

Белорусский национальный технический университет
Факультет информационных технологий и робототехники
Кафедра «Высшая математика»

Заведующий кафедрой
Щукин М.В. _____ 2025 г.

Декан факультета
Авсиевич А.М. _____ 2025 г.

ЭЛЕКТРОННЫЙ УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО
УЧЕБНОЙ ДИСЦИПЛИНЕ

Дискретная математика

для специальностей 6-05-0611-01 «Информационные системы и технологии»
6-05-0612-01 «Программная инженерия»

Составитель: Б.А. Бадак

Минск
БНТУ
2025

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ЭУМК предназначен для изучения дисциплины «Дискретная математика» для студентов факультета информационных технологий и робототехники БНТУ специальностей: 6-05-0611-01 «Информационные системы и технологии» профилизации «Информационные системы и технологии в проектировании и производстве», 6-05-0612-01 «Программная инженерия». Он содержит набор методических материалов по этой дисциплине. ЭУМК состоит из четырех частей.

Теоретический раздел содержит набор методических материалов по этому предмету: рекомендаций студенту для работы с дисциплиной, кратких теоретических материалов, посвященных изложению в наглядном виде основных определений, свойств, формул и теорем, сопровождающихся подробными примерами. Приведённые примеры сторителлинга в начале каждого теоретического раздела способствуют более интересному и увлекательному восприятию материала студентами, что позволяет повысить мотивацию обучающихся к изучению предмета.

Практический раздел содержит практикум по дисциплине, состоящий из материалов для проведения аудиторных занятий по математике и включает задания как практико-ориентированного, так и проблемного, эвристического характера. Каждое занятие содержит задачи для самостоятельной работы.

Раздел контроля знаний содержит пример контрольной работы для организации текущего контроля знаний студентов.

Вспомогательный раздел содержит программу дисциплины, перечень вопросов к экзамену, задачи эвристического и практико-ориентированного содержания, список рекомендуемой литературы.

Тестовые задания при текущем контроле могут быть выполнены как в аудитории, так и в компьютерной системе тестирования.

Оглавление

Введение	4
1 Теоретический раздел	5
1.1 Комбинаторика. Теория множеств и отношения. Кодирование	5
1.1.1 Основные комбинаторные операции и принципы	6
1.1.2 Множества и операции над ними	16
1.1.3 Упорядоченные множества. Отношения на множествах	20
1.1.4 Мультимножества	23
1.1.5 Кодирование	27
1.1.6 Шифрование	32
1.2 Математическая логика	36
1.2.1 Соответствия и функции	36
1.2.2 Элементарные булевы функции, канонические способы задания	40
1.2.3 Высказывания, методы проверки логического следования	53
1.2.4. Высказывания, методы проверки логического следования	64
1.3 Теория графов	89
1.3.1 Основные понятия теории графов, типы графов, обходы графов, способы задания	89
1.3.2 Ориентированные ациклические графы и деревья	94
1.3.3 Маршруты, пути, циклы. Эйлеровы циклы	99
1.3.4 Двудольные и планарные графы	100
1.3.5 Задача о кратчайшем пути. Алгоритм Дейкстры	105
2 Практический раздел	108
2.1. Комбинаторика. Теория множеств и отношения. Кодирование	108
2.2. Математическая логика	109
2.3. Теория графов	112
Вопросы по курсу «Дискретная математика»	114
Программа курса	115
Список использованных источников	117
Приложение 1	118
Приложение 2	140

Введение

Дискретная математика является одной из основополагающих дисциплин для специалистов в области информационных систем, технологий и программной инженерии. Она предоставляет теоретические основы, необходимые для понимания алгоритмов, структур данных, а также методов анализа и обработки информации. В условиях стремительного развития технологий и увеличения объемов данных, знание принципов дискретной математики становится особенно актуальным для будущих инженеров-программистов.

Комплекс содержит основные математические методы, знание которых необходимо для специалистов в области инженерии и программирования: комбинаторные задачи; операции над множествами; кодирование и шифрование; методы проверки логических высказываний; оптимизационные задачи на графах. Вместе с теоретическими материалами в электронном учебно-методическом комплексе рассматривается достаточное количество примеров решения задач с практико-ориентированным содержанием, приведены упражнения для самостоятельной работы студентов, задачи проблемного характера (задачи с собеседований по направлению DataScience).

