} \cap B \cup \overline{A} - C$

Вариант 3

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x > y \text{ или } y > 7)\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } 10x + y \text{ кратно } 3\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x \text{ — четно и } y \text{ — нечетно}\}$
6) $D = A \circ B \cup C^2 \Delta \overline{A}$

Вариант 4

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x \text{ — четно и } x + y < 10\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + 2y < 20\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x > 7 \text{ или } y \text{ кратно } 3)\}$
6) $D = (A - B)^{-1} \cup C \cap \overline{A}$

Вариант 5

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + y \text{ — четно}\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } |x - y| > 5\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } \text{если } x \text{ — четно, то } y < x, \text{ иначе } y > x\}$
6) $D = A \cup B - A \circ B \Delta C^{-1}$

Вариант 6

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + y \text{ — четно и } x + y > 8\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + 2y > 20\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x,y) \in \{1,2,4,8\} \times \{3,5,7,10\}\}$
6) $D = A \cap B^{-1} \Delta A \circ C \circ B$

Вариант 7

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + y \text{ кратно трем}\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (2 < x < 8 \text{ или } 2 < y < 8)\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x^2 + y^2 < 100\}$
6) $D = A^2 - B \cup A^{-1} \circ C$

Вариант 8

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (y > x + 5 \text{ или } x > y + 5)\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x \text{ и } y \text{ четные})\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } |x - y| > 5\}$
6) $D = A \circ B^{-1} \cap A - C$

Вариант 9

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } y < x + 1 \text{ и } x \neq 2 \cdot y\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x \text{ — четно и } x + y < 10\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x,y) \in \{2,3,8\} \times \{1,4,5,8\}\}$
- б) $D = \overline{A} \Delta A^{-1} \cap B \circ C$

Вариант 10

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x < y < (9-x) \text{ или } (9-x) < y < x)\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x \text{ — четно и } y \text{ — нечетно}\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x \text{ у кратно трем}\}$
- б) $D = A \circ B^2 - \overline{C} \cup C^1$

Вариант 11

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (y > x + 3 \text{ или } y < x - 3)\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } \text{если } x < 9, \text{ то } y \text{ кратно 3}\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + 3 \cdot y > 25\}$
- б) $D = A \cup A^{-1} \cap \overline{B} \circ C$

Вариант 12

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + y \text{ кратно } x\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x,y) \in \{2,4,6,8\} \times \{1,7,9\}\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x + y \text{ — четно и } x \cdot y < 20\}$
- б) $D = A \circ B \circ C - A^{-1} \Delta C$

Вариант 13

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } \text{если } x \text{ — четно, то } y < x, \text{ иначе } y > x\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (y < 7 \text{ или } x \text{ кратно 3})\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x^2 + y^2 > 100 \text{ или } x = y)\}$
- б) $D = A \cap B \Delta C^2 - B^{-1}$

Вариант 14

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (y - x > 5 \text{ или } x - y > 5)\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } 10 \cdot x + 2 \cdot y + 1 \text{ кратно 3}\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } ((x+3) < y < (11-x) \text{ или } (9-x) < y < (x+1))\}$
- б) $D = (A \cup B) \circ (\overline{A} - C)^{-1}$

Вариант 15

- a) $A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } x \cdot y < y^2\}$
 $B = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } ((x,y) \in \{2,4,6,8,10\} \times \{1,3,5,7,9\} \text{ или } x = y)\}$
 $C = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (1 < x < 7 \text{ или } 3 < y < 9)\}$
- б) $D = \overline{A \Delta B} \cup (B \circ C)^{-1}$

Практическое занятие 3.2

Транзитивное замыкание отношения

Цель занятия: изучить и выполнить сравнительный анализ алгоритмов вычисления транзитивного замыкания отношения.

Задания

- Изучить и программно реализовать алгоритмы 3.10 — 3.12 вычисления транзитивного замыкания.
 - Усовершенствовать алгоритм 3.11 объединения степеней (АОС). Доказать, что $A \circ (I \cup A)^{N^2} = A \cup A^2 \cup \dots \cup A^{N-1}$ и вычислить транзитивное замыкание C отношения A по формуле $A \circ (I \cup A)^{N^2}$.
 - Усовершенствовать алгоритм 3.12 Уоршалла (АУ) следующим образом. Рассмотрим выражение $C_{x,y} := C_{x,y} \text{ or } C_{x,z} \text{ and } C_{z,y}$ из алгоритма Уоршалла. Здесь $C_{x,y}$ может изменить значение с ложного на истинное только в том случае, если истинны $C_{x,z}$ и $C_{z,y}$. Поэтому, если $C_{x,z}$ ложно, то $C_{x,y}$ значение не изменит и нет смысла в поиске пары (z,y) .
 - Разработать и программно реализовать генератор отношений на множестве мощности N и содержащих заданное число пар.
 - Разработать и написать программу, которая генерирует 100 отношений на множестве мощности N с заданным числом пар, для каждого отношения вычисляет транзитивное замыкание пятью алгоритмами и подсчитывает количество k выполнений операции сравнения. Значение k при обработке различных отношений на множестве мощности N с заданным числом пар может быть разным, поэтому программа должна определять минимальное и максимальное значение k . Отношение, при обработке которого получено минимальное (максимальное) k , и его транзитивное замыкание, сохранить в файле. Выполнить программу при $N = 5, 10$ и 15 . Результат для каждого N представить в виде таблицы (табл. 3.2), а сохраненные отношения — в виде графа.

Таблица 3.2

Практическое занятие 3.3

Фактормножества

Цель занятия: научиться программно формировать фактормножества для заданного отношения эквивалентности и находить отношение эквивалентности для заданного разбиения.

Задания

1. Отношение на множестве $\{1,2,3,4,5,6,7,8,9,10\}$ (табл. 3.3) представить графом и характеристической функцией в матричной форме. Найти разбиение Φ , определяемое заданным отношением эквивалентности.

Таблица 3.3

Варианты заданий

Вариант	Отношение
1	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } ((x \text{ и } y \text{ четные}) \text{ или } x=y)\}$
2	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } ((x \text{ и } y \text{ четные}) \text{ или } (x \text{ и } y \text{ нечетные}))\}$
3	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x-y \text{ кратно } 5 \text{ или } x=y)\}$
4	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x \text{ и } y \text{ кратно } 4 \text{ или } x \text{ и } y \text{ кратно } 5 \text{ или } x=y)\}$
5	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x+y \text{ — четно})\}$
6	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x-y \text{ — четно})\}$
7	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x-y \text{ кратно } 3 \text{ или } x=y)\}$
8	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x \text{ и } y \text{ меньше } 3 \text{ или } x \text{ и } y \text{ большие } 3 \text{ или } x=y)\}$
9	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x-y \text{ кратно } 4 \text{ или } x=y)\}$
10	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x \text{ и } y \text{ кратно } 3 \text{ или } x \text{ и } y \text{ кратно } 5 \text{ или } x=y)\}$
11	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x \text{ и } y \text{ четно и меньше } 5 \text{ или } x \text{ и } y \text{ нечетно и больше } 5 \text{ или } x=y)\}$
12	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } (x \text{ и } y \text{ четно и меньше } 7 \text{ или } x=y)\}$
13	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } ((x,y) \in \{1,2,3\}^2 \text{ или } (x,y) \in \{6,7,8,9\}^2 \text{ или } x=y)\}$
14	$A = \{(x,y) / x \in N \text{ и } y \in N \text{ и } x < 11 \text{ и } y < 11 \text{ и } ((x,y) \in \{1,2,5\}^2 \text{ или } (x,y) \in \{3,7,8,9\}^2 \text{ или } x=y)\}$
15	$A = \{(x,y) / (x,y) \in \{1,3,10\}^2 \text{ или } (x,y) \in \{2,7,8,9\}^2 \text{ или } (x,y) \in \{4,5,6\}^2\}$

2. Программно реализовать алгоритм построения отношения эквивалентности R по разбиению S множества M .

Практическое занятие 3.4

Упорядоченные множества

Цель занятия: изучить упорядоченные множества, алгоритм топологической сортировки, научиться представлять множества диаграммами Хассе, находить минимальные (максимальные) и наименьшие (наибольшие) элементы упорядоченного множества.

Задания

Даны множества точек на плоскости M_1 (рис. 3.23), M_2 (рис. 3.24) и отношение порядка (табл. 3.5). Для определения отношения на множестве точек примем следующие обозначения: a_x — абсцисса точки a ; a_y — ордината a . На рис. 3.23 координаты правой верхней точки считать $(1,1)$. На рис. 3.24 координаты самой верхней точки считать $(0,2)$, а координаты самой правой точки считать $(2,0)$.

1. Написать программы, формирующие матрицы отношения порядка, в соответствии с вариантом задания (табл. 3.5), на множествах M_1 и M_2 .
2. Написать программы, формирующие матрицы отношения доминирования по матрицам отношения порядка.
3. Написать программу, реализующую алгоритм топологической сортировки по матрице отношения доминирования.
4. Изобразить диаграмму Хассе отношения доминирования на множествах M_1 и M_2 .
5. Найти минимальные и максимальные элементы множеств M_1 и M_2 .
6. Найти, если существуют, наименьший и наибольший элементы множеств M_1 и M_2 .

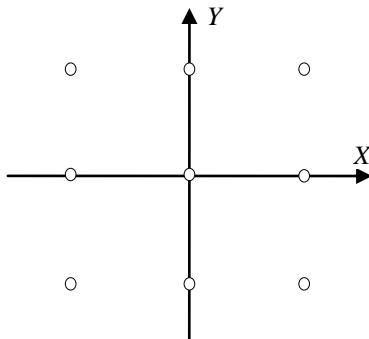


Рис. 3.23. Множество M_1

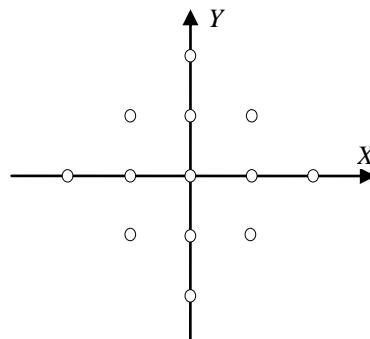


Рис. 3.24. Множество M_2

Таблица 3.5

Варианты заданий

Вариант	Отношение
1	$A = \{(a,b) / a_x \leq b_x \text{ и } a_y \leq b_y\}$
2	$A = \{(a,b) / a_x \leq b_y \text{ и } b_x = a_y\}$
3	$A = \{(a,b) / a_x \leq b_x\}$
4	$A = \{(a,b) / a_x - a_y \leq b_x - b_y\}$
5	$A = \{(a,b) / a_x \leq b_x \text{ и } a_y < b_y\}$
6	$A = \{(a,b) / a_x - b_x \leq a_y - b_y\}$
7	$A = \{(a,b) / a_y \leq b_y\}$
8	$A = \{(a,b) / a_x^2 + a_y^2 < b_x^2 + b_y^2\}$
9	$A = \{(a,b) / a_x < b_x \text{ и } a_y \leq b_y\}$
10	$A = \{(a,b) / a_x - b_x \leq b_y - a_y\}$
11	$A = \{(a,b) / a_x \leq b_y\}$
12	$A = \{(a,b) / a_x \leq a_y \text{ и } b_x \leq b_y\}$
13	$A = \{(a,b) / a_x < b_x \text{ и } a_y < b_y\}$
14	$A = \{(a,b) / a_x + a_y < b_x + b_y\}$
15	$A = \{(a,b) / a_x = b_y \text{ и } b_x \leq a_y\}$

Контрольные вопросы и задания

1. Что называется упорядоченной парой? Чем различаются упорядоченная пара и двухэлементное множество? Какое отличие в обозначениях упорядоченной пары и двухэлементного множества?
2. Что является прямым (декартовым) произведением множеств?
3. Определите мощность прямого (декартова) произведения двух данных конечных множеств.
4. Что называется бинарным соответствием?
5. Что является областью отправления и областью определения бинарного соответствия? Что является областью прибытия и областью значений бинарного соответствия?
6. Определите область отправления, область определения, область прибытия и область значений заданного бинарного соответствия.
7. Что называется образом и прообразом элемента при бинарном соответствии?
8. Определите образ каждого элемента из области определения заданного соответствия. Определите прообраз каждого элемента из области прибытия заданного соответствия.
9. Какое бинарное соответствие называется функциональным?

10. Какое соответствие называется полностью определенным? Что называется отображением?

11. Какая функция называется сюръективной, а какая — инъективной?

12. Что называется взаимно-однозначным отображением?

13. Что называется бинарным отношением?

14. Какое бинарное отношение называется пустым, тождественным, универсальным?

15. Приведите примеры задания отношений различными способами.

16. Определите, принадлежит ли заданная упорядоченная пара композиции заданных отношений. Назовите упорядоченную пару, принадлежащую композиции заданных отношений и упорядоченную пару, не принадлежащую композиции этих отношений.

17. Дайте определения основным и производным свойствам отношений. Определите свойства заданного отношения.

18. Что называется замыканием отношения относительно заданного свойства?

19. Получите транзитивное замыкание отношения, заданного графом.

20. Определите порядок функции временной сложности изученных алгоритмов вычисления транзитивного замыкания.

21. Что называется классом эквивалентности и фактормножеством?

22. Постройте отношение эквивалентности, определяемое заданным разбиением.

23. Что называется упорядоченным множеством?

24. Какие элементы упорядоченного множества называются сравнимыми, а какие — несравнимыми?

25. Какое множество называется линейно упорядоченным?

26. Постройте отношение строгого порядка, ассоциированного с заданным отношением порядка. Используйте матричное и графовое представление отношений.

27. Постройте отношение доминирования, ассоциированного с заданным отношением порядка. Используйте матричное и графовое представление отношений.

28. Что представляет собой транзитивное замыкание отношения доминирования, ассоциированного с заданным отношением порядка?

29. Определите основные свойства отношения доминирования.

30. Является ли отношение доминирования отношением порядка?

31. Что такая диаграмма Хассе?

32. Что такое топологическая сортировка? Примените алгоритм топологической сортировки к отношению, граф которого имеет циклы.

4. ГРАФЫ

4.1. Основные понятия

Первая работа по теории графов принадлежит Л. Эйлеру, и опубликована она была в 1736 г., однако термин “граф” впервые появился в книге венгерского математика Д. Кенига в 1936 г.

Дадим определение графа.

Пусть V — непустое множество, $V^{(2)}$ — множество всех его двухэлементных подмножеств. Пара (V,E) , где $E \subseteq V^{(2)}$ называется графом (неориентированным графом).

Граф называется *конечным*, если множество V конечно. В дальнейшем под термином “граф” будем понимать конечные графы.

Элементы множества V называются *вершинами* графа, а элементы множества E называются *ребрами* графа.

Число вершин графа $G = (V,E)$ называется его *порядком*.

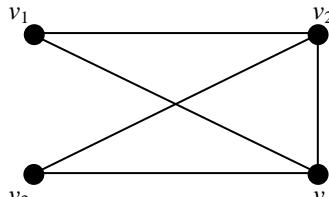
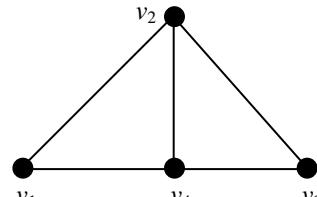
Исходя из определения графа, каждому ребру соответствует двухэлементное подмножество вершин. Если подмножество $\{v_1, v_2\}$ соответствует ребру e , то ребро e *инцидентно* вершинам v_1 и v_2 , а вершины v_1 и v_2 *смежны*. Вершины v_1 и v_2 называются *концевыми* вершинами ребра e . Два ребра, инцидентные одной вершине, называются *смежными*. Таким образом, смежность есть отношение между однородными элементами графа, а инцидентность — отношение между разнородными элементами графа.

Множество вершин, смежных с вершиной v , называется *множеством смежности* вершины v и обозначается $\Gamma(v)$. Если $A \subset V$, то $\Gamma(A)$ — множество всех вершин, смежных с вершинами из A .

Количество ребер, инцидентных вершине v , называется *степенью* вершины v и обозначается $d(v)$. $d(v) = |\Gamma(v)|$. Если степени всех вершин равны k , то граф называется *регулярным* степени k .

Граф можно представить графически в виде *диаграммы*: каждая вершина представляется точкой или окружностью, а каждое ребро представляется линией, соединяющей его концевые вершины.

Граф $G = (V,E)$, $V = \{v_1, v_2, v_3, v_4\}$, $E = \{\{v_1, v_2\}, \{v_1, v_4\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}\}$ представлен диаграммой на рис.4.1. Один и тот же граф может быть представлен различными диаграммами, так как расположение вершин не имеет значения. На рис.4.2 представлена другая диаграмма графа G .

Рис.4.1. Диаграмма графа G Рис.4.2. Диаграмма графа G

Часто рассматриваются следующие родственные графам объекты.

1. Если элементами множества E являются *упорядоченные пары*, то граф называется *ориентированным* (или *орграфом*). В этом случае элементы множества E называются *дугами*. Если дуга $(v_1, v_2) \in E$, то вершины v_1 и v_2 называются ее началом и концом соответственно. Дуга имеет направление от начала к концу. На диаграмме направление дуги указывается стрелочкой. На рис.4.3 приведен пример диаграммы орграфа. Число дуг, выходящих из вершины, называется *полустепенью исхода*, а число входящих дуг — *полустепенью захода*. Орграф называется *симметричным*, если полустепень исхода каждой вершины равна полу-степени захода.

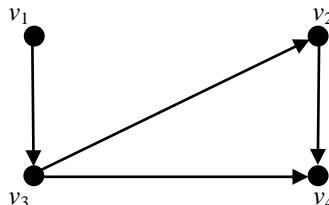


Рис.4.3. Диаграмма орграфа

2. Если элементом множества E может быть пара *одинаковых* (не различных) элементов множества V , то такой элемент множества E называется *петлей*, а граф называется *псевдографом*. На рис.4.4 приведен пример диаграммы псевдографа.

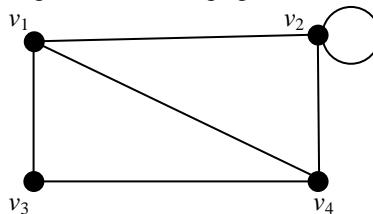


Рис.4.4. Диаграмма псевдографа

3. Если E является *мультимножеством*, содержащим несколько одинаковых элементов, то эти элементы называются *кратными ребрами*, а граф называется *мультиграфом*. На рис.4.5 приведен пример диаграммы мультиграфа.

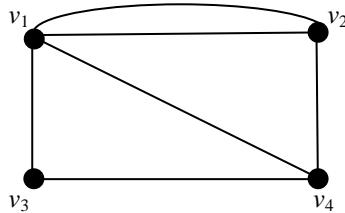


Рис.4.5. Диаграмма мультиграфа

4. Если элементом множества E являются не обязательно двухэлементные, а *любые* подмножества множества V , то такие элементы множества E называются *гиперребрами*, а граф называется *гиперграфом*. На диаграмме гиперграфа гиперребро представляет собой замкнутую линию, охватывающую инцидентные гиперребру вершины. Диаграмма гиперграфа $GG = (V, E)$, $V = \{v_1, v_2, v_3, v_4\}$, $E = \{\{v_1, v_2, v_4\}, \{v_1, v_2\}, \{v_3, v_4\}\}$ представлена на рис.4.6.

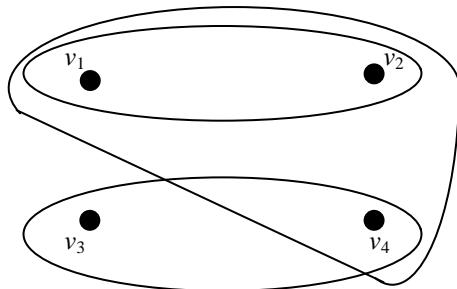


Рис.4.6. Диаграмма гиперграфа

5. Если задана функция $F: V \rightarrow M$ и/или $H: E \rightarrow M$, то множество M называется *множеством весов*, а граф называется *взвешенным*. На диаграмме весами отмечаются вершины и/или ребра. В качестве весов обычно используются числа или символы. На рис.4.7 приведен пример диаграммы взвешенного графа.

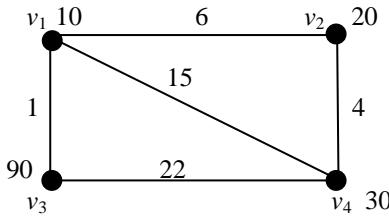


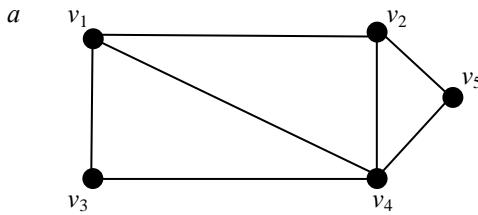
Рис.4.7. Диаграмма взвешенного графа

4.2. Матричные способы представления графов

Любой граф может быть представлен в матричной форме.

Матрицей смежности графа $G = (V, E)$ называется матрица A порядка $n \times n$, где $n = |V|$. Элемент матрицы смежности $a_{ij} = 1$, если $(v_i, v_j) \in E$, $v_i, v_j \in V$ и $a_{ij} = 0$, если $(v_i, v_j) \notin E$. В матрице смежности строки и столбцы соответствуют вершинам графа.

На рис. 4.8 представлены граф в виде диаграммы и его матрица смежности.

*а*

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Рис.4.8. Граф в виде диаграммы и матрица смежности:

- а* — диаграмма графа;
- б* — матрица смежности

Для неориентированных графов матрица смежности симметрична относительно главной диагонали.

Матрицей инцидентности графа $G = (V, E)$ называется матрица B порядка $n \times m$, где $n = |V|$, $m = |E|$. Элементы матрицы инцидентности b_{ij} определяются следующим образом: $b_{ij} = 1$, если i -я вершина является началом j -й дуги, $b_{ij} = -1$, если i -я вершина является концом j -й дуги, $b_{ij} = 0$, если i -я вершина и j -я дуга не инцидентны.

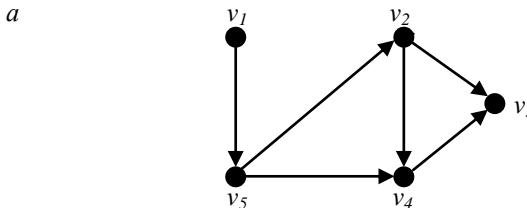
Для неориентированных графов в матрице инцидентности элементам достаточно присваивать только два символа (1 и 0).

В матрице инцидентности строки соответствуют вершинам графа, а столбцы — ребрам (дугам).

Для графа, представленного на рис.4.8, матрица инцидентности имеет вид:

$$B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

На рис.4.9 приведен пример диаграммы орграфа и его матрицы смежности и инцидентности.



$$\begin{array}{l} \delta \\ A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \end{array} \quad \begin{array}{l} \delta \\ B = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & -1 \\ -1 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Рис.4.9. Диаграмма орграфа, матрицы смежности и инцидентности:

α — диаграмма орграфа;

β — матрица смежности;

γ — матрица инцидентности

Матрицей дуг (списком дуг) графа $G = (V, E)$ называется матрица C порядка $2 \times m$, где $m = |E|$. Элементы матрицы C определяются следующим образом: $c_{1k} = v_i$ и $c_{2k} = v_j$, если $e_k = (v_i, v_j)$, т.е. если i -я вершина является началом k -й дуги, а j -я вершина является концом k -й дуги.

Для графа, представленного на рис.4.9, матрица дуг имеет вид:

$$C = \begin{pmatrix} 1 & 2 & 2 & 4 & 5 \\ 5 & 3 & 4 & 3 & 4 \end{pmatrix}$$

Для представления взвешенного графа с взвешенными дугами вместо единичных элементов матрицы смежности удобно записать веса соответствующих дуг, а веса несуществующих дуг полагать равными ∞ . Такая матрица называется матрицей весов. Если график представлен матрицей инцидентности или матрицей дуг, то задать веса дуг можно m -разрядным вектором (m — количество дуг в графике), в котором i -й элемент определяет вес i -й дуги. Если в графике взвешены вершины, то задать веса вершин можно n -разрядным вектором (n — количество вершин в графике), в котором i -й элемент определяет вес i -й вершины.

Для хранения в памяти ЭВМ матриц смежности, инцидентности, матриц дуг и матриц весов удобно использовать двумерные массивы.

4.3. Изоморфизм графов

Два графа $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ называются *изоморфными*, если существует взаимнооднозначное соответствие $p : V_1 \rightarrow V_2$ между вершинами графов, сохраняющее смежность, т.е. если вершины v_i и v_j смежны в графике G_1 , то вершины $p(v_i)$ и $p(v_j)$ смежны в графике G_2 , если же вершины v_i и v_j не смежны в графике G_1 , то вершины $p(v_i)$ и $p(v_j)$ не смежны в графике G_2 .

Изоморфные графы можно изобразить диаграммами (рис.4.10), которые различаются только обозначениями вершин.



Рис.4.10. Диаграммы изоморфных графов

Взаимнооднозначное соответствие между вершинами графов $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ можно задать вектором p , в котором номер элемента представляет собой вершину графа G_1 , а его значение — соответ-

ствующую вершину графа G_2 . Вектор p можно рассматривать как перестановку множества V_2 . Если одна из перестановок определяет соответствие, сохраняющее смежность, то графы G_1 и G_2 изоморфны. Пусть граф G_1 задан матрицей A , а граф G_2 — матрицей B . Соответствие p сохраняет смежность в том случае, если $A_{i,j} = B_{p_i,p_j}$ для всех элементов. Таким образом для определения изоморфизма графов можно использовать алгоритм порождения перестановок, в котором для каждой порождаемой перестановки проверяется условие: определяет ли она соответствие между вершинами графов, сохраняющее смежность.

4.4. Маршруты

Маршрутом в графе называется чередующаяся последовательность вершин и ребер, начинающаяся и кончаящаяся вершиной, в которой любые два соседних элемента инцидентны, причем однородные элементы (вершины, ребра) через один смежны или совпадают. Очевидно, маршрут может быть задан только последовательностью вершин (или ребер). Произвольная последовательность вершин является маршрутом только тогда, когда каждая пара соседних вершин представляет собой пару смежных вершин.

Длина маршрута определяется количеством ребер в нем. Маршрут длины l может быть задан последовательностью из $l + 1$ вершины.

Получить все маршруты W длины l в графе G , начинающиеся вершиной v можно, используя метод поиска с возвращением. Сначала на первое место маршрута W поставим вершину v . Затем, начиная со второго элемента, последовательно формируем элементы маршрута. Формирование i -го элемента маршрута опишем рекурсивным алгоритмом 4.1, блок-схема которого представлена на рис.4.11. В цикле перебираются вершины, смежные вершине, стоящей на $i - 1$ -м месте в маршруте W , т.е. элементы множества $\Gamma(W_{i-1})$, которые можно поставить на i -е место. Если заполнено последнее, $i + 1$ -е место, то маршрут получен и выводим его, иначе заполняем следующее $i + 1$ -е место по алгоритму 4.1.

Алгоритм 4.1 получения всех маршрутов W длины l в графе G , начинающиеся вершиной v .

Вход: i — заполняемое место в маршруте W .

Выход: последовательность всех маршрутов W длины l в графе G , начинающиеся вершиной v .

Глобальные параметры: W — формируемый маршрут;

l — длина маршрута.

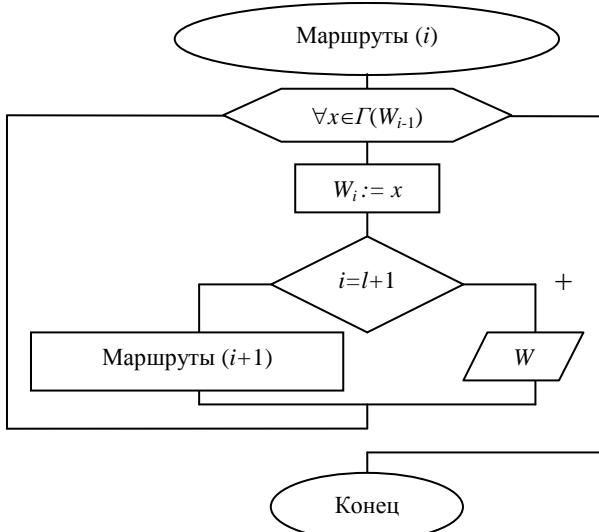


Рис.4.11. Рекурсивный алгоритм получения всех маршрутов W длины l в графе G , начинающихся вершиной v

Для определения количества маршрутов длины l между каждой парой вершин графа можно использовать алгоритм 4.1:

1. Обнулить все элементы матрицы R порядка $n \times n$, где n — количество вершин в графе.
2. $v := 1$.
3. Пока $v \leq n$ выполнять пп.4, 5.
4. Получить все маршруты W длины l в графе G , начинающиеся вершиной v , используя алгоритм 4.1. После получения очередного маршрута W увеличить на единицу элемент матрицы $R_{v,u}$, где $u = W_{l+1}$.
5. $v := v + 1$.
6. Элемент $R_{i,j}$ содержит количество маршрутов длины l между вершинами i и j .

Если граф G задан матрицей смежности A , то $R = A^l$, т.е. для определения количества маршрутов длины l между каждой парой вершин графа G достаточно матрицу смежности A возвести в l -ю степень.

Маршрут, в котором все ребра различны, называется *цепью*. Цепь, так же как и маршрут, может быть задана последовательностью вершин. Пусть задана последовательность вершин и требуется определить, является ли она цепью в заданном графе. Для этого будем использовать множество E' (в исходном состоянии оно пустое) ребер. Последователь-

но перебирая пары соседних вершин в заданной последовательности, проверяем условие, являются ли эти вершины смежными и принадлежит ли ребро, образованное этими вершинами множеству E' , и, если вершины смежны и ребро, образованное ими, не принадлежит множеству E' , то ребро включаем в множество E' и переходим к следующей паре вершин. Если же на некотором шаге обнаруживаем, что вершины не смежны или ребро, образованное ими, принадлежит множеству E' , то заданная последовательность вершин не является цепью в заданном графе.

Получить все цепи W длины l в графе G , начинающиеся вершиной v можно, используя метод поиска с возвращением. Сначала на первое место цепи W поставим вершину v . Множество ребер E' , включенных в цепь, пусто. Затем, начиная со второго элемента, последовательно формируем элементы цепи. Формирование i -го элемента цепи опишем рекурсивным алгоритмом 4.2, блок-схема которого представлена на рис.4.12. В цикле перебираются вершины, смежные вершине, стоящей на $i - 1$ -м месте в цепи W и не образующие с ней ребро, принадлежащее множеству E' , т.е. элементы множества $\Gamma(W_{i-1}) - \{y \mid \{W_{i-1}, y\} \in E'\}$, которые можно поставить на i -е место. Если заполнено последнее, $i + 1$ -е место, то цепь получена и выводим ее, иначе заполняем следующее $i + 1$ -е место по алгоритму 4.2 с учетом последнего включенного в цепь ребра.

Алгоритм 4.2 получения всех цепей W длины l в графе G , начинающихся вершиной v .

Вход: i — заполняемое место в цепи W ;

E' — множество ребер, включенных в цепь.

Выход: последовательность всех цепей W в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемая цепь;

l — длина цепи.

Маршрут, в котором все вершины (а значит, и ребра) различны, называется *простой цепью*. Пусть задана последовательность вершин и требуется определить, является ли она простой цепью в заданном графе. Для этого будем использовать множество V' вершин (в исходном состоянии оно содержит только первую вершину заданной последовательности). Последовательно перебирая пары соседних вершин в заданной последовательности, проверяем условие, являются ли эти вершины смежными и принадлежит ли вторая вершина пары множеству V' , и, если вершины смежны и вторая вершина пары не принадлежит множе-

ству V' , то вторую вершину пары включаем в множество V' и переходим к следующей паре вершин. Если же на некотором шаге обнаруживаем, что вершины не смежны или вторая вершина пары принадлежит множеству V' , то заданная последовательность вершин не является простой цепью в заданном графе.

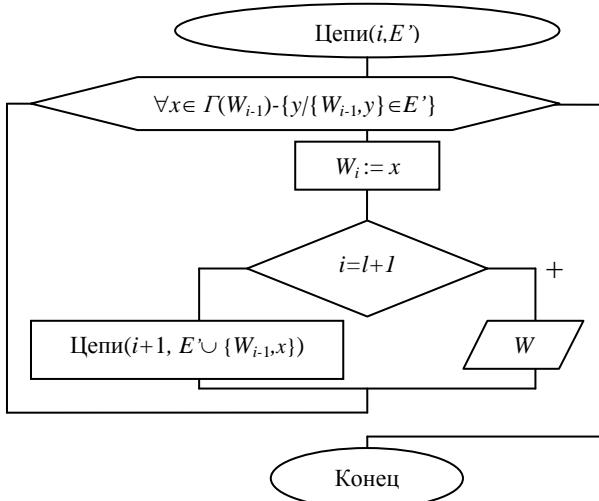


Рис.4.12. Рекурсивный алгоритм получения всех цепей W длины l в графе G , начинающихся вершиной v

Получить все простые цепи W длины l в графе G , начинающиеся вершиной v можно, используя метод поиска с возвращением. Сначала на первое место цепи W поставим вершину v и запомним ее в множестве V' . Затем, начиная со второго элемента, последовательно формируем элементы цепи. Формирование i -го элемента цепи опишем рекурсивным алгоритмом 4.3, блок-схема которого представлена на рис.4.13. В цикле перебираются вершины, смежные вершине, стоящей на $i - 1$ -м месте в цепи W за исключением тех, которые уже включены в цепь, т.е. элементы множества $\Gamma(W_{i-1}) - V'$, которые можно поставить на i -е место. Если заполнено последнее, $i + 1$ -е место, то простая цепь длины l получена и выводим ее, иначе заполняем следующее $i + 1$ -е место по алгоритму 4.3 с учетом последней включенной в цепь вершины.

Алгоритм 4.3 получения всех простых цепей W длины l в графе G , начинающихся вершиной v .

Вход: i — заполняемое место в цепи W ;

V' — множество вершин, включенных в цепь.

Выход: последовательность всех простых цепей W длины l в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемая простая цепь;
 l — длина цепи.

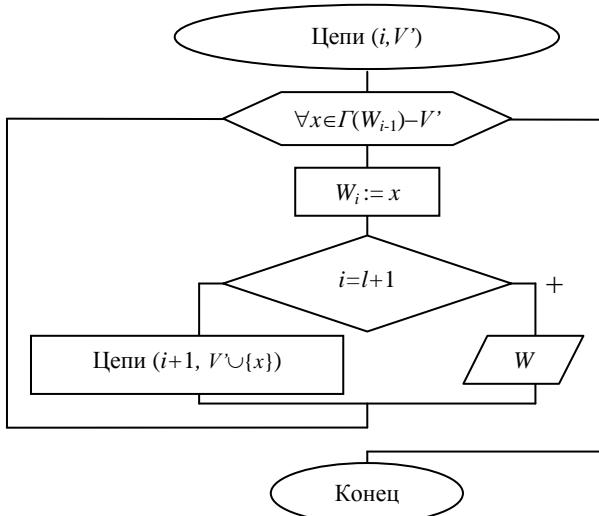


Рис.4.13. Рекурсивный алгоритм получения всех простых цепей W длины l в графе G , начинающихся вершиной v

Простая цепь, начинающаяся вершиной v , называется *максимальной простой цепью*, если она не является частью ни какой другой простой цепи, начинающейся вершиной v . Пусть задана последовательность вершин и требуется определить, является ли она максимальной простой цепью в заданном графе. Последовательность вершин будет максимальной простой цепью, если она является простой цепью и все вершины, смежные с последней вершиной последовательности, содержатся в этой последовательности. Это означает, что в максимальную простую цепь нельзя добавить ни одной вершины так, чтобы последовательность оставалась простой цепью.

Получить все максимальные простые цепи W в графе G , начинающиеся вершиной v , можно, используя метод поиска с возвращением. Сначала на первое место цепи W поставим вершину v и запомним ее в множестве V' . Затем, начиная со второго элемента, последовательно формируем элементы цепи. Формирование i -го элемента цепи опишем рекурсивным алгоритмом 4.4, блок-схема которого представлена на

рис.4.14. В цикле перебираются вершины, смежные вершине, стоящей на $i - 1$ -м месте в цепи W за исключением тех, которые уже включены в цепь, т.е. элементы множества $\Gamma(W_{i-1}) - V'$, которые можно поставить на i -е место. Если все вершины, смежные вершине, установленной на i -е место, уже находятся в цепи, то максимальная простая цепь получена и выводим ее, иначе заполняем следующее $i + 1$ -е место по алгоритму 4.4 с учетом последней включенной в цепь вершины.

Алгоритм 4.4 получения всех максимальных простых цепей W в графе G , начинающихся вершиной v .

Вход: i — заполняемое место в цепи W ;

V' — множество вершин, включенных в цепь.

Выход: последовательность всех максимальных простых цепей W в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемая максимальная простая цепь, начинающаяся вершиной v .

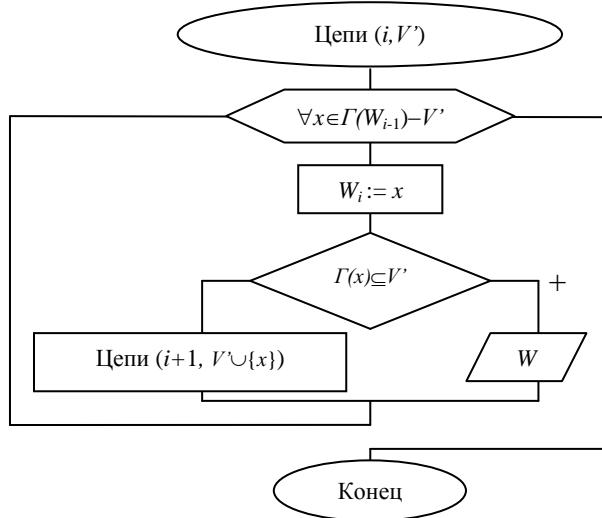


Рис.4.14. Рекурсивный алгоритм получения всех максимальных простых цепей W в графе G , начинающихся вершиной v

4.5. Метрические характеристики графа

Пусть v_1 и v_2 — две вершины графа $G = (V, E)$. Длина кратчайшего маршрута, соединяющего вершины v_1 и v_2 , называется *расстоянием между вершинами v_1 и v_2* , обозначается через $d(v_1, v_2)$. Для любой вершины v величина $e(v) = \max_{y \in V} d(v, y)$ называется *эксцентриситетом вершины v* .

Максимальный из всех эксцентриситетов вершин называется *диаметром графа* и обозначается $d(G)$.

Минимальный из всех эксцентриситетов вершин называется *радиусом графа* и обозначается $r(G)$.

Вершина v называется *периферийной*, если ее эксцентриситет равен диаметру графа, т.е. $e(v) = d(G)$.

Простая цепь, в которой расстояние между концевыми вершинами равно диаметру графа, называется *диаметральной цепью*.

Вершина v называется *центральной*, если ее эксцентриситет равен радиусу графа, т.е. $e(v) = r(G)$. Множество всех центральных вершин графа называется его *центром*.

Рассмотрим граф, диаграмма которого представлена на рис.4.15.

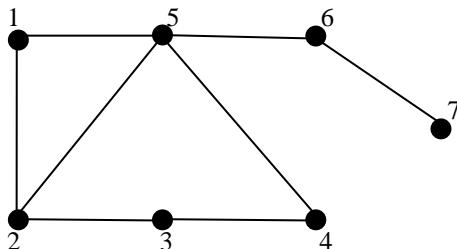


Рис.4.15. Диаграмма графа

Эксцентриситеты вершин этого графа представлены в табл. 4.1.