Электронный учебно-методический комплекс предназначен для специальностей 6-05-0611-01 «Информационные системы и технологии» профилизации «Информационные системы и технологии в проектировании и производстве», 6-05-0612-01 «Программная инженерия» факультета информационных технологий и робототехники БНТУ и отвечает требованиям учебного плана и содержанию учебной программы курса.

1 Теоретический раздел

1.1 Комбинаторика. Теория множеств и отношения. Кодирование

Сторителлинг «Путешествие в мир комбинаторики, теории множеств и кодирования»:

История 1: «Знакомство с миром возможностей»

В небольшом городке, где каждый день проходил в рутинной работе и учебе, жил юноша по имени Алекс. Он всегда мечтал стать инженером-программистом, создавая удивительные приложения и решая сложные задачи. Однажды, готовясь к важному экзамену по программированию, он наткнулся на книгу по дискретной математике, и его внимание привлекла глава о комбинаторике.

Алекс открыл книгу и погрузился в мир, где возможности были безграничны. Он узнал, что комбинаторика – это не просто набор формул, а целая наука о том, как считать, комбинировать и упорядочивать объекты. В его голове зародилась идея: «Как же много всего можно создать, если правильно использовать комбинации!».

История 2: «Теория множеств – основа всего»

После изучения комбинаторики Алекс решил углубиться в теорию множеств. Он понял, что множества – это основа для работы с данными. Каждое приложение, каждая программа начинается с определения того, какие данные ему нужны. Алекс представил себе множество пользователей, множество функций и множество возможных ошибок, которые могут возникнуть.

Он начал визуализировать свои идеи. Каждый пользователь был элементом множества, а функции – подмножествами. Он понял, что при помощи теории множеств можно не только организовать данные, но и находить новые решения. Например, если он знает, что у него есть множество пользователей с разными интересами, он может создать алгоритм, который будет рекомендовать им именно то, что им нужно.

История 3: «Отношения – связь между элементами»

Погружаясь глубже в теорию множеств, Алекс столкнулся с теорией отношений. Он осознал, что отношения между элементами множеств имеют огромное значение. Например, если один пользователь рекомендует другому фильм, это создает отношение между ними. Алекс начал разрабатывать свою первую программу – приложение для рекомендаций. Он использовал графы для представления отношений между пользователями и фильмами. Каждый узел графа был пользователем или фильмом, а ребра представляли собой отношения. Он понял, что чем больше данных он соберет о пользователях и их предпочтениях, тем точнее будет его алгоритм.

История 4: «Кодирование – язык общения с машинами»

После успешного создания приложения Алекс задумался о следующем шаге — кодировании. Он вспомнил уроки о том, как важно правильно кодировать информацию для передачи и хранения данных. Он изучил различные методы кодирования и понял, что от этого зависит эффективность работы его приложения. Алекс выбрал метод Хаффмана для сжатия данных о фильмах и пользователях. Он осознал, что правильное кодирование позволяет не только экономить место на сервере, но и ускорять обработку запросов. Это было критически важно для его приложения, которое могло стать популярным среди пользователей.

История 5: «Применение знаний на практике»

Собрав все свои знания о комбинаторике, теории множеств и кодировании, Алекс запустил свое приложение. Оно быстро завоевало популярность среди друзей и знакомых. Пользователи были в восторге от точности рекомендаций и удобного интерфейса. Но самое главное – Алекс понял, что все эти математические положения не просто теоретические знания. Они стали основой его успеха как программиста и инженера. Он осознал важность глубокого понимания математики в своей будущей профессии и решил делиться своими знаниями с другими.

История-эпилог: «Путь к новым вершинам»

Алекс продолжал развиваться как специалист. Он стал преподавать основы дискретной математики своим сверстникам и вдохновлять их на изучение комбинаторики, теории множеств и кодирования. Каждый новый проект становился для него возможностью применить свои знания на практике. И так, в маленьком городке появился инженер-программист, который знал: математика – это не просто цифры и формулы; это ключ к пониманию мира технологий и созданию инновационных решений. Каждый новый день открывал перед ним новые горизонты возможностей, а его путешествие только начиналось.

1.1.1 Основные комбинаторные операции и принципы

Определение 1: Комбинаторика – это раздел математики, в котором изучаются вопросы о том, сколько различных комбинаций, подчиненных тем или иным условиям, можно составить из заданных объектов.