Таблица 4.1

Эксцентриситеты вершин графа

Вершина	1	2	3	4	5	6	7
Эксцентриситет	3	3	4	3	2	3	4

Диаметр этого графа равен 4, радиус равен 2. Периферийные вершины: 3 и 7, диаметральные цепи: (3,2,5,6,7) и (3,4,5,6,7), центральная вершина 5.

4.6. Циклы

Циклом в графе называется цепь, в которой начальная и конечная вершины совпадают (замкнутая цепь). Простая замкнутая цепь называется *простым циклом*. В простом цикле ни какая вершина, за исключением начальной, не встречается более одного раза. Простой цикл не содержит внутри себя других циклов. Граф, содержащий в себе хотя бы один цикл, называется *циклическим*, в противном случае — *ациклическим*. Циклический граф может иметь несколько простых циклов с заданной начальной вершиной. Алгоритм 4.5 получения всех простых циклов W в графе G , начинающихся вершиной v , отличается от алгоритма 4.3 только условием получения решения. Простой цикл получен, если на очередное i -е место поставлена вершина v (в исходном состоянии вершина v в множество V' не включается). Блок-схема алгоритма 4.5 представлена на рис.4.16.

Алгоритм 4.5 получения всех простых циклов W в графе G , начинающихся вершиной v .

Вход: i — заполняемое место в цикле W ;

V' — множество вершин, включенных в цикл.

Выход: последовательность всех простых циклов W в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемый цикл;

v — исходная вершина.

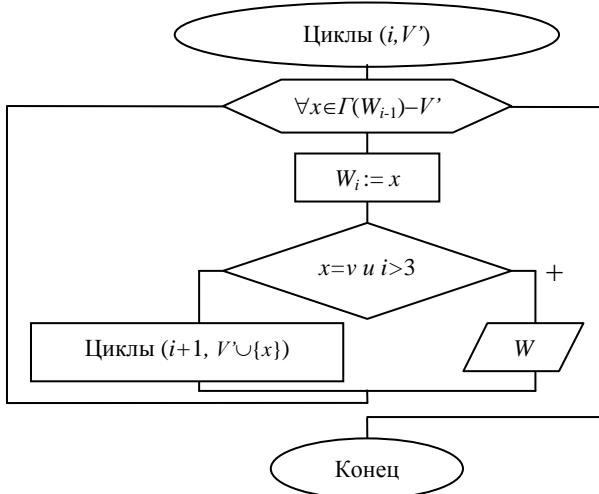


Рис.4.16. Рекурсивный алгоритм получения всех простых циклов W в графе G , начинающихся вершиной v

Если граф имеет простой цикл, содержащий все вершины графа (по одному разу), то такой цикл называется *гамильтоновым* циклом, а граф — *гамильтоновым* графом. Вопрос о принадлежности графов к классу гамильтоновых решается, как правило, очень трудно. Известны лишь *достаточные* условия гамильтоновости графов, например, теорема Оре: если для любой пары x и y несмежных вершин графа G порядка $n \geq 3$ выполняется условие $d(x) + d(y) \geq n$, то G — гамильтонов граф; или теорема Дирака: если для любой вершины x графа G порядка $n \geq 3$ выполняется условие $d(x) \geq n/2$, то G — гамильтонов граф. Убедиться в том, что граф гамильтонов можно, например, отыскав в нем один из гамильтоновых циклов, используя метод поиска с возвращением. На рис.4.17 представлена блок-схема рекурсивного алгоритма 4.6 нахождения всех гамильтоновых циклов графа. Этот алгоритм отличается от алгоритма 4.5 нахождения всех простых циклов только условием получения решения, так как гамильтонов цикл — это простой цикл, содержащий все вершины графа.

Алгоритм 4.6 получения всех гамильтоновых циклов W в графе G .

Вход: i — заполняемое место в цикле W ;

V' — множество вершин, включенных в цикл.

Выход: последовательность всех гамильтоновых циклов W в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемый цикл.

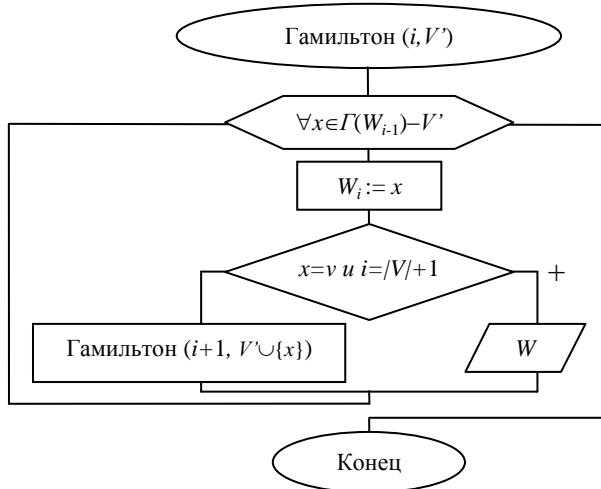


Рис.4.17. Рекурсивный алгоритм получения всех гамильтоновых циклов W в графе G

Если граф имеет цикл (не обязательно простой), содержащий все ребра графа по одному разу, то такой цикл называется *эйлеровым циклом*, а граф — *эйлеровым графом*. Эйлеров цикл содержит не только все ребра, но и все вершины графа (возможно, по несколько раз). Принадлежность графа классу эйлеровых легко устанавливается теоремой Эйлера: *связный граф является эйлеровым тогда и только тогда, когда степени всех его вершин четны.*

Эйлеров цикл представляет собой замкнутую цепь, содержащую все ребра графа, поэтому можно получить все эйлеровы циклы используя алгоритм 4.2 получения всех цепей заданной длины, изменив в нем условие получения решения. Алгоритм 4.7 получения всех эйлеровых циклов представлен блок-схемой на рис.4.18.

Алгоритм 4.7 получения всех эйлеровых циклов W в графе G .

Вход: i — заполняемое место в цикле W ;

E' — множество ребер, включенных в цикл.

Выход: последовательность всех эйлеровых циклов W в графе G , начинающихся вершиной v .

Глобальные параметры: W — формируемый цикл;

v — исходная вершина.

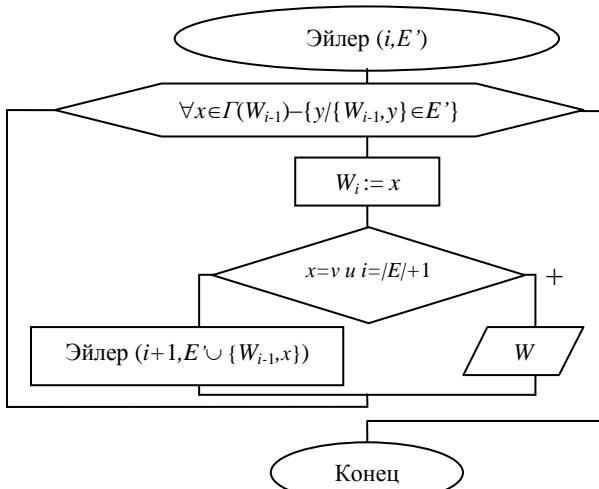


Рис.4.18. Рекурсивный алгоритм получения всех эйлеровых циклов W в графе G , начинающихся вершиной v

В эйлеровом графе найти один из эйлеровых циклов можно следующим образом:

1. Стек пуст. Множество отмеченных ребер пусто. Произвольную вершину v положить в стек.

2. Пока стек не пуст, выполнять п.3.

3. $x :=$ верхний элемент стека.

Если все ребра, инцидентные вершине x , отмечены, то взять вершину x из стека и поставить ее на очередное место цикла, иначе выбрать неотмеченное ребро $\{x,y\}$, отметить его и вершину y положить в стек.

Процесс нахождения эйлерова цикла в эйлеровом графе, диаграмма которого изображена на рис.4.19, представлен в табл. 4.2.

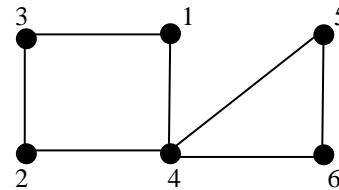


Рис.4.19. Диаграмма эйлерова графа

Таблица 4.2

Процесс нахождения эйлерова цикла

Шаг	Стек	Отмеченные ребра	Эйлеров цикл
1	1		
2	1,4	{1,4}	
3	1,4,2	{1,4},{4,2}	
4	1,4,2,3	{1,4},{4,2},{2,3}	
5	1,4,2,3,1	{1,4},{4,2},{2,3},{3,1}	
6	1,4,2,3	{1,4},{4,2},{2,3},{3,1}	1
7	1,4,2	{1,4},{4,2},{2,3},{3,1}	1,3
8	1,4	{1,4},{4,2},{2,3},{3,1}	1,3,2
9	1,4,5	{1,4},{4,2},{2,3},{3,1},{4,5}	1,3,2
10	1,4,5,6	{1,4},{4,2},{2,3},{3,1},{4,5},{5,6}	1,3,2
11	1,4,5,6,4	{1,4},{4,2},{2,3},{3,1},{4,5},{5,6},{6,4}	1,3,2
12	1,4,5,6	{1,4},{4,2},{2,3},{3,1},{4,5},{5,6},{6,4}	1,3,2,4
13	1,4,5	{1,4},{4,2},{2,3},{3,1},{4,5},{5,6},{6,4}	1,3,2,4,6
14	1,4	{1,4},{4,2},{2,3},{3,1},{4,5},{5,6},{6,4}	1,3,2,4,6,5
15	1	{1,4},{4,2},{2,3},{3,1},{4,5},{5,6},{6,4}	1,3,2,4,6,5,4
16		{1,4},{4,2},{2,3},{3,1},{4,5},{5,6},{6,4}	1,3,2,4,6,5,4,1

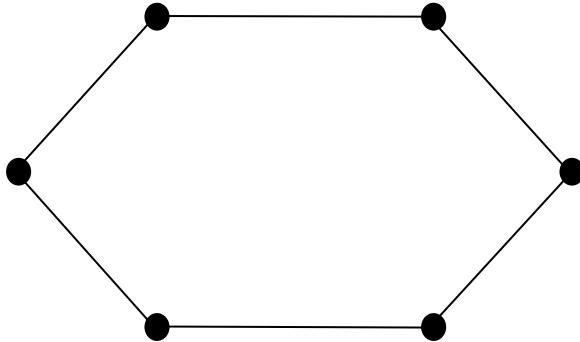
4.7. Связность

Задача 4.1.

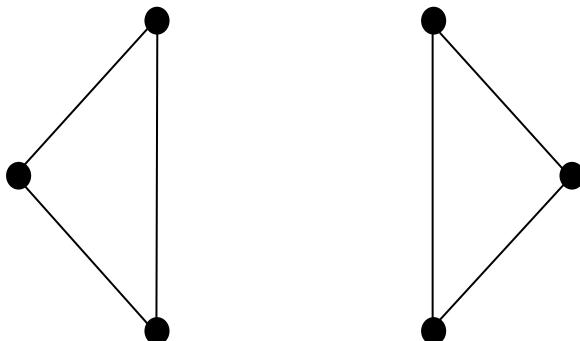
Из шести человек каждый знаком с двумя другими. Может ли каждый из них через своих знакомых познакомиться со всеми остальными?

Решение.

Отношение знакомства представим графом, в котором каждая вершина соответствует человеку и две вершины смежны, если соответствующие им люди знакомы. Если диаграмма полученного графа имеет вид, например:



то ответ на вопрос задачи будет положительным, а если диаграмма полученного графа имеет вид:



то ответ будет отрицательным.

В первом случае граф является связным, а во втором — несвязным.

Граф называется *связным*, если в нем для каждой пары вершин найдется соединяющая их цепь, иначе — *несвязным*. Две вершины графа

называются *связанными*, если существует соединяющая их цепь. В связном графе все пары вершин связаны. Отношение связанности вершин является отношением эквивалентности: оно рефлексивно (каждая вершина соединена с собой цепью нулевой длины), симметрично (всякая цепь, записанная в обратном порядке, также является цепью) и транзитивно (если имеется цепь, соединяющая вершины x и y и цепь, соединяющая вершины y и z , то соединение этих цепей в вершине y даст маршрут из вершины x в вершину z , из которого можно выделить цепь, соединяющую вершины x и z), поэтому оно разбивает множество вершин на классы эквивалентных вершин. Подграф, построенный на множестве вершин одного класса эквивалентности, представляет собой *связную компоненту* графа. Связный граф имеет одну компоненту связности, несвязный — более одной.

Отношение связанности вершин графа $G=(V,E)$ можно задать матрицей связанности C порядка $n \times n$, где $n=|V|$. Элемент матрицы связанности $c_{ij}=1$, если вершины i и j связаны и $c_{ij}=0$, если вершины i и j не связаны. Рассмотрим два способа получения матрицы связанности.

Первый способ.

Получить матрицу связанности C можно, используя алгоритм 4.4 нахождения всех максимальных простых цепей, основанный на методе поиска с возвращением. Для этого необходимо инициализировать матрицу C нулями, а затем последовательно формировать строки матрицы. При формировании i -й строки находятся все максимальные простые цепи с начальной вершиной i . Если цепь найдена и содержит вершину j , то элементу c_{ij} присвоить значение 1.

Второй способ.

Вычислить матрицу связанности C можно по формуле $C = I + M^+$, где I — бинарная матрица, содержащая единицы только на главной диагонали;

M — матрица смежности графа;

M^+ — транзитивное замыкание отношения, представленного матрицей смежности графа, которое можно вычислить, используя алгоритмы 3.10 — 3.11 или их модификации.

По матрице связанности C можно построить разбиение множества вершин графа на классы эквивалентности, используя алгоритм 3.14, тем самым найдем подмножества вершин, образующих связные компоненты графа.

Найти разбиение S множества вершин графа $G = (V,E)$ на подмножества вершин, образующих связные компоненты, можно и без формиро-

вания матрицы связанности, используя алгоритм 4.4 нахождения всех максимальных простых цепей следующим образом:

1. $A := V; S := \emptyset$.
2. Пока $A \neq \emptyset$ выполнять пп.3 — 5.
3. Выбрать вершину $v \in A; M := \{v\}$.
4. Найти все максимальные простые цепи с начальной вершиной v и все вершины, входящие в максимальные простые цепи, включить в множество M . M — множество вершин, образующих связную компоненту графа.
5. $S := S \cup \{M\}; A := A - M$.

Мостом называется ребро, удаление которого увеличивает число связных компонент графа. Существует несколько признаков мостов:

- 1) ребро $\{v_i, v_j\}$ является мостом в том и только в том случае, если (v_i, v_j) — единственный маршрут, соединяющий вершины v_i и v_j ;
- 2) ребро $\{v_i, v_j\}$ является мостом в том и только в том случае, если найдутся две вершины v_k и v_l такие, что каждый маршрут, соединяющий их, содержит v_i и v_j ;
- 3) ребро $\{v_i, v_j\}$ является мостом в том и только в том случае, если оно не принадлежит ни одному циклу.

Множество ребер, исключение которых увеличивает число связных компонент графа, называется *разрезом*.

Реберной связностью связного графа называется наименьшее число ребер, удаление которых приводит к несвязному графу.

Вершина графа называется *точкой сочленения*, если ее удаление увеличивает число связных компонент. Концевые вершины мостов являются точками сочленения, но точка сочленения может не являться концевой вершиной моста. Граф без точек сочленения называется *двусвязным*.

Вершинной связностью связного графа называется наименьшее число вершин, удаление которых приводит к несвязному графу.

4.8. Деревья

Связный ациклический граф называется *деревом*. Несвязный ациклический граф называется *лесом*. Лес представляет собой множество деревьев. Дерево и лес обладают следующими свойствами:

- 1) количество ребер любого дерева на единицу меньше количества вершин;
- 2) количество ребер любого леса меньше количества вершин на количество деревьев в нем;
- 3) любые две вершины, принадлежащие различным деревьям — не смежны;

- 4) любую пару вершин дерева соединяет единственная простая цепь;
- 5) если в дереве любую пару несмежных вершин соединить ребром, то полученный граф будет содержать ровно один цикл;
- 6) если две вершины леса, принадлежащих различным деревьям, соединить ребром, то количество деревьев уменьшится на единицу.

Вершина дерева, степень которой равна единице, называется *висячей вершиной* или *листом*. Центральная вершина дерева (эксцентрикитет которой равен радиусу) называется *корнем*.

Если из дерева D , имеющего радиус r , удалить все его листья, то получим дерево D_1 , имеющее радиус $r - 1$ и те же самые корни. Продолжая процесс, получим дерево с одной либо с двумя вершинами, которое имеет те же корни, что и исходное дерево, следовательно, любое дерево имеет один, либо два корня.

На рис.4.20 представлена диаграмма дерева, в котором вершины 1, 3, 5, 6, 8, 9 и 11 являются листьями, а вершины 4 и 7 – корнями.

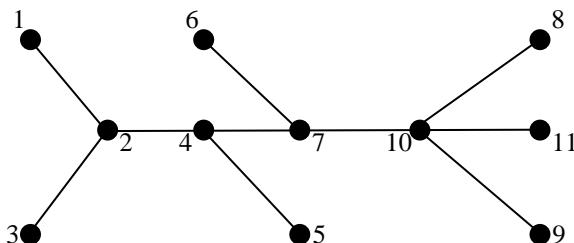


Рис.4.20. Диаграмма дерева

Каждому дереву с n вершинами можно поставить во взаимно однозначное соответствие последовательность длины $n - 2$, элементы которой берутся из множества вершин. Получить такую последовательность для заданного дерева, множество вершин которого $V = \{1, 2, \dots, n\}$, можно следующим образом:

1. Выполнить п.2 $n - 2$ раза.
2. Найти лист с минимальным номером, исключить его, вершину, смежную с ним, занести в последовательность.

Рассмотрим дерево, диаграмма которого изображена на рис.4.20. Выбираем у него лист с наименьшим номером. Это вершина 1. Удалим ее вместе с инцидентным ребром и вершину 2, смежную с ней, запишем на первое место последовательности. После этого листом с наименьшим номером будет вершина 3. Удалим ее вместе с инцидентным ребром и

вершину 2, смежную с ней, запишем на второе место последовательности. Далее листом с наименьшим номером будет вершина 2. Удалим ее вместе с инцидентным ребром и вершину 4, смежную с ней, запишем на третье место последовательности. Продолжая выполнять такие действия, получим для данного дерева определенную единственным образом последовательность $(2, 2, 4, 4, 7, 7, 10, 10, 10)$.

Каждая последовательность P длины $n - 2$, элементы которой берутся из множества вершин, однозначно определяет дерево с n вершинами. Построить такое дерево для заданной последовательности P можно следующим образом:

1. Составить множество N натуральных чисел $(1, 2, \dots, n)$.
2. $i := 1$.
3. Выполнять п.4, пока $|N| > 2$.
4. Найти в множестве N минимальное число m , которого нет в последовательности P . $\{m, P_i\}$ — ребро дерева. Число m удалить из множества N , а P_i — из последовательности P . $i := i + 1$.
5. Последнее ребро образуется оставшимися элементами множества N .

Построим дерево, которое определяется последовательностью $P = (4, 2, 4, 7, 7, 7, 10, 11, 11)$. В этой последовательности девять элементов, поэтому в дереве будет 11 вершин. Составим множество $N = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Найдем в этом множестве минимальное число, которого нет в последовательности P . Это число 1. Оно с первым элементом последовательности образует ребро $\{1, 4\}$. Удалим из множества N число 1, из последовательности P — число 4 и получим $P = (, 2, 4, 7, 7, 7, 10, 11, 11)$ и $N = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Теперь снова найдем в множестве N минимальное число, которого нет в последовательности P . Это число 3. Оно со вторым элементом последовательности образует ребро $\{3, 2\}$. Удалим из множества N число 3, из последовательности P — число 2 и получим $P = (, , 4, 7, 7, 7, 10, 11, 11)$ и $N = \{2, 4, 5, 6, 7, 8, 9, 10, 11\}$. Продолжая выполнять такие действия будем получать новые ребра дерева. Когда все элементы из последовательности P будут исключены, в множестве N останутся два элемента $N = \{10, 11\}$, которые образуют последнее ребро дерева. Диаграмма полученного дерева представлена на рис.4.21.

Так как каждому дереву с n вершинами можно поставить во взаимно однозначное соответствие последовательность длины $n - 2$, элементы которой берутся из множества вершин, то количество различных деревьев, которые можно построить на n вершинах равно количеству таких последовательностей (размещений с повторениями из n элементов по $n - 2$ местам), т.е. n^{n-2} .

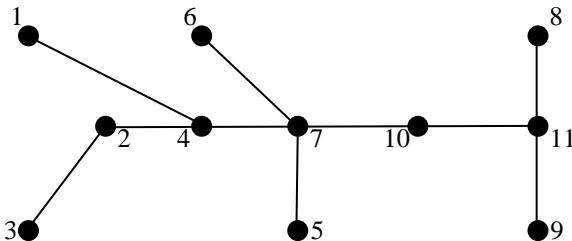


Рис.4.21. Диаграмма дерева, которое определяется последовательностью
 $P = (4, 2, 4, 7, 7, 10, 11, 11)$

4.9. Покрывающие деревья графа

Покрывающим (остовным) деревом графа G называется дерево, являющееся подграфом графа G и содержащее все его вершины. Если граф G несвязный, то множество, состоящее из остовных деревьев каждой связной компоненты, называется *покрывающим (остовным) лесом*.

Для того, чтобы получить покрывающее дерево связного графа, из него нужно удалить ребра, образующие циклы. Количество ребер связного графа, которые нужно удалить из него для получения покрывающего дерева, не зависит от порядка удаления ребер и равно $m - n + 1$, где m — количество ребер графа, n — количество вершин, так как дерево с n вершинами содержит $n - 1$ ребро и это на $m - n + 1$ меньше, чем m . Удаление не каждого подмножества из $m - n + 1$ ребер превращает связный граф в покрывающее дерево.

Для того, чтобы получить покрывающий лес несвязного графа, содержащего k связных компонент, из него нужно удалить $m - n + k$ ребра, так как из каждой i -й связной компоненты нужно удалить $m_i - n_i + 1$ ребра для получения покрывающего дерева этой компоненты. Удаление не каждого подмножества из $m - n + k$ ребер превращает несвязный граф, содержащий k связных компонент, в покрывающий лес.

Рассмотрим алгоритм построения покрывающего леса, предложенный Дж. Краскалом в 1957 г.

Суть алгоритма состоит в просмотре в произвольном порядке ребер исходного графа. Для каждого ребра принимается решение о том, будет ли оно включено в покрывающий лес или нет. Очередное ребро включается в лес, если оно с уже включенными ранее в лес ребрами не образует цикла.

Для определения, образует ли ребро с ранее включенными в лес ребрами цикл, будем использовать понятие “букет”. *Букет* — это множество вершин, принадлежащих одной связной компоненте. Некоторое

ребро образует цикл с ребрами, уже включенными в лес, если обе его концевые вершины принадлежат одному и тому же букету.

Если граф связный, то после просмотра всех ребер будет сформировано покрывающее дерево и один букет, содержащий все вершины графа. Если граф несвязный, то после просмотра всех ребер будет сформирован покрывающий лес из k деревьев и k букетов, где k — количество связных компонент в графе.

Алгоритм Краскала.

1. Начальная установка. Лес не содержит ни одного ребра и ни один из букетов не сформирован.

2. Выбрать любое ребро, включить его в лес и сформировать букет, включив в него концевые вершины выбранного ребра.

3. Пока в графе есть необработанные ребра, выполнять п.4.

4. Выбрать необработанное ребро. После этого выбора возможны четыре различных случая:

А. Обе концевые вершины выбранного ребра принадлежат одному и тому же букету. В этом случае ребро не включается в лес.

Б. Одна из концевых вершин выбранного ребра принадлежит некоторому букету, а другая концевая вершина не принадлежит ни одному из уже сформированных букетов. В этом случае выбранное ребро включается в лес и его концевая вершина, не принадлежащая ранее ни одному букету, включается в букет, которому принадлежит другая концевая вершина рассматриваемого ребра.

В. Ни одна из концевых вершин не принадлежит ни одному из сформированных букетов. В этом случае выбранное ребро включается в лес и формируется новый букет из его концевых вершин.

Г. Концевые вершины выбранного ребра принадлежат различным букетам. В этом случае выбранное ребро включается в лес, а оба букета, которым принадлежат его концевые вершины, объединяются в один новый букет и количество букетов при этом уменьшается.

Не нарушая понятия букета, можно упростить алгоритм Краскала, если при начальной установке сформировать n букетов по одной вершине в каждом (n — количество вершин в графе). В этом случае в п.4 после выбора очередного ребра возможны только два случая — А и Г.

При программной реализации алгоритма Краскала для хранения букетов целесообразно использовать массив B натуральных чисел, в котором количество элементов равно количеству вершин в графе. Индекс i элемента массива B_i соответствует i -й вершине графа, а значение B_i — номеру букета, которому принадлежит i -я вершина. Если i и j — концевые вершины ребра, то для распознавания случая А или Г пункта 4 ал-

горитма, достаточно сравнить на равенство значения элементов B_i и B_j . Для объединения букетов B_i и B_j необходимо просмотреть все элементы массива B и значения, равные B_j , заменить на B_i . При этом количество букетов уменьшится.

Рассмотрим работу "упрощенного" алгоритма Краскала на примере графа (рис.4.22). Процесс работы алгоритма представим в табл. 4.3.

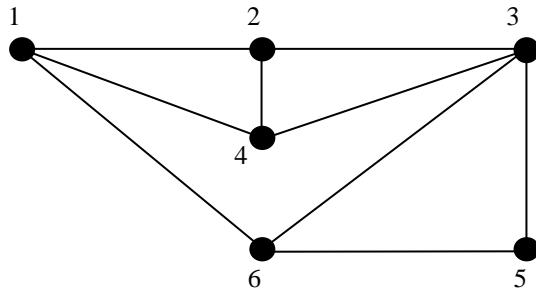


Рис.4.22. Диаграмма графа

Таблица 4.3

Работа алгоритма Краскала

Шаг	Ребро	Решение	Массив В						Примечание
			1	2	3	4	5	6	
1			1	2	3	4	5	6	Начальная установка
2	{1,2}	Включать в лес	1	1	3	4	5	6	На шаге 1 $B_1 \neq B_2$
3	{1,4}	Включать в лес	1	1	3	1	5	6	На шаге 2 $B_1 \neq B_4$
4	{2,3}	Включать в лес	1	1	1	1	5	6	На шаге 3 $B_2 \neq B_3$
5	{2,4}	Не включать	1	1	1	1	5	6	На шаге 4 $B_2 = B_4$
6	{3,4}	Не включать	1	1	1	1	5	6	На шаге 5 $B_3 = B_4$
7	{3,5}	Включать в лес	1	1	1	1	1	6	На шаге 6 $B_3 \neq B_6$
8	{3,6}	Включать в лес	1	1	1	1	1	1	Дерево построено
9	{5,6}	Не включать	1	1	1	1	1	1	На шаге 8 $B_5 = B_6$
10	{6,1}	Не включать	1	1	1	1	1	1	На шаге 9 $B_6 = B_1$

Связный граф может иметь несколько покрывающих деревьев. Количество всех различных покрывающих деревьев равно определителю матрицы $|B \cdot B'|$, где B — матрица инцидентности с одной удаленной строкой, B' — транспонированная матрица к B .

Рассмотрим алгоритм, позволяющий найти все покрывающие деревья. Пусть задан связный граф, содержащий n вершин и m ребер. Покрывающее дерево содержит $k = n - 1$ ребер и представляет собой k -элементное подмножество m -элементного множества ребер, т.е. сочетание из m по k . Но не каждое сочетание из m по k представляет собой покрывающее дерево. Поэтому для получения всех покрывающих деревьев можно использовать алгоритм 2.5 порождения сочетаний и, когда сочетание получено, нужно проверить, является ли полученное сочетание деревом или циклическим графом.

Для того, чтобы порождать только те сочетания, которые являются покрывающими деревьями, выполним следующие действия:

1. Выберем произвольную вершину v .
2. Ребрам, инцидентным вершине v , припишем номера $1, 2, \dots, |\Gamma(v)|$, а остальным — $|\Gamma(v)| + 1, |\Gamma(v)| + 2, \dots, m$.
3. Покрывающее дерево будем представлять последовательностью T ребер в возрастающем порядке.
4. На первое место последовательности ставить только ребра, инцидентные вершине v .
5. Очередное i -е место последовательности T заполняется по алгоритму 4.8.

Алгоритм 4.8 формирования всех покрывающих деревьев в графе G .

Вход: i — заполняемое место в последовательности T ;

b — минимальный элемент, который можно поставить на i -е место последовательности T .

Выход: последовательность всех покрывающих деревьев.

Глобальные параметры: T — формируемая последовательность ребер;

m — количество ребер в графе G ;

k — количество ребер в покрывающем дереве.

Алгоритм 4.8 формирования всех покрывающих деревьев представлен блок-схемой на рис.4.23. В этом алгоритме в цикле перебираются ребра с большим номером, чем ребра, включенные в частично сформированное дерево, и такие, добавление которых к частично сформированному дереву не образует цикл. Для определения, образует ли ребро с ранее включенными в дерево ребрами цикл, можно использовать понятие “букет” аналогично тому, как это делалось в алгоритме Краскала.

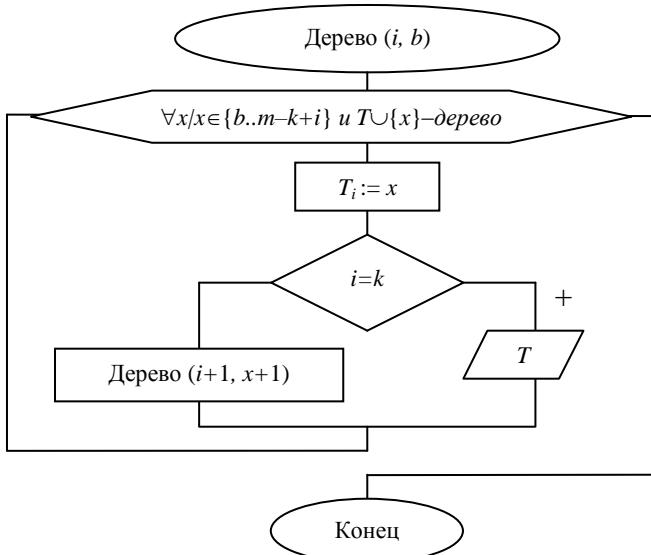


Рис.4.23. Рекурсивный алгоритм формирования всех покрывающих деревьев

Процесс формирования всех покрывающих деревьев графа, диаграмма которого дана на рис.4.24, представлен в виде дерева на рис.4.25. Вершины дерева соответствуют ребрам графа, а ветви от корня до листа содержат ребра покрывающего дерева. Диаграммы покрывающих деревьев изображены на рис.4.25.

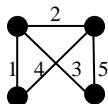


Рис.4.24. Диаграмма графа

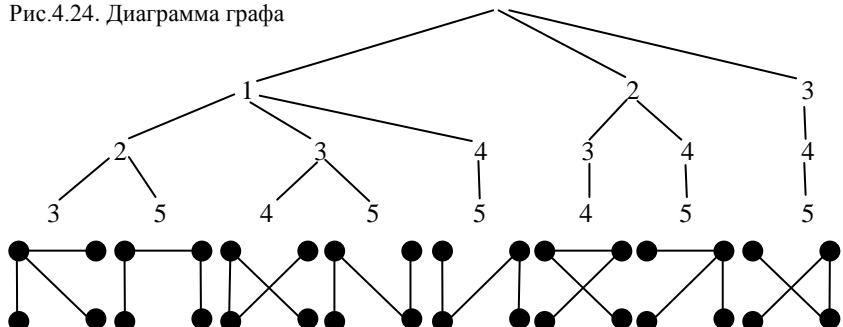


Рис.4.25. Процесс формирования всех покрывающих деревьев

4.10. Покрывающие деревья минимальной стоимости

Пусть задан граф со взвешенными ребрами и его покрывающее дерево. Сумма весов ребер, принадлежащих покрывающему дереву, называется *стоимостью дерева*. Покрывающее дерево графа, имеющее наименьшую стоимость по сравнению со всеми другими покрывающими деревьями этого же графа, называется *покрывающим деревом минимальной стоимости*.

Рассмотрим задачи, решение которых сводится к нахождению покрывающего дерева минимальной стоимости.

Задача 4.2. В управлении шоссейных дорог рассматривается проект строительства новых дорог, которые должны связать несколько городов некоторого района (причем не обязательно непосредственно каждую пару городов). Стоимость прокладки дороги между каждой парой городов известна. Необходимо определить, между какими городами нужно проложить дороги, чтобы суммарная стоимость прокладки дорог была минимальной.

Задача 4.3. Необходимо соединить проводами клеммы электрической сети, минимизируя длину проводника. Расстояния между каждой парой клемм известно.

Для нахождения покрывающего дерева минимальной стоимости можно получить все покрывающие деревья, определить их стоимость и выбрать дерево минимальной стоимости. Такой подход не является эффективным.

Алгоритм Краскала строит покрывающее дерево минимальной стоимости, если ребра графа просматривать в порядке неубывания их весов. Поэтому для нахождения покрывающего дерева минимальной стоимости можно предварительно использовать алгоритм сортировки ребер, а затем применить алгоритм Краскала. Тогда на каждом шаге будет выбираться ребро с минимальным весом, которое можно включить в покрывающее дерево. Такой выбор можно осуществить и в процессе выполнения алгоритма Краскала без предварительной сортировки ребер. В этом случае при выборе ребра используем алгоритм нахождения ребра минимального веса среди ребер, не включенных в дерево, при этом все ребра, образующие цикл с ребрами, включенными в дерево, исключаются из дальнейшего рассмотрения.

Рассмотрим еще один алгоритм построения покрывающего дерева минимальной стоимости, не требующий предварительной сортировки ребер. Это алгоритм Прима (алгоритм ближайшего соседа). Он отличается от алгоритма Краскала тем, что в нем формируется только один букет. На очередном шаге выбирается ребро минимального веса среди

ребер, у которых одна концевая вершина принадлежит букету, а другая — не принадлежит. Такое ребро включается в дерево, а концевая вершина, не принадлежащая букету, включается в него.

Для эффективного выбора ребра в алгоритме Прима каждой вершине v_i графа, не принадлежащей букету B приписывается метка $t(v_i)$ — “ближайшая” к v_i вершина, принадлежащая букету B , и число $d(v_i)$ — вес ребра $\{t(v_i), v_i\}$. На каждом шаге выполнения алгоритма вершина v_i с наименьшим числом $d(v_i)$ добавляется к букету B , а ребро $\{t(v_i), v_i\}$ — к покрывающему дереву. После добавления новой вершины v_i к букету B метка $t(v_j)$ и число $d(v_j)$ вершины v_j , не принадлежащей букету, изменяются, если вес ребра $\{v_i, v_j\}$ меньше числа $d(v_j)$.

Алгоритм Прима.

Вход: $G = (V, E)$ — взвешенный граф, где

V — множество вершин,

E — множество ребер.

Выход: T — покрывающее дерево минимальной стоимости.

1. Начальная установка.

Каждой вершине v_i приписать $t(v_i) = 0$;

$B := \{v\}$, v — произвольная вершина графа включается в букет B ;

каждой вершине v_i , не принадлежащей букету, приписать $d(v_i) = \infty$,

$y := v$, y — последняя вершина, включенная в букет;

$T := \emptyset$, дерево пусто.

2. Пересчет чисел $d(v_i)$ и меток $t(v_i)$.

$\forall v_i \in \Gamma(y) - B$ пересчитать $d(v_i)$:

$$d(v_i) := \min(d(v_i), \text{вес}\{y, v_i\}).$$

Если число $d(v_i)$ изменилось (уменьшилось), то $t(v_i) := y$.

3. Из всех вершин $v_i \notin B$ выбрать вершину v_j с наименьшим числом $d(v_j)$;

$B := B \cup \{v_j\}$, включить выбранную вершину в букет;

$y := v_j$, y — последняя вершина, включенная в букет;

$T := T \cup \{t(v_i), v_i\}$, добавить ребро в дерево.

4. Если все вершины графа включены в букет ($B = V$), то конец алгоритма, иначе перейти к п.2

При программной реализации алгоритма Прима букет можно представить бинарным массивом B , i -й элемент которого соответствует вершине v_i и, если вершина v_i принадлежит букету, то $B_i = 1$, иначе $B_i = 0$. Числа $d(v_i)$ целесообразно хранить в массиве D , i -й элемент которого соответствует вершине v_i , а его значение D_i — числу $d(v_i)$. Для хранения меток $t(v_i)$ можно использовать массив T , i -й элемент которого соответствует вершине v_i , а его значение T_i — метке $t(v_i)$. Заметим, что мас-

сив T по окончании алгоритма хранит полную информацию о структуре построенного покрывающего дерева минимальной стоимости: пара $\{T_i, i\}$ соответствует ребру дерева, если $T_i \neq 0$, а сумма всех элементов массива D равна стоимости дерева.

Работу алгоритма Прима при построении покрывающего дерева минимальной стоимости графа, диаграмма которого изображена на рис.4.26, представим в табл. 4.4, отображающей изменения значений массивов B , D и T на каждом шаге выполнения алгоритма.

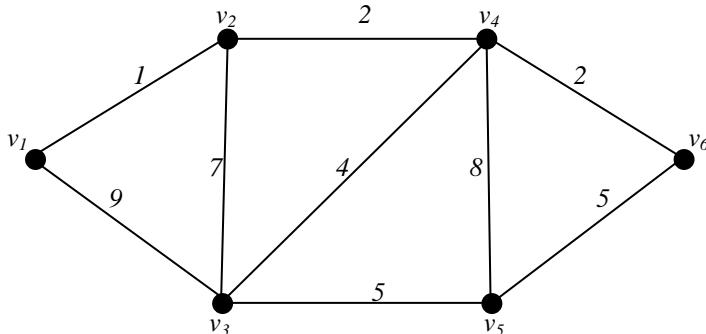


Рис.4.26. Диаграмма графа

Таблица 4.4

Работа алгоритма Прима

Номер шага	Пункт алгоритма	Состояние массивов B , D и T	Вершина у
1	2	$B=(1, 0, 0, 0, 0, 0)$	4
1	1	$D=(\infty, \infty, \infty, \infty, \infty)$ $T=(0, 0, 0, 0, 0, 0)$	1
2	2	$B=(1, 0, 0, 0, 0, 0)$ $D=(0, 1, 9, \infty, \infty, \infty)$ $T=(0, 1, 1, 0, 0, 0)$	1
3	3	$B=(1, 1, 0, 0, 0, 0)$ $D=(0, 1, 9, \infty, \infty, \infty)$ $T=(0, 1, 1, 0, 0, 0)$	2
4	4	$B \neq V$, продолжить	2
5	2	$B=(1, 1, 0, 0, 0, 0)$ $D=(0, 1, 7, 2, \infty, \infty)$ $T=(0, 1, 2, 2, 0, 0)$	2
6	3	$B=(1, 1, 0, 1, 0, 0)$ $D=(0, 1, 7, 2, \infty, \infty)$ $T=(0, 1, 2, 2, 0, 0)$	4

Окончание табл.4.4

1	2	3	4
7	4	$B \neq V$, продолжить	4
8	2	$B=(1, 1, 0, 1, 0, 0)$ $D=(0, 1, 4, 2, 8, 2)$ $T=(0, 1, 4, 2, 4, 4)$	4
9	3	$B=(1, 1, 0, 1, 0, 1)$ $D=(0, 1, 4, 2, 8, 2)$ $T=(0, 1, 4, 2, 4, 4)$	6
10	4	$B \neq V$, продолжить	6
11	2	$B=(1, 1, 0, 1, 0, 1)$ $D=(0, 1, 4, 2, 5, 2)$ $T=(0, 1, 4, 2, 6, 4)$	6
12	3	$B=(1, 1, 1, 1, 0, 1)$ $D=(0, 1, 4, 2, 5, 2)$ $T=(0, 1, 4, 2, 6, 4)$	3
13	4	$B \neq V$, продолжить	3
14	2	$B=(1, 1, 1, 1, 0, 1)$ $D=(0, 1, 4, 2, 5, 2)$ $T=(0, 1, 4, 2, 6, 4)$	3
15	3	$B=(1, 1, 1, 1, 1, 1)$ $D=(0, 1, 4, 2, 5, 2)$ $T=(0, 1, 4, 2, 6, 4)$	5
16	4	$B=V$, конец	5

Диаграмма построенного покрывающего дерева минимальной стоимости представлена на рис.4.27.