Слово «комбинаторика» происходит от латинского слова «combinare», что в переводе на русский означает – «сочетать», «соединять». Комбинаторные задачи возникли и в связи с такими играми, как шашки, шахматы, домино, карты, кости и т.д. Термин «комбинаторика» был введен знаменитым Готфридом Вильгельмом Лейбницем, - всемирно известным немецким учёным.

Комбинаторные задачи делятся на: задачи на перестановки, задачи на размещение, задачи на сочетание.

Определение 2: Факториал – это произведение всех натуральных чисел от 1 до n . Обозначение: $n! = 1 \cdot 2 \cdot 3 \cdot \dots \cdot n$. Читается: «эн факториал».

Пример 1.1. $4! = 1 \cdot 2 \cdot 3 \cdot 4 = 24$.

Кроме того: $0! = 1$.

А. Перестановки

Определение 3: Перестановками из n элементов называются комбинации из n элементов, отличающиеся друг от друга только порядком расположения в них элементов.

Формула $P_n = n!$

Типичная смысловая нагрузка: «Сколькими способами можно переставить n объектов?»

Пример 1.2. Какое количество уникальных перестановок можно создать для 3 различных объектов (книг) в фиксированном массиве (книжной полке)?

Решение: Выбираем один из различных объектов (одну из 3-х книг) и ставим на первое место. Это можно сделать 3-мя способами. Вторую книгу мы можем выбрать из 2-х оставшихся двумя способами, получаем $3 \cdot 2$ способов. Третью книгу мы можем выбрать 1 способом. Получится $3 \cdot 2 \cdot 1 = 6$ способов.

Ответ: 6.

Пример 1.3. Сколько уникальных способов можно организовать расстановку 8 участников финального забега на 8 различных беговых дорожках?

Решение: $P_8 = 8! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 = 40320$.

Ответ: 40320.

Пример 1.4. Сколько различных способов можно расположить буквы слова «МАТН», если буквы «М» и «А» должны находиться рядом?

Решение: Если буквы «М» и «А» должны находиться рядом, мы можем рассматривать их как один элемент (блок). Таким образом, у нас будет 3 элемента: «МА», «Т», «Н». $P_3 = 3! = 1 \cdot 2 \cdot 3 = 6$. Однако, внутри блока «МА» буквы могут быть расположены двумя способами (МА или АМ). Поэтому общее количество перестановок $6 \cdot 2 = 12$.

Ответ: 12 различных способов.

Пример 1.5. Сколько различных шестизначных чисел, кратных 5, можно составить из цифр 1, 2, 3, 4, 5, 6 при условии, что цифры в числе не повторяются?

Решение: Чтобы число было кратным 5, цифра 5 должна стоять на последнем месте. Остальные цифры могут стоять на оставшихся пяти местах в любом порядке. Следовательно, искомое количество шестизначных чисел, кратных 5, равно числу перестановок из 5 элементов, т.е. $P_5 = 5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$.

Ответ: 120.

В. Размещения

Определение 4: Размещением из n элементов по k ($k \leq n$) называется любое множество, состоящее из k элементов, взятых в определённом порядке из данных n элементов.

Формула: $A_n^k = \frac{n!}{(n-k)!}$

Типичная смысловая нагрузка: «Сколькими способами можно выбрать k объектов и в каждой выборке переставить их местами?»

Пример 1.6. Расписание студентов в определённый день недели должно состоять из 4 пар. Занятия могут быть проведены по любым из 9 дисциплин. Следует определить число возможных вариантов расписания.

Решение: Так как при составлении расписания учитывается как состав дисциплин, так и их порядок, надо воспользоваться размещениями:

$$A_9^4 = \frac{9!}{5!} = \frac{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} = 6 \cdot 7 \cdot 8 \cdot 9 = 3024.$$

Ответ: 3024.

Пример 1.7. Сколько трехзначных чисел можно составить из цифр 2, 4, 6, 7, 9?

Решение: $A_5^3 = \frac{5!}{2!} = 3 \cdot 4 \cdot 5 = 60.$

Ответ: 60.

Пример 1.8. В соревнованиях высшей лиги по футболу участвуют 18 команд. Борьба идет за золотые, серебряные и бронзовые медали. Сколькими способами могут быть распределены медали между командами?