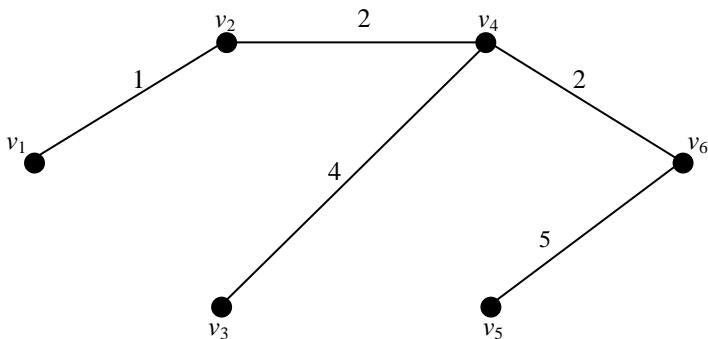


Рис.4.27. Диаграмма покрывающего дерева минимальной стоимости

4.11. Поиск в орграфе

Задача 4.4. Система состоит из конечного числа объектов. Между некоторыми из них установлены одно- или двусторонние каналы связи. Определить, каким объектам системы может передать сообщение по каналам связи заданный объект.

Решение. Рассматриваемую систему можно представить орграфом, в котором вершины соответствуют объектам, а дуги – каналам связи. Объект, соответствующий вершине v_i , может передать сообщение объекту, соответствующему вершине v_j , если в орграфе существует путь из v_i в v_j , т.е. если вершина v_j достижима из v_i . Пусть заданному объекту соответствует вершина v_i . Тогда для решения задачи необходимо найти все вершины орграфа, достижимые из v_i . Процесс нахождения вершин, достижимых из заданной, называется *поиском в орграфе*.

Одним из методов поиска в орграфе является *поиск в глубину*. Порядок “посещения” вершин при поиске в глубину определяется следующими правилами:

1. Посещаем начальную вершину v_i и считаем ее текущей.
2. Если v_t — текущая вершина, v_j — вершина, смежная вершине v_t и еще не посещалась, то посещаем ее и считаем текущей.
3. Если исходная вершина v_i — текущая вершина и все смежные с ней уже посещались, то поиск заканчивается.
4. Если v_t — текущая вершина и все смежные с ней уже посещались, то текущей считаем вершину, из которой пришли в вершину v_t .

Процесс поиска в глубину можно представить как построение ориентированного дерева, корнем которого является начальная вершина орграфа, остальные вершины дерева — вершины орграфа, достижимые из начальной. Дуги дерева соответствуют дугам орграфа, по которым осуществлялось “движение” во время поиска. На рис.4.28 представлен орграф, а на рис.4.29 — дерево поиска в глубину с начальной вершиной 1. Нижний индекс около номера вершины дерева определяет порядок посещения вершин.

Для программной реализации метода поиска в глубину можно использовать рекурсивный или итеративный алгоритм. В алгоритмах используется множество V' вершин, достижимых из начальной. В исходном состоянии оно содержит только начальную вершину. В итеративном алгоритме для запоминания вершин, предшествующих текущей, используется стек. Дерево поиска можно сохранить в массиве T , количество элементов которого равно количеству вершин орграфа. Элемент T_i представляет собой вершину, предшествующую вершине i в дереве. Если вершина i — корень дерева, то $T_i = 0$. Если вершина i недостижима

из начальной, $T_i = -1$. В исходном состоянии элемент массива T , соответствующий начальной вершине, равен 0, а все остальные равны -1 .

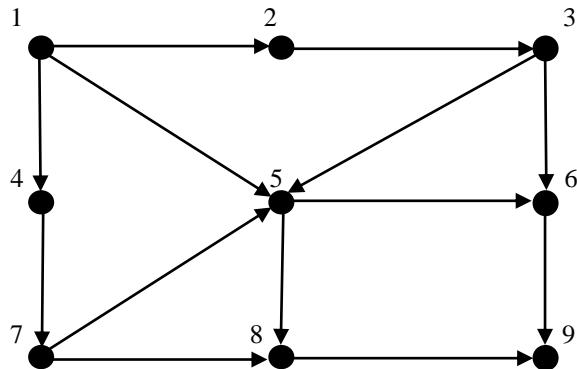


Рис.4.28. Диаграмма графа

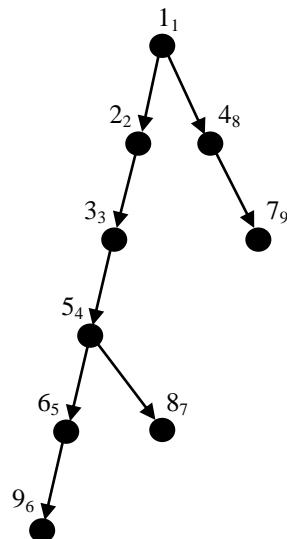


Рис.4.29. Дерево поиска в глубину

Рекурсивный алгоритм 4.9 (рис.4.30) поиска в глубину.

Вход: v — текущая вершина.

Выход: последовательность вершин орграфа в порядке их посещения в процессе поиска в глубину (начальная вершина выводится перед первым обращением);

T — дерево поиска в глубину.

Глобальные параметры: V' — множество вершин, достижимых из начальной.

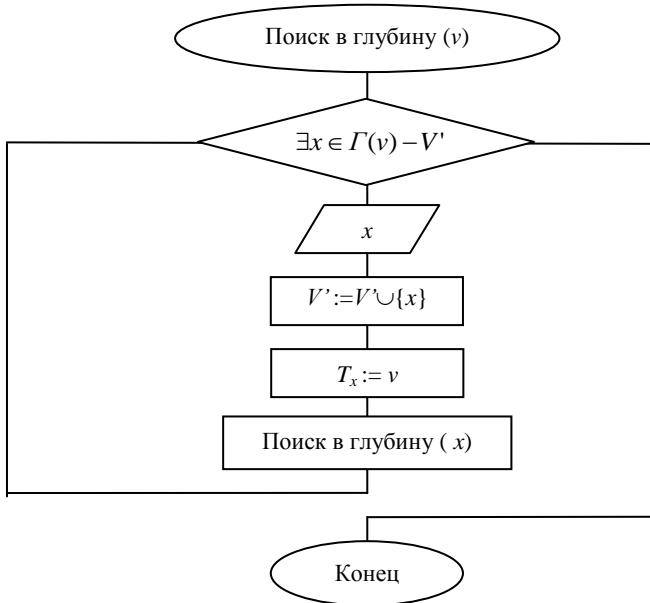


Рис.4.30. Рекурсивный алгоритм поиска в глубину

Итеративный алгоритм 4.10 (рис.4.31) поиска в глубину.

Вход: v — начальная вершина.

Выход: последовательность вершин орграфа в порядке их посещения в процессе поиска в глубину;

T — дерево поиска в глубину.

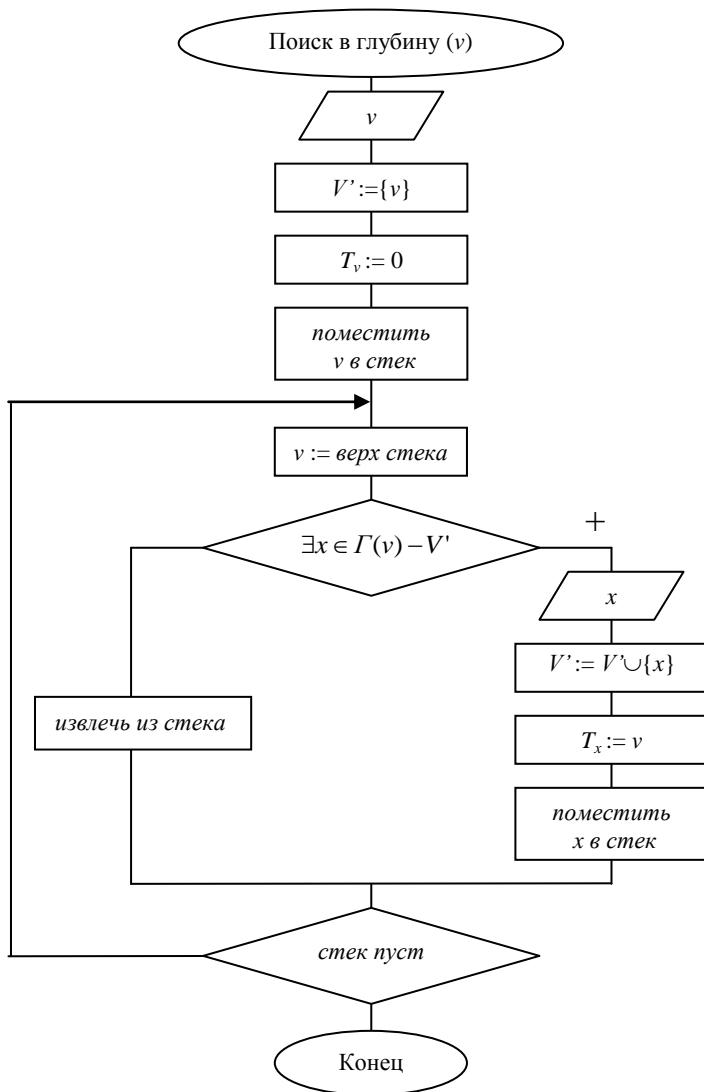


Рис.4.31. Итеративный алгоритм поиска в глубину

Задача 4.5. Система состоит из конечного числа объектов. Между некоторыми из них установлены одно- или двусторонние каналы связи. Определить пути передачи сообщения по каналам связи от заданного объекта ко всем достижимым, содержащие минимальное число промежуточных объектов.

Решение. Рассматриваемую систему можно представить орграфом, в котором вершины соответствуют объектам, а дуги – каналам связи. Пути передачи сообщения от заданного объекта ко всем достижимым можно представить деревом поиска. Поиск в глубину не позволяет найти пути, содержащие минимальное число промежуточных объектов. Для нахождения таких путей используется *поиск в ширину*.

Порядок посещения вершин при поиске в ширину определяется следующими правилами:

1. Посещаем начальную вершину v_i и считаем ее текущей.
2. Если v_i – текущая вершина, $\Gamma(v_i)$ – множество вершин, смежных с v_i , то вершины из множества $\Gamma(v_i)$, которые еще не посещались, последовательно посещаем, а вершину, которую посетили вслед за v_i , считаем текущей.
3. Если v_i – последняя вершина, которая посещалась при поиске, является текущей и все вершины, смежные с ней, уже посещались, то поиск заканчивается.

Дерево поиска в ширину с начальной вершиной 1 для орграфа (рис.4.28) представлено на рис.4.32. Нижний индекс около номера вершины дерева определяет порядок посещения вершин.

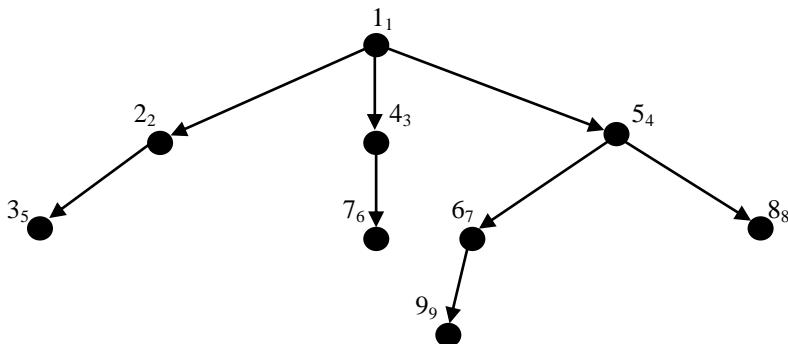


Рис.4.32. Дерево поиска в ширину

При программной реализации метода поиска в ширину для запоминания порядка посещения вершин используется очередь, а множество V' – для хранения вершин, достижимых из начальной. Дерево поиска

в ширину можно сохранить в массиве T , в котором элемент T_i представляет собой вершину, предшествующую вершине i в дереве.

Алгоритм 4.11 (рис.4.33) поиска в ширину.

Вход: v — начальная вершина.

Выход: последовательность вершин орграфа в порядке их посещения в процессе поиска в глубину;

T — дерево поиска в ширину.

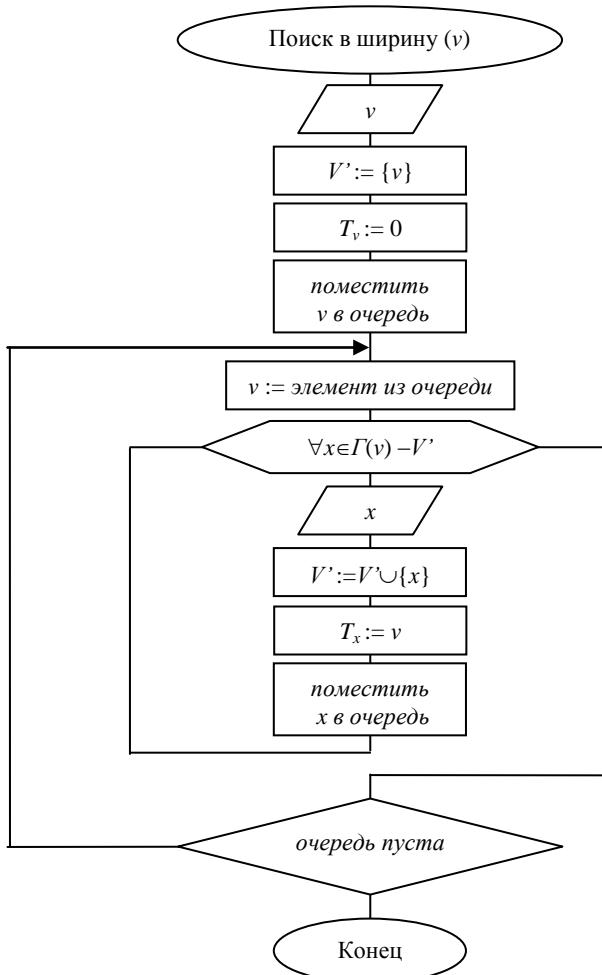


Рис.4.33. Алгоритм поиска в ширину

4.12. Связность в орграфе. Компоненты сильной связности

Две вершины v_i и v_j орграфа $G = (V, E)$ называются *сильно связанными*, если v_i достижима из v_j и v_j достижима из v_i , т.е. вершины v_i и v_j взаимодостижимы.

Две вершины v_i и v_j орграфа $G = (V, E)$ называются *односторонне связанными*, если v_i достижима из v_j или v_j достижима из v_i .

Две вершины v_i и v_j орграфа $G = (V, E)$ называются *слабо связанными*, если они связаны в графе G' , полученном из орграфа G' путем отмены ориентации дуг.

Если все вершины в орграфе сильно (односторонне, слабо) связаны, то орграф называется *сильно (односторонне, слабо) связанным*. На рис.4.34 показаны диаграммы сильно, односторонне и слабо связанных орграфов.

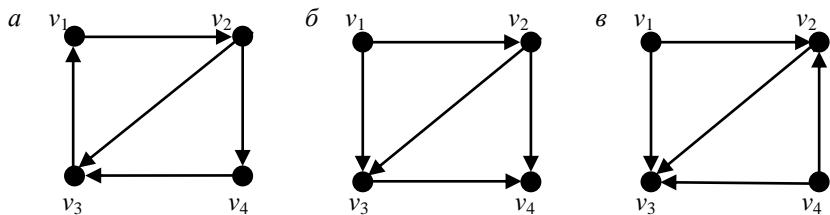


Рис.4.34. Диаграммы орграфов:
а — диаграмма сильно связного орграфа;
б — диаграмма односторонне связного орграфа;
в — диаграмма слабо связного орграфа

На множестве вершин орграфа можно определить *отношение взаимодостижимости*. Пара вершин (v_i, v_j) принадлежит отношению взаимодостижимости, если v_i и v_j взаимодостижимы. Отношение взаимодостижимости является отношением эквивалентности: оно рефлексивно, симметрично и транзитивно, поэтому оно разбивает множество вершин на классы эквивалентных вершин. Подграф, построенный на множестве вершин одного класса эквивалентности, представляет собой *компоненту сильной связности* орграфа.

Для нахождения отношения H взаимодостижимости используются отношения R достижимости и Q контрдостижимости. Пара вершин (v_i, v_j) принадлежит отношению R достижимости, если v_j достижима из v_i . Пара вершин (v_i, v_j) принадлежит отношению Q контрдостижимости, если v_i достижима из v_j . Отношения достижимости, контрдостижимости и взаимодостижимости будем задавать матрицами R , Q и H соответственно.

Рассмотрим два способа получения матрицы R достижимости.

Первый способ.

Получить матрицу R достижимости можно, используя поиск в орграфе. Для этого необходимо инициализировать матрицу R нулями, а затем последовательно формировать строки матрицы. При формировании i -й строки, используя поиск в орграфе, находится множество V' вершин, достижимых из вершины i . Если вершина j принадлежит множеству V' , то элементу c_{ij} присвоить значение 1.

Второй способ.

Вычислить матрицу R достижимости можно по формуле $R = I + M^+$, где I — бинарная матрица, содержащая единицы только на главной диагонали;

M — матрица смежности орграфа;

M^+ — транзитивное замыкание отношения, представленного матрицей смежности орграфа, которое можно вычислить, используя алгоритм 3.10 объединения степеней или алгоритм 3.11 Уоршалла.

Отношение Q контрудостигимости вычисляется по формуле $Q := R^{-1}$, а отношение H взаимодости — по формуле $H := R \cap Q$.

По матрице H взаимодости можно построить разбиение множества вершин орграфа на классы эквивалентности, используя алгоритм 3.13, тем самым найдем подмножества вершин, образующих компоненты сильной связности орграфа.

На рис.4.35 представлена диаграмма орграфа, на рис.4.36 — матрицы отношений достижимости, контрудостигимости и взаимодости, а на рис.4.37 — компоненты сильной связности.

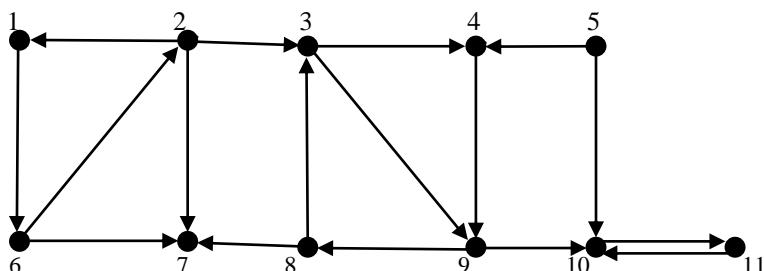


Рис.4.35. Диаграмма орграфа

a

$$R = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

b

$$Q = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

c

$$H = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Рис.4.36. Матрицы отношений:

a — достижимости;*b* — контрдостигимости;*c* — взаимодостигимости

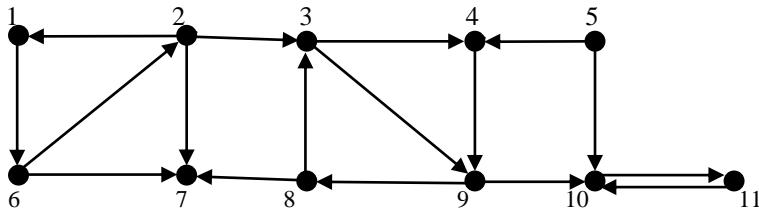


Рис.4.37. Компоненты сильной связности

Множество V вершин орграфа (рис.4.35) разбивается отношением H взаимодостижимости на пять классов эквивалентности: $V_1 = \{1,2,6\}$, $V_2 = \{3,4,8,9\}$, $V_3 = \{5\}$, $V_4 = \{7\}$ и $V_5 = \{10,11\}$, следовательно, орграф имеет пять компонент сильной связности.

Орграф $G^* = (V^*, E^*)$, в котором

$V^* = \{V_1, V_2, \dots, V_n\}$, где n — количество компонент сильной связности,

V_i — множество вершин i -й компоненты сильной связности,

$E^* = \{(V_i, V_j) / v_i \in V_i \text{ и } v_j \in V_j \text{ и } (v_i, v_j) \in E\}$,

называется *конденсацией* орграфа $G = (V, E)$. Конденсация G^* орграфа G получается “стягиванием” в одну вершину каждой компоненты сильной связности. Конденсация G^* орграфа G (рис.4.35) показана на рис.4.38.

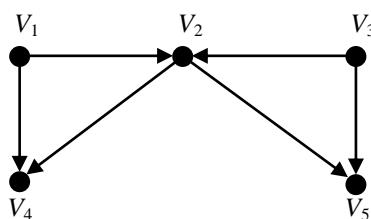


Рис.4.38. Конденсация орграфа

Базой B орграфа G называется такое множество вершин, которое удовлетворяет следующим двум условиям:

1) каждая вершина орграфа G достижима хотя бы из одной вершины множества B ;

2) в B нет вершины, которая достижима из другой вершины множества B .

Из этих двух условий следует, что:

1) в множестве B нет двух вершин, которые принадлежат одной и той же компоненте сильной связности орграфа G ;

2) в любом орграфе без циклов существует единственная база — она состоит из всех таких вершин орграфа, полустепени захода которых равны нулю.

Конденсация G^* орграфа G не содержит циклов, поэтому имеет единственную базу B^* , состоящую из вершин орграфа G^* , полустепени захода которых равны нулю. Базы орграфа G можно строить так: из каждой компоненты сильной связности, соответствующей вершине базы B^* конденсации G^* надо взять по одной вершине.

Базой орграфа G^* (рис.4.38) является множество $B^* = \{V_1, V_3\}$, а базами орграфа G (рис.4.35) являются множества $B_1 = \{1, 5\}$, $B_2 = \{2, 5\}$ и $B_3 = \{6, 5\}$.

Антибазой \bar{B} орграфа G называется такое множество вершин, которое удовлетворяет следующим двум условиям:

1) из любой вершины орграфа G достижима хотя бы одна вершина множества \bar{B} ;

2) в \bar{B} нет вершины, из которой достижима другая вершина множества \bar{B} .

Из этих двух условий следует, что:

1) в множестве \bar{B} нет двух вершин, которые принадлежат одной и той же компоненте сильной связности орграфа G ;

2) в любом орграфе без циклов существует единственная антибаза — она состоит из всех таких вершин орграфа, полустепени исхода которых равны нулю.

Конденсация G^* орграфа G не содержит циклов, поэтому имеет единственную антибазу \bar{B}^* , состоящую из вершин орграфа G^* , полустепени исхода которых равны нулю. Антибазы орграфа G можно строить так: из каждой компоненты сильной связности, соответствующей вершине антибазы \bar{B}^* конденсации G^* надо взять по одной вершине.

Антибазой орграфа G^* (рис.4.38) является множество $\bar{B}^* = \{V_4, V_5\}$, а антибазами орграфа G (рис.4.35) являются множества $\bar{B}_1 = \{7, 10\}$ и $\bar{B}_2 = \{7, 11\}$.

Задача 4.6. Руководство организации должно иметь возможность получать информацию от каждого сотрудника и иметь власть над каждым сотрудником. Необходимо сформировать эффективное руководство для управления организацией из наименьшего числа людей.

Решение. 1. Построим орграф $G = (V, E)$, в котором вершины соответствуют членам организации и $(v_i, v_j) \in E$, если член организации, со-

ответствующий вершине v_i , имеет возможность непосредственно связываться с членом организации, соответствующим вершине v_j . Наименьшее число лиц, которые знают или могут получать все сведения об организации, образуют одну из антибаз \bar{B} орграфа G .

2. Построим орграф $G' = (V, E')$, в котором вершины соответствуют членам организации и $(v_i, v_j) \in E'$, если член организации, соответствующий вершине v_i , имеет непосредственное влияние на члена организации, соответствующего вершине v_j . Наименьшее число лиц, имеющих власть над каждым сотрудником, образуют одну из баз B' орграфа G' .

3. Руководство может представлять собой множество $\bar{B} \cup B'$ наименьшей мощности.

4.13. Кратчайший путь в орграфе

Рассмотрим задачу на графе.

Дан взвешенный орграф (взвешены дуги, веса дуг положительные) и две его вершины — S и F . Длиной пути от вершины S до вершины F называется сумма весов всех дуг, принадлежащих этому пути. Требуется найти в орграфе путь от вершины S до вершины F минимальной длины, т.е. путь, длина которого не больше любого другого пути от вершины S до вершины F . Путь минимальной длины будем также называть *кратчайшим*, а длину этого пути — *кратчайшим* или *минимальным расстоянием* от начальной вершины пути до конечной.

Многие реальные задачи сводятся к рассматриваемой задаче на графе естественным образом. Например, дано множество населенных пунктов, между некоторыми из них проложены автомобильные дороги с одно- или двусторонним движением. Известны длины дорог, соединяющих населенные пункты. Требуется найти путь минимальной длины от населенного пункта A до пункта B .

Для того чтобы эту задачу свести к задаче на графе, построим граф, в котором вершины соответствуют населенным пунктам, а дуги — парам населенных пунктов, между которыми проложены дороги, направление дуги определяется направлением разрешенного движения по соответствующей дороге. Вес дуги определяется длиной соответствующей ей дороги.

Сведение же других реальных задач к рассматриваемой задаче на графе не так уж очевидно.

Предположим, что вам необходимо иметь в своем распоряжении автомобиль в течение нескольких лет. Имеется большой выбор автомобилей. Автомобили имеют различные сроки эксплуатации и разную стоимость. Каким образом на заданном временном интервале выбрать вари-

ант покупок автомобилей, имеющий минимальную суммарную стоимость покупаемых автомобилей?

Для того, чтобы свести данную задачу к задаче на графе, представим моменты времени возможных сделок на заданном временном интервале, связанные с покупкой автомобиля, вершинами некоторого графа. Для упрощения в качестве моментов времени сделок можно рассматривать лишь первые дни каждого месяца, квартала или другого временного отрезка. Изобразим в данном графе факт приобретения автомобиля дугой, соединяющей вершину, соответствующую моменту покупки, с вершиной, соответствующей моменту окончания срока службы автомобиля. Длина каждой дуги построенного графа совпадает со стоимостью соответствующей сделки.

Чтобы выбрать вариант с минимальной суммарной стоимостью, необходимо среди всех возможных путей из вершины, соответствующей начальному моменту времени, в вершину, соответствующую конечному моменту времени, найти кратчайший путь.

Задачу отыскания кратчайшего пути во взвешенном орграфе от вершины S до вершины F можно решить различными способами.

1. Кратчайший путь от вершины S до вершины F представляет собой простую цепь (любой другой путь, содержащий в себе цикл, будет иметь большую длину), поэтому для отыскания кратчайшего пути достаточно выполнить полный перебор простых цепей от вершины S до вершины F и выбрать цепь минимальной длины. Для этого введем в алгоритм 4.3 нахождения простых цепей методом поиска с возвращением дополнения, позволяющие выбрать цепь минимальной длины. Цепь W представлена последовательностью вершин с вершиной S на первом месте. В процессе формирования цепи будем вычислять ее длину L . Если в цепь из $i - 1$ вершин длины L добавить i -ю вершину x , то получим цепь из i вершин длины $L' = L + \text{длина_дуги}(W_{i-1}, x)$. Будем использовать переменную L_{min} для хранения длины пути, принятого за минимальный, а переменную W_{min} — для хранения этого пути. Инициализируем L_{min} бесконечно большим числом. После нахождения очередной цепи W от вершины S до вершины F сравним ее длину L с L_{min} и, если окажется, что L меньше L_{min} , запомним цепь W в переменной W_{min} , а ее длину L — в L_{min} .

Алгоритм 4.12 (рис.4.39) поиска кратчайшего пути от вершины S до вершины F во взвешенном орграфе G .

Вход: i — заполняемое место в цепи W ;

V' — множество вершин, включенных в цепь;

L — длина цепи.

Выход: W_{min} — кратчайший путь от вершины S до вершины F ;

L_{min} — длина кратчайшего пути.

Глобальные параметры: W_{min}, L_{min} .

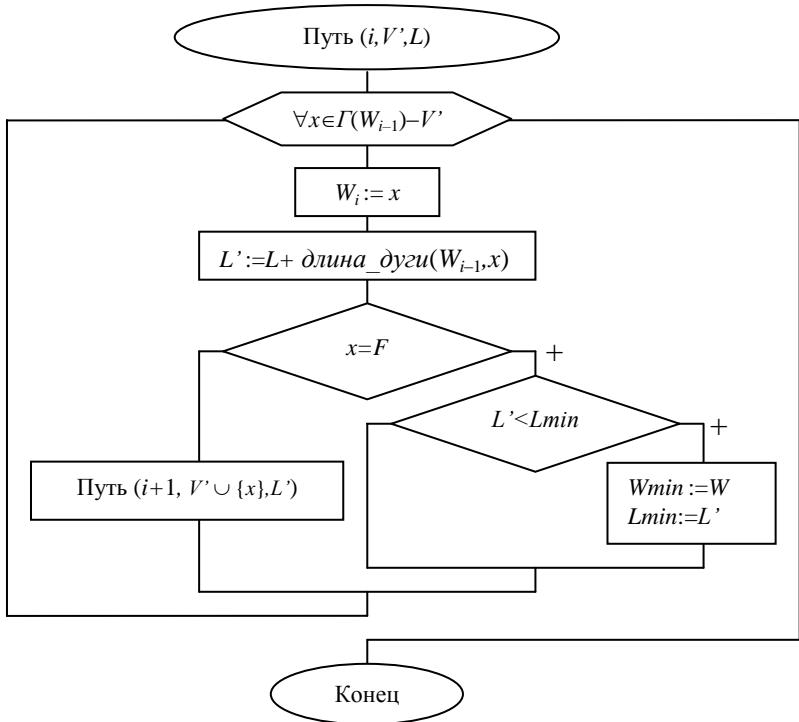


Рис.4.39. Алгоритм поиска кратчайшего пути от вершины S до вершины F

2. Для сокращения количества анализируемых простых цепей воспользуемся идеей метода *ветвей и границ*. Этот метод позволяет сократить число анализируемых решений, когда требуется выбрать из множества всех решений наилучшее, обладающее заданным свойством. В нашем случае множество всех решений — это множество всех простых цепей от вершины S до вершины F , а наилучшее — кратчайший путь. В методе ветвей и границ решение представляет собой "ветвь", получение решения — построение "ветви". Если на некотором шаге построения "ветви" станет ясно, что продолжение построения "ветви" не даст наилучшего решения, то на этом шаге получения решения "ветвь" ограничивается, выполняется шаг назад и делается попытка построить

”ветвь” в другом направлении. Пусть L_{min} — длина пути от вершины S до вершины F , принятого за минимальный. Если на некотором шаге добавление дуги (W_{i-1}, x) дает цепь, длина которой $L' \geq L_{min}$, то дальнейшее построение этой цепи прекращается и выполняется шаг назад.

Алгоритм 4.13 (рис.4.40) поиска кратчайшего пути от вершины S до вершины F во взвешенном орграфе G на основе метода ветвей и границ.

Вход: i — заполняемое место в цепи W ;

V' — множество вершин, включенных в цепь;

L — длина цепи.

Выход: W_{min} — кратчайший путь от вершины S до вершины F ;

L_{min} — длина кратчайшего пути.

Глобальные параметры: W_{min}, L_{min} .

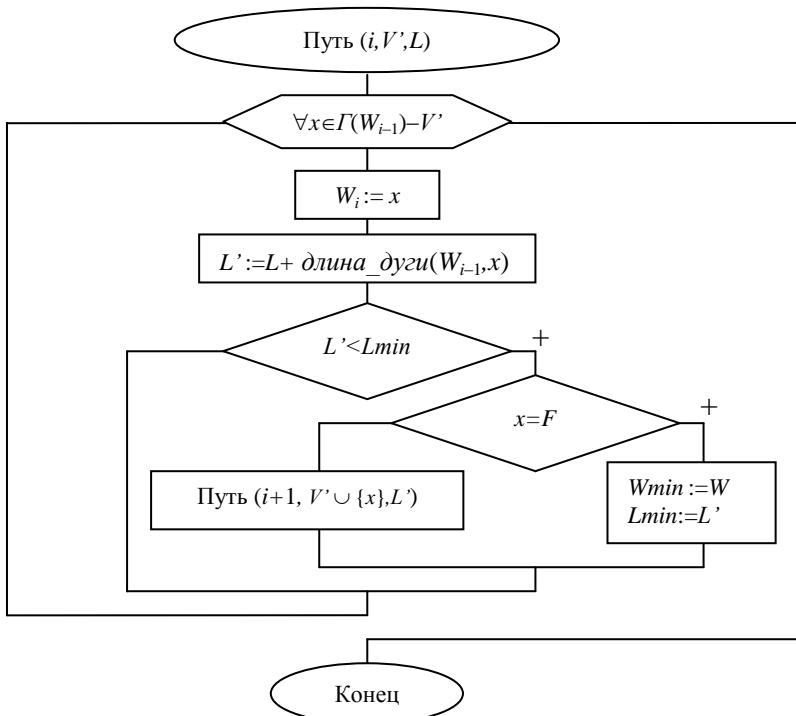


Рис.4.40. Алгоритм поиска кратчайшего пути от вершины S до вершины F на основе метода ветвей и границ

3. Один из эффективных алгоритмов нахождения кратчайшего пути во взвешенном орграфе между двумя вершинами, при условии, что все дуги орграфа имеют положительный вес, предложил в 1959 г. Едсгер Дейкстра.

В алгоритме Дейкстры множество вершин V орграфа разбивается на два непересекающихся подмножества — V' и V'' , таких, что $V' \cup V'' = V$. Множество V' содержит вершины орграфа, до которых найдено кратчайшее расстояние (и кратчайший путь) из исходной вершины S , а множество V'' — вершины орграфа, до которых еще не найдено кратчайшее расстояние (и кратчайший путь) из исходной вершины S . Вершины, принадлежащие множеству V' , будем называть *отмеченными*. Каждой вершине $x_i \in V$ орграфа приписывается число $d(x_i)$, которое служит оценкой длины кратчайшего пути от вершины S к вершине x_i . Если вершине x_i приписано число $d(x_i)$, то это означает, что в орграфе существует путь из S в x_i , имеющий длину $d(x_i)$. Если вершина $x_i \in V'$, то это означает, что $d(x_i)$ — кратчайшее расстояние от вершины S к вершине x_i и кратчайший путь от вершины S к вершине x_i найден.

На каждом шаге алгоритма числа $d(x_i)$ для вершин из множества V'' пересчитываются, и среди всех вершин $x_i \in V''$ выбирается вершина x_i , к которой приписано минимальное число $d(x_i)$. До этой вершины кратчайшее расстояние (и кратчайший путь) на этом шаге найдено, она исключается из множества V'' ($V'' := V'' - \{x_i\}$) и заносится в множество V' ($V' := V' \cup \{x_i\}$). Алгоритм заканчивает работу, если на очередном шаге найдено кратчайшее расстояние (и путь) до конечной вершины F или если нельзя выбрать вершину с минимальным числом (ко всем вершинам из множества V'' приписано бесконечно большое число, в этом случае пути от вершины S ко всем вершинам из множества V'' не существует).

Алгоритм Дейкстры.

Вход: взвешенный орграф;

S — начальная вершина пути;

F — конечная вершина пути;

Выход: кратчайший путь из S в F и его длина $d(F)$.

1. Начальная установка.

$V' := \{S\}$, $V'' := V - \{S\}$, $d(S) := 0$, $\forall x_i \in V'', d(x_i) := \infty$,

$y := S$ (у — последняя вершина, до которой найден кратчайший путь).

2. Пересчет чисел $d(x_i)$.

$\forall x_i \in \Gamma(y) \cap V''$ пересчитать $d(x_i)$:

$$d(x_i) := \min(d(x_i), d(y) + \text{длина_дуги}(y, x_i)).$$

Если величина $d(x_i)$ изменилась (уменьшилась), то это означает, что кратчайший путь в вершину x_i , вероятно, содержит дугу (y, x_i) , поэтому ее нужно запомнить как претендента на включение в кратчайший путь.

3. Поиск вершины x_i , до которой найден кратчайший путь.

Из всех вершин $x_i \in V''$ выбрать вершину x_j с наименьшим значением $d(x_j)$. Если $d(x_j) = \infty$, то пути от вершины S ко всем вершинам из множества V'' не существует, иначе $y := x_j$, $V' := V' \cup \{y\}$, $V'' := V'' - \{y\}$, отметить дугу претендента, ведущую в вершину y .

4. Проверка на завершение.

Если $y = F$, то конец алгоритма, кратчайший путь из S в F найден, его длина $d(y)$.

Если $d(x_j) = \infty$, то пути из S в F нет, конец алгоритма.

Иначе перейти к п.2 .

В каждую вершину из множества V' , за исключением вершины S , входит только одна отмеченная дуга. Отмеченные дуги не могут образовывать в исходном графе цикл, так как в алгоритме не может отмечаться дуга, конец которой принадлежит множеству V' . Следовательно, отмеченные дуги образуют в исходном графе ориентированное дерево с корнем в вершине S . Такое дерево называется деревом кратчайших путей .

Из дуг (не обязательно из всех) дерева, складывается кратчайший путь из S в F . В дереве кратчайших путей только одна дуга входит в вершину F . Пусть это будет дуга (x_i, F) . Далее, только одна дуга входит в вершину x_i и т.д. Таким образом, поднимаясь к корню, восстанавливается кратчайший путь из S в F (в обратном порядке).

При программной реализации алгоритма Дейкстры числа $d(x_i)$ целесообразно хранить в массиве D , i -й элемент которого соответствует вершине x_i , а его значение D_i — числу $d(x_i)$. Для хранения дерева кратчайших путей можно использовать массив T , i -й элемент которого соответствует вершине x_i , а его значение T_i — вершине x_j , предшествующей вершине x_i на кратчайшем пути из S в x_i ; таким образом пара (T_i, i) соответствует дуге дерева кратчайших путей. Значение T_s равно нулю, так как в вершину S не входит ни одна дуга дерева, она является корнем. На первом шаге алгоритма, при начальной установке, необходимо элементу T_s присвоить ноль, остальным — неопределенность (например, минус один). На втором шаге, если величина $d(x_i)$ изменилась (уменьшилась), то элементу T_i нужно присвоить значение y , что соответствует запоминанию дуги (y, x_i) как претендента на включение в кратчайший путь. Для хранения множеств V' и V'' можно использовать бинарный массив V , в котором i -й элемент соответствует вершине x_i , $V_i = 1$, если $x_i \in V'$ и $V_i = 0$, если $x_i \in V''$.

Работу алгоритма Дейкстры при поиске кратчайшего пути от вершины v_1 до вершины v_5 во взвешенном орграфе, диаграмма которого изображена на рис.4.41, представим в табл. 4.5, отображающей изменения значений массивов V , D и T на каждом шаге выполнения алгоритма.

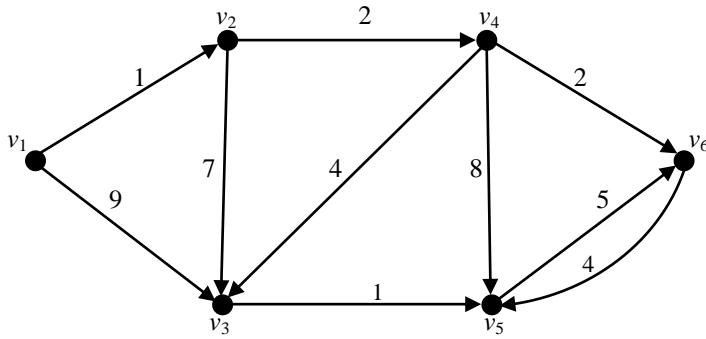


Рис.4.41. Диаграмма орграфа

Таблица 4.5

Работа алгоритма Дейкстры

Номер шага	Пункт алгоритма	Состояние массивов V , D и T	Вершина у
1	2	3	4
1	1	$V=(1, 0, 0, 0, 0, 0)$ $D=(0, \infty, \infty, \infty, \infty, \infty)$ $T=(0, -1, -1, -1, -1, -1)$	1
2	2	$V=(1, 0, 0, 0, 0, 0)$ $D=(0, 1, 9, \infty, \infty, \infty)$ $T=(0, 1, 1, -1, -1, -1)$	1
3	3	$V=(1, 1, 0, 0, 0, 0)$ $D=(0, 1, 9, \infty, \infty, \infty)$ $T=(0, 1, 1, -1, -1, -1)$	2
4	4	$2 \neq 5$, продолжить	2
5	2	$V=(1, 1, 0, 0, 0, 0)$ $D=(0, 1, 8, 3, \infty, \infty)$ $T=(0, 1, 2, 2, -1, -1)$	2
6	3	$V=(1, 1, 0, 1, 0, 0)$ $D=(0, 1, 8, 3, \infty, \infty)$ $T=(0, 1, 2, 2, -1, -1)$	4
7	4	$4 \neq 5$, продолжить	4
8	2	$V=(1, 1, 0, 1, 0, 0)$ $D=(0, 1, 7, 3, 11, 5)$ $T=(0, 1, 4, 2, 4, 4)$	4

Окончание табл.4.5

1	2	3	4
9	3	$V=(1, 1, 0, 1, 0, 1)$ $D=(0, 1, 7, 3, 11, 5)$ $T=(0, 1, 4, 2, 4, 4)$	6
10	4	$6 \neq 5$, продолжить	6
11	2	$V=(1, 1, 0, 1, 0, 1)$ $D=(0, 1, 7, 3, 9, 5)$ $T=(0, 1, 4, 2, 5, 4)$	6
12	3	$V=(1, 1, 1, 1, 0, 1)$ $D=(0, 1, 7, 3, 9, 5)$ $T=(0, 1, 4, 2, 5, 4)$	3
13	4	$3 \neq 5$, продолжить	3
14	2	$V=(1, 1, 1, 1, 0, 1)$ $D=(0, 1, 7, 3, 8, 5)$ $T=(0, 1, 4, 2, 3, 4)$	3
15	3	$V=(1, 1, 1, 1, 1, 1)$ $D=(0, 1, 7, 3, 8, 5)$ $T=(0, 1, 4, 2, 3, 4)$	5
16	4	$5=5$, конец	5

В результате выполнения алгоритма Дейкстры при поиске кратчайшего пути от вершины v_1 до вершины v_5 построено дерево кратчайших путей (рис.4.42) с корнем v_1 . Дерево кратчайших путей покрывает все вершины орграфа и определяет кратчайшие пути от вершины v_1 не только до вершины v_5 , но и до всех остальных вершин орграфа. Длина кратчайшего пути от вершины v_1 до вершины v_i указана в скобках рядом с вершиной v_i на рис.4.42.

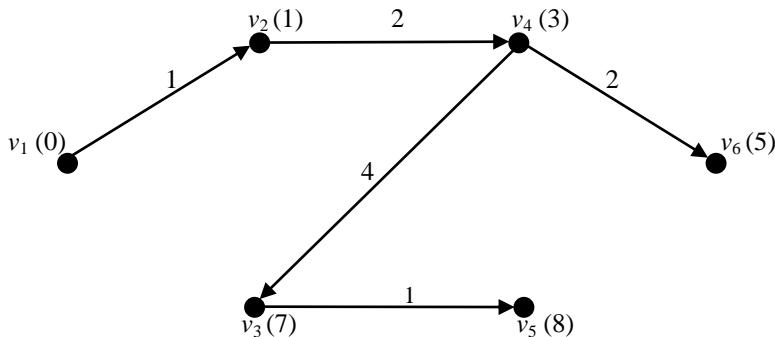


Рис.4.42. Дерево кратчайших путей

4.14. Кратчайшие пути между каждой парой вершин в орграфе

Кратчайшие пути и их длины между каждой парой вершин во взвешенном орграфе будем хранить в матрице W , в которой элемент W_{ij} состоит из двух частей:

$W_{ij}.d$ — длина кратчайшего пути от вершины i до вершины j ;

$W_{ij}.t$ — вершина, предшествующая вершине j на кратчайшем пути от вершины i до вершины j .

Сформировать матрицу W по матрице весов M взвешенного орграфа можно различными способами.

1. Для нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе применим алгоритм Дейкстры. В алгоритме Дейкстры (см. выше) изменим п.4:

4. Проверка на завершение.

Если $V' = V$ или $d(x_j) = \infty$, то конец алгоритма,
иначе перейти к п.2.

В результате такого изменения после выполнения алгоритма получим кратчайшие пути (массив T) и их длины (массив D) от исходной вершины до всех достижимых из нее вершин орграфа.

Теперь матрицу W можно получить следующим образом:

1. Для всех i от 1 до $|V|$ выполнить п.2.

2. Сформировать i -ю строку матрицы W :

2.1. Выполнить алгоритм Дейкстры, считая вершину i исходной.

2.2. Для всех j от 1 до $|V|$ выполнить: $W_{ij}.d := D_j$, $W_{ij}.t := T_j$.

2. Метод Шимбелла.

Матрица весов M взвешенного орграфа определяет длины кратчайших путей между каждой парой вершин орграфа, состоящих из одной дуги.

Для того чтобы получить матрицу длин кратчайших путей между каждой парой вершин орграфа, состоящих из двух дуг, возведем матрицу M в квадрат, но при этом заменим операцию умножения $(a \cdot b)$ операцией сложения $(a + b)$, а операцию сложения $(a + b)$ — выбором минимального элемента $(\min(a, b))$. Матрица M^k будет содержать длины кратчайших путей, состоящих из k дуг. Матрица $M + M^2 + \dots + M^k$ содержит длины кратчайших путей, состоящих не более чем из k дуг (здесь операция сложения также заменена выбором минимального элемента). Учитывая, что кратчайший путь может содержать не более чем $n - 1$ дуг, где n — количество вершин в орграфе, длины кратчайших путей между каждой парой вершин орграфа можно вычислить по формуле: $M + M^2 + \dots + M^{n-1}$.

Чтобы найти не только длины кратчайших путей между каждой парой вершин орграфа, но и сами пути, модифицируем матрицу весов M : примем, что элемент M_{ij} состоит из двух частей:

$$M_{ij}.d — \text{длина дуги из вершины } i \text{ в вершину } j;$$

$$M_{ij}.t = \begin{cases} i, & \text{если есть дуга из вершины } i \text{ в вершину } j; \\ 0, & \text{если } i = j; \\ -1, & \text{если нет дуги из вершины } i \text{ в вершину } j. \end{cases}$$

Матрицу W кратчайших путей и их длин между каждой парой вершин орграфа можно вычислить по формуле:

$$W = M + M^2 + \dots + M^{n-1},$$

при этом результатом операции умножения величин a и b будет величина c , которая определяется: $c.d = a.d + b.d,$

$$c.t = b.t;$$

а результатом операции сложения величин a и b будет величина c , которая определяется:

$$c = \begin{cases} a, & \text{если } a.d < b.d; \\ b, & \text{иначе.} \end{cases}$$

3. Алгоритм Флойда.

Алгоритм Флойда схож с алгоритмом Уоршелла нахождения транзитивного замыкания. В этом алгоритме последовательно рассматривают ся вершины орграфа и, если существует путь в две дуги от некоторой вершины x до вершины y через рассматриваемую вершину z и его длина меньше длины пути между вершинами x и y через рассмотренные ранее вершины, то в качестве кратчайшего пути из x в y нужно принять путь через вершину z . В начальный момент, когда ни одна из вершин графа не рассмотрена, информацию о кратчайших путях и их длинах между каждой парой вершин орграфа представляет модифицированная матрица весов M (см. выше метод Шимбелла). Алгоритм Флойда представлен блок-схемой на рис.4.43.

Алгоритм 4.14 нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе.

Вход: M — модифицированная матрица весов;

N — количество вершин в орграфе.

Выход: W — матрица кратчайших путей и их длин между каждой парой вершин орграфа.

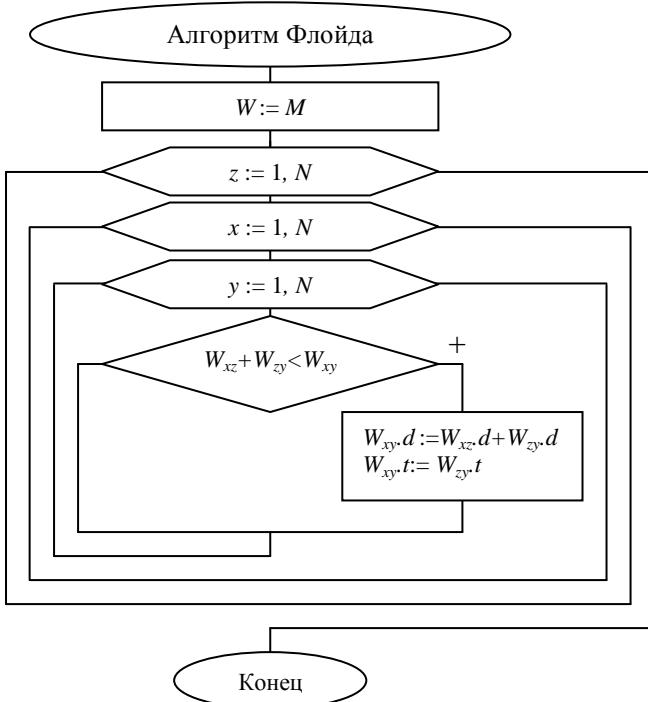


Рис.4.43. Алгоритм Флойда нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе

4.15. Центры и медианы во взвешенном орграфе

Определим для каждой вершины v_i взвешенного орграфа число $s_o(v_i)$ как максимум среди кратчайших расстояний от вершины v_i до каждой вершины орграфа, число $s_t(v_i)$ как максимум среди кратчайших расстояний до вершины v_i от каждой вершины орграфа и число $s_{ot}(v_i)$ как сумму чисел $s_o(v_i)$ и $s_t(v_i)$: $s_{ot}(v_i) = s_o(v_i) + s_t(v_i)$. Число $s_o(v_i)$ называется *числом внешнего разделения* вершины v_i , число $s_t(v_i)$ — *числом внутреннего разделения* вершины v_i и число $s_{ot}(v_i)$ — *числом внешне-внутреннего разделения* вершины v_i .

Вершина с минимальным числом внешнего разделения называется *внешним центром*, вершина с минимальным числом внутреннего разделения называется *внутренним центром* и вершина с минимальным числом внешне-внутреннего разделения называется *внешне-внутренним центром*.

Определим для каждой вершины v_i взвешенного орграфа число $f_o(v_i)$ как сумму кратчайших расстояний от вершины v_i до каждой вершины орграфа, число $f_t(v_i)$ как сумму среди кратчайших расстояний до вершины v_i от каждой вершины орграфа и число $f_{ot}(v_i)$ как сумму чисел $f_o(v_i)$ и $f_t(v_i)$: $f_{ot}(v_i) = f_o(v_i) + f_t(v_i)$. Число $f_o(v_i)$ называется *внешним передаточным числом* вершины v_i , число $f_t(v_i)$ — *внутренним передаточным числом* вершины v_i и число $f_{ot}(v_i)$ — *внешне-внутренним передаточным числом* вершины v_i .

Вершина с минимальным внешним передаточным числом называется *внешней медианой*, вершина с минимальным внутренним передаточным числом называется *внутренней медианой* и вершина с минимальным внешне-внутренним передаточным числом называется *внешне-внутренней медианой*.

На рис.4.44 представлен взвешенный орграф, матрица D кратчайших расстояний, числа внешнего s_o , внутреннего s_t и внешне-внутреннего s_{ot} разделения, а также внешние f_o , внутренние f_t и внешне-внутренние f_{ot} передаточные числа вершин орграфа. В этом орграфе вершины 2 и 5 являются внешними центрами, вершины 3 и 5 — внутренними центрами, вершина 5 — внешне-внутренним центром, вершины 1 и 2 — внешними медианами, вершина 3 — внутренней медианой и вершины 1, 2 и 6 — внешне-внутренней медианой.

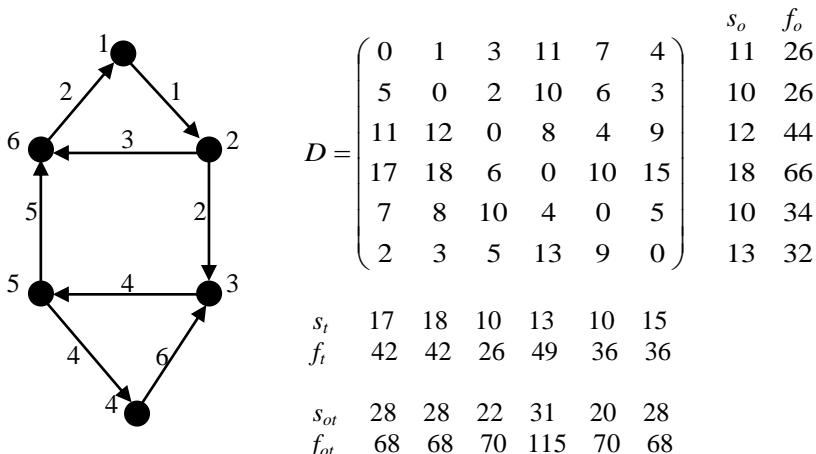


Рис.4.44. Определение центров и медиан орграфа

Рассмотрим задачи, решение которых сводится к нахождению центров и медиан взвешенного орграфа.

Задача 4.7. Несколько жилых районов связаны дорожной сетью. Необходимо разместить в одном из них пожарное депо так, чтобы самый отдаленный пункт находился от него на минимально возможном расстоянии.

Решение. Построим взвешенный орграф, в котором вершины соответствуют районам, дуги соответствуют дорогам между районами, вес дуги — длина дороги. Пожарное депо можно разместить в районе, который соответствует вершине орграфа, являющейся внешним центром.

Задача 4.8. Несколько жилых районов связаны дорожной сетью. Необходимо разместить в одном из них больницу так, чтобы доставка пациента в больницу машиной скорой помощи из самого отдаленного района занимала минимально возможное время.

Решение. Построим взвешенный орграф, в котором вершины соответствуют районам, дуги соответствуют дорогам между районами, вес дуги — время проезда машины скорой помощи по соответствующей дороге. Больницу можно разместить в районе, который соответствует вершине орграфа, являющейся внешне-внутренним центром.

Задача 4.9. Потребители обслуживаются товарами со склада. Потребитель посыпает машину на склад, она загружается товаром и возвращается. Необходимо разместить склад на территории одного из потребителей так, чтобы общее расстояние, проходимое транспортом, было бы минимально возможным.

Решение. Построим взвешенный орграф, в котором вершины соответствуют потребителям, дуги соответствуют дорогам между потребителями, вес дуги — длина дороги. Склад можно разместить на территории потребителя, который соответствует вершине орграфа, являющейся внешне-внутренней медианой.

Задача 4.10. Электросеть должна охватывать заданные районы. Необходимо разместить в одном из них подстанцию и протянуть линии электропередач от подстанции к каждому району так, чтобы суммарная длина проводников была минимально возможной.

Решение. Построим полный взвешенный граф, в котором вершины соответствуют районам, а вес ребра — расстоянию между соответствующими районами. Подстанцию можно разместить в районе, который соответствует вершине орграфа, являющейся медианой.

4.16. Клики и независимые множества

Множество вершин M графа $G = (V, E)$ называется *кликой*, если любые две вершины из этого множества смежны. Подграф графа G , построенный на множестве вершин M , являющимся кликой, представляет собой *полный подграф*.

Обычно в графе можно найти несколько клик, например, каждая пара смежных вершин является кликой. Клика называется *максимальной*, если она не является собственным подмножеством никакой другой клики. В максимальную клику нельзя добавить ни одной вершины графа так, чтобы полученное множество было кликой. Максимальных клик в графе может быть несколько. Различные максимальные клики могут не пересекаться или находиться в общем положении, могут содержать одинаковое или различное число вершин. В графе, диаграмма которого представлена на рис.4.45, максимальных клик шесть: $\{1,2,5\}$, $\{2,3,6,7\}$, $\{3,4,7\}$, $\{5,6,2\}$, $\{5,8\}$, $\{7,8\}$.

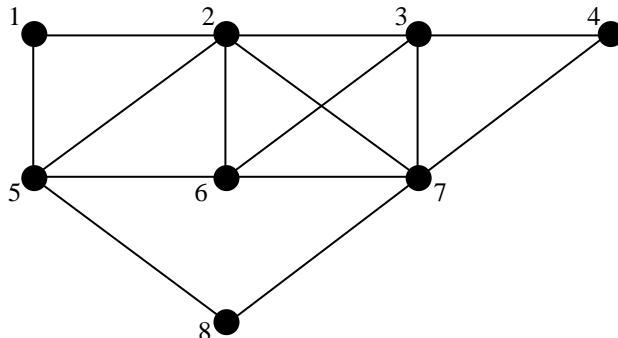


Рис.4.45. Диаграмма графа

В полном графе есть только одна максимальная клика, состоящая из всех вершин графа. Наибольшее количество максимальных клик имеют графы Муна — Мозера. Граф Муна — Мозера с $3n$ вершинами строится следующим образом. Множество вершин разбивается на n непересекающихся классов по три вершины в каждом. Каждая вершина класса K_i смежна только со всеми вершинами, не принадлежащими классу K_i . Взяв в таком графе по одной вершине из каждого класса, получим максимальную клику. Учитывая, что каждый класс состоит из трех вершин, количество максимальных клик в графе Муна — Мозера с $3n$ вершинами равно 3^n . На рис.4.46 представлена диаграмма графа Муна — Мозера с двенадцатью вершинами. Множество вершин разбито на четыре класса:

$\{1,2,3\}$, $\{4,5,6\}$, $\{7,8,9\}$, $\{10,11,12\}$. Выбирая из каждого класса по одной вершине, получим максимальную клику. Всего в этом графе 81 максимальная клика.

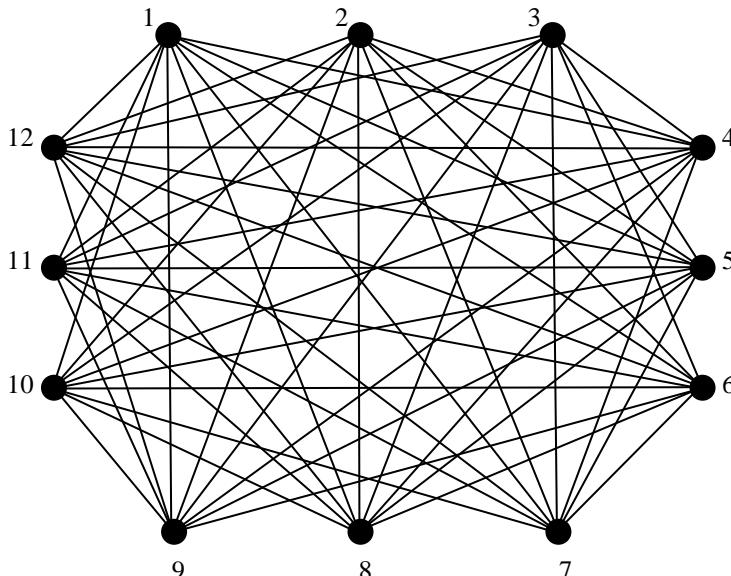


Рис.4.46. Диаграмма графа Муна — Мозера

Найти одну из максимальных клик можно по следующему алгоритму.

Алгоритм 4.15 поиска одной из максимальных клик.

Вход: граф $G = (V, E)$.

Выход: M — максимальная клика.

1. $M := \{x\}, x \in V$;
2. $V' := V - \Gamma(x)$;
3. Пока $V' \neq \emptyset$ выполнять п.4;
4. $M := M \cup \{x\}, x \in V'; V' := V' - \overline{\Gamma(x)}$;
5. M — максимальная клика;
6. Конец.

В этом алгоритме на первом шаге в множество M включается одна из вершин графа. На втором шаге определяется множество вершин V' , элементы которого могут быть включены в формируемую клику M . На четвертом шаге вершина из множества V' добавляется в M и V' пересчитывается.

Все максимальные клики можно найти, используя метод поиска с возвращением.

Алгоритм 4.16 (рис.4.47) поиска всех максимальных клик.

Вход: граф $G = (V, E)$;

M — клика;

V' — множество вершин, смежных с каждой вершиной множества M .

Выход: последовательность максимальных клик.

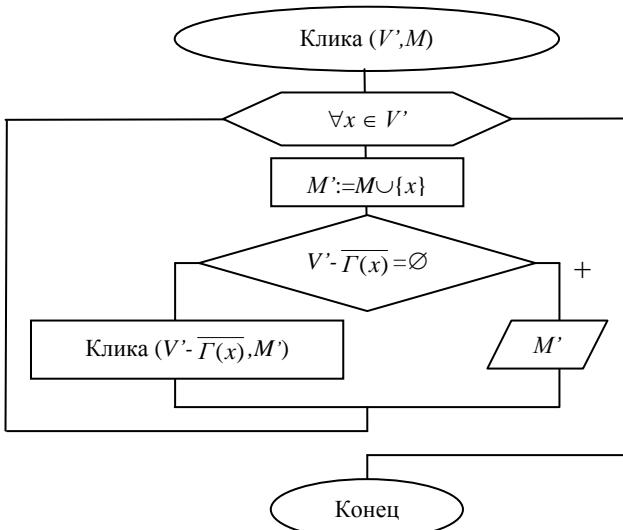


Рис.4.47. Рекурсивный алгоритм получения всех максимальных клик

Этот алгоритм имеет огромный недостаток: каждая n -элементная максимальная клика будет повторяться $n!$ раз. Процесс поиска всех максимальных клик в графе (рис.4.48) можно представить в виде дерева (рис.4.49), в котором корень дерева соответствует исходной ситуации (ни одна вершина не включена в клику), а листья — максимальным кликам. В этом дереве двухэлементные максимальные клики $\{1,2\}$ и $\{2,5\}$ повторяются по 2 раза, а трехэлементная клика $\{2,3,4\}$ — 6 раз.

Наиболее эффективным алгоритмом получения всех максимальных клик, основанном на методе поиска с возвращением, является алгоритм Брана — Кербоша. В этом алгоритме шаг назад выполняется в том случае, если продолжение поиска решения может привести только к получению максимальных клик, которые уже были получены ранее. В результате этого дерево поиска, а следовательно, и время выполнения алгоритма, значительно сокращается.

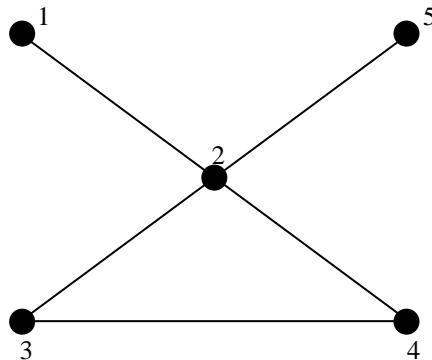


Рис.4.48. Диаграмма графа

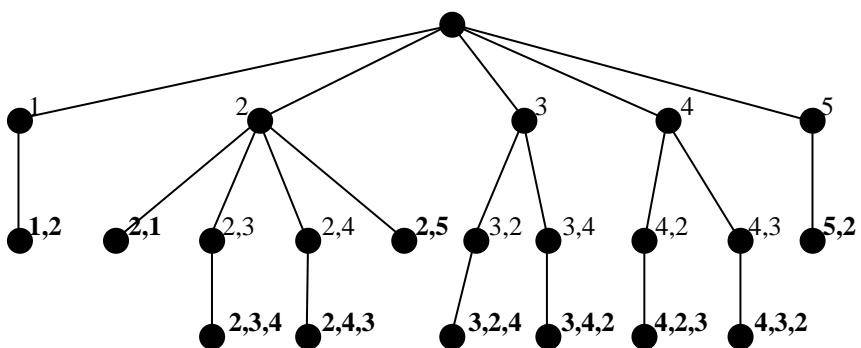


Рис.4.49. Дерево поиска всех максимальных клик

На каждом шаге выполнения алгоритма Брана — Кербоша используются три множества:

- 1) M — формируемая клика;
- 2) K — множество кандидатов, т.е. вершин, каждая из которых может быть добавлена в формируемую клику M ;
- 3) P — множество просмотренных вершин, каждая из которых не может быть добавлена в текущую клику, так как уже добавлялась ранее.

Алгоритм 4.17 (рис.4.50) Брана — Кербоша поиска всех максимальных клик.

Вход: граф $G = (V, E)$;

Выход: последовательность максимальных клик.

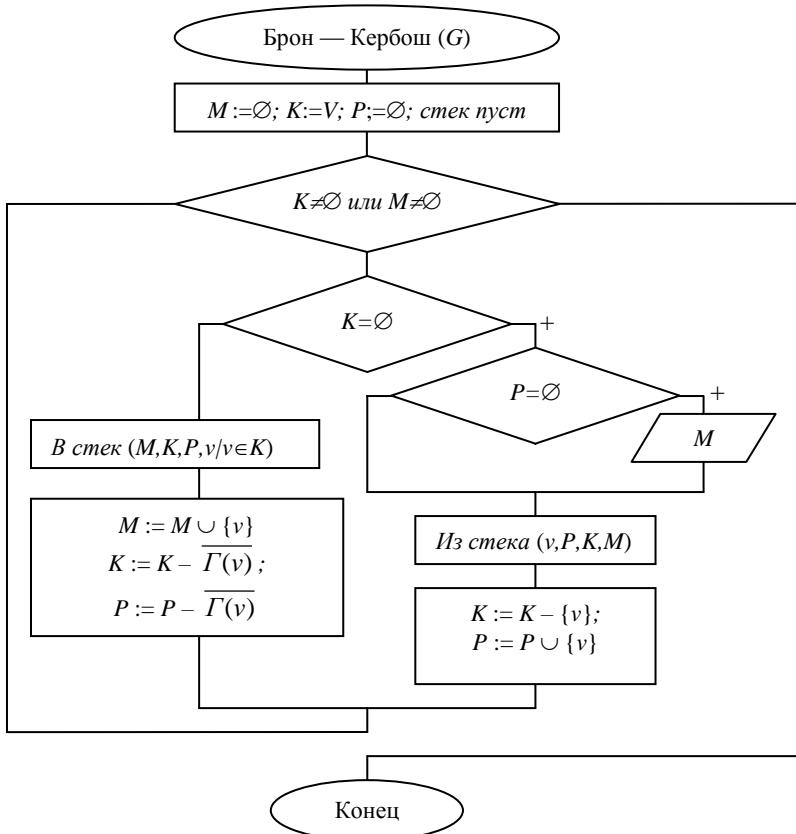


Рис.4.50. Блок-схема алгоритма Брана — Кербоша

Процесс выполнения алгоритма Брана — Кербоша при поиске всех максимальных клик в графе, диаграмма которого изображена на рис.4.48, можно представить в виде дерева (рис.4.51).

Максимальная клика, содержащая наибольшее количество вершин по отношению ко всем другим максимальным кликам, называется *наибольшей*. Для поиска наибольшей клики можно перебрать все максимальные клики, используя алгоритм Брана — Кербоша, и выбрать среди них клику наибольшей мощности. Учитывая, что количество максимальных клик в графе с n вершинами может достигать $3^{n/3}$, этот подход не является эффективным. Эффективный алгоритм поиска наибольшей клики еще не найден.

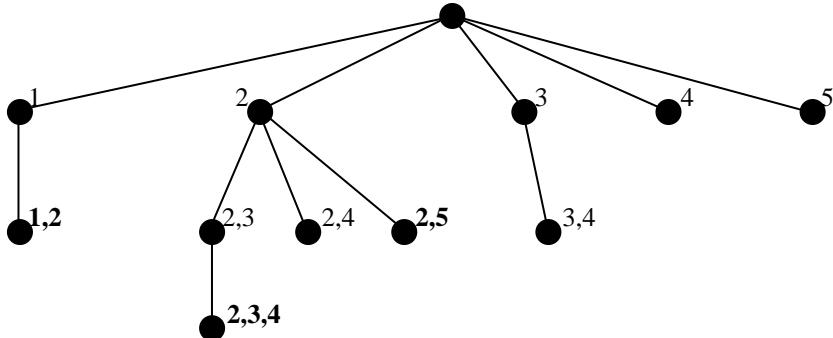


Рис.4.51. Дерево поиска всех максимальных клик по алгоритму Брана — Кербоша

Для поиска максимальной клики, близкой по мощности к наибольшей, применяют эвристические алгоритмы, которые позволяют быстро получить решение. В большинстве случаев это решение представляет собой наибольшую клику, а иногда — максимальную клику меньшей мощности, чем наибольшая. Рассмотрим три алгоритма поиска максимальной клики, близкой по мощности к наибольшей.

В первом алгоритме сначала в множество M , в котором по окончании выполнения алгоритма будет находиться максимальная клика, включаются все вершины графа. Затем из этого множества последовательно исключаются вершины, которые не смежны с наибольшим числом вершин из множества M . Исключения вершин продолжается до тех пор, пока все вершины в множестве M будут попарно смежны. Это и будет максимальная клика, близкая по мощности к наибольшей.

Алгоритм 4.18 поиска максимальной клики, близкой по мощности к наибольшей.

Вход: граф $G = (V, E)$.

Выход: M — максимальная клика, близкая по мощности к наибольшей.

1. Определить $G' = (V', E')$, где $V' = V$, $E' = E$.
2. $M := V'$.
3. Вычислить степени вершин в графе G' .
Пусть x — вершина, степень которой минимальна,
 $d(x)$ — степень вершины x .
4. Пока $d(x) < |V'| - 1$ выполняять:
 - 4.1. $M := M - \{x\}$.
 - 4.2. Удалить из графа G' вершину x с инцидентными ребрами.
 - 4.3. Выполнить п.3.
5. Конец алгоритма.

Во втором алгоритме в исходном состоянии множество M пусто и на каждом шаге добавляется в него вершина наибольшей степени, смежная всем вершинам этого множества. Если на некотором шаге не найдется вершины, которую можно добавить в множество M , то множество M содержит максимальную клику.

Алгоритм 4.19 поиска максимальной клики, близкой по мощности к наибольшей.

Вход: граф $G = (V, E)$.

Выход: M — максимальная клика, близкая по мощности к наибольшей.

1. Определить $G' = (V', E')$, где $V' = V$, $E' = E$.
2. $M := \emptyset$.
3. Пока $V' \neq \emptyset$ выполнять:
 - 3.1. Вычислить степени вершин в графе G' .
Пусть x — вершина, степень которой максимальна.
 - 3.2. $M := M \cup \{x\}$.
 - 3.3. Вершины $\overline{G(x)}$ удалить из графа G' с инцидентными им ребрами.
4. Конец алгоритма.

В третьем алгоритме для каждой пары смежных вершин (каждого ребра) $e = \{v_i, v_j\}$ вычисляется характеристика $h(e) = \min(d(v_i), d(v_j))$, где $d(v_i)$ и $d(v_j)$ — степени концевых вершин v_i и v_j ребра e . Вероятно, пара вершин, имеющая большую характеристику, может быть включена в максимальную клику большей мощности, чем пара вершин, имеющая меньшую характеристику. Максимальная клика, близкая по мощности к наибольшей, строится по алгоритму 4.19, в котором во втором пункте в множество M включается пара вершин с наибольшей характеристикой и вершины, не смежные этим вершинам, удаляются с инцидентными им ребрами из графа G' . Пары вершин, характеристики которых меньше мощности полученной клики не могут быть включены в клику большей мощности, поэтому такие пары вершин исключаются из дальнейшего рассмотрения. Из оставшихся пар смежных вершин снова выбирается пара с наибольшей характеристикой и относительно нее строится новая максимальная клика, возможно, большей мощности. Процесс заканчивается когда все пары смежных вершин будут рассмотрены. Этот алгоритм позволяет быстро находить наибольшую клику в графах, в которых никакой полный подграф не является подмножеством объединения всех остальных полных подграфов.

Алгоритм 4.20 поиска максимальной клики, близкой по мощности к наибольшей.

Вход: граф $G = (V, E)$.

Выход: M — максимальная клика, близкая по мощности к наибольшей.

1. Построить последовательность P пар смежных вершин, упорядоченных по неубыванию характеристик.
2. $M := \emptyset$.
3. Пока последовательность P не пуста, выполнять:
 - 3.1. Найти максимальную клику M' , содержащую первую пару $\{v_i, v_j\}$ вершин из последовательности P по алгоритму 4.19, в котором
 2. $M := \emptyset$
 заменить на
 - 3.2. $M := \{v_i, v_j\}$. Вершины, не смежные с v_i и v_j , удалить из графа G' с инцидентными им ребрами.
 - 3.3. Исключить из последовательности P пары вершин, характеристика которых меньше $|M|$.
4. Конец алгоритма.

Множество вершин M графа $G = (V, E)$ называется *независимым*, если никакие две вершины из этого множества не смежны. Независимое множество является некоторой противоположностью клики.

Граф $\bar{G} = (V, E')$ называется *дополнением* графа $G = (V, E)$, если в нем две вершин смежны только в том случае, если эти вершины не смежны в графе G . Независимое множество вершин графа $G = (V, E)$ представляет собой клику в графе $\bar{G} = (V, E')$ и клика графа $G = (V, E)$ представляет собой независимое множество в графе $\bar{G} = (V, E')$.

Независимое множество называется *максимальным*, если оно не является собственным подмножеством ни какого другого независимого множества. Максимальное независимое множество, содержащее наибольшее количество вершин по отношению ко всем другим максимальным независимым множествам, называется *наибольшим*. Для поиска максимальных и наибольших независимых множеств в графе G могут применяться рассмотренные выше алгоритмы поиска максимальных и наибольших клик в графе \bar{G} .

Рассмотрим задачи, решение которых сводится к нахождению наибольшего независимого множества.

Задача 4.11. Передатчик может передавать сигналы приемнику из конечного множества $V = \{v_1, v_2, \dots, v_n\}$. При передаче возникают искажения сигналов и приемник может перепутать некоторые сигналы, какие именно — известно. Для кодирования передаваемых сообщений необходимо найти наибольшее количество сигналов из множества V , которые приемник не перепутает.

Решение. Построим граф, вершины которого соответствуют сигналам из множества V , пара вершин образует ребро, если соответствующие им сигналы приемник может перепутать. Наибольшее по мощности множество сигналов, которое можно использовать для кодирования сообщений, соответствует наибольшему независимому множеству.

Задача 4.12. Имеется n проектов, которые должны быть выполнены. Для выполнения каждого проекта требуется определенное подмножество ресурсов. Один и тот же ресурс не может быть использован одновременно в выполнении различных проектов. Требуется выбрать максимальное подмножество проектов, которые могут быть выполнены одновременно.

Решение. Построим граф, вершины которого соответствуют проектам, пара вершин образует ребро, если для выполнения соответствующих им проектов требуется хотя бы один общий ресурс. Такие проекты нельзя выполнять одновременно. Максимальное подмножество проектов, которые могут быть выполнены одновременно, соответствует наибольшему независимому множеству.

Задача 4.13. Имеется n проектов, которые должны быть выполнены. Для выполнения каждого проекта требуется определенное подмножество ресурсов. Один и тот же ресурс не может быть использован одновременно в выполнении различных проектов. Известна стоимость каждого проекта. Требуется выбрать подмножество проектов с максимальной суммарной стоимостью, которые могут быть выполнены одновременно.

Решение. Построим граф, вершины которого соответствуют проектам, пара вершин образует ребро, если соответствующие им проекты нельзя выполнять одновременно. Найдем все максимальные независимые множества. Определим суммарную стоимость подмножеств проектов, соответствующих максимальным независимым множествам и выберем среди них подмножество с максимальной суммарной стоимостью.

4.17. Раскраска графа

Пусть задан граф $G = (V, E)$ и натуральное число k . Вершинной k -раскраской графа называется функция, ставящая в соответствие каждой вершине из множества V натуральное число (цвет) из множества $\{1.. k\}$. Другими словами, раскраска задает цвет каждой вершине графа. Раскраска называется *правильной*, если никакие две смежные вершины не окрашены в один цвет. В дальнейшем под раскраской будем понимать только правильную раскраску. Граф называется k -раскрашиваемым, если для него существует правильная k -раскраска. Хроматическим числом графа называется минимальное число цветов, требующееся для правильной раскраски графа. Правильная раскраска графа в хроматическое число цветов называется *минимальной*.

Очевидны следующие факты.

1. Хроматическое число графа $G = (V, E)$, в котором $E = \emptyset$, равно 1.
2. Хроматическое число полного графа $G = (V, E)$ равно $n = |V|$.
3. Если мощность наибольшей клики графа $G = (V, E)$ равна n , то его хроматическое число не меньше n .
4. Множество вершин графа $G = (V, E)$, окрашенных в один цвет является независимым множеством.

Пусть требуется получить все раскраски графа $G = (V, E)$ не более чем в k цветов. Раскраску будем хранить в массиве m , m_i — цвет i -й вершины графа. Если i -я вершина графа не окрашена, то $m_i = 0$. Обозначим через $Color(\Gamma(i))$ множество цветов, которыми окрашены вершины, смежные i -й вершине графа. Тогда цвет, в который может быть окрашена i -я вершина, принадлежит множеству $\{1.. k\} - Color(\Gamma(i))$.

Получить все раскраски графа не более чем в k цветов можно, используя метод поиска с возвращением. В исходном состоянии все вершины графа не окрашены и все элементы массива m равны нулю. Вершины графа будем окрашивать последовательно, начиная с первой. Окраску i -й вершины опишем рекурсивным алгоритмом 4.21, блок-схема которого представлена на рис.4.52. В цикле перебираются цвета, в которые может быть окрашена i -я вершина. Если окрашена последняя n -я вершина ($n = |V|$), то раскраска получена и выводим ее, иначе окрашиваем следующую $i + 1$ -ю вершину по алгоритму 4.21. Когда выполняется шаг назад, к окраске $i - 1$ -й вершине, i -я вершина становится неокрашенной, поэтому процедура имеет дополнительный параметр m .

Алгоритм 4.21 получения всех раскрасок графа $G = (V, E)$ не более чем в k цветов.

Вход: i — окрашиваемая вершина;

m — текущая раскраска.

Выход: последовательность всех раскрасок графа в k цветов, если граф G k -раскрашиваемый.

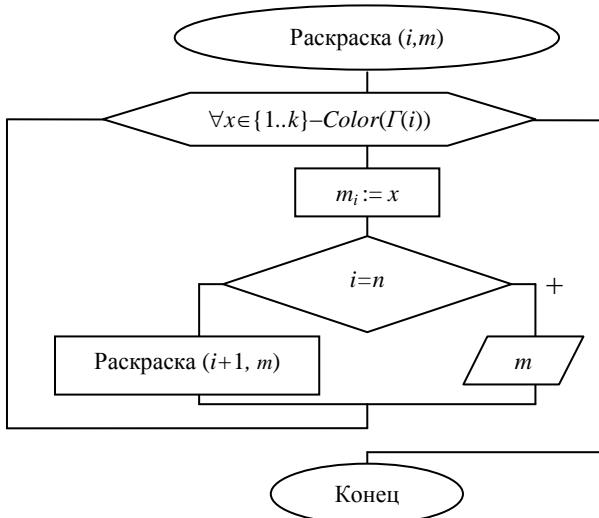


Рис.4.52. Рекурсивный алгоритм получения всех раскрасок графа в k цветов

Пусть требуется получить одну раскраску графа $G = (V, E)$ не более чем в k цветов. Для решения этой задачи можно использовать алгоритм 4.21 с небольшой модификацией: после получения первой раскраски закончить алгоритм. Такая модификация позволит получить решение, но для повышения эффективности можно учитывать следующее:

1. i -ю вершину ($i < k$) можно окрашивать в цвет, не превышающий i .
2. В первую очередь окрашивать вершины, принадлежащие наибольшей клике.
3. Остальные вершины окрашивать в порядке невозрастания степеней.

Учитывая вышесказанное, получим более эффективный алгоритм 4.22 получения одной раскраски графа $G = (V, E)$ не более чем в k цветов. Отметим, что если граф не является k -раскрашиваемым, то раскраска в k цветов не будет получена, в остальных же случаях будет получена раскраска, близкая к минимальной.

Для получения минимальной раскраски графа можно использовать следующий алгоритм.

Алгоритм 4.23 получения минимальной раскраски графа.

Вход: граф $G = (V, E)$.

Выход: M_{min} — минимальная раскраска.

- 1 Получить раскраску m не более чем в $k = |V|$ цветов по алгоритму 4.22. Присвоить k наибольший цвет в раскраске m .
2. Если раскраска m в k цветов получена, выполнять:
 - 2.1. Сохранить в M_{min} раскраску m .
 - 2.2. $k := k - 1$.
 - 2.3. Получить раскраску m не более чем в k цветов по алгоритму 4.22.
4. Конец алгоритма.

Любой граф может быть раскрашен в число цветов, равное количеству вершин в графе. В п.1 по алгоритму 4.22 получаем раскраску m , близкую к минимальной и сохраняем количество цветов k в ней. В п.2 пытаемся получить раскраску в меньшее число цветов, предварительно запомнив ранее полученную в качестве претендента на минимальную. Минимальной раскраской будет последняя полученная.

Рассмотрим задачи, решение которых сводится к нахождению минимальной раскраски графа.

Задача 4.14. Имеется город G , который связан авиамаршрутами с городами G_1, G_2, \dots, G_n . Маршрут GG_iG обслуживается в интервале времени $[x_i, y_i]$. Требуется определить минимальное количество самолетов, достаточное для обслуживания всех маршрутов.

Решение. Построим граф, множество вершин которого соответствует множеству маршрутов. Две вершины смежны тогда, когда временные интервалы соответствующих маршрутов пересекаются. Хроматическое число этого графа равно минимальному количеству самолетов, достаточному для обслуживания всех маршрутов, а минимальная раскраска определяет распределение самолетов по маршрутам: маршруты, соответствующие вершинам, окрашенным в один цвет, могут обслуживаться одним самолетом.

Задача 4.15. Требуется составить расписание уроков так, чтобы заданное множество занятий было проведено за минимально возможное время.

Решение. Построим граф, множество вершин которого соответствует множеству занятий. Две вершины смежны тогда, когда соответствующие занятия нельзя проводить одновременно (один и тот же преподаватель, кабинет или класс). Минимальная раскраска графа определяет требуемое расписание: на первом уроке проводятся занятия, соответст-

вующие вершинам, окрашенным в первый цвет, на втором — занятия, соответствующие вершинам, окрашенным во второй цвет и т.д.

Задача 4.16. Заданы множества работ и механизмов. Для выполнения работ требуется время (одинаковое для всех работ) и некоторые механизмы. Никакой из механизмов не может быть одновременно занят в нескольких работах. Нужно организовать процесс выполнения всех работ за минимально возможное время.

Решение. Построим граф, множество вершин которого соответствует множеству работ. Две вершины смежны, если для соответствующих работ нужен хотя бы один общий механизм. Получим минимальную раскраску графа. Работы, соответствующие вершинам графа, окрашенным в первый цвет, выполняются одновременно в первую очередь, затем одновременно выполняются работы, соответствующие вершинам, окрашенным во второй цвет и т.д.

Задача 4.17. Граф на рис.4.53 представляет схему электрических соединений; вершины соответствуют клеммам, ребра — прямым металлическим полоскам проводников. Проводники не должны пересекать друг друга, поэтому необходимо их распределить по нескольким параллельным платам, в каждой из которых проводники не пересекались бы. Клеммы доступны на всех платах.

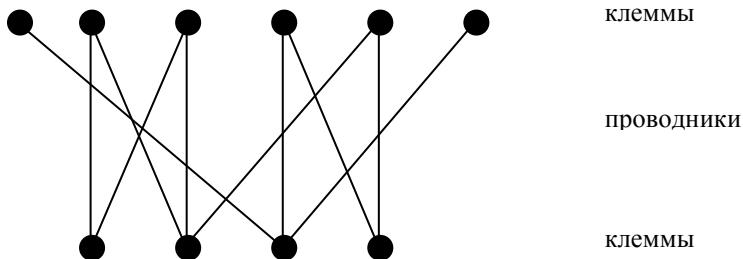


Рис.4.53. Схема электрических соединений

Необходимо определить наименьшее число плат и распределить проводники по платам.

Решение. Построим граф, множество вершин которого соответствует множеству проводников. Две вершины смежны, если соответствующие им проводники пересекаются. Получим минимальную раскраску графа. Проводники, соответствующие вершинам графа, окрашенным в первый цвет, располагаются на первой плате, проводники, соответствующие вершинам, окрашенным во второй цвет, располагаются на второй плате и т.д.

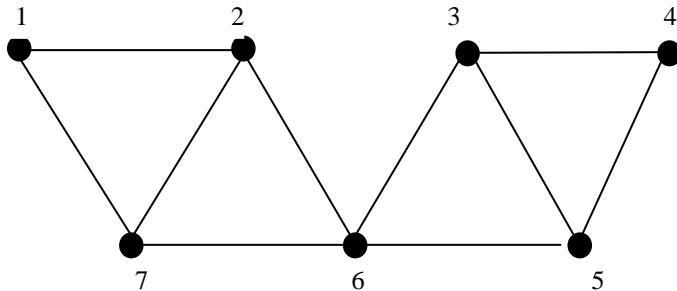
Практическое занятие 4.1

Маршруты

Цель занятия: изучить основные понятия теории графов, способы задания графов, научиться программно реализовывать алгоритмы получения и анализа маршрутов в графах.

Задания

1. Представить графы G_1 и G_2 (см."Варианты заданий", п.а) матрицей смежности, матрицей инцидентности, диаграммой.
2. Определить, являются ли последовательности вершин (см. "Варианты заданий", п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G_1 и G_2 (см."Варианты заданий", п.а).
3. Написать программу, определяющую, является ли заданная последовательность вершин (см. "Варианты заданий", п.б) маршрутом, цепью, простой цепью, циклом, простым циклом в графах G_1 и G_2 (см."Варианты заданий", п.а).
4. Написать программу, получающую все маршруты заданной длины, выходящие из заданной вершины. Использовать программу для получения всех маршрутов заданной длины в графах G_1 и G_2 (см."Варианты заданий", п.а).
5. Написать программу, определяющую количество маршрутов заданной длины между каждой парой вершин графа. Использовать программу для определения количества маршрутов заданной длины между каждой парой вершин в графах G_1 и G_2 (см."Варианты заданий", п.а).
6. Написать программу, определяющую все маршруты заданной длины между заданной парой вершин графа. Использовать программу для определения всех маршрутов заданной длины между заданной парой вершин в графах G_1 и G_2 (см."Варианты заданий", п.а).
7. Написать программу, получающую все простые максимальные цепи, выходящие из заданной вершины графа. Использовать программу для получения всех простых максимальных цепей, выходящих из заданной вершины в графах G_1 и G_2 (см."Варианты заданий", п.а).

Варианты заданий**Вариант 1**а) диаграмма графа G_1 матрица смежности графа G_2

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

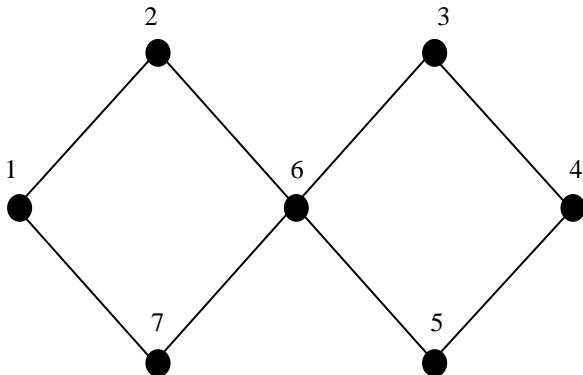
б) последовательности вершин

1. (2, 1, 7, 6, 3, 5)
2. (7, 6, 3, 5, 6, 2, 1)
3. (6, 5, 4, 3, 6)
4. (7, 6, 5, 3, 6, 5)
5. (3, 6, 7, 1, 6, 5, 3)

Вариант 2

а) матрица смежности графа G_1

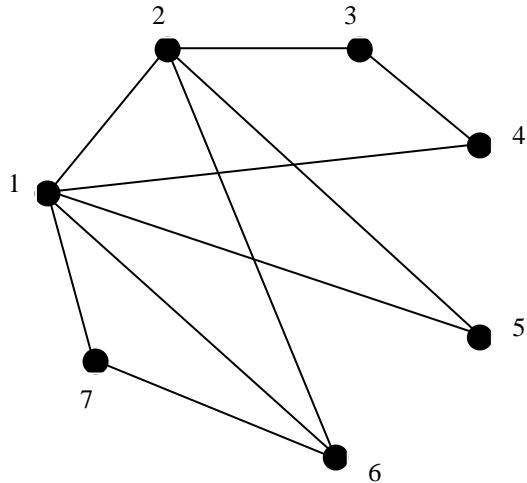
$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

диаграмма графа G_2 

б) последовательности вершин

1. (6, 2, 1, 7)
2. (1, 2, 6, 5, 3, 6, 2)
3. (1, 2, 6, 7, 1)
4. (6, 7, 1, 2, 5, 1, 6)
5. (1, 2, 6, 7, 2, 6, 3)

Вариант 3

а) диаграмма графа G_1 матрица инцидентности графа G_2

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

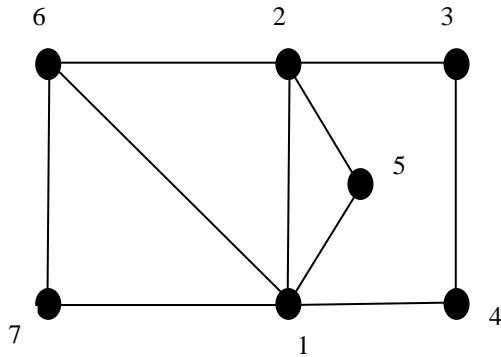
б) последовательности вершин

1. (6, 7, 1, 4, 3, 2)
2. (2, 1, 7, 6, 1, 4)
3. (1, 2, 3, 4, 1)
4. (1, 2, 3, 4, 2, 1)
5. (2, 1, 6, 7, 1, 4, 2)

Вариант 4

а) матрица инцидентности графа G_1

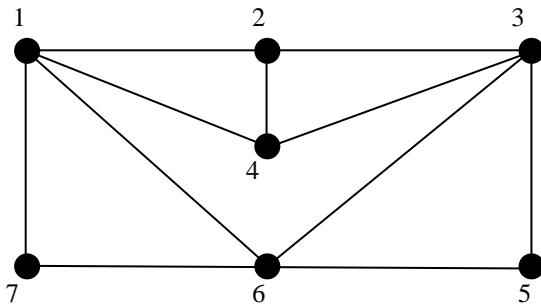
$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

диаграмма графа G_2 

б) последовательности вершин

1. (2, 3, 4, 1, 5)
2. (4, 3, 6, 5, 3, 4, 2)
3. (6, 1, 2, 5, 1, 7)
4. (1, 7, 6, 2, 1)
5. (2, 1, 7, 6, 1, 4, 3, 2)

Вариант 5

а) диаграмма графа G_1 матрица смежности графа G_2

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

б) последовательности вершин

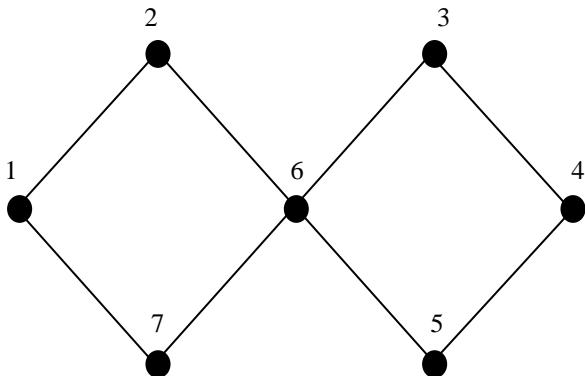
1. (5, 3, 6, 7, 1, 2)
2. (1, 2, 6, 5, 3, 6, 7)
3. (6, 3, 4, 5, 6)
4. (5, 6, 3, 5, 6, 7)
5. (3, 5, 6, 1, 7, 6, 3)

Вариант 6

а) матрица смежности графа G_1

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

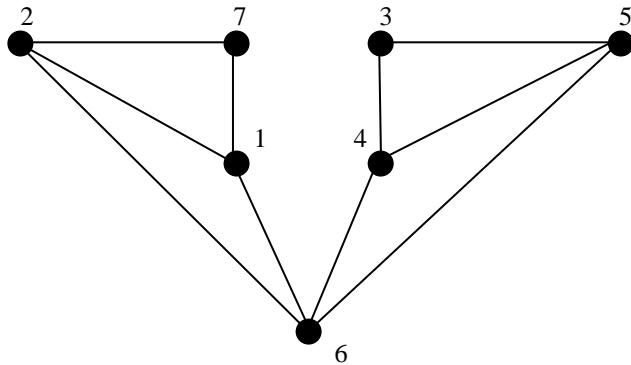
диаграмма графа G_2



б) последовательности вершин

1. (1, 7, 6, 3, 5)
2. (6, 5, 3, 6, 7, 1)
3. (7, 6, 3, 2, 1, 7)
4. (4, 3, 5, 3, 6, 5)
5. (1, 2, 7, 6, 2, 1)

Вариант 7

а) диаграмма графа G_1 матрица инцидентности графа G_2

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

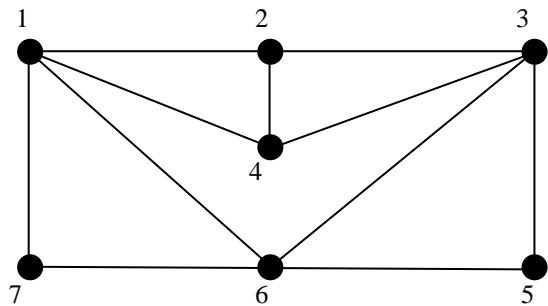
б) последовательности вершин

1. (7, 6, 5, 3, 4)
2. (1, 7, 6, 2, 5, 6)
3. (7, 1, 2, 3, 6, 7)
4. (5, 6, 3, 5, 3, 4)
5. (1, 2, 6, 7, 2, 1)

Вариант 8

а) матрица смежности графа G_1

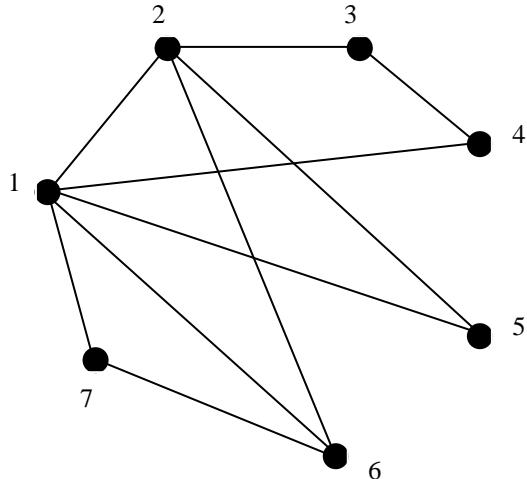
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

диаграмма графа G_2 

б) последовательности вершин

1. (3, 5, 6, 7, 1, 2)
2. (3, 6, 7, 1, 2, 6, 5)
3. (4, 3, 6, 5, 4)
4. (5, 5, 7, 6, 3, 5)
5. (5, 3, 6, 7, 1, 6, 5)

Вариант 9

а) диаграмма графа G_1 матрица инцидентности графа G_2

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

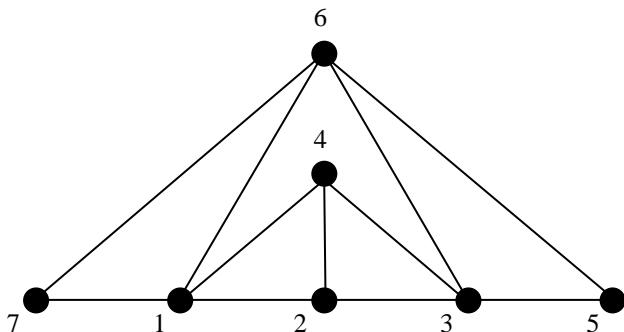
б) последовательности вершин

1. (1, 2, 6, 7)
2. (2, 6, 1, 2, 3, 4)
3. (7, 6, 2, 1, 7)
4. (6, 2, 1, 7, 2, 6)
5. (7, 6, 2, 1, 6, 2, 3)

Вариант 10

а) матрица инцидентности графа G_1

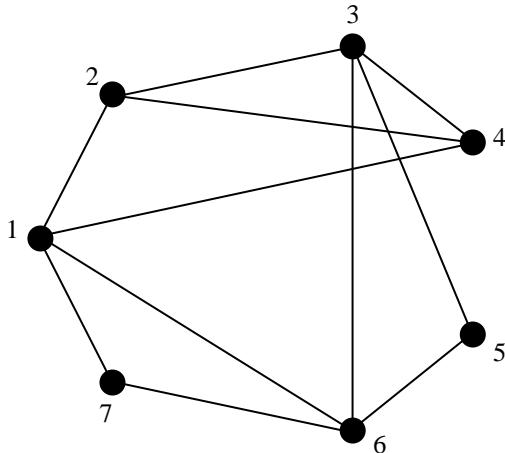
$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

диаграмма графа G_2 

б) последовательности вершин

1. (2, 3, 4, 1, 7, 6)
2. (4, 1, 6, 7, 1, 2)
3. (1, 4, 3, 2, 1)
4. (1, 2, 4, 3, 2, 1)
5. (2, 4, 1, 7, 6, 1, 2)

Вариант 11

а) диаграмма графа G_1 матрица инцидентности графа G_2

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

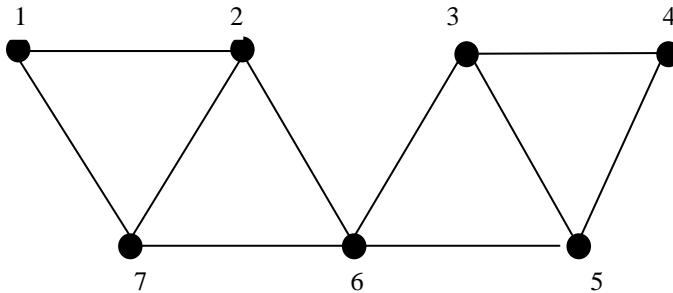
б) последовательности вершин

1. (5, 1, 4, 3, 2)
2. (2, 4, 3, 5, 6, 3, 4)
3. (7, 1, 5, 2, 1, 6)
4. (1, 2, 6, 7, 1)
5. (2, 3, 4, 1, 6, 7, 1, 2)

Вариант 12

а) матрица смежности графа G_1

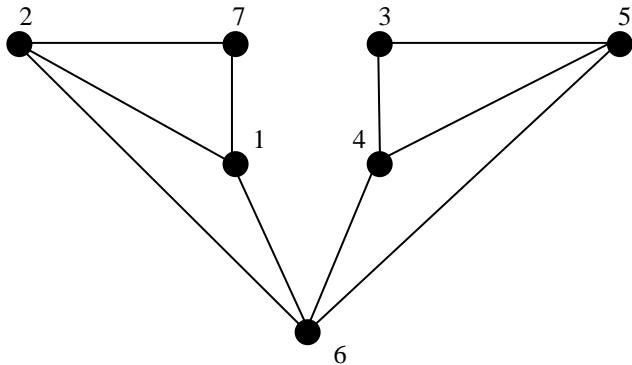
$$\begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

диаграмма графа G_2 

б) последовательности вершин

1. (7, 6, 2, 1)
2. (2, 6, 5, 3, 6, 2, 1)
3. (6, 7, 1, 2, 6)
4. (7, 1, 5, 2, 1, 6, 7)
5. (3, 6, 2, 7, 6, 2, 1)

Вариант 13

а) диаграмма графа G_1 матрица инцидентности графа G_2

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

б) последовательности вершин

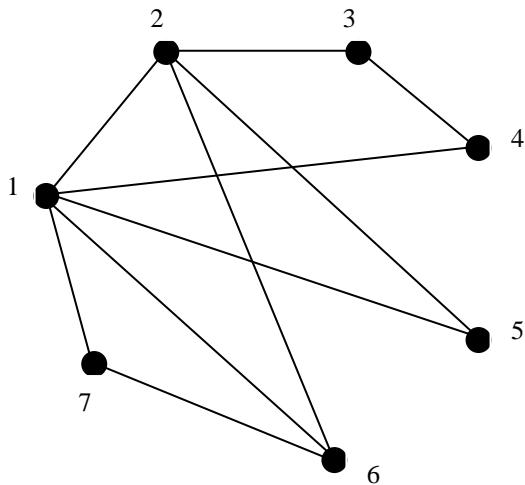
1. (7, 1, 2, 6)
2. (2, 6, 3, 5, 6, 2, 1)
3. (1, 7, 6, 2, 1)
4. (6, 1, 5, 2, 1, 7, 6)
5. (3, 6, 2, 7, 6, 2, 1)

Вариант 14

a) матрица инцидентности графа G_1

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

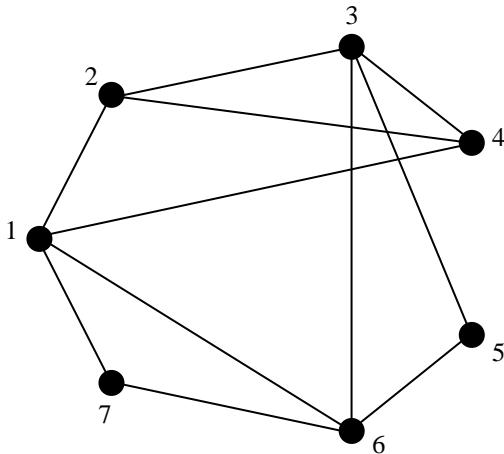
диаграмма графа G_2



б) последовательности вершин

1. (2, 6, 7, 1)
2. (4, 3, 2, 1, 6, 2)
3. (7, 1, 2, 6, 7)
4. (6, 2, 7, 1, 2, 6)
5. (3, 2, 6, 1, 2, 6, 7)

Вариант 15

а) диаграмма графа G_1 матрица инцидентности графа G_2

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

б) последовательности вершин

1. (3, 4, 1, 6, 7)
2. (5, 3, 4, 1, 6, 7, 1, 2)
3. (2, 3, 4, 1, 6, 7, 1, 2)
4. (6, 7, 1, 2, 6)
5. (1, 4, 3, 2, 1, 7, 6, 1)

Практическое занятие 4.2

Циклы

Цель занятия: изучить разновидности циклов в графах, научиться генерировать случайные графы, определять их принадлежность к множеству эйлеровых и гамильтоновых графов, находить все эйлеровы и гамильтоновы циклы в графах.

Задания

1. Разработать и реализовать алгоритм генерации случайного графа, содержащего n вершин и m ребер.
2. Написать программу, которая:
 - а) в течение десяти секунд генерирует случайные графы, содержащие n вершин и m ребер;
 - б) для каждого полученного графа определяет, является ли он эйлеровым или гамильтоновым;
 - в) подсчитывает общее количество сгенерированных графов и количество графов каждого типа.

Результат работы программы представить в виде таблицы (табл. 4.6).

Таблица 4.6

Результат работы программы

Количество вершин	Количество ребер	Количество графов		
		эйлеровых	гамильтоновых	всех
n	n			
n	$n + 1$			
n	$n + 2$			
n	C_n^2			

3. Выполнить программу при $n = 8, 9, 10$ и сделать выводы.
4. Привести пример диаграммы графа, который является эйлеровым, но не гамильтоновым. Найти в нем все эйлеровы циклы.
5. Привести пример диаграммы графа, который является гамильтоновым, но не эйлеровым. Найти в нем все гамильтоновы циклы.
6. Привести пример диаграммы графа, который является эйлеровым и гамильтоновым. Найти в нем все эйлеровы и гамильтоновы циклы.
7. Привести пример диаграммы графа, который не является ни эйлеровым, ни гамильтоновым.

Практическое занятие 4.3

Связность

Цель занятия: изучить алгоритм Краскала построения покрывающего леса, научиться использовать его при решении различных задач.

Задания

1. Реализовать алгоритм Краскала построения покрывающего леса.
2. Используя алгоритм Краскала, разработать и реализовать алгоритм решения задачи (см. варианты заданий).
3. Подобрать тестовые данные. Результат представить в виде диаграммы графа.

Варианты заданий

1. Найти все мосты в связном графе.
2. Найти минимальное множество ребер, удаление которых из связного графа делает его несвязанным.
3. Найти минимальное множество вершин, удаление которых из связного графа разбивает его на три связные компоненты.
4. Получить все покрывающие деревья связного графа $G = (V, E)$, исключая всеми способами $|E| - |V| + 1$ ребер графа.
5. Найти в эйлеровом графе эйлеров цикл, используя алгоритм *Флери*, по которому выбираются и нумеруются непронумерованные ребра графа по следующим правилам:
 1. Произвольно выбрать некоторую вершину v_1 и ребро $\{v_1, v_2\}$, инцидентное вершине v_1 . Этому ребру присвоить номер 1. Перейти в вершину v_2 .
 2. Находясь в вершине v_i , следует не выбирать ребро $\{v_i, v_1\}$, если имеется возможность другого выбора.
 3. Находясь в вершине v_i , следует не выбирать ребро $\{v_i, v_j\}$, которое является мостом, если имеется возможность другого выбора.
 4. После того как в графе будут занумерованы все ребра, образуется эйлеров цикл, порядок нумерации соответствует последовательности обхода ребер.
 6. Найти все множества вершин, исключение которых из связного графа разбивает его на две связные компоненты, причем каждая компонента не содержит изолированных вершин.
 7. Найти все k -элементные множества ребер, исключение которых из связного графа разбивает его на две связные компоненты.

8. Найти все множества ребер, исключение которых из связного графа разбивает его на две связные компоненты, причем каждая компонента не содержит изолированных вершин.
9. Найти максимальное множество ребер, исключение которых из связного графа разбивает его на две связные компоненты.
10. Найти минимальное множество ребер, удаление которых из связного графа разбивает его на три связные компоненты.
11. Найти минимальное множество вершин, удаление которых из связного графа делает его несвязным.
12. Найти все точки сочленения в связном графе.
13. Найти все k -элементные множества вершин, исключение которых из связного графа разбивает его на две связные компоненты.
14. Найти максимальное множество вершин, исключение которых из связного графа разбивает его на две связные компоненты.
15. Найти все максимальные множества ребер, исключение которых из связного графа разбивает его на две связные компоненты.

Практическое занятие 4.4

Анализ алгоритмов построения покрывающего дерева минимальной стоимости

Цель занятия: изучить алгоритмы построения покрывающего дерева минимальной стоимости и выполнить их сравнительный анализ.

Задания

1. Разработать и реализовать алгоритм построения случайного связного взвешенного графа с заданным числом вершин n и ребер m .
2. Реализовать алгоритмы построения оптимального покрывающего дерева:
 1. Сортировка + алгоритм Краскала.
 2. Алгоритм Краскала с выбором очередного ребра минимального веса.
 3. Алгоритм Прима.
3. Разработать и написать программу, которая генерирует 100 случайных связных взвешенных графов с заданным числом вершин n и ребер m , для каждого графа строит покрывающее дерево минимальной стоимости тремя алгоритмами и определяет время выполнения каждого алгоритма. Выполнить программу при $n = 10, 15$ и 20 . Результат для каждого n представить в виде таблицы (табл.4.7).

Таблица 4.7

Время выполнения алгоритмов

Практическое занятие 4.5

Анализ алгоритмов поиска кратчайшего пути во взвешенном орграфе между заданной парой вершин

Цель занятия: изучить алгоритмы поиска кратчайшего пути во взвешенном орграфе между заданной парой вершин и выполнить их сравнительный анализ.

Задания

1. Разработать и реализовать алгоритм построения случайного слабо связного взвешенного орграфа с заданным числом вершин n и дуг m .
 2. Реализовать алгоритмы поиска кратчайшего пути во взвешенном орграфе между заданной парой вершин:
 1. Перебор простых цепей.
 2. Метод ветвей и границ.
 3. Алгоритм Дейкстры.
 3. Разработать и написать программу, которая генерирует 100 случайных слабо связных взвешенных орграфов с заданным числом вершин n и дуг m , для каждого орграфа находит кратчайший путь между каждой парой вершин тремя алгоритмами и определяет время выполнения каждого алгоритма. Выполнить программу при $n = 8, 9$ и 10 . Результат для каждого n представить в виде таблицы (табл.4.8).

Таблица 4.8

Время выполнения алгоритмов

Практическое занятие 4.6

Кратчайшие пути во взвешенном орграфе

Цель занятия: изучить алгоритм Дейкстры нахождения кратчайших путей между вершинами взвешенного орграфа, научиться рационально использовать его при решении различных задач.

Задания

1. Изучить алгоритм Дейкстры нахождения кратчайших путей между вершинами взвешенного орграфа.
2. Используя алгоритм Дейкстры, разработать и реализовать алгоритм решения задачи (см. варианты заданий).
3. Подобрать тестовые данные. Результат представить в виде диаграммы графа.

Варианты заданий

1. Найти множество вершин взвешенного орграфа, до которых длина кратчайшего пути от заданной вершины не превосходит заданной величины. Вывести кратчайшие пути от заданной вершины до каждой из найденного множества и длины путей.
2. Найти вершину взвешенного орграфа, через которую проходит наибольшее число кратчайших путей от заданной вершины до всех остальных. Вывести кратчайшие пути от заданной вершины до каждой вершины орграфа.
3. Найти множество вершин взвешенного орграфа, от которых длина кратчайшего пути до заданной вершины превосходит заданную величину. Вывести кратчайшие пути от каждой из вершины из найденного множества до заданной вершины и длины путей.
4. Найти путь наименьшей длины во взвешенном орграфе от вершины x до вершины y , проходящий через вершину z . Вывести найденный путь и его длину.
5. Найти кратчайший путь во взвешенном орграфе от вершины x до вершины y , проходящий сначала через вершину v , а затем — через вершину w . Вывести найденный путь и его длину.
6. Найти кратчайший путь во взвешенном орграфе от вершины x до вершины y , содержащий в себе вершины v и w (в любом порядке). Вывести найденный путь и его длину.

7. В орграфе заданы вероятности успешного прохождения дуг. Вероятность успешного прохождения пути определяется как произведение вероятностей составляющих его дуг. Найти наиболее надежный (имеющий наибольшую вероятность успешного прохождения) путь от вершины x до вершины y . Вывести путь и вероятность успешного его прохождения.

8. Найти множество вершин взвешенного орграфа, до которых длина кратчайшего пути от первой заданной вершины больше, чем длина кратчайшего пути от второй заданной вершины. Вывести найденное множество вершин, кратчайшие пути от заданных вершин до вершин найденного множества и длины этих путей.

9. Во взвешенном орграфе найти путь между вершинами x и y , в котором длина кратчайшей дуги максимальна (временная метка вершины z — это длина кратчайшей дуги на пути от x до z). Вывести найденный путь и длины дуг этого пути.

10. Во взвешенном орграфе найти простой цикл наименьшей длины с начальной вершиной v . Вывести найденный путь и его длину.

11. Во взвешенном орграфе найти путь наименьшей длины между двумя вершинами, проходящий по заданной дуге. Вывести найденный путь и его длину.

12. Во взвешенном орграфе найти простой цикл наименьшей длины из вершины A , проходящий через вершину B . Вывести найденный цикл и его длину.

13. Во взвешенном орграфе найти путь наименьшей длины между двумя вершинами, проходящий по заданной дуге дважды. Вывести найденный путь и его длину.

14. Определить, существуют ли во взвешенном орграфе две вершины, до которых длины кратчайших путей от заданной вершины одинаковы. Если существуют, то вывести кратчайшие пути от заданной вершины до найденных.

15. Найти во взвешенном орграфе вершину, до которой есть хотя бы два пути наименьшей длины от заданной вершины. Вывести длину кратчайшего пути от заданной вершины до найденной и два пути этой длины между ними.

16. Найти вершину взвешенного орграфа, через которую проходит наибольшее число кратчайших путей до заданной вершины от всех остальных. Вывести кратчайшие пути от каждой вершины орграфа до заданной вершины.

17. Во взвешенном орграфе найти путь наименьшей длины между двумя вершинами, не проходящий по заданной дуге. Вывести найденный путь и его длину.

18. Найти множество вершин взвешенного орграфа, до которых длина кратчайшего пути от первой заданной вершины равна длине кратчайшего пути от второй заданной вершины. Вывести найденное множество вершин, кратчайшие пути от заданных вершин до вершин найденного множества и длины этих путей.
 19. Найти путь наименьшей длины во взвешенном орграфе от вершины x до вершины y , проходящий через вершину z . Вывести найденный путь и его длину.

Практическое занятие 4.7

Анализ алгоритмов поиска кратчайших путей во взвешенном орграфе между каждой парой вершин

Цель занятия: изучить алгоритмы поиска кратчайших путей во взвешенном орграфе между каждой парой вершин и выполнить их сравнительный анализ.

Задания

1. Разработать и реализовать алгоритм построения случайного слабо связного взвешенного орграфа с заданным числом вершин n и дуг m .
 2. Реализовать алгоритмы поиска кратчайших путей во взвешенном орграфе между каждой парой вершин:
 1. Алгоритм Дейкстры.
 2. Алгоритм Шимбелла.
 3. Алгоритм Флойда.
 3. Разработать и написать программу, которая генерирует 100 случайных слабо связных взвешенных орграфов с заданным числом вершин n и дуг m , для каждого орграфа находит кратчайшие пути между каждой парой вершин тремя алгоритмами и определяет время выполнения каждого алгоритма. Выполнить программу при $n = 8, 9$ и 10 . Результат для каждого n представить в виде таблицы (табл.4.9).

Таблица 4.9

Практическое занятие 4.8

Кратчайшие пути между каждой парой вершин во взвешенном орграфе

Цель занятия: изучить алгоритмы нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе, научиться использовать их при решении различных задач.

Задания

1. Изучить алгоритмы нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе.
2. Разработать и реализовать алгоритм решения задачи (см. варианты заданий).
3. Подобрать тестовые данные. Результат представить в виде диаграммы графа.

Варианты заданий

1. Во взвешенном орграфе найти все пары вершин v_i и v_j , такие, что кратчайшее расстояние от v_i до v_j равно кратчайшему расстоянию от v_j до v_i . Вывести кратчайшие пути между найденными парами вершин.
2. Дерево кратчайших путей назовем покрывающим, если все вершины взвешенного орграфа принадлежат дереву. Сумму длин дуг дерева назовем стоимостью. Построить все покрывающие деревья кратчайших путей минимальной стоимости.
3. Во взвешенном орграфе найти все такие вершины v_i , что сумма кратчайших расстояний от всех вершин орграфа до v_i равна сумме кратчайших расстояний от v_i до всех вершин орграфа.
4. Определить, есть ли в заданном взвешенном орграфе вершина, которая является и внешней медианой, и внутренним центром. Привести пример орграфа с такой вершиной.
5. Во взвешенном орграфе найти все такие вершины v_i , что кратчайшее расстояние от самой “удаленной” вершины орграфа до v_i равно кратчайшему расстоянию от v_i до самой “удаленной” от нее вершины.
6. Используя алгоритм нахождения кратчайших расстояний между каждой парой вершин в орграфе определить:
 - а) является ли заданный орграф сильно связным;
 - б) является ли заданный орграф односторонне связным.

7. Найти все внешние, внутренние и внешне-внутренние центры взвешенного орграфа. Привести примеры орграфов, которые имеют более одного центра каждого вида.

8. Найти все пары вершин взвешенного орграфа, кратчайший путь между которыми содержит заданное число дуг.

9. Определить, есть ли в заданном взвешенном орграфе вершина, которая является и внешней, и внутренней медианой. Привести пример орграфа с такой вершиной.

10. Во взвешенном орграфе найти все пары вершин v_i и v_j , такие, что кратчайшее расстояние от v_i до v_j меньше кратчайшего расстояния от v_j до v_i . Вывести кратчайшие пути между найденными парами вершин.

11. Дерево кратчайших путей назовем покрывающим, если все вершины взвешенного орграфа принадлежат дереву. Сумму длин дуг дерева назовем стоимостью. Построить все покрывающие деревья кратчайших путей максимальной стоимости.

12. Во взвешенном орграфе найти все такие вершины v_i , что сумма кратчайших расстояний от всех вершин орграфа до v_i меньше суммы кратчайших расстояний от v_i до всех вершин орграфа.

13. Определить, есть ли в заданном взвешенном орграфе вершина, которая является и внешним центром, и внутренней медианой. Привести пример орграфа с такой вершиной.

14. Во взвешенном орграфе найти все такие вершины v_i , что кратчайшее расстояние от самой “удаленной” вершины орграфа до v_i меньше кратчайшего расстояния от v_i до самой “удаленной” от нее вершины.

15. Найти множество вершин во взвешенном орграфе, внешние и внутренние передаточные числа которых совпадают.

16. Найти все внешние, внутренние и внешне-внутренние медианы взвешенного орграфа. Привести примеры орграфов, которые имеют более одной медианы каждого вида.

17. Найти все пары вершин взвешенного орграфа, кратчайший путь между которыми содержит не более заданного числа дуг.

18. Найти множество вершин во взвешенном орграфе, числа внешнего и внутреннего разделения которых совпадают.

19. Определить, есть ли в заданном взвешенном орграфе вершина, которая является и внешним, и внутренним центром. Привести пример орграфа с такой вершиной.

20. Найти все пары вершин взвешенного орграфа, кратчайший путь между которыми содержит более заданного числа дуг.

Контрольные вопросы и задания

1. Дайте определение графа, орграфа, псевдографа, мультиграфа, гиперграфа, взвешенного графа. Что такое степень вершины, полустепень исхода и полустепень захода? Дайте определение понятиям смежности и инцидентности.
2. Что такое матрица смежности и матрица инцидентности? Предложите различные способы хранения взвешенных графов и орграфов в памяти ЭВМ. Разработайте алгоритмы преобразования одного способа хранения графа в другой.
3. Какие графы называются изоморфными? Сколько существует графов, изоморфных заданному? Разработайте алгоритмы проверки изоморфизма двух графов.
4. Дайте определение маршрута, цепи, простой цепи. Опишите алгоритмы получения всех маршрутов, цепей и простых цепей заданной длины. Как можно подсчитать количество маршрутов заданной длины между каждой парой вершин графа?
5. Что называется расстоянием между заданными вершинами, эксцентризитетом вершины, диаметром и радиусом графа? Какая вершина графа называется переферийной, центральной? Что называется центром графа? Какая цепь называется диаметральной?
6. Дайте определение цикла и простого цикла. Опишите алгоритм получения всех простых циклов в графе.
7. Какой цикл называется гамильтоновым? Как определить, является ли граф гамильтоновым? Какой цикл называется эйлеровым? Как определить, является ли граф эйлеровым? Опишите алгоритмы получения всех гамильтоновых и эйлеровых циклов.
8. Какой граф называется связным?
9. Что такое матрица связности вершин? Как ее вычислить?
10. Что называется мостом, точкой сочленения, разрезом, реберной и вершинной связностью?
11. Дайте определение дереву и лесу. Какими свойствами обладают дерево и лес?
12. Что такое лист и корень дерева? Сколько корней может иметь дерево? Докажите.
13. Сколько различных деревьев можно построить на n вершинах? Докажите.
14. Дайте определение покрывающему дереву и покрывающему лесу. Сколько ребер нужно удалить из связного графа, чтобы получить покрывающее дерево? Сколько ребер нужно удалить из несвязного графа, чтобы получить покрывающий лес?

15. Опишите алгоритм Краскала. Что такое “букет”?
16. Опишите алгоритм формирования всех покрывающих деревьев связного графа.
17. Как определяется стоимость покрывающего дерева взвешенного графа? Опишите алгоритмы построения покрывающего дерева минимальной стоимости.
18. Что называется поиском в орграфе? Опишите алгоритмы поиска в глубину и в ширину.
19. Какие виды связности существуют в орграфе?
20. Дайте определения отношениям достижимости, контрдостижимости и взаимодостижимости вершин орграфа.
21. Что называется компонентой сильной связности орграфа? Опишите алгоритм нахождения компонент сильной связности орграфа.
22. Что называется конденсацией орграфа? Что называется базой и антибазой орграфа?
23. Что называется кратчайшим путем и кратчайшим расстоянием между двумя вершинами во взвешенном орграфе? Опишите алгоритмы нахождения кратчайших путей между заданной парой вершин во взвешенном орграфе.
24. Что такое дерево кратчайших путей? Как его построить?
25. Опишите различные алгоритмы нахождения кратчайших путей между каждой парой вершин во взвешенном орграфе.
26. Что называется числом внешнего, внутреннего и внешне-внутреннего разделения вершины? Что называется внешним, внутренним и внешне-внутренним центром взвешенного орграфа?
27. Что называется внешним, внутренним и внешне-внутренним передаточным числом вершины во взвешенном орграфе? Что называется внешней, внутренней и внешне-внутренней медианой взвешенного орграфа?
28. Дайте определение клики, максимальной клики и наибольшей клики. Опишите алгоритмы поиска всех максимальных клик и наибольшей клики.
29. Дайте определение независимому множеству вершин, максимальному и наибольшему независимому множеству. Установите связь между кликами и независимыми множествами вершин.
30. Что такое раскраска графа, правильная раскраска, хроматическое число, минимальная раскраска?
31. Опишите алгоритм получения всех раскрасок графа в число цветов, не превышающее заданного.
32. Опишите алгоритм получения минимальной раскраски.

5. БУЛЕВЫ ФУНКЦИИ

5.1. Табличные способы задания булевых функций

Функция f , зависящая от n переменных x_1, x_2, \dots, x_n называется *булевой*, или *переключательной*, если функция f и любой из ее аргументов x_i , $i = (1, \dots, n)$ принимают значения только из множества $\{1, 0\}$. Аргументы булевой функции также называются *булевыми*.

Произвольная булева функция может быть задана табличным способом. При таком способе булева функция $f(x_1, \dots, x_n)$ задается *таблицей истинности* (табл. 5.1), в левой части которой представлены все возможные двоичные наборы длины n , а в правой указывается значение функции на этих наборах. Под *двоичным набором* $y = (y_1, y_2, \dots, y_n)$, $y_i \in \{1, 0\}$ понимается совокупность значений аргументов x_1, x_2, \dots, x_n булевой функции f . Двоичный набор имеет длину n , если он представлен n цифрами из множества $\{0, 1\}$.

Иногда двоичные наборы в таблице истинности булевой функции удобно представлять *номерами* наборов. Запишем аргументы x_1, x_2, \dots, x_n в порядке возрастания их индексов. Тогда любой двоичный набор $y = (y_1, y_2, \dots, y_n)$, $y_i \in \{1, 0\}$ можно рассматривать как двоичное число N :

$$N = y_1 \cdot 2^{n-1} + y_2 \cdot 2^{n-2} + \dots + y_n,$$

называемое номером набора y . Например, двоичные наборы 0101 и 1000 имеют номера 5 и 8 соответственно. Очевидно, любая булева функция может быть задана таблицей истинности, в которой двоичные наборы заменены своими номерами (табл. 5.2).

Таблица 5.1
Таблица истинности

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Таблица 5.2
Таблица истинности

Номер набора	f
0	0
1	1
2	0
3	0
4	1
5	1
6	0
7	1

Булевы функции, зависящие от большого числа переменных, задавать таблицей истинности не удобно в силу ее громоздкости. Например, таблица истинности булевой функции 8 переменных будет содержать $2^8 = 256$ строк. Поэтому для задания функции многих переменных удобно использовать *модифицированную таблицу истинности*. В этом случае множество из n переменных функции разбивается на два подмножества: $\{x_1, x_2, \dots, x_j\}$ и $\{x_{j+1}, \dots, x_n\}$. Переменными x_1, x_2, \dots, x_j отмечают строки таблицы истинности, задавая в каждой строке значение соответствующего двоичного набора длины j , а переменными x_{j+1}, \dots, x_n отмечают столбцы таблицы, задавая в каждом столбце значение соответствующего двоичного набора длины $n - j$. Значение функции записывается в клетке на пересечении соответствующей строки и столбца (табл. 5.3).

Таблица 5.3

Модифицированная таблица истинности

x_1, x_2, \dots, x_j	x_{j+1}, \dots, x_n			
	00...0	0...1	...	
00...0			...	
00...1			...	
...
11...1				

5.2. Элементарные булевые функции

Булева функция, зависящая от двух аргументов, называется *элементарной*. Таблица истинности элементарной булевой функции содержит четыре двоичных набора, а значения функции на этих наборах представляют собой четырёхразрядный двоичный вектор. Всего существует $2^4 = 16$ различных четырёхразрядных двоичных векторов и каждый из них определяет одну элементарную булеву функцию, следовательно, всего существует 16 различных элементарных булевых функций.

Построим таблицу всех элементарных булевых функций. Таблица состоит из 18 строк и 4 столбцов. Первые две строки содержат значения аргументов x_1 и x_2 соответственно, таких, что столбцы этих строк представляют собой различные наборы значений аргументов. Следующие 16 строк содержат значения функций на всех четырех наборах значений аргументов. Набор значений функции будем рассматривать как четырехразрядное двоичное число j , изменяющееся в пределах от $0000_{(2)} = 0_{(10)}$ до $1111_{(2)} = 15_{(10)}$. Присвоив это число, но уже в десятичной системе, соответствующей функции в качестве нижнего индекса при ее буквенном изображении: f_0, \dots, f_{15} , расположим все функции в таблице по порядку в соответствии с их индексами (табл. 5.4).

Таблица 5.4
Элементарные булевые функции

x_1	0	0	1	1	Наименование	Обозначение
x_2	0	1	0	1		
f_0	0	0	0	0	Константа нуль (функция 0)	$f_0 = 0$
f_1	0	0	0	1	Конъюнкция (функция И)	$f_1 = x_1 \wedge x_2 = x_1 x_2$
f_2	0	0	1	0	Запрет 1-го аргумента (нет)	$f_2 = x_1 \leftarrow x_2$
f_3	0	0	1	1	Повторение 1-го аргу- мента (да)	$f_3 = x_1$
f_4	0	1	0	0	Запрет 2-го аргумента (нет)	$f_4 = x_2 \leftarrow x_1$
f_5	0	1	0	1	Повторение 2-го аргу- мента (нет)	$f_5 = x_2$
f_6	0	1	1	0	Неравнозначность, сложение по модулю 2 (или-или)	$f_6 = x_1 \oplus x_2$
f_7	0	1	1	1	Дизъюнкция (или)	$f_7 = x_1 \vee x_2$
f_8	1	0	0	0	Операция Пирса (или-не)	$f_8 = x_1 \downarrow x_2$
f_9	1	0	0	1	Эквивалентность (функция и-и)	$f_9 = x_1 \equiv x_2$
f_{10}	1	0	1	0	Отрицание 2-го аргу- мента (не)	$f_{10} = \overline{x_2} = \neg x_2$
f_{11}	1	0	1	1	Импликация от 2-го аргумента к 1-му (нет-не)	$f_{11} = x_2 \rightarrow x_1$
f_{12}	1	1	0	0	Отрицание 1-го аргу- мента (не)	$f_{12} = \overline{x_1} = \neg x_1$
f_{13}	1	1	0	1	Импликация от 1-го аргумента ко 2-му (нет-не)	$f_{13} = x_1 \rightarrow x_2$
f_{14}	1	1	1	0	Операция Шеффера (и-не)	$f_{14} = x_1 x_2$
f_{15}	1	1	1	1	Константа единица (1)	$f_{15} = 1$

Заметим, что функция с четным индексом j может быть получена как
отрицание функции с нечетным индексом $15 - j$: $f_j = \overline{f_{15-j}}$.

5.3. Функциональная полнота систем булевых функций

Функционально полным набором называется множество таких элементарных булевых функций $\{f_1, f_2, \dots, f_k\}$, что произвольная булева функция f может быть записана в виде формулы через функции этого множества.

Решение задачи формирования функциональной полных наборов элементарных функций с использованием принципов необходимости и достаточности основано на понятии замкнутого относительно операции суперпозиции класса функций. *Операция суперпозиции* заключается в подстановке вместо аргументов других булевых функций. Функция $h(x_1, \dots, x_n) = g(f_1(x_1, \dots, x_n), \dots, f_k(x_1, \dots, x_n))$ называется *суперпозицией* функций g и f_1, \dots, f_k . Суперпозиция функций одного аргумента порождает функции одного аргумента. Суперпозиция функций двух аргументов дает возможность строить функции любого числа аргументов. *Классом функций* называется множество функций, обладающих общими свойствами. *Классом булевых функций, функционально замкнутый по операции суперпозиции*, есть множество функций, любая суперпозиция которых дает функцию, также принадлежащую этому множеству.

Все элементарные функции подразделяются по своим свойствам на пять замкнутых классов:

- 1) функции, сохраняющие 0;
- 2) функции, сохраняющие 1;
- 3) самодвойственные;
- 4) монотонные;
- 5) линейные.

1. Булева функция называется функцией, *сохраняющей 0*, если на нулевом наборе аргументов она равна нулю, т.е. $f(0,0,\dots,0) = 0$. Элементарными функциями, сохраняющими 0, являются восемь первых функций: $f_0 - f_7$.

2. Булева функция называется функцией, *сохраняющей 1*, если на единичном наборе аргументов она равна единице, т.е. $f(1,1,\dots,1) = 1$. Элементарными функциями, сохраняющими 1, являются функции с нечетными индексами: $f_1, f_3, f_5, f_7, f_9, f_{11}, f_{13}, f_{15}$.

3. Булева функция называется *самодвойственной*, если на каждой паре противоположных наборов аргументов она принимает противоположные значения, т.е. $f(x_1, x_2, \dots, x_n) = \bar{f}(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)$. Элементарными самодвойственными функциями являются функции f_3, f_5, f_{10}, f_{12} .

4. Булева функция называется *монотонной*, если при возрастании набора аргументов значения этой функции не убывают. Набор аргументов считается больше другого набора, если в нем есть хотя бы один аргумент, больший по значению, и нет ни одного, меньшего по значению. Если у двух наборов есть аргументы и больше, и меньше, то наборы считаются несравнимыми. У функций двух переменных несравнимыми являются наборы 0, 1 и 1, 0. Таким образом, проверку на монотонность для каждой элементарной функции надо провести по двум путям:

$$f(0, 0) \Rightarrow f(0, 1) \Rightarrow f(1, 1) \text{ и } f(0, 0) \Rightarrow f(1, 0) \Rightarrow f(1, 1).$$

Монотонными функциями являются $f_0, f_1, f_3, f_5, f_7, f_{15}$.

5. Булева функция называется *линейной*, если ее можно представлять в виде $f(x_1, x_2, \dots, x_n) = a_n \wedge x_n \oplus \dots \oplus a_2 \wedge x_2 \oplus a_1 \wedge x_1 \oplus a_0$, где $a_i \in \{0, 1\}; i = 0, 1, \dots, n$.

Существуют восемь линейных элементарных функций (табл. 5.5).

Таблица 5.5

Линейные элементарные булевые функции

a_2	a_1	a_0	Линейные функции
0	0	0	$0=f_0$
0	0	1	$1=f_{15}$
0	1	0	$x_1=f_3$
0	1	1	$x_1 \oplus 1=f_{12}$
1	0	0	$x_2=f_5$
1	0	1	$x_2 \oplus 1=f_{10}$
1	1	0	$x_2 \oplus x_1=f_6$
1	1	1	$x_2 \oplus x_1 \oplus 1=f_9$

Для того, чтобы набор элементарных булевых функций был функционально полным, необходимо и достаточно, чтобы в него входили:

- 1) хотя бы одна функция, не сохраняющая 0;
- 2) хотя бы одна функция, не сохраняющая 1;
- 3) хотя бы одна не самодвойственная;
- 4) хотя бы одна не монотонная;
- 5) хотя бы одна не линейная.

При формировании функционально полных наборов элементарных функций целесообразно рассматривать только двенадцать функций из шестнадцати, так как пары f_2 и f_4 , f_3 и f_5 , f_{10} и f_{12} , f_{11} и f_{13} представляют собой с точностью до нумерации аргументов одну функцию: запрет, повторение, отрицание и импликацию соответственно.

Для удобства подбора функционально полных наборов используют таблицу (табл. 5.6), в которой каждой строке соответствует элементарная функция, а столбцам — свойства функций: несохраняемость 0, несохраняемость 1, несамодвойственность, немонотонность и нелинейность. На пересечении строки и столбца ставится крестик, если функция, соответствующая строке, обладает свойством, соответствующим столбцу.

Признаком функциональной полноты являются наличие крестика в каждом столбце таблицы хотя бы для одной из составляющих систему булевых функций.

Из таблицы видно, что операции Пирса и Шеффера каждая в отдельности являются функционально полными наборами, а операция повторения не играет никакой роли при формировании функционально полных наборов.

Таблица 5.6

Свойства элементарных булевых функций

Функция	Свойства				
	Несох-раняе-мость 0	Несох-раняе-мость 1	Несамо-двойст-венност	Немо-нотон-ность	Нели-ней-ность
Константа 0 (f_0)	x	x			
Конъюнкция (f_1)		x			x
Запрет (f_2, f_4)	x	x		x	x
Повторение (f_3, f_5)					
Неравнозначность (f_6)	x	x	x		
Дизъюнкция (f_7)			x		x
Операция Пирса (f_8)	x	x	x	x	x
Равнозначность (f_9)	x		x	x	
Отрицание (f_{10}, f_{12})	x	x		x	
Импликация (f_{11}, f_{13})	x		x	x	x
Операция Шеффера (f_{14})	x	x	x	x	x
Константа 1 (f_{15})	x		x		

5.4. Аналитические способы задания булевых функций

Среди аналитических способов задания функций (задание функций формулами) широкое распространение получили совершенная дизъюнктивная нормальная форма, совершенная полиномиальная нормальная форма, совершенная конъюнктивная нормальная форма, дизъюнктивная нормальная форма, конъюнктивная нормальная форма, произвольная скобочная форма и полином Жегалкина.

Совершенная дизъюнктивная нормальная форма (СДНФ) представляет собой дизъюнкцию конституент единицы.

Конституентой единицы называется функция, принимающая значение 1 только на единственном наборе.

Конституента единицы записывается как логическое произведение всех аргументов функции, некоторые из них могут быть с отрицаниями. Например, функция $f = \overline{x_1} \overline{x_2} x_3 x_4$ является конституентой единицы и принимает значение 1 на единственном наборе 1001.

Для того, чтобы получить аналитическое выражение в СДНФ функции, заданной таблично, нужно составить дизъюнкцию конституент единицы для тех наборов значений аргументов, для которых значение функции равно 1, причем символ любой переменной в некоторой конституенте берется со знаком отрицания, если конкретное значение переменной в рассматриваемом наборе имеет значение 0.

Пример 5.1. Для функций, заданных таблицей истинности (табл. 5.7), СДНФ имеет вид:

$$\begin{aligned}f_1 &= \overline{x_1} \overline{x_2} x_3 \vee \overline{x_1} \overline{x_2} \overline{x_3} \vee x_1 \overline{x_2} \overline{x_3} \vee x_1 x_2 x_3 \\f_2 &= \overline{x_1} x_2 x_3 \vee x_1 \overline{x_2} x_3 \vee x_1 x_2 \overline{x_3}\end{aligned}$$

Таблица 5.7
Таблица истинности

x_1	x_2	x_3	f_1	f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Если в СДНФ заменить операции дизъюнкции операциями сложения по модулю 2, то равенство сохранится. Такая форма называется *совершенной полиномиальной нормальной формой* (СПНФ).

Пример 5.2. Для функций, заданных таблицей истинности (см. табл. 5.7), СПНФ имеет вид:

$$\begin{aligned}f_1 &= \overline{x_1} \overline{x_2} x_3 \oplus \overline{x_1} x_2 \overline{x_3} \oplus x_1 \overline{x_2} \overline{x_3} \oplus x_1 x_2 x_3 \\f_2 &= \overline{x_1} x_2 x_3 \oplus x_1 \overline{x_2} x_3 \oplus x_1 x_2 \overline{x_3}\end{aligned}$$

Совершенная конъюнктивная нормальная форма (СКНФ) представляет собой конъюнкцию конституент нуля.

Конституентой нуля называется функция, принимающая значение 0 на единственном наборе.

Конституента нуля записывается в виде дизъюнкции всех аргументов, некоторые из них могут быть с отрицаниями. Каждому набору соответствует своя конституента 0. Например, набору 0110 переменных x_1, x_2, x_3, x_4 соответствует конституента нуля $x_1 \vee \overline{x}_2 \vee \overline{x}_3 \vee x_4$.

Для того, чтобы получить аналитическое выражение функции, заданной таблично, в СКНФ, нужно составить конъюнкцию конституент нуля для тех наборов значений аргументов, на которых значение функции равно 0, причем символ любой переменной в некоторой конституенте берется со знаком отрицания, если ее конкретное значение в рассматриваемом наборе равно 1.

Пример 5.3. Для функций, заданных таблицей истинности (см. табл. 5.7), совершенная конъюнктивная нормальная форма имеет вид:

$$\begin{aligned}f_1 &= (x_1 \vee x_2 \vee x_3)(x_1 \vee \overline{x}_2 \vee \overline{x}_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3) \\f_2 &= (x_1 \vee x_2 \vee x_3)(x_1 \vee x_2 \vee \overline{x}_3)(x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_3)\end{aligned}$$

Кроме совершенных нормальных форм булевых функций существуют другие нормальные формы.

Дизъюнктивной нормальной формой (ДНФ) называется дизъюнкция элементарных конъюнкций.

Элементарной конъюнкцией называется конъюнкция конечного числа различных между собой переменных, каждая из которых может быть с отрицанием.

Рангом элементарной конъюнкции называется число переменных в данной конъюнкции.

Конституента единицы представляет собой элементарную конъюнкцию, ранг которой равен количеству аргументов функции. ДНФ может содержать элементарные конъюнкции различного ранга, например:

$$f = \overline{x}_1 \overline{x}_2 x_3 \vee \overline{x}_2 \overline{x}_3 \vee x_1 \overline{x}_3 \vee x_2.$$

Конъюнктивной нормальной формой (КНФ) называется конъюнкция элементарных дизъюнкций.

Элементарной дизъюнкцией называется дизъюнкция конечного числа различных между собой переменных, каждая из которых может быть с отрицанием.

Рангом элементарной дизъюнкции называется число переменных в данной дизъюнкции.

Конституента нуля представляет собой элементарную дизъюнкцию, ранг которой равен количеству аргументов функции. КНФ может содержать элементарные дизъюнкции различного ранга, например:

$$f = (x_1 \vee x_2 \vee x_3)(\overline{x}_1 \vee \overline{x}_2)\overline{x}_3.$$

Кроме совершенных форм булевых функций существует *произвольная скобочная форма* (инфиксная запись), например:

$$f = \overline{x}_1(\overline{x}_2x_3 \vee \overline{x}_3(x_2 \vee \overline{x}_4)) \vee x_2(x_3 \vee x_1\overline{x}_4).$$

Еще одно представление булевых функций получается с использованием операций конъюнкции \wedge , сложения по модулю 2 \oplus и константы 1. Алгебра над множеством булевых функций с бинарными операциями \wedge и \oplus и константой 1 называется *алгеброй Жегалкина*. В алгебре Жегалкина выполняются следующие соотношения:

$$x_1 \oplus x_2 = x_2 \oplus x_1; \quad (5.1)$$

$$x_1(x_2 \oplus x_3) = x_1x_2 \oplus x_1x_3; \quad (5.2)$$

$$x_1 \oplus x_1 = 0; \quad (5.3)$$

$$x_1 \oplus 0 = x_1; \quad (5.4)$$

$$x_1(x_2x_3) = (x_1x_2)x_3; \quad (5.5)$$

$$x_1x_2 = x_2x_1; \quad (5.6)$$

$$x_1x_1 = x_1; \quad (5.7)$$

$$x_1 \wedge 1 = x_1; \quad (5.8)$$

$$x_1 \wedge 0 = 0. \quad (5.9)$$

Отрицание и дизъюнкция выражаются так:

$$\overline{x}_1 = x_1 \oplus 1; \quad (5.10)$$

$$\overline{\overline{x}_1 \vee x_2} = \overline{x_1}x_2 = (x_1 \oplus 1)(x_2 \oplus 1) \oplus 1 = x_1x_2 \oplus x_1 \oplus x_2; \quad (5.11)$$

Если в произвольной формуле алгебры Жегалкина раскрыть скобки и произвести все возможные упрощения по соотношениям (5.1) — (5.9), то получится формула, имеющая вид суммы произведений, т.е. полинома по модулю 2. Такая формула называется *полиномом Жегалкина* для данной функции.

От произвольной скобочной записи булевой функции или от нормальной формы всегда можно перейти к форме алгебры Жегалкина и, следовательно, полиному Жегалкина, используя равенства (5.10) и (5.11), а также равенство, вытекающее из (5.11): если $f_1f_2 = 0$, то $f_1 \vee f_2 = f_1 \oplus f_2$. Оно, в частности, позволяет заменять знак дизъюнкции знаком “ \oplus ” в случае, когда исходная формула — СДНФ.

Пример 5.4. Преобразование произвольной скобочной формы булевой функции в полином Жегалкина:

$$\begin{aligned}
 (x_1 \vee x_2)(\overline{x}_2 \vee x_1 x_3) &= (x_1 x_2 \oplus x_1 \oplus x_2)(x_1 \overline{x}_2 x_3 \oplus \overline{x}_2 \oplus x_1 x_3) = \\
 &= (x_1 x_2 \oplus x_1 \oplus x_2)(x_1(x_2 \oplus 1)x_3 \oplus (x_2 \oplus 1) \oplus x_1 x_3) = \\
 &= (x_1 x_2 \oplus x_1 \oplus x_2)(x_1 x_2 x_3 \oplus x_1 x_3 \oplus x_2 \oplus 1 \oplus x_1 x_3) = \\
 &= (x_1 x_2 \oplus x_1 \oplus x_2)(x_1 x_2 x_3 \oplus x_2 \oplus 1) = \\
 &= x_1 x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_2 \oplus x_1 x_2 \oplus x_2 \oplus x_1 x_2 \oplus x_1 \oplus x_2 = \\
 &= x_1 x_2 x_3 \oplus x_1 x_2 \oplus x_1
 \end{aligned}$$

Пример 5.5. Преобразование СДНФ булевой функции в полином Жегалкина:

$$\begin{aligned}
 \overline{x}_1 x_2 x_3 \vee x_1 \overline{x}_2 x_3 &= \overline{x}_1 x_2 x_3 \oplus x_1 \overline{x}_2 x_3 = (x_1 \oplus 1)x_2 x_3 \oplus x_1(x_2 \oplus 1)x_3 = \\
 &= x_1 x_2 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3 \oplus x_1 x_3 = x_1 x_3 \oplus x_2 x_3
 \end{aligned}$$

5.5. Основные законы булевой алгебры и их следствия

Основные законы булевой алгебры.

1. Коммутативность:

$$\text{а)} \quad x_1 x_2 = x_2 x_1; \quad \text{б)} \quad x_1 \vee x_2 = x_2 \vee x_1.$$

2. Ассоциативность:

$$\text{а)} \quad x_1(x_2 x_3) = (x_1 x_2) x_3; \quad \text{б)} \quad (x_1 \vee x_2) \vee x_3 = x_1 \vee (x_2 \vee x_3).$$

3. Дистрибутивность конъюнкции относительно дизъюнкции:

$$x_1(x_2 \vee x_3) = x_1 x_2 \vee x_1 x_3.$$

4. Дистрибутивность дизъюнкции относительно конъюнкции:

$$x_1 \vee (x_2 x_3) = (x_1 \vee x_2)(x_1 \vee x_3).$$

5. Идемпотентность:

$$\text{а)} \quad x x = x; \quad \text{б)} \quad x \vee x = x.$$

6. Двойное отрицание:

$$\bar{\bar{x}} = x.$$

7. Свойства констант:

$$\begin{array}{lll}
 \text{а)} \quad x \wedge 1 = x; & \text{б)} \quad x \wedge 0 = 0; & \text{в)} \quad x \vee 1 = 1; \\
 \text{г)} \quad x \vee 0 = x; & \text{д)} \quad \bar{0} = 1; & \text{е)} \quad \bar{1} = 0.
 \end{array}$$

8. Законы де Моргана:

$$\text{а) } \overline{x_1 x_2} = \overline{x_1} \vee \overline{x_2}; \quad \text{б) } \overline{x_1 \vee x_2} = \overline{x_1} \overline{x_2}.$$

9. Закон противоречия:

$$\overline{x} \overline{x} = 0.$$

10. Закон "исключённого третьего":

$$x \vee \overline{x} = 1.$$

Следствиями основных законов булевой алгебры являются: правило склеивания, правило поглощения, правило вычеркивания и ряд других соотношений, позволяющих преобразовывать, в частности упрощать, формулы булевых функций.

1. Правила склеивания:

$$\text{а) } x_1 x_2 \vee x_1 \overline{x_2} = x_1; \quad \text{б) } (x_1 \vee x_2)(x_1 \vee \overline{x_2}) = x_1.$$

2. Правила поглощения:

$$\text{а) } x_1 x_2 \vee x_1 = x_1; \quad \text{б) } (x_1 \vee x_2)x_1 = x_1.$$

3. Правила вычеркивания:

$$\text{а) } x_1 x_2 \vee \overline{x_1} = x_2 \vee \overline{x_1}; \quad \text{б) } \overline{x_1} x_2 \vee x_1 = x_2 \vee x_1.$$

В булевой алгебре существует дизъюнктивное разложение Шеннона:

$$f(x_1, x_2, \dots, x_i, \dots, x_n) = \overline{x_i} y_0 \vee x_i y_1,$$

где x_i – переменная разложения;

y_0 и y_1 – коэффициенты разложения:

$$y_0 = f(x_1, x_2, \dots, 0, \dots, x_n);$$

$$y_1 = f(x_1, x_2, \dots, 1, \dots, x_n).$$

Дизъюнктивное разложение Шеннона позволяет выразить булеву функцию через переменную разложения (с отрицанием и без отрицания) и функции (коэффициенты разложения), зависящие от меньшего количества аргументов.

Если выполнить разложение Шеннона булевой функции, заданной в любой аналитической форме, по всем переменным, то получим СДНФ этой функции.

5.6. Минимизация полностью определённых булевых функций

Областью определения булевой функции n аргументов является множество всех возможных двоичных наборов длины n . Если значение функции определено на всех двоичных наборах длины n , то булева функция является *полностью определённой*.

Общая задача минимизации булевых функций может быть сформулирована следующим образом: найти аналитическое выражение заданной булевой функции в форме, содержащей минимально возможное число букв (переменных и операций). В общей постановке данная задача пока не решена, однако достаточно хорошо исследована в классе дизъюнктивно-конъюнктивных нормальных форм.

5.6.1. Основные понятия

Минимальной дизъюнктивной нормальной формой булевой функции называется ДНФ, содержащая наименьшее число букв (по отношению ко всем другим ДНФ, представляющим заданную булеву функцию).

Булева функция $g(x_1, x_2, \dots, x_n)$ называется *импликантой* булевой функции $f(x_1, x_2, \dots, x_n)$, если для любого набора переменных, на котором $g = 1$, справедливо $f = 1$.

Пример 5.6. Булева функция f задана табл. 5.8. Там же приведены все ее импликанты. Запишем функцию f и ее импликанты в СДНФ и выполним все возможные упрощения, применяя правило склеивания:

$$f = \overline{x_1}x_2x_3 \vee x_1\overline{x_2}\overline{x_3} \vee x_1x_2x_3 = g_7;$$

$$g_1 = x_1x_2x_3;$$

$$g_2 = x_1x_2\overline{x_3};$$

$$g_3 = x_1\overline{x_2}x_3 \vee x_1x_2x_3 = x_1x_2;$$

$$g_4 = \overline{x_1}x_2x_3;$$

$$g_5 = \overline{x_1}x_2x_3 \vee x_1x_2\overline{x_3} = x_2x_3;$$

$$g_6 = x_1x_2x_3 \vee x_1\overline{x_2}\overline{x_3}.$$

Импликанта g булевой функции f , являющаяся элементарной конъюнкцией, называется *простой*, если никакая часть импликанты g не является импликантой функции f .

Из примера видно, что импликанты $g_3 = x_1x_2$ и $g_5 = x_2x_3$ являются простыми импликантами функции f . Импликанты g_1, g_2, g_4, g_6 не являются простыми, так как их части являются импликантами функции f , например g_3 является частью g_1 . Приведем без доказательства два утверждения, полезные при получении минимальной ДНФ.

1. Дизъюнкция любого числа импликант логической функции f также является импликантой этой функции.
2. Любая булева функция f эквивалентна дизъюнкции всех своих простых импликант. Такая форма представления булевой функции называется *сокращенной ДНФ*.

Таблица 5.8
Импликанты булевой функции

x_1	x_2	x_3	f	g_1	g_2	g_3	g_4	g_5	g_6	g_7
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
1	1	0	1	0	1	1	0	0	1	1
1	1	1	1	1	0	1	0	1	0	1

Перебор всех возможных импликант для булевой функции f из рассмотренного примера дает возможность убедиться, что простых импликант всех две: g_3 и g_5 . Следовательно, сокращенная ДНФ функции f имеет вид:

$$f = g_3 \vee g_5 = x_1x_2 \vee x_2x_3$$

Как видно из табл. 5.8 импликанты g_3 и g_5 в совокупности покрывают своими единицами все единицы функции f . Получение сокращенных ДНФ является первым этапом отыскания минимальных форм булевых функций. Как уже отмечалось, в сокращенную ДНФ входят все простые импликанты булевой функции. Иногда из сокращенной ДНФ можно убрать одну или несколько простых импликант, не нарушая эквивалентности исходной функции. Такие простые импликанты называются *лишними*. Исключение лишних простых импликант на сокращенных ДНФ — второй этап минимизации.

Сокращенная ДНФ булевой функции называется *туниковой*, если в ней отсутствует лишние простые импликанты.

Устранение лишних простых импликант из сокращенной ДНФ булевой функции не является однозначным процессом, т.е. булева функция может иметь несколько тупиковых ДНФ. Тупиковые ДНФ булевой функции f , содержащие минимальное число букв, являются минимальными. Минимальных ДНФ то же может быть несколько.

В общем случае минимизация булевой функции производится в три этапа:

- первый — переход от СДНФ к сокращенной ДНФ;
- второй — переход от сокращенной ДНФ к миникальной ДНФ;
- третий — переход от миникальной ДНФ к скобочной форме.

5.6.2. Получение сокращенной ДНФ булевой функции

Метод Квайна

Метод Квайна применим для получения сокращенной ДНФ булевой функции из СДНФ. Для этого производятся все возможные склеивания друг с другом сначала конституент единицы, а затем всех производных элементарных конъюнкций более низкого ранга.

Пусть имеется СДНФ булевой функции:

$$f = \overline{x_1} \overline{x_2} \overline{x_3} x_4 \vee \overline{x_1} \overline{x_2} x_3 \overline{x_4} \vee \overline{x_1} x_2 \overline{x_3} \overline{x_4} \vee \overline{x_1} x_2 x_3 \overline{x_4} \vee x_1 \overline{x_2} x_3 \overline{x_4} \vee x_1 x_2 x_3 \overline{x_4}$$

1 2 3 4 5 6

Пометим каждую конституенту единицы из СДНФ функции f десятичным номером (записан под конституентой). Выполняем склеивания. Конституента 1 склеивается только с конституентой 2 (по переменной x_3) и с конституентой 3 (по переменной x_2), конституента 2 с конституентой 4 и т.д. В результате получаем:

$$\begin{array}{ll} 1 - 2 : \overline{x_1} \overline{x_2} x_4; & 3 - 4 : \overline{x_1} x_2 \overline{x_4}; \\ 1 - 3 : \overline{x_1} x_3 \overline{x_4}; & 4 - 6 : x_2 x_3 x_4; \\ 2 - 4 : \overline{x_1} x_3 x_4; & 5 - 6 : x_1 x_2 x_3. \end{array}$$

Далее склеиваем полученные элементарные конъюнкции. Возможны два случая склеивания:

$$\begin{aligned} \overline{x_1} \overline{x_2} x_4 \vee \overline{x_1} x_2 \overline{x_4} &= \overline{x_1} x_4; \\ \overline{x_1} x_3 \overline{x_4} \vee x_1 x_3 x_4 &= \overline{x_1} x_4 \end{aligned}$$

с появлением одной и той же элементной конъюнкции $\overline{x_1} x_4$. Дальнейшие склеивания невозможны.

Не участвовавшие в склеивании конституенты единицы и элементарные конъюнкции более низкого ранга включаются в сокращенную ДНФ:

$$\overline{x_2x_3x_4} \vee \overline{x_1x_2x_3} \vee \overline{x_1x_4}$$

Метод Квайна — Мак-Класски

Метод Квайна — Мак-Класски позволяет сократить количество парных проверок на возможность склеивания по сравнению с методом Квайна и выполняется следующим образом:

1. Все конституенты единицы из СДНФ записываются их двоичными номерами.

2. Все номера разбиваются на непересекающиеся группы. Признак образования i -й группы: i единицы в каждом двоичном номере конституенты единицы.

3. Склейивание производят только между номерами соседних групп. Склейываемые номера отмечаются.

4. Склейивания производят всевозможные, как и в методе Квайна. Неотмеченные после склейивания номера являются простыми импликантами.

Рассмотрим метод на примере булевой функции, заданной в СДНФ:

$$f = \overline{x_1}\overline{x_2}\overline{x_3}\overline{x_4} \vee \overline{x_1}\overline{x_2}\overline{x_3}x_4 \vee \overline{x_1}x_2\overline{x_3}\overline{x_4} \vee \overline{x_1}\overline{x_2}x_3\overline{x_4} \vee x_1\overline{x_2}\overline{x_3}\overline{x_4} \vee x_1x_2\overline{x_3}\overline{x_4} \vee x_1x_2x_3\overline{x_4}$$

1. В СДНФ функции f заменим все конституенты единицы их двоичными номерами:

$$f = 0001 \vee 0011 \vee 0101 \vee 0111 \vee 1110 \vee 1111$$

2. Образуем группы двоичных номеров. Результат заносим в таблицу:

Номер группы				
0	1	2	3	4
	0001+	0011+	0111+	1111+
	0101+	1110+		

3. Склейим номера из соседних групп этой таблицы. Склейываемые номера отметим. Результаты склейивания занесем в следующую таблицу:

Номер группы				
0	1	2	3	4
	00*1+	0*11+	*111	
	0*01+	01*1+	111*	

. 5.10. Склейм номера из соседних групп полученной таблицы. Склейваться могут только номера, имеющие звездочки в одинаковых позициях. Склейиваемые номера отмечаем. Результаты склейивания занесем в таблицу:

Номер группы				
0	1	2	3	4
	0**1			

4. Неотмеченные номера соответствуют простым импликантам:

111, 111, 0**1 или, в другой форме, $x_2x_3x_4 \vee x_1x_2x_3 \vee \overline{x_1}x_4$.

Разбиение конституент на группы позволяет уменьшить число парных сравнений при склейивании.

Метод Нельсона

Метод позволяет получить сокращенную ДНФ булевой функции из ее произвольной КНФ. Суть метода заключается в использовании следующего утверждения: если в произвольной КНФ булевой функции f раскрыть скобки и произвести все поглощения, то в результате будет получена сокращенная ДНФ булевой функции f .

Пусть булева функция задана в произвольной КНФ:

$$f = (x_1 \vee x_2)(\overline{x}_1 \vee x_3)(x_1 \vee x_2 \vee \overline{x}_3)$$

После раскрытия скобок получаем:

$$f = (x_1x_3 \vee \overline{x}_1\overline{x}_2 \vee \overline{x}_2x_3)(x_1 \vee x_2 \vee \overline{x}_3) = x_1x_3 \vee x_1x_3x_2 \vee \overline{x}_1\overline{x}_2\overline{x}_3 \vee x_1\overline{x}_2x_3$$

После проведения всех поглощений имеем сокращенную ДНФ:

$$f = x_1x_3 \vee \overline{x}_1\overline{x}_2x_3$$

5.6.3. Получение минимальной ДНФ булевой функции

Для получения минимальной ДНФ необходимо убрать из сокращенной ДНФ все лишние простые импликанты. Это делается с помощью специальной импликантной матрицы *Квайна*. Строки такой матрицы отмечаются простыми импликантами булевой функции, т. е. членами сокращенной ДНФ, а столбцы — конституентами единицы, т. е. членами СДНФ булевой функции. Клетка импликантной матрицы на пересечении строки i и столбца j отмечается крестиком, если i -я импликанта покрывает j -ю конституенту. Простая импликанта поглощает некоторую конституенту единицы, если является ее собственной частью.

Минимальные ДНФ строятся по импликантной матрице следующим образом: рассматриваются различные варианты выбора подмножества простых импликант, которые покроют крестиками все столбцы импликантной матрицы и выбираются варианты с минимальным суммарным числом букв в такой совокупности импликант.

Этот метод является трудоемким, так как существует $2^n - 1$ различных вариантов выбора подмножеств простых импликант, где n — количество простых импликант.

Сократить перебор можно следующим образом:

1. Найти столбцы импликантной матрицы, имеющие только один крестик. Соответствующие этим крестикам простые импликанты называются *базисными* и составляют *ядро Квайна*, которое обязательно входит в минимальную ДНФ.

2. Рассмотреть различные варианты выбора подмножеств простых импликант, которые покроют крестиками остальные столбцы импликантной матрицы, и выбрать варианты с минимальным суммарным числом букв в таком подмножестве импликант.

В этом случае существуют $2^{n-r} - 1$ вариантов, где r — количество базисных импликант, входящих в ядро Квайна.

Метод Петрика

Метод Петрика позволяет получить все тупиковые ДНФ по импликантной матрице. Этот метод основан на *конъюнктивном представлении матрицы*. Для получения конъюнктивного представления матрицы все простые импликанты обозначаются разными буквами. После этого для каждого i -го столбца матрицы строится дизъюнкция всех букв, обозначающих строки матрицы, пересечение которых с i -м столбцом отмечено крестиком, конъюнктивное представление F импликантной матрицы образуется как конъюнкция построенных функций для всех столбцов матрицы. После раскрытия скобок и выполнения всех возможных поглощений получится дизъюнкция элементарных конъюнкций, каждая из которых содержит все импликанты тупиковой ДНФ.

Для матрицы (табл. 5.9) конъюнктивное представление следующее:

$$F = (A \vee B)(A \vee C)(C \vee D)DB.$$

После упрощения получим: $F = ABD \vee BCD$, что определяет две тупиковые ДНФ:

$$f = x_1x_3 \vee x_1x_2 \vee \overline{x_1}\overline{x_2} \quad \text{и} \quad f = x_1x_2 \vee \overline{x_2}x_3 \vee \overline{x_1}\overline{x_2},$$

первая из которых является минимальной ДНФ.

Таблица 5.9
Импликантная матрица Квайна

Простые импликанты	Конституенты единицы				
	$x_1x_2x_3$	$\bar{x}_1\bar{x}_2x_3$	$\bar{x}_1x_2\bar{x}_3$	$\bar{x}_1\bar{x}_2\bar{x}_3$	$x_1\bar{x}_2\bar{x}_3$
x_1x_3	+	+			
x_1x_2	+				+
\bar{x}_2x_3		+	+		
\bar{x}_1x_2				+	+

Эвристический метод

С целью уменьшения трудоемкости процесса нахождения минимальной ДНФ может быть использован эвристический метод, который позволяет найти тупиковую ДНФ, как правило, близкую по сложности к минимальной ДНФ, и заключается в следующем:

1. Выделяется ядро Квайна и включается в искомую ДНФ.
2. Вычеркиваются строки и столбцы, покрываемые импликантами, входящими в ядро Квайна. Если в полученной матрице имеются столбцы, содержащие по одному крестику, то операцию выделения ядра Квайна повторяют.
3. Если не все столбцы покрыты, то выбрать простую импликанту, покрывающую максимальное количество невычеркнутых столбцов. Если таких импликант несколько, то выбирается та, которая содержит минимальное количество букв. Выбранная импликанта включается в искомую ДНФ, а покрываемые ею строки и столбцы вычеркиваются. Если все столбцы покрыты, то искомая ДНФ найдена.

Эвристический метод не гарантирует нахождения действительно минимальной ДНФ и позволяет получить только приближенное решение. Степень приближенности достаточно высокая для практического использования рассмотренного метода.

5.6.4. Скобочная минимизация булевых функций

Скобочная минимизация булевых функций (факторизация) в ряде случаев позволяет получить скобочную форму, значительно более простую, чем минимальная ДНФ. Процесс факторизации заключается в многократном выполнении операции вынесения за скобки общих членов. Работу факторного алгоритма рассмотрим на примере.

Пусть минимальная ДНФ функции f имеет вид:

$$f = x_1 \bar{x}_2 x_6 \vee x_1 x_2 x_5 \bar{x}_6 \vee x_1 \bar{x}_2 \bar{x}_4 \bar{x}_5 \vee x_1 \bar{x}_2 x_3 x_5$$

Обозначим каждую элементарную конъюнкцию буквами A, B, C, D . Тогда функцию f можно представить множеством: $F = (A, B, C, D)$.

На первом этапе работы алгоритма ищутся все попарные пересечения элементов множества F . В нашем случае имеем:

$$\begin{aligned} A \cap B &= x_1; & A \cap C &= x_1 \bar{x}_2; & A \cap D &= x_1 \bar{x}_2; \\ B \cap C &= x_1; & B \cap D &= x_1 x_5; & C \cap D &= x_1 \bar{x}_2. \end{aligned}$$

Выбирается та пара элементов множества F , пересечение которых дает конъюнкцию наибольшей длины. Для рассматриваемого примера выберем либо пару (A, C) , либо (A, D) , либо (C, D) , что с точки зрения результатов работы алгоритма безразлично. Выберем пару (C, D) и запишем функцию f аналитически, вынося общий элемент $x_1 \bar{x}_2$ за скобки по отношению к C и D . Получим:

$$f = x_1 \bar{x}_2 x_6 \vee x_1 x_2 x_5 \bar{x}_6 \vee x_1 \bar{x}_2 (x_4 \bar{x}_5 \vee x_3 x_5)$$

$$\text{Обозначим } A_1 = x_1 \bar{x}_2 x_6; \quad B_1 = x_1 x_2 x_5 \bar{x}_6; \quad C_1 = x_1 \bar{x}_2 (x_4 \bar{x}_5 \vee x_3 x_5).$$

Получим $F_1 = (A_1, B_1, C_1)$.

На втором этапе алгоритма ищутся все попарные пересечения элементов множества F_1 . В нашем случае имеем:

$$A_1 \cap B_1 = x_1; \quad A_1 \cap C_1 = x_1 \bar{x}_2; \quad B_1 \cap C_1 = x_1.$$

Вновь выбирается пара элементов множества F_1 , пересечение которых дает конъюнкцию наибольшей длины. Выберем пару (A_1, C_1) и запишем функцию f аналитически, вынося общий элемент $x_1 \bar{x}_2$ за скобки по отношению к A_1 и C_1 . Получим:

$$f = x_1 \bar{x}_2 (x_6 \vee x_4 \bar{x}_5 \vee x_3 x_5) \vee x_1 x_2 x_5 \bar{x}_6.$$

Обозначив $A_2 = x_1 \bar{x}_2 (x_6 \vee x_4 \bar{x}_5 \vee x_3 x_5)$, $B_2 = x_1 x_2 x_5 \bar{x}_6$, получим множество $F_2 = (A_2, B_2)$. Ищем пересечение элементов множества F_2 . Получаем $A_2 \cap B_2 = x_1$. Окончательно имеем:

$$f = x_1 (\bar{x}_2 (x_6 \vee x_4 \bar{x}_5 \vee x_3 x_5) \vee x_2 x_5 \bar{x}_6).$$

5.7. Минимизация частично определенных булевых функций

Булева функция n аргументов называется *частично определенной* (*частичной*), если ее значение определено не на всех двоичных наборах длины n . Минимизация частично определенных функций, так же как и полностью определенных, выполняется в три этапа.

Первый этап — получение сокращенной ДНФ, членами которой являются простые импликанты частичной функции.

Простой импликантой частичной функции f называется элементарная конъюнкция K , если:

- 1) она покрывает хотя бы один набор значений аргументов, на котором функция принимает единичное значение;
- 2) не покрывает ни одного набора, на котором функция принимает нулевое значение;
- 3) любая конъюнкция, полученная из нее вычеркиванием переменных, покрывает некоторый набор, на котором функция принимает нулевое значение.

Для построения системы всех простых импликант удобно пользоваться заданием частичной функции f в виде таблиц T_1 и T_0 :

x_1	x_2	x_3	x_4
0	1	0	0
0	1	1	0
1	0	1	0
1	1	1	1

x_1	x_2	x_3	x_4
0	0	0	1
1	0	0	0
1	1	0	0
1	1	1	0

в первой из которых перечислены наборы значений аргументов, на которых функция f принимает единичное значение, а во второй — нулевое.

Метод построения всех простых импликант проиллюстрирован таблицей:

0	1	0	0	+	0	-	0	0	+	0	-	-	0
0	1	1	0	+	0	1	-	0	+	0	1	-	-
1	0	1	0	+	0	1	0	-	+	0	-	1	-
1	1	1	1	+	0	-	1	0	+	-	0	1	-
					0	1	1	-	+	-	-	1	1
					-	0	1	0	+	-	1	-	1
					1	0	1	-	+	1	-	-	1
					-	1	1	1	+				
					1	-	1	1	+				
					1	1	-	1	+				

В 1-й колонке помещены все наборы таблицы T_1 , во 2-й колонке содержатся всевозможные наборы, образованные из наборов 1-й колонки заменой одного разряда прочерком и обладающие тем свойством, что они не покрывают никаких наборов таблицы T_0 (не совпадают с ними в не вычеркнутых разрядах). Они могут быть найдены просмотром всех наборов из 1-й колонки и проверкой возможности вычеркивания в каждом из них каждого из символов. Наборы 1-й колонки, покрываемые наборами 2-й колонки, помечаются знаком “+”. В 3-й колонке находятся все наборы, полученные из наборов 2-й колонки заменой одного разряда прочерком и не покрывающие наборов таблицы T_0 . Покрытые ими наборы 2-й колонки помечаются знаком “+”. Из наборов 3-й колонки уже не удается образовать наборов с тремя прочерками, не покрывающих наборов из таблицы T_0 . На этом процедура завершается. Все непомеченные наборы таблицы соответствуют простым импликантам. В данном случае это

$$\overline{x_1} \overline{x_4}, \quad \overline{x_1} x_2, \quad \overline{x_1} x_3, \quad \overline{x_2} x_3, \quad x_3 x_4, \quad x_2 x_4, \quad x_1 x_4.$$

Второй этап минимизации — получение минимальной ДНФ.

Для получения минимальной ДНФ необходимо убрать из сокращенной ДНФ все лишние простые импликанты. Это делается, как и в случае полностью определенных функций, с помощью специальной импликантой матрицы Квайна. Строки этой матрицы отмечаются простыми импликантами, полученными на первом этапе, а столбцы — конституентами единицы, соответствующими наборам таблицы T_1 . Импликантная матрица представлена в табл. 5.10.

Таблица 5.10
Импликантная матрица Квайна

Простые импликанты	Конституенты единицы			
	$\overline{x_1} \overline{x_2} \overline{x_3} \overline{x_4}$	$\overline{x_1} \overline{x_2} x_3 x_4$	$x_1 \overline{x_2} \overline{x_3} \overline{x_4}$	$x_1 x_2 x_3 x_4$
$\overline{x_1} x_4$	+	+		
$\overline{x_1} x_2$	+	+		
$\overline{x_1} x_3$		+		
$\overline{x_2} x_3$			+	
$x_3 x_4$				+
$x_2 x_4$				+
$x_1 x_4$				+

Исключая лишние простые импликанты, получаем минимальную ДНФ частично определенной функции. В данном случае примером минимальной ДНФ может служить $f = \overline{x_1}x_2 \vee \overline{x_2}x_3 \vee x_3x_4$.

Трудоемкость процесса минимизации частично определенной булевой функции значительно зависит от степени определенности функции. Рассмотренный метод целесообразно использовать при минимизации слабо определенных булевых функций, когда таблицы T_1 и T_0 содержат небольшое количество строк. Если степень определенности частичной функции высока, то для минимизации функции можно использовать *модифицированный метод Квайна — Мак-Класски*. Модификация метода Квайна — Мак-Класски заключается в том, что частичная функция на неопределенных наборах доопределяется единичным значением. Для доопределенной функции методом Квайна — Мак-Класски находятся все простые импликации. При составлении матрицы Квайна строки отмечаются полученными простыми импликантами, а столбцы — конституентами единицы частичной функции, соответствующими строкам таблицы T_1 . Исключая лишние простые импликанты, получаем минимальную ДНФ частично определенной функции.

Используем модифицированный метод Квайна — Мак-Класски для нахождения минимальной ДНФ частичной функции, заданной таблицами T_1 и T_0 :

x_1	x_2	x_3	x_4
0	1	0	0
0	1	1	0
1	0	1	0
1	1	1	1

x_1	x_2	x_3	x_4
0	0	0	1
1	0	0	0
1	1	0	0
1	1	1	0

Частичную функцию на неопределенных наборах доопределим единичным значением и запишем ее в СДНФ, используя двоичное представление конституент единицы:

$$\begin{aligned} f = & 0100 \vee 0110 \vee 1010 \vee 1111 \vee 0000 \vee 0010 \vee 0011 \vee 0101 \vee \\ & 0111 \vee 1001 \vee 1011 \vee 1101 \end{aligned}$$

Процесс нахождения простых импликант методом Квайна — Мак-Класски представлен в виде таблиц:

$0\ 0\ 0\ 0 +$	$0\ 1\ 0\ 0 +$	$0\ 1\ 1\ 0 +$	$0\ 1\ 1\ 1 +$	$1\ 1\ 1\ 1 +$
	$0\ 0\ 1\ 0 +$	$1\ 0\ 1\ 0 +$	$1\ 0\ 1\ 1 +$	
		$0\ 0\ 1\ 1 +$	$1\ 1\ 0\ 1 +$	
		$0\ 1\ 0\ 1 +$		

0 - 0 0 +	0 1 - 0 +	0 1 1 - +	- 1 1 1 +
0 0 - 0 +	0 1 0 - +	- 0 1 1 +	1 - 1 1 +
	0 - 1 0 +	0 1 - 1 +	1 1 - 1 +
	- 0 1 0 +	- 1 0 1 +	
	0 0 1 - +	1 0 - 1 +	
		1 - 0 1 +	

0 - - 0	0 1 - -	- - 1 1	
	- 0 1 -	- 1 - 1	
	0 - 1 -	1 - - 1	

Последняя таблица содержит только простые импликанты. Матрица Квайна для нахождения минимальной ДНФ частичной функции показана в таблице 5.10.

Третий этап минимизации частично определенной булевой функции — переход от минимальной ДНФ к скобочной форме, не отличается от соответствующего этапа минимизации полностью определенной булевой функции.

5.8. Минимизация систем полностью определенных булевых функций

Система полностью определенных булевых функций представляет собой множество полностью определенных булевых функций, имеющих общее множество аргументов.

Пусть задана система полностью определенных булевых функций, представленных в ДНФ:

$$f_1(x_1, x_2, x_3) = x_1 \bar{x}_3 \vee x_1 \bar{x}_2 \vee \bar{x}_1 x_3;$$

$$f_2(x_1, x_2, x_3) = x_1 \bar{x}_2 \vee \bar{x}_1 x_3 \vee \bar{x}_1 x_2;$$

$$f_3(x_1, x_2, x_3) = x_1 x_2 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3.$$

Все различные элементарные конъюнкции системы функций объединим в множество K , которое назовем *полным множеством элементарных конъюнкций* системы функций. В нашем случае:

$$K = \{x_1 \bar{x}_3, x_1 \bar{x}_2, x_1 x_3, \bar{x}_1 x_2, x_1 x_2, \bar{x}_1 \bar{x}_2 \bar{x}_3\}.$$

Система ДНФ булевых функций называется *минимальной*, если ее полное множество элементарных конъюнкций содержит минимальное количество букв, а каждая ДНФ системы включает минимальное число элементарных конъюнкций наименьшего ранга. При этом ДНФ представления булевой функции в минимальной системе в общем случае не совпадает с ее минимальной ДНФ.

Минимизация систем полностью определенных булевых функций может производиться по алгоритму, аналогичному алгоритму метода Квайна с небольшими отличиями. Алгоритм минимизации следующий.

1. Построить полное множество K элементарных конъюнкций минимизируемой системы функций (каждая из функций системы представлена в СДНФ). Каждой конституенте единицы множества K присвоить признак, содержащий номера функций системы, в которые входит рассматриваемая конституента.

2. Произвести минимизацию СДНФ функции f , конституентами единицы которой являются все элементы множества K . При выполнении склеивания двух конституент единицы каждой вновь образуемой элементарной конъюнкции присвоить признак, состоящий из номеров функций, общих для двух склеиваемых конституент (см. примеры). Последнее справедливо и для двух склеиваемых элементарных конъюнкций с признаками. Если признаки склеиваемых конституент не содержат общих номеров, то склеивание не производится. Поглощение производится только для элементарных конъюнкций с одинаковыми признаками. Полученные в результате склеивания и поглощения конъюнкции называются *простыми импликантами системы функций*.

3. Построить импликантную матрицу функции f , аналогичную матрице Квайна, с той разницей, что для каждой конституенты единицы выделяется столько столбцов, сколько различных номеров функций содержит ее признак. Покрытие матрицы импликантами производится аналогично методу Квайна.

Пусть система булевых функций задана таблицей истинности (табл. 5.11). Найдем минимальную ДНФ системы булевых функций.

Таблица 5.11
Таблица истинности

x_1	x_2	x_3	f_1	f_2
0	0	0	1	1
0	0	1	0	0
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

Представим каждую из функции системы в СДНФ:

$$f_1 = \overline{x_1} \overline{x_2} \overline{x_3} \vee \overline{x_1} \overline{x_2} x_3 \vee x_1 \overline{x_2} \overline{x_3} \vee x_1 x_2 x_3;$$

$$f_2 = \overline{x_1} \overline{x_2} x_3 \vee \overline{x_1} x_2 \overline{x_3} \vee \overline{x_1} x_2 x_3 \vee x_1 \overline{x_2} \overline{x_3}.$$

1. Построим полное множество K элементарных конъюнкций полученной системы, приписывая каждой конституенте единицы признак входления в функции f_1 и f_2 :

$$K = \{\overline{x_1} \overline{x_2} \overline{x_3}(1,2), \overline{x_1} \overline{x_2} x_3(1,2), \overline{x_1} x_2 \overline{x_3}(1), x_1 x_2 x_3(1), \overline{x_1} x_2 \overline{x_3}(2), \overline{x_1} x_2 x_3(2)\}.$$

2. Построим СДНФ функции f . Для удобства выполнения склеивания пронумеруем каждую конституенту единицы из СДНФ функции f и выполним все склеивания:

$$f = \overline{x_1} \overline{x_2} \overline{x_3}(1,2) \vee \overline{x_1} \overline{x_2} x_3(1,2) \vee x_1 \overline{x_2} \overline{x_3}(1) \vee x_1 \overline{x_2} x_3(1) \vee \overline{x_1} x_2 \overline{x_3}(2) \vee \overline{x_1} x_2 x_3(2).$$

$$1-5: \overline{x_1} \overline{x_2} \overline{x_3}(1,2) \vee \overline{x_1} x_2 \overline{x_3}(2) \vee \overline{x_1} x_3(2);$$

$$2-4: x_1 \overline{x_2} x_3(1,2) \vee x_1 x_2 \overline{x_3}(1) \vee x_1 x_3(1);$$

$$3-4: x_1 x_2 \overline{x_3}(1) \vee x_1 x_2 x_3(1) \vee x_1 x_2(1);$$

$$5-6: \overline{x_1} x_2 \overline{x_3}(2) \vee \overline{x_1} x_2 x_3(2) \vee \overline{x_1} x_2(2).$$

После проведения всех поглощений, с учетом признака каждой конъюнкции, получим:

$$f = \overline{x_1} \overline{x_2} \overline{x_3}(1,2) \vee \overline{x_1} \overline{x_3}(2) \vee x_1 \overline{x_2} \overline{x_3}(1,2) \vee x_1 x_3(1) \vee x_1 x_2(1) \vee \overline{x_1} x_2(2).$$

Дальнейшие склеивания и поглощения невозможны. Получены простые импликанты минимизируемой системы булевых функций.

3. Строим импликантную матрицу (табл. 5.12). Столбцы матрицы помечены конституентами единицы из СДНФ функции f . Для каждой конституенты единицы отводим столько столбцов матрицы, сколько различных номеров функций содержит признак конституенты. Строки матрицы помечаем простыми импликантами системы булевых функций. Импликанты $\overline{x_1} \overline{x_2} \overline{x_3}(1,2)$, $x_1 \overline{x_2} x_3(1,2)$, $x_1 x_2(1)$, $\overline{x_1} x_2(2)$ покрывают все конституенты единицы из СДНФ функции f . В соответствии с этим имеем

$$f = \overline{x_1} \overline{x_2} \overline{x_3}(1,2) \vee x_1 \overline{x_2} x_3(1,2) \vee x_1 x_2(1) \vee \overline{x_1} x_2(2).$$

Выделив для функции f_i импликанты с признаком, включающим i , получим следующую минимальную дизъюнкцию нормальной формуры системы функций:

$$f_1 = \overline{x_1} \overline{x_2} \overline{x_3} \vee \overline{x_1} x_2 \overline{x_3} \vee x_1 x_2;$$

$$f_2 = \overline{x_1} x_2 \overline{x_3} \vee x_1 \overline{x_2} x_3 \vee \overline{x_1} x_2.$$

Таблица 5.12
Импликантная матрица Квайна

Простые импликанты системы функций	Конституанты единицы функции f						
	$\overline{x_1 x_2 x_3}$		$\overline{x_1} \overline{x_2} x_3$		$x_1 \overline{x_2} \overline{x_3}$		$x_1 x_2 \overline{x_3}$
	1	2	1	2	1	1	2
$\overline{x_1} \overline{x_2} \overline{x_3}$ (1,2)	+	+					
$\overline{x_1} x_3$ (2)		+					+
$x_1 \overline{x_2} x_3$ (1,2)			+	+			
$x_1 x_3$ (1)			+			+	
$x_1 x_2$ (1)					+	+	
$\overline{x_1} x_2$ (2)							+
							+

5.9. Минимизация систем частично определенных булевых функций

Минимизация систем частично определенных булевых функций заключается в преобразовании исходной системы к виду, содержащему минимальное число аргументов, минимальное число конъюнкций наименьшего ранга в полном множестве конъюнкций.

Возможность уменьшения числа аргументов системы булевых функций появляется в том случае, если значения функций системы не изменяются при произвольных изменениях значений некоторых аргументов. Такие аргументы называются *фиктивными*.

Для частично определенных булевых функций одни и те же аргументы могут быть фиктивными и нефикативными. Например, для системы функций, заданной табл. 5.13, это все, кроме x_3 . Действительно, если вычеркнуть любой столбец таблицы, кроме третьего, то среди наборов значений оставшихся аргументов не найдется таких двух одинаковых наборов, на которых функции принимают различные значения. После вычеркивания некоторых столбцов отдельные аргументы могут стать нефикативными.

Если вычеркнуть столбцы аргументов x_4 , x_5 , x_6 в табл. 5.13, то аргументы x_1 и x_2 перестают быть фиктивными, а таблица истинности сокращается до табл. 5.14 и система функций становится полностью определенной.

Таблица 5.13
Система функций

№	x_1	x_2	x_3	x_4	x_5	x_6	f_1	f_2
1	0	0	1	0	1	1	0	1
2	0	1	1	0	0	1	1	0
3	0	0	0	1	0	0	0	0
4	0	1	0	0	0	0	1	1
5	1	0	0	0	0	1	1	1
6	1	1	0	1	0	0	0	0
7	1	0	1	0	0	1	1	0
8	1	1	1	0	0	0	0	1

Минимально возможное число аргументов системы булевых функций можно найти следующим образом. По исходной табл. 5.13 образуем все пары входных наборов, на которых хотя бы одна из функций системы принимает противоположное значение. Такими парами являются:

(1, 2); (1, 3); (1, 4); (1, 5); (1, 6); (1, 7); (2, 3); (2, 4); (2, 5); (2, 6); (2, 8); (3, 4); (3, 5); (3, 6); (3, 7); (3, 8); (4, 6); (4, 7); (4, 8); (5, 6); (5, 7); (5, 6); (6, 7); (6, 8); (7, 8).

Для каждой полученной пары наборов укажем множество тех аргументов, значениями которых различаются наборы рассматриваемой пары. Так, наборы, входящие в пару (1, 2), отмечаются значениями переменных x_2, x_5 . Сведем полученные результаты в матрицу, столбцы которой отмечаются парами наборов, а строки — аргументами системы поставим крестик, если переменная, отмечающая 1-ю строку, имеет различные значения в паре наборов, отмечающих 1-й столбец матрицы. Полученная матрица называется матрицей различий, а каждая ее строка — вектором различий. Для рассматриваемого примера матрица различий приведена в табл. 5.15.

Таблица 5.15
Матрица различий

	1	1	1	1	1	1	2	2	2	2	2	3	3	3	3	4	4	4	5	5	5	6	6	7	
x_1																									
x_2	+		+		+		+		+		+		+		+		+		+		+		+		
x_3		+	+	+	+		+	+	+		+		+		+		+		+		+		+		
x_4	+			+		+		+		+	+		+		+		+		+		+		+		
x_5	+	+	+	+	+	+																			
x_6	+	+		+		+	+		+	+		+	+			+		+		+	+	+	+	+	

Таблица 5.14
Система функций

№	x_1	x_2	x_3	f_1	f_2
1	0	0	1	0	1
2	0	1	1	1	0
3	0	0	0	0	0
4	0	1	0	1	1
5	1	0	0	1	1
6	1	1	0	0	0
7	1	0	1	0	1
8	1	1	1	0	1

Найдем минимальное покрытие столбцов матрицы строками. Для этой цели можно, например, использовать метод Петрика. Конъюнктивное представление таблицы имеет вид:

$$\begin{aligned}
 F = & (x_2 \vee x_5)(x_2 \vee x_4 \vee x_5 \vee x_6)(x_2 \vee x_3 \vee x_5 \vee x_6)(x_2 \vee x_3 \vee x_5) \wedge \\
 & \wedge (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)(x_1 \vee x_5)(x_2 \vee x_3 \vee x_4 \vee x_6)(x_3 \vee x_6) \wedge \\
 & \wedge (x_1 \vee x_2 \vee x_3)(x_1 \vee x_3 \vee x_4 \vee x_6)(x_1 \vee x_6)(x_2 \vee x_4)(x_1 \vee x_4 \vee x_6) \wedge \\
 & \wedge (x_1 \vee x_3 \vee x_4 \vee x_6)(x_1 \vee x_2 \vee x_3 \vee x_4)(x_1 \vee x_4)(x_1 \vee x_2 \vee x_3 \vee x_6) \wedge \\
 & \wedge (x_1 \vee x_3)(x_2 \vee x_4 \vee x_6)x_3(x_2 \vee x_3 \vee x_6)(x_2 \vee x_3 \vee x_4 \vee x_6)(x_3 \vee x_4) \wedge \\
 & \wedge (x_2 \vee x_4).
 \end{aligned}$$

После проведения преобразований получим:

$$F = x_1 x_2 x_3 \vee x_3 x_4 x_5 x_6$$

Следовательно, минимальным по числу строк покрытием является множество $\{x_1, x_2, x_3\}$. Минимальное покрытие и представляет минимально возможное число аргументов системы булевых функций.

Система булевых функций от аргументов x_1, x_2, x_3 является полностью определенной, поэтому для ее минимизации может быть использован метод минимизации систем полностью определенных булевых функций. В результате применения метода получим

$$\begin{aligned}
 f_1 &= \overline{x_1} \overline{x_2} \vee x_1 \overline{x_2}; \\
 f_2 &= \overline{x_1} \overline{x_2} x_3 \vee \overline{x_1} x_2 \overline{x_3} \vee x_1 \overline{x_2} \overline{x_3} \vee x_1 x_2 x_3.
 \end{aligned}$$

Если в качестве нефиксивных аргументов принять x_3, x_4, x_5, x_6 , то получим систему частично определенных булевых функций, представленную в табл. 5.16.

Минимизацию системы частично определенных булевых функций выполним в три этапа:

- 1) нахождение простых импликант системы частично определенных булевых функций;
- 2) нахождение минимального покрытия импликантной матрицы Квайна системы частично определенных булевых функций;
- 3) формирование минимальной системы булевых функций.

ЭТАП 1. Простой импликантой системы частично определенных булевых функций F называется конъюнкция K , если:

- 1) она накрывает хотя бы один набор a , на котором $f_i(a) = 1$;
- 2) не накрывает ни одного набора a , на котором $f_i(a) = 0$;
- 3) любая конъюнкция, полученная из нее вычеркиванием переменных, накрывает некоторый набор a , на котором $f_i(a) = 0$.

Здесь $i \in \{1, 2, \dots, n\}$, где n — количество функций в системе.

Для нахождения всех простых импликант удобно пользоваться представлением системы частично определенных булевых функций в виде таблиц T_1 и T_0 (табл. 5.17), в первой из которых перечислены наборы аргументов, на которых хотя бы одна из функций системы принимает единичное значение, а во второй — наборы аргументов, на которых хотя бы одна из функций системы принимает нулевое значение. Наборы таблицы T_1 отмечены номерами функций, которые на этих наборах принимают единичное значение. Наборы таблицы T_0 отмечены номерами функций, которые на этих наборах принимают нулевое значение.

Таблица 5.16
Система функций

x_3	x_4	x_5	x_6	f_1	f_2
1	0	1	1	0	1
1	0	0	1	1	0
0	1	0	0	0	0
0	0	0	0	1	1
0	0	0	1	1	1
1	0	0	0	0	1

Таблица 5.17
Таблицы T_1 и T_0 системы функций

x_3	x_4	x_5	x_6		x_3	x_4	x_5	x_6	
1	0	1	1	(2)	1	0	1	1	(1)
1	0	0	1	(1)	1	0	0	1	(2)
0	0	0	0	(1,2)	0	1	0	0	(1, 2)
0	0	0	1	(1, 2)	0	0	0	0	(1)
1	0	0	0	(2)	1	0	0	0	

Процесс нахождения всех простых импликант системы частично определенных булевых функций проиллюстрирован таблицей:

+1	0	1	1	(2)	+	-	0	1	1	(2)	+	-	-	1	-	(2)
+1	0	0	1	(1)	+	1	-	1	1	(2)	+	-	0	1	-	(2)
+0	0	0	0	(1, 2)	+	1	0	1	-	(2)	+	1	-	1	-	(2)
+0	0	0	1	(1, 2)	+	-	0	0	1	(1)	-	-	0	1	(1)	
+1	0	0	0		+	1	-	0	1	(1)	0	0	-	-	(1, 2)	
					+	0	0	-	0	(1, 2)	0	-	-	1	(1, 2)	
					+	0	0	0	-	(1, 2)	-	0	-	0	(2)	
					+	0	-	0	1	(1, 2)	1	-	-	0	(2)	
					+	0	0	-	1	(1, 2)						
					+	-	0	0	0	(2)						
					+	-	0	0	0	(2)						
					+	1	0	-	0	(2)						

В первой колонке помещены все наборы таблицы T_1 с признаками. Во второй колонке содержатся все возможные наборы, образованные из наборов первой колонки заменой одного разряда прочерком и обладающие тем свойством, что они не покрывают ни одного набора из таблицы T_0 , имеющего хотя бы один общий признак с полученным набором. При замене разряда прочерком признак набора не изменяется. В наборе 1011 (2) нельзя поставить прочерк в третьем разряде, так как полученный

набор 10-1 (2) покрывает набор 1001 (2) из таблицы T_0 , но можно поставить прочерк во втором разряде, несмотря на то, что полученный набор 1-11 (2) покрывает набор 1011 (1) из таблицы T_0 . Это покрытие не имеет значения, так как признаки наборов 1-11(2) и 1011(1) не пересекаются. Наборы первой колонки, покрываемые наборами второй, помечаются знаком “+”, аналогично заполняются третья и четвертая колонки. В четвертой колонке нельзя вычеркнуть ни одного разряда, и процесс завершается. Все помеченные наборы таблицы соответствуют простым импликантам системы частично определенных булевых функций.

ЭТАП 2. Строим импликантную матрицу Квайна системы частично определенных булевых функций. Столбцы матрицы помечаем наборами таблицы T_1 . Для каждого набора отводим столько столбцов матрицы, сколько различных номеров функций содержит его признак. Строки матрицы помечаем простыми импликантами. Матрицу заполняем с учетом признаков. Матрица Квайна представлена в табл. 5.18.

Таблица 5.18
Импликантная матрица Квайна

	1011	1001	0000		0001		1000
	2	1	1	2	1	2	2
--01 (1)		+			+		
00-- (1, 2)			+	+	+	+	
0-1 (1, 2)					+	+	
-0-0 (2)				+			+
1--0 (2)							+
--1- (2)	+						

Импликанты --1-(2), --01(1), 00--(1,2), 1--0(2) образуют минимальное покрытие столбцов матрицы Квайна.

ЭТАП 3. Выделив для функции f_i импликанты с признаком, включающим i , получаем минимальную ДНФ системы частично определенных булевых функций:

$$f_1 = \neg\neg 1 \vee 00--;$$

$$f_2 = \neg\neg 1 \vee 00-- \vee 1--$$

или в буквенном виде:

$$f_1 = \overline{x_5} \overline{x_6} \vee \overline{x_3} \overline{x_4};$$

$$f_2 = x_5 \vee \overline{x_3} \overline{x_4} \vee x_3 \overline{x_6}.$$

Для минимизации системы частично определенных булевых функций можно использовать *модифицированный метод минимизации систем полностью определенных булевых функций*. Модификация ме-

тода заключается в том, что все функции системы на неопределенных наборах доопределяются единичным значением. Для доопределенной системы находятся все ее простые импликанты. При составлении матрицы Квайна строки отмечаются полученными простыми импликантами, а столбцы — конституентами единицы системы частично определенных булевых функций, т.е. строками таблицы T_1 с учетом признаков. Из импликант, образующих минимальное покрытие столбцов матрицы Квайна, формируются функции минимальной ДНФ системы частично определенных булевых функций.

В заключении следует отметить, что нахождение минимальной ДНФ системы частично определенных булевых функций при наличии фиктивных аргументов не является решением задачи о минимальном числе аргументов, несмотря на то, что их количество может уменьшиться. Так, если для системы функций (табл. 5.13) сразу начинать поиск минимальной ДНФ, то можно получить систему ДНФ, зависящую от четырех переменных и содержащую четыре простых импликанты.

5.10. Графовые способы представления булевых функций

Графовыми представлениями булевых функций являются бинарные деревья, бинарные графы и синтаксические деревья.

5.10.1. Бинарные деревья и бинарные графы

Бинарное дерево булевой функции содержит начальную вершину (корень), в которую не входит ни одна дуга и выходят две дуги, множество условных вершин, в которые входит одна дуга и выходят две дуги, и множество заключительных вершин (листьев), в которые входит одна дуга и ни одна не выходит. Начальная и условные вершины на диаграмме изображаются кружочками, а заключительные — прямоугольниками. В начальной и в условных вершинах записываются аргументы функции, а в заключительных — значения функции. Дуги, выходящие из начальной и условных вершин, отмечаются нулем и единицей. На рис.5.1 приведен пример бинарного дерева булевой функции четырех аргументов.

Бинарное дерево булевой функции может быть построено исходя из любой рассмотренной ранее формы представления булевой функции. Алгоритм построения бинарного дерева основан на многократном разложении Шеннона булевой функции по одной переменной и заключается в следующем:

- 1) построить корень дерева и приписать к нему булеву функцию f ;
- 2) выбрать переменную разложения x_i , записать ее в корень дерева и выполнить разложение Шеннона по этой переменной

$$f(x_1, x_2, \dots, x_i, \dots, x_n) = \bar{x}_i f(x_1, x_2, \dots, 0, \dots, x_n) \vee x_i f(x_1, x_2, \dots, 1, \dots, x_n).$$

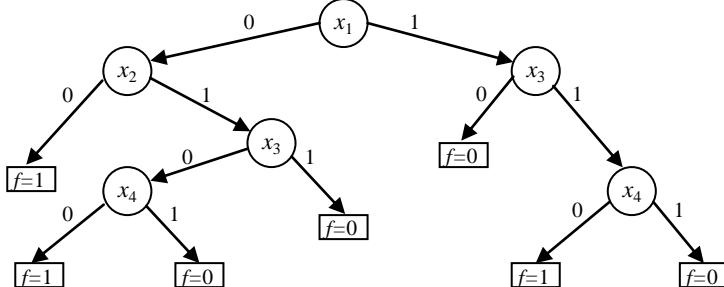


Рис.5.1. Бинарное дерево булевой функции

Если коэффициент $f(x_1, x_2, \dots, 0, \dots, x_n)$ представляет собой константную функцию, то ввести лист дерева, записать в нем значение $f(x_1, x_2, \dots, 0, \dots, x_n)$, соединить дугой корень дерева с введенной вершиной и отметить ее нулем. Если коэффициент $f(x_1, x_2, \dots, 1, \dots, x_n)$ представляет собой не константную функцию, то ввести условную вершину дерева, приписать к ней функцию $f(x_1, x_2, \dots, 1, \dots, x_n)$, соединить дугой корень дерева с введенной вершиной и отметить ее единицей. Если коэффициент $f(x_1, x_2, \dots, 1, \dots, x_n)$ представляет собой константную функцию, то ввести лист дерева, записать в нем значение $f(x_1, x_2, \dots, 1, \dots, x_n)$, соединить дугой корень дерева с введенной вершиной и отметить его единицей, если коэффициент $f(x_1, x_2, \dots, 1, \dots, x_n)$ представляет собой не константную функцию, то ввести условную вершину дерева, приписать к ней функцию $f(x_1, x_2, \dots, 1, \dots, x_n)$, соединить дугой корень дерева с введенной вершиной и отметить ее единицей;

3) пока существуют концевые условные вершины, для каждой из них выполнить действия, аналогичные действиям, выполненным над корнем дерева.

Бинарное дерево булевой функции, зависящей от n аргументов, обладает следующими свойствами:

- количество условных вершин лежит в пределах от n до $2^n - 1$;
- количество листьев лежит в пределах от $n + 1$ до 2^n ;
- глубина дерева лежит в пределах от $\lceil \log_2(n + 1) \rceil$ до n .

Бинарный граф булевой функции отличается от бинарного дерева тем, что он содержит только две заключительные вершины и в каждую условную и заключительную вершины входят не менее одной дуги. На рис.5.2 приведен пример бинарного графа булевой функции четырех аргументов.

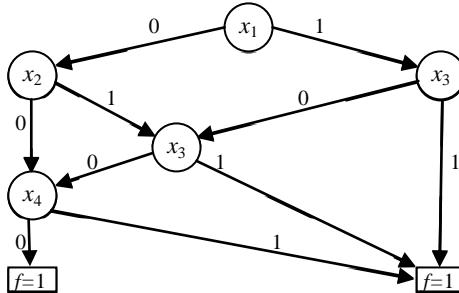


Рис.5.2. Бинарный граф булевой функции

Алгоритм построения бинарного графа булевой функции следующий:

- 1) строится начальная вершина, к которой приписывается функция f ;
- 2) выбирается переменная разложения x_i и записывается в вершине, выполняется разложение Шеннона функции f по переменной x_i

$$f(x_1, x_2, \dots, x_i, \dots, x_n) = x_i y_0 \vee \bar{x}_i y_1,$$

где y_0 и y_1 — коэффициенты разложения:

$$y_0 = f(x_1, x_2, \dots, 0, \dots, x_n);$$

$$y_1 = f(x_1, x_2, \dots, 1, \dots, x_n).$$

- 3) если в графе существует условная вершина, к которой приписана функция y_0 (или y_1), то проводится дуга в эту вершину и отмечается нулем (единицей); если в графе не существует условной вершины, к которой приписана функция y_0 (или y_1) и функция y_0 (или y_1) не является константой, то вводится новая условная вершина, к этой вершине проводится дуга, отмеченная нулем (или единицей) и к ней приписывается функция y_0 (или y_1); если функция y_0 (или y_1) является константой, то проводится дуга, отмеченная нулем (или единицей) в заключительную вершину $f=y_0$ ($f=y_1$), если такая существует, или, в противном случае, вводится такая заключительная вершина;
- 4) пп. 2 и 3 выполняются для всех концевых условных вершин.

Процесс построения бинарного графа функции $f = x_1x_3 \vee \bar{x}_1\bar{x}_2x_3$ представлен на рис.5.3.

Бинарный граф булевой функции, зависящей от n аргументов, обладает следующими свойствами:

- бинарный граф не содержит циклов;
- количество условных вершин лежит в пределах от n до $2^n - 1$;
- имеет две заключительные вершины;
- глубина графа лежит в пределах от $\lceil \log_2(n + 1) \rceil$ до n .

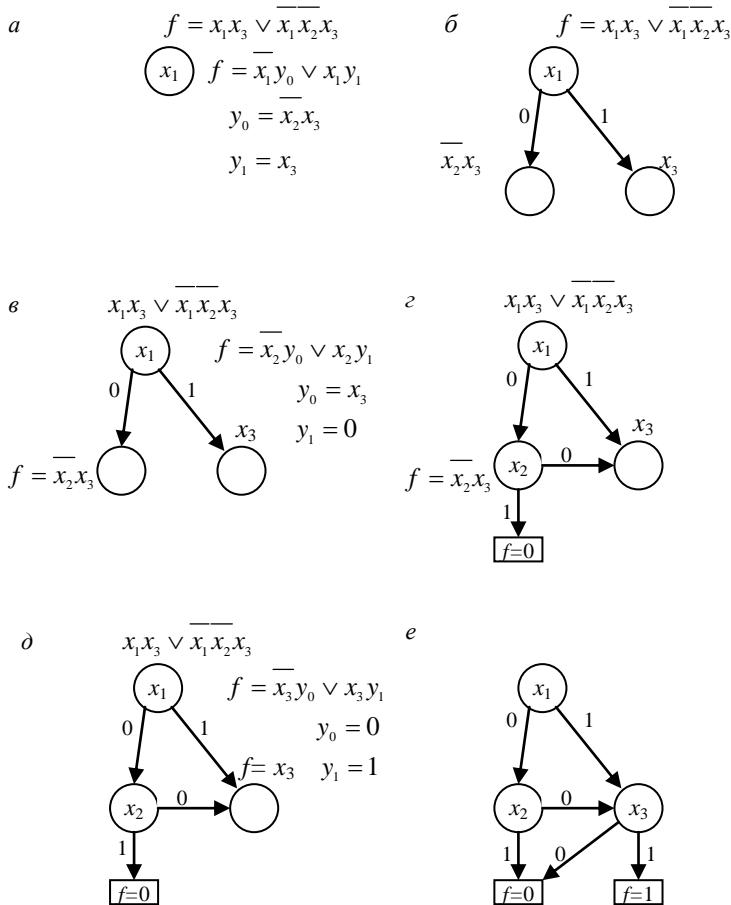


Рис.5.3. Процесс построения бинарного графа

Бинарный граф системы булевых функций отличается от бинарного графа одной функции тем, что он может содержать более двух заключительных вершин, в которых записаны значения всех функций системы. На рис.5.4 приведен пример бинарного графа системы из трех булевых функций четырех аргументов.

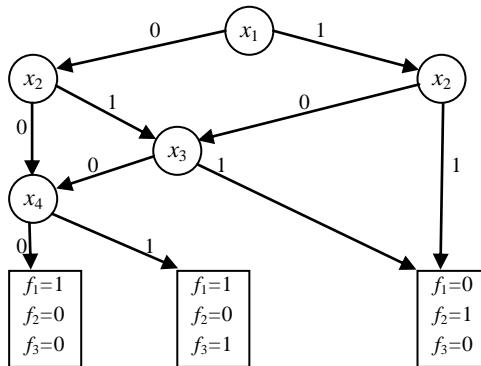


Рис.5.4. Бинарный граф системы булевых функций

Алгоритм построения бинарного графа системы булевых функций аналогичен алгоритму построения бинарного графа одной функции, за исключением того, что переменная разложения выбирается для всей системы и выполняется разложение Шеннона по выбранной переменной каждой функции системы, образуя две новые системы функций, включающие в себя коэффициенты разложения всех функций. К вводимым условным вершинам приписываются новые системы функций, в которых хотя бы одна функция не является константой. Система из константных функций образует заключительную вершину графа.

Бинарный граф системы из k булевых функций, зависящих от n аргументов, обладает следующими свойствами:

- бинарный граф не содержит циклов;
- количество условных вершин не зависит от количества функций в системе и лежит в пределах от n до $2^n - 1$;
- количество заключительных вершин зависит от количества функций в системе и лежит в пределах от 2 до 2^k ;
- глубина графа не зависит от количества функций в системе и лежит в пределах от $\lceil \log_2(n + 1) \rceil$ до n .

Булеву функцию можно представить различными бинарными графами, которые могут различаться количеством условных вершин. Количество вершин зависит от выбора переменной разложения на каждом шаге. Количество вершин в графе вероятно будет меньше, если на каждом шаге в качестве переменной разложения будем выбирать такую переменную, которая позволяет получить коэффициенты разложения с минимальным количеством букв.

5.10.2. Синтаксические деревья

Булеву функцию, заданную формулой, например, в произвольной скобочной форме, можно представить *синтаксическим деревом*, в листьях которого записываются аргументы функции, а в остальных вершинах — операции. Не нарушая общности, будем считать, что в формулах могут использоваться операции конъюнкция, дизъюнкция и отрицание.

Построить синтаксическое дерево булевой функции по формуле E можно по следующим правилам:

1. Построить вершину — корень дерева, и приписать к нему формулу E . Построенную вершину считать необработанной.

2. Пока есть необработанные вершины, выбрать вершину для обработки, к которой приписана формула E и обработать ее следующим образом:

1. Если $E = x$, где x — аргумент функции, то записать в вершину аргумент x и считать ее листом:

$$\textcircled{ } \stackrel{x}{\Rightarrow} \textcircled{x}$$

2. Если формулу E можно представить в виде $E_1 \wedge E_2$, то в вершину записать знак конъюнкции \wedge , добавить к вершине левого и правого сына, к левому сыну приписать E_1 , а к правому — E_2 :

$$\begin{array}{ccc} \textcircled{ } & \stackrel{E_1 \wedge E_2}{\Rightarrow} & \begin{array}{c} \textcircled{\wedge} \\ \swarrow \quad \searrow \\ \textcircled{E_1} \quad \textcircled{E_2} \end{array} \end{array}$$

3. Если формулу E можно представить в виде $E_1 \vee E_2$, то в вершину записать знак дизъюнкции \vee , добавить к вершине левого и правого сына, к левому сыну приписать E_1 , а к правому — E_2 :

$$\begin{array}{ccc} \textcircled{ } & \stackrel{E_1 \vee E_2}{\Rightarrow} & \begin{array}{c} \textcircled{\vee} \\ \swarrow \quad \searrow \\ \textcircled{E_1} \quad \textcircled{E_2} \end{array} \end{array}$$

4. Если формулу E можно представить в виде \bar{E} , то в вершину записать знак отрицания \neg , добавить к вершине сына и приписать к нему E :

$$\begin{array}{ccc} \textcircled{ } & \stackrel{\bar{E}}{\Rightarrow} & \begin{array}{c} \textcircled{\neg} \\ \downarrow \\ \textcircled{E} \end{array} \end{array}$$

Рассмотрим пример построения синтаксического дерева булевой функции, заданной формулой $x_1(\overline{x_2x_3} \vee \overline{x_2}x_3)$.

Строим корень дерева и приписываем к нему исходную формулу (рис. 5.5, а). Эту формулу можно представить как конъюнкцию E_1 и E_2 , где $E_1 = x_1$ и $E_2 = \overline{x_2x_3} \vee \overline{x_2}x_3$. В вершину записываем знак конъюнкции, к вершине пристраиваем левого и правого сына, к левому сыну приписываем x_1 , а к правому — $\overline{x_2x_3} \vee \overline{x_2}x_3$ (рис. 5.5, б).

Обрабатываем вершину, к которой приписана формула x_1 , представляющая собой аргумент функции. Записываем в вершину x_1 (рис. 5.5, в).

Обрабатываем вершину, к которой приписана формула $\overline{x_2x_3} \vee \overline{x_2}x_3$. Этую формулу можно представить в виде дизъюнкции E_1 и E_2 , где $E_1 = \overline{x_2x_3}$ и $E_2 = \overline{x_2}x_3$. В вершину записываем знак дизъюнкции, к вершине пристраиваем левого и правого сына, к левому сыну приписываем $\overline{x_2x_3}$, а к правому — $\overline{x_2}x_3$ (рис. 5.5, г).

Следующие три шага построения синтаксического дерева булевой функции представлены на рис. 5.5, д — ж, и результат — на рис. 5.5, з.

Синтаксическое дерево можно преобразовать в формулу по следующим правилам:

1) если в вершине записан аргумент x , то приписать к вершине формулу x ;

2) если в вершине записана операция дизъюнкция, к левому сыну приписана формула E_1 , к правому — формула E_2 , то к вершине приписать формулу $E_1 \vee E_2$;

3) если в вершине записана операция отрицание и к сыну приписана формула E , то к вершине приписать формулу \overline{E} ;

4) если в вершине записана операция конъюнкция, к левому сыну приписана формула E_1 , к правому — формула E_2 , то к вершине приписать формулу E_1E_2 , при этом формулы E_1 или E_2 заключаются в скобки, если в вершине, к которой приписана формула E_1 или E_2 записан знак дизъюнкции (операция, меньшего приоритета, чем конъюнкция)

Описанные выше правила применяются, пока в дереве есть хотя бы одна вершина, к которой не приписана формула.

Формула, соответствующая синтаксическому дереву, приписана к корню дерева.

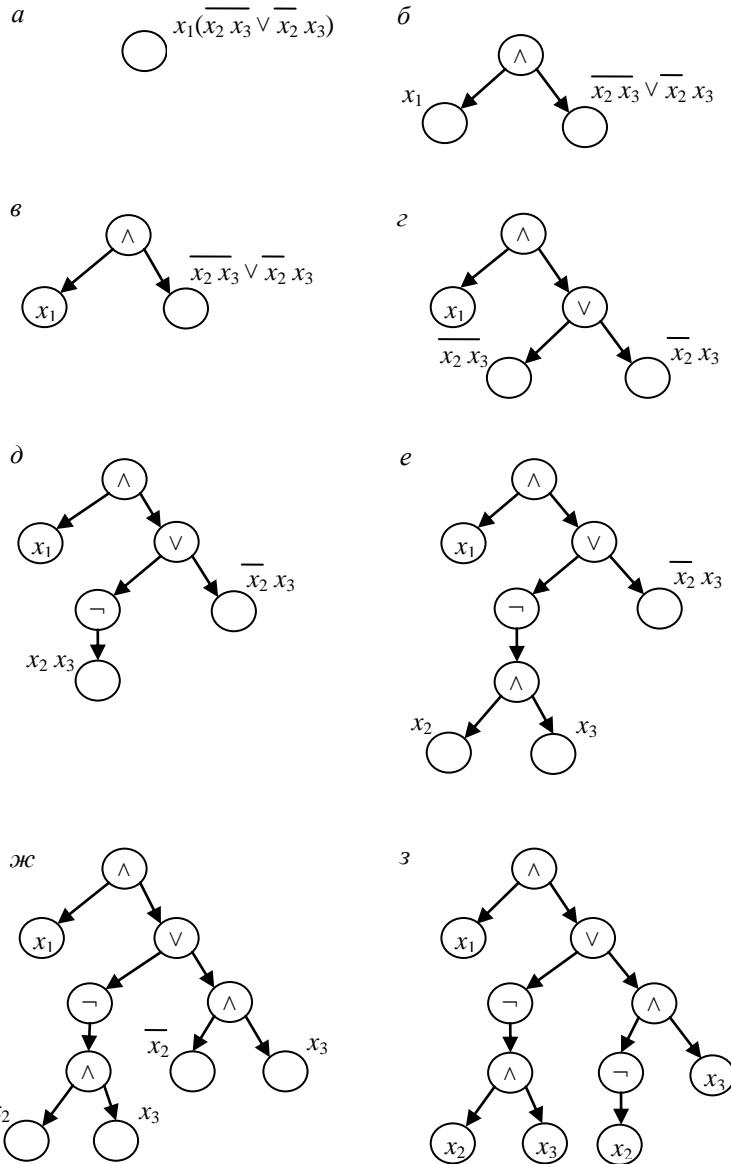


Рис. 5.5. Процесс построения синтаксического дерева

5.11. Программная реализация булевых функций

Под программной реализацией булевой функции будем понимать программу, вычисляющую значение заданной булевой функции на заданном наборе значений аргументов. Пусть набор значений аргументов хранится в массиве X из n элементов и элемент $X[i]$ содержит значение аргумента x_i . Значение функции должно быть получено в переменной F . Алгоритм вычисления зависит от способа представления булевой функции.

5.11.1. Вычисление значения булевой функции по таблице истинности

Если булева функция задана таблицей истинности, то ее можно сохранить в одномерном массиве T из 2^n элементов с индексами от 0 до $2^n - 1$. Элемент $T[i]$ содержит значение функции на i -м наборе аргументов. Для вычисления значения функции необходимо набор значений аргументов (массив X) преобразовать в номер набора. Пусть это преобразование выполняет функция $Index$. Тогда значение булевой функции определяется выражением $F := T[Index(x)]$.

Для того, чтобы избежать вычисление номера набора аргументов, значения булевой функции можно сохранить в n -мерном массиве T , i -й индекс которого соответствует аргументу x_i . Тогда значение булевой функции определяется выражением $F := T[X[1], X[2], \dots, X[n]]$.

5.11.2. Вычисление значения булевой функции по ДНФ

Булеву функцию, заданную в ДНФ, можно закодировать последовательностью целых чисел следующим образом. Просматриваем ДНФ слева направо. Если встречаем аргумент x_i без отрицания, то в последовательность дописываем число i . Если встречаем аргумент x_i с отрицанием, то в последовательность дописываем число $-i$. Если встречаем знак дизъюнкции, то в последовательность дописываем число 0. Знак конъюнкции пропускаем. Так, например, булева функция $f = \overline{x_1}x_2x_4 \vee x_2\overline{x_3} \vee x_3x_4$ кодируется последовательностью $(-1, 2, 4, 0, -2, 3, 0, 3, 4)$. Полученную последовательность можно сохранить в массиве T и использовать для вычисления значения булевой функции по следующему алгоритму.

Алгоритм (рис.5.6) вычисления значения булевой функции по ДНФ.

Вход: X — набор значений аргументов;

T — массив, содержащий ДНФ булевой функции;

KT — количество элементов в массиве T .

Выход: F — значение булевой функции.

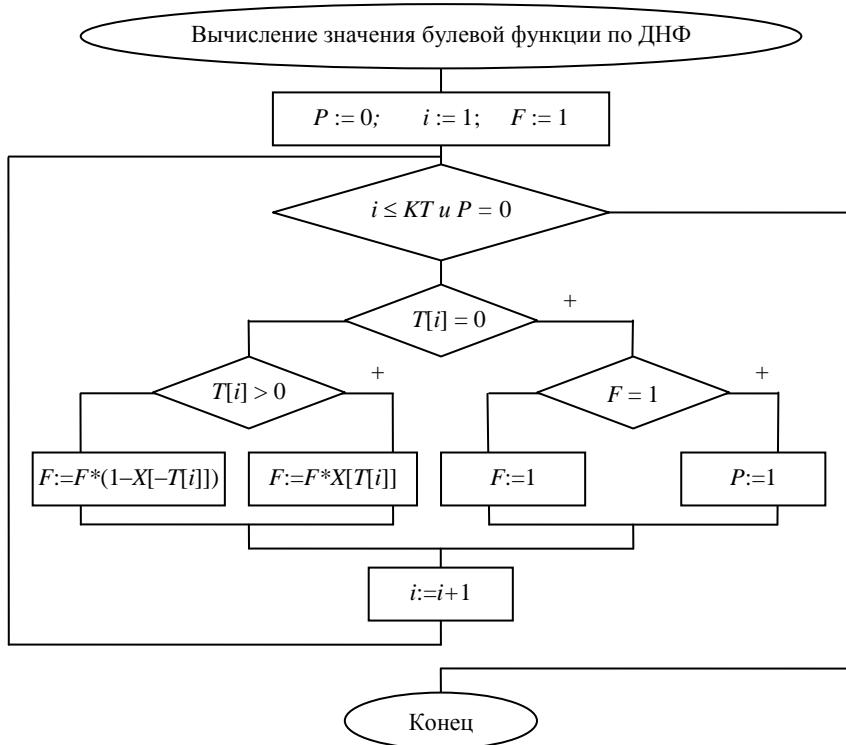


Рис.5.6. Алгоритм вычисления значения булевой функции по ДНФ

5.11.3. Вычисление значения булевой функции по бинарному графу

Если булева функция задана бинарным графом (или деревом), то можно написать программу вычисления ее значения с использованием только команд двух типов:

<условная команда> → <метка> если <аргумент> то <метка> иначе <метка>
 <операторная команда> → <метка> F:=<значение> конец

Для того, чтобы написать программу, необходимо произвольным образом пронумеровать вершины бинарного графа, начиная с начальной вершиной (рис.5.7). Каждой вершине графа соответствует команда программы. Начальной и условным вершинам соответствуют условные команды, а заключительным — операторные. Если в начальной или условной вершине с номером m записан аргумент x_i , дуга, отмеченная нулем,

идет в вершину с номером n , а дуга, отмеченная единицей, идет в вершину с номером k , то такой вершине соответствует команда:

m если x_i то k иначе n

Заключительной вершине с номером l , в которой записано $f = 0$, соответствует команда:

$l F := 0$ конец

Заключительной вершине с номером p , в которой записано $f = 1$, соответствует команда:

$p F := 1$ конец

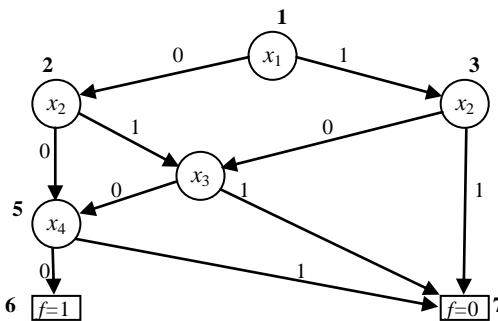


Рис.5.7. Бинарный граф булевой функции с пронумерованными вершинами

Программа вычисления значения булевой функции по бинарному графу (рис.5.7) будет такой:

1 если x_1 то 3 иначе 2

2 если x_2 то 4 иначе 5

3 если x_2 то 7 иначе 4

4 если x_3 то 7 иначе 5

5 если x_4 то 7 иначе 6

6 $F := 1$ конец

7 $F := 0$ конец

Рассмотрим еще один способ вычисления значения булевой функции по бинарному графу. Представим бинарный граф таблицей. Таблица имеет три столбца, нумерация которых начинается с нуля. Строки таблицы соответствуют вершинам графа. Если в начальной или условной вершине с номером m записан аргумент x_i , дуга, отмеченная нулем, идет в вершину с номером n , а дуга, отмеченная единицей, идет в вершину с номером k , то такой вершине соответствует строка таблицы с номером m , нулевой столбец которой содержит n , первый — k и второй — i .

Заключительной вершине с номером l , в которой записано $f = 0$, соответствует строка таблицы с номером l , в которой все столбцы содержат нули. Заключительной вершине с номером p , в которой записано $f = 1$, соответствует строка таблицы с номером p , в которой нулевой и первый столбцы содержат нули, а второй — единицу. Другими словами, если строка соответствует начальной или условной вершине, то нулевой и первый столбцы таблицы содержат номера вершин, в которые идут дуги из вершины, отмеченные соответственно нулем и единицей, а второй — содержит номер аргумента. Если же строка соответствует заключительной вершине, то нулевой и первый столбцы содержат нули, а второй — значение функции. Бинарный граф (рис.5.7) представлен табл. 5.19.

Таблица 5.19
Таблица бинарного графа

	0	1	2
1	2	3	1
2	5	4	2
3	4	7	2
4	5	7	3
5	6	7	4
6	0	0	1
7	0	0	0

Такую таблицу можно сохранить в двумерном массиве T и использовать для вычисления значения булевой функции по следующему алгоритму.

Алгоритм вычисления значения булевой функции по бинарному графу.
Вход: X — набор значений аргументов;

T — таблица бинарного графа булевой функции.

Выход: F — значение булевой функции.

1. $v := 1$;
2. Пока $T[v,0] \neq 0$ выполнять $v := T[v,X[T[v,2]]]$;
3. $F := T[v,2]$.

5.11.4. Вычисление значения булевой функции по синтаксическому дереву

Вычислить значение булевой функции по синтаксическому дереву можно с помощью рекурсивной функции *Вычислить*, в которую передается вершина, являющаяся корнем дерева.

Алгоритм функции *Вычислить (вершина)* следующий:

если в *вершине* записан аргумент функции, то значение функции равно значению аргумента;

если в *вершине* записана операция отрицания, то значение функции равно значению выражения *Вычислить(сын)*;

если в *вершине* записана операция конъюнкции, то значение функции равно значению выражения *Вычислить (левый сын)* \wedge *Вычислить (правый сын)*;

если в *вершине* записана операция дизъюнкции, то значение функции равно значению выражения *Вычислить (левый сын)* \vee *Вычислить (правый сын)*;

Практическое занятие 5.1

Полностью определенные булевые функции

Цель занятия: изучить способы задания булевых функций, методы их минимизации и программной реализации.

Задания

1. Представить булеву функцию f_i (табл. 5.31), где i — номер варианта, совершенной дизъюнктивной нормальной формой, совершенной конъюнктивной нормальной формой и полиномом Жегалкина.
2. Получить сокращенную ДНФ функции f_i методом Квайна — МакКлакси.
3. Получить минимальную ДНФ функции f_i , используя матрицу Квайна.
4. Выполнить скобочную минимизацию минимальной ДНФ функции f_i .
5. Построить бинарный граф функции f_i .
6. Написать программу, вычисляющую значение функции f_i на заданном наборе аргументов по минимальной ДНФ.
7. Написать программу, вычисляющую значение функции f_i на заданном наборе аргументов по бинарному графу.
8. Используя программу п.6, написать программу, строящую таблицу истинности функции f_i . Результат работы программы сравнить с заданием (табл. 5.31).
9. Используя программу п.7, написать программу, строящую таблицу истинности функции f_i . Результат работы программы сравнить с заданием (табл. 5.31).

Практическое занятие 5.2

Чистично определенные булевые функции

Цель занятия: изучить методы минимизации чистично определенных булевых функций.

Задания

1. Получить минимальную ДНФ чистично определенной булевой функции f_i (табл. 5.20), где i — номер варианта, различными способами.
2. Написать программу, строящую таблицу истинности функции f_i . Результат работы программы сравнить с заданием (табл. 5.12).

Таблица 5.20

Таблица истинности полностью определенных булевых функций

x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	1	1	1	0	0	0	1	1	0	1	0	0	0	1
0	0	0	1	1	0	0	0	1	1	1	0	0	1	0	1	1	0	
0	0	1	0	1	0	1	0	1	0	1	0	1	1	0	1	1	0	
0	0	1	1	0	1	1	0	1	0	0	1	1	1	0	0	0	1	
0	1	0	0	0	0	1	0	1	0	1	1	0	1	0	0	1	0	
0	1	0	1	0	1	1	1	0	1	0	1	0	1	1	0	1	0	
0	1	1	0	0	0	0	0	1	1	1	1	0	0	1	0	0	1	
0	1	1	1	1	0	0	1	1	1	1	0	0	0	1	1	0	0	
1	0	0	0	1	1	1	0	0	1	1	0	0	1	0	0	0	1	
1	0	0	1	1	1	0	1	0	0	0	1	1	0	1	1	1	0	
1	0	1	0	0	1	0	1	1	0	0	0	1	1	1	0	0	1	
1	0	1	1	1	0	0	0	1	1	1	0	0	0	1	1	0	0	
1	1	0	0	1	1	0	1	0	0	0	1	1	1	0	1	0	0	
1	1	0	1	0	1	1	0	0	0	1	0	1	0	0	1	0	0	
1	1	1	0	0	1	0	0	0	1	1	0	1	0	1	0	0	1	
1	1	1	1	0	1	1	1	0	1	0	1	0	1	0	0	1	1	

Таблица 5.21

Таблица истинности частично определенных булевых функций

x_1	x_2	x_3	x_4	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	1	0	1	1	0	1	0	0	0	1	1	0
0	0	0	1	1	1	1	0	1	0	0	1	0	1	1	1	0	0	
0	0	1	0	1	-	1	0	-	0	0	1	-	1	-	1	-	0	
0	0	1	1	-	0	-	-	0	-	-	0	1	-	0	-	1	-	
0	1	0	0	-	-	1	1	0	1	0	0	-	1	1	-	-	0	
0	1	0	1	0	1	-	1	-	1	-	0	1	-	-	0	1	1	
0	1	1	0	-	-	-	-	0	1	0	-	1	-	-	-	1	-	
0	1	1	1	-	1	0	1	1	0	-	0	-	1	0	0	-	1	
1	0	0	0	1	-	-	0	-	0	-	1	1	-	1	1	0	-	
1	0	0	1	-	1	0	1	-	1	1	-	0	0	-	0	1	-	
1	0	1	0	0	-	-	1	-	-	1	0	1	0	0	-	-	1	
1	0	1	1	-	1	-	-	1	0	-	0	1	-	-	0	0	1	
1	1	0	0	0	0	-	0	-	0	-	0	-	-	-	0	1	1	
1	1	0	1	1	-	-	0	1	-	-	0	-	-	-	0	1	-	
1	1	1	0	-	-	0	-	1	1	-	1	0	-	1	0	-	0	
1	1	1	1	-	1	-	0	0	1	-	1	1	0	-	0	1	0	

Практическое занятие 5.3

Минимизация систем полностью определенных булевых функций

Цель занятия: научиться минимизировать системы полностью определенных булевых функций, закрепить навыки минимизации полностью определенных булевых функций.

Задания

В табл. 5.20 (см. выше) представлено множество $\{f_1, f_2, \dots, f_{15}\}$ полностью определенных булевых функций. В табл. 5.22 каждому варианту поставлена в соответствие система булевых функций, представляющая собой подмножество множества $\{f_1, f_2, \dots, f_{15}\}$ (см. табл. 5.20).

1. Минимизировать систему полностью определенных булевых функций в соответствии с вариантом задания (см. табл. 5.22).
2. Получить систему минимальных ДНФ булевых функций и сравнить с результатом п.1.

Таблица 5.22

Варианты заданий

Вариант	Система булевых функций
1	f_1, f_2, f_3, f_4
2	f_1, f_3, f_4, f_5
3	f_4, f_5, f_6, f_7
4	f_2, f_3, f_5, f_6
5	f_5, f_6, f_7, f_8
6	f_1, f_2, f_6, f_7
7	f_1, f_4, f_5, f_9
8	f_3, f_4, f_8, f_{10}
9	f_5, f_7, f_9, f_{10}
10	f_8, f_9, f_{10}, f_{11}
11	$f_{10}, f_{11}, f_{12}, f_{14}$
12	$f_{11}, f_{12}, f_{13}, f_{15}$
13	f_7, f_8, f_{13}, f_{14}
14	f_4, f_6, f_{10}, f_{13}
15	f_2, f_4, f_{11}, f_{14}
16	f_3, f_5, f_9, f_{13}
17	f_5, f_6, f_8, f_{15}
18	f_5, f_8, f_{11}, f_{15}
19	f_6, f_7, f_{10}, f_{12}
20	f_7, f_9, f_{13}, f_{14}

Практическое занятие 5.4

Вычисление систем булевых функций

Цель занятия: научиться минимизировать системы частично определенных булевых функций, строить бинарные графы систем булевых функций и использовать их для вычисления значений функций системы.

Задания

В табл. 5.21 (см. выше) представлено множество $\{f_1, f_2, \dots, f_{15}\}$ частично определенных булевых функций. В табл. 5.22 (см. выше) каждому варианту поставлена в соответствие система булевых функций, представляющая собой подмножество множества $\{f_1, f_2, \dots, f_{15}\}$ (см. табл. 5.21).

1. Минимизировать систему частично определенных булевых функций в соответствии с вариантом задания (см. табл. 5.22).
2. Построить бинарный граф системы булевых функций.
3. Написать программу, вычисляющую значение системы булевых функций на заданном наборе аргументов по бинарному графу.

Заключительные вершины бинарного графа системы булевых функций дополнительльно пронумеруем натуральными числами от 1 до k , где k — количество заключительных вершин, и построим таблицу, которая каждой заключительной вершине поставит в соответствие набор значений функций системы. Такую таблицу можно представить двумерным массивом F , в котором элемент F_{ij} содержит значение функции f_j , записанной в заключительной вершине с дополнительным номером i . В основной же таблице в строке, соответствующей заключительной вершине с дополнительным номером i , во втором столбце запишем i . Алгоритм вычисления значения системы булевых функций на заданном наборе аргументов по бинарному графу отличается от алгоритма вычисления значения одной функции тем, что при достижении заключительной вершины (после выхода из цикла) выводятся значения всех функций системы из массива F с использованием дополнительной метки заключительной вершины.

4. Используя программу п.3, написать программу, строящую таблицу истинности системы булевых функций. Результат работы программы сравнить с заданием (табл. 5.21).

Контрольные вопросы и задания

1. Дайте определение булевой функции.
2. Приведите примеры различных таблиц истинности, задающих одну и ту же булеву функцию.
3. Какая булева функция называется элементарной? Сколько их?
4. Что называется функционально полным набором булевых функций?
5. Что называется операцией суперпозиции и суперпозицией функций?
6. Дайте определение классу булевых функций, функционально замкнутого по операции суперпозиции.
7. Дайте определение свойствам булевых функций. Определите свойства заданной булевой функции.
8. Сформулируйте необходимые и достаточные условия функциональной полноты системы булевых функций. Определите, обладает ли функциональной полнотой заданная система функций?
9. Дайте определение конституенты единицы и конституенты нуля.
10. Что представляет собой СДНФ, СКНФ, СПНФ?
11. Что называется элементарной конъюнкцией, элементарной дизъюнкцией? Что представляет собой ДНФ и КНФ? Чем они отличаются от СДНФ и СКНФ.
12. Что называется полиномом Жегалкина? Представьте полиномом Жегалкина булеву функцию, заданную таблицей истинности.
13. Запишите основные законы булевой алгебры. Запишите и докажите истинность правил склеивания, поглощения и вычеркивания.
14. Что такое дизъюнктивное разложение Шеннона? Выполните разложение Шеннона заданной булевой функции по различным аргументам.
15. Что называется импликантой булевой функции? Какая импликанта называется простой?
16. Дайте определение сокращенной ДНФ. Получите сокращенную ДНФ булевой функции, заданной таблицей истинности, различными методами.
17. Дайте определение тупиковой и минимальной ДНФ булевой функции.
18. Что представляет собой матрица Квайна, ядро Квайна?
19. Что такое конъюнктивное представление матрицы Квайна? Как его использовать для получения всех тупиковых ДНФ?
20. Выполните скобочную минимизацию заданной минимальной ДНФ булевой функции.

21. Какая булева функция называется частично определенной?
22. Что называется простой импликантой частично определенной булевой функции? Как найти все простые импликанты частично определенной булевой функции?
23. Какая система ДНФ булевых функций называется минимальной?
24. Что называется простой импликантой системы булевых функций?
25. Модифицируйте метод Квайна — Мак-Класски для получения всех простых импликант системы булевых функций.
26. Как строится импликантная матрица Квайна для системы булевых функций?
27. Какие аргументы системы частично определенных булевых функций называются фиктивными? Как их найти?
28. Что называется простой импликантой системы частично определенных булевых функций? Как найти все простые импликанты системы частично определенных булевых функций?
29. Приведите пример бинарного графа булевой функции. Запишите эту функцию в ДНФ.
30. Постройте различные бинарные графы одной и той же булевой функции.
31. Сколько условных вершин может быть в бинарном графе булевой функции? Какова может быть глубина бинарного графа?
32. Приведите пример бинарного графа системы булевых функций. Запишите эти функции в ДНФ.
33. Постройте различные бинарные графы одной и той же системы булевых функций.
34. Сколько условных и заключительных вершин может быть в бинарном графе системы булевых функций?
35. Какова может быть глубина бинарного графа системы булевых функций?
36. Напишите программы для вычисления значения булевой функции по таблице истинности при различных способах ее хранения.
37. Опишите алгоритм вычисления значения булевой функции по ДНФ.
38. Напишите различные программы для вычисления значения булевой функции по заданному бинарному графу.
39. Разработайте способ хранения и алгоритм вычисления системы булевых функций по таблице истинности.
40. Разработайте способ хранения и алгоритм вычисления системы булевых функций по минимальной системе ДНФ булевых функций.

Заключение

В данном пособии рассмотрены вопросы, соответствующие содержанию дисциплины «Дискретная математика» для студентов, обучающихся по направлениям 09.03.01 «Информатика и вычислительная техника» и 09.03.04 «Программная инженерия». В целом же, дискретная математика представляет собой обширную область знаний, которая значительно превосходит содержание одноименной дисциплины. Например, в пособии не рассмотрены вопросы, касающиеся кодирования информации, логических исчислений, теории алгоритмов, алгебраических структур, формальных языков и грамматик, теории автоматов, сетей Петри и др. Ответы на эти вопросы можно получить при изучении смежных дисциплин или обратившись к специальной литературе.

Библиографический список

1. Асанов, М. Дискретная математика: графы, матроиды, алгоритмы / М.О. Асанов, В.А. Баранский, В.В. Расин. — Ижевск: НИЦ «Регуляя и хаотическая динамика», 2001. — 288 с.
2. Белоусов, А.И. Дискретная математика: учеб. для вузов / А.И. Белоусов, С.Б. Ткачев, под ред. В.С. Зарубина, А.П. Крищенко. — 3-е изд., стер. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. — 744 с. (Сер. Математика в техническом университете; Вып. XIX).
3. Горбатов, В.А. Фундаментальные основы дискретной математики. Информационная математика / В.А. Горбатов.— М.: Физматлит, 2000. — 544 с.
4. Иванов, Б.Н. Дискретная математика. Алгоритмы и программы: учеб. пособие / Б.Н. Иванов. — М.: Лаборатория Базовых Знаний, 2003. — 288 с.
5. Поздняков, С.Н. Дискретная математика: учебник для студ. вузов / С.Н. Поздняков, С.В. Рыбин. — М. : Издательский центр «Академия», 2008. — 448 с.
6. Прилуцкий, М.Х. Дискретная математика. Методические пособия для студентов факультета ВМК, специальности 351400 «Прикладная информатика». Часть 1 /Н. Новгород.Нижег.гос.ун-т, 2007, с.63.
7. Кристофидес, Н. Теория графов. Алгоритмический подход / Н. Кристофидес. — М.: Мир, 1978. — 429 с.
8. Кузнецов, О.П. Дискретная математика для инженера / О.П. Кузнецов, Г.М. Адельсон-Вельский. — 2-е изд., перераб. и доп. — М.: Энергоатомиздат, 1988. — 480с.
9. Кузнецов, О.П.. Дискретная математика для инженера / О.П. Кузнецов. — 3-е изд., перераб. и доп. — СПб.: Лань, 2004. — 400 с.: ил. — (Учебники для вузов. Специальная литература).
10. Липский, В. Комбинаторика для программистов / В. Липский. — М.: Мир, 1988. — 201 с.
11. Муромцев, В.В. Проектирование полнопереборных алгоритмов: учеб. пособие./ В.В. Муромцев. — Белгород: Изд-во БелГТАСМ, 2001. — 67 с.
12. Новиков, Ф.А. Дискретная математика для программистов: учеб. для вузов / Ф.А. Новиков. — 3-е изд. — СПб.: Питер, 2008. — 384 с.: ил. — (Серия «Учебник для вузов»).
13. Прикладная теория цифровых автоматов / К.Г. Самофалов, А.М. Романкевич, В.Н. Валуйский и др. — Киев: Вища школа, 1987. — 357 с.
14. Рейнгольд, Э. Комбинаторные алгоритмы. Теория и практика / Э. Рейнгольд, Ю. Нивергельт, Н. Дэо. — М.: Мир, 1980. — 476 с.

15. Рязанов, Ю.Д. Булевы функции. Способы задания и реализация: учеб. пособие / Ю.Д. Рязанов. — Белгород: Изд-во БТИСМ, 1993. — 124 с.
16. Рязанов, Ю.Д. Множества и комбинаторные объекты: учеб. пособие / Ю.Д. Рязанов — Белгород: Изд-во БГТУ, 2008. — 99 с.
17. Рязанов, Ю.Д. Быстрый алгоритм решения теоретико-множественных уравнений / Ю.Д. Рязанов // Научное творчество XXI века: Материалы IV Всероссийской научно-практической конференции с международным участием. Приложение к журналу "В мире научных открытий". — Красноярск, 2011. — Вып. 2, с. 78 — 79.
18. Рязанов, Ю.Д. Интуитивные и формальные методы доказательства теоретико-множественных тождеств/ Ю.Д. Рязанов // Наука в решении региональных проблем: сб. науч. трудов с международным участием. — Пермь: Березниковский филиал Пермского национального исследовательского политехнического университета, 2012. — Вып. 8., с. 26 — 31.
19. Рязанов, Ю.Д. Генератор заданий для изучения преобразования теоретико-множественных выражений в нормальные формы Кантора / Ю.Д. Рязанов, А.В. Штырь // Инновационные технологии в производстве, науке и образовании. Сборник трудов II Международной научно-практической конференции. Часть 2. — Махачкала: Изд-во «ООО «Риасофт», 2012., с. 293 — 297.
20. Седжвик, Р. Фундаментальные алгоритмы на C++. Ч. 5: Алгоритмы на графах: пер. с англ / Р. Седжвик. — СПб.: ООО «Диа-СофтЮП», 2002. — 496 с.
21. Хаггарти, Р. Дискретная математика для программистов / Р. Хаггарти. — М.: Техносфера, 2005. — 400 с.
22. Шапорев, С.Д. Дискретная математика: курс лекций и практических занятий / С.Д. Шапорев. — СПб.: БХВ-Петербург, 2007. — 400 с.
23. Яблонский, С.В. Введение в дискретную математику: учеб. пособие для вузов / под ред. В.А. Садовничего. — 4-е изд., стер. — М.: Высш. шк., 2003. — 384 с.
24. Литература по дискретной математике — Не решается алгебра / высшая математика ?.. ПОМОЖЕМ! URL: <http://www.diary.ru/~eek/p49631731.htm#> (дата обращения: 15.09.2015)