Решение: $A_{18}^3 = \frac{18!}{15!} = 16 \cdot 17 \cdot 18 = 4896.$

Ответ: 4896.

Пример 1.9. Сколькими способами можно опустить 5 писем в 11 почтовых ящиков, если в каждый ящик опускают не более одного письма?

Решение: $A_{11}^5 = \frac{11!}{6!} = 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11 = 55440.$

Ответ: 55440.

Пример 1.10. Боря, Дима и Володя сели играть в карты. Сколькими способами им можно сдать по одной карте? (колода содержит 36 карт).

Решение:

$A_{36}^3 = \frac{36!}{33!} = 34 \cdot 35 \cdot 36 = 42840$ – способами можно раздать 3 карты игрокам.

Ответ: 42840.

Пример 1.11. В пассажирском поезде 9 вагонов. Сколькими способами можно рассадить в поезде 4 человека, при условии, что все они должны ехать в различных вагонах?

Решение:

$$A_9^4 = \frac{9!}{5!} = 6 \cdot 7 \cdot 8 \cdot 9 = 3024 - \text{способами можно рассадить в поезде 4 человека.}$$

Ответ: 3024.

С. Сочетания

Определение 5: Сочетанием из n элементов по k ($k < n$) называется любое множество, составленное из k элементов, выбранных из данных n элементов (не имеет значения, в каком порядке указаны элементы).

Формула: $C_n^k = \frac{n!}{(n-k)! \cdot k!}$

Типичная смысловая нагрузка: «Сколькими способами можно выбрать k объектов из n ?»

Пример 1.12 В учебной группе 7 человек успешно занимаются дискретной математикой. Сколькими способами можно выбрать из них двоих для участия в математической олимпиаде?

Решение: $C_7^2 = \frac{7!}{2! \cdot 5!} = \frac{5! \cdot 6 \cdot 7}{2! \cdot 5!} = \frac{6 \cdot 7}{2!} = 21.$

Ответ: 21.

Пример 1.13. На тренировках занимаются 12 баскетболистов. Сколько может быть организовано тренером разных стартовых пятёрок?

Решение: $C_{12}^5 = \frac{12!}{5! \cdot 7!} = \frac{7! \cdot 8 \cdot 9 \cdot 10 \cdot 11 \cdot 12}{5! \cdot 7!} = \frac{8 \cdot 9 \cdot 10 \cdot 11 \cdot 12}{120} = 8 \cdot 9 \cdot 11 = 792.$

Ответ: 792.

Пример 1.14. В ящике находится 15 деталей. Сколькими способами можно взять 4 детали?

Решение: $C_{15}^4 = \frac{15!}{4! \cdot 11!} = \frac{11! \cdot 12 \cdot 13 \cdot 14 \cdot 15}{4! \cdot 11!} = \frac{12 \cdot 13 \cdot 14 \cdot 15}{24} = 1365.$

Ответ: 1365.

Пример 1.15. Сколькими способами из колоды в 36 карт можно выбрать 3 карты?

Решение: $C_{36}^3 = \frac{36!}{33! \cdot 3!} = \frac{33! \cdot 34 \cdot 35 \cdot 36}{33! \cdot 3!} = \frac{34 \cdot 35 \cdot 36}{6} = 7140.$

Ответ: 7140.

Пример 1.16. Для участия в команде тренер отбирает 5 парней из 10. Сколькими способами он может сформировать команду, если 2 определенных парня должны войти в команду?

Решение: Т.к. двое парней войдут в команду, то остается отобрать 3 из 8. Для выборки важен только состав (по условию все члены команды не различаются по ролям). $C_8^3 = \frac{8!}{3! \cdot 5!} = \frac{5! \cdot 6 \cdot 7 \cdot 8}{3! \cdot 5!} = \frac{6 \cdot 7 \cdot 8}{1 \cdot 2 \cdot 3} = 56$. **Ответ:** 56.

Пример 1.17. В шахматном турнире принимали участие 15 студентов, причем каждый из них сыграл только одну партию с каждым из остальных. Сколько всего партий было сыграно в этом турнире?

Решение: В одной игре участвуют 2 человека, следовательно, нужно вычислить, сколькими способами можно отобрать 2-х человек из 15, причем порядок в таких парах не важен. $C_{15}^2 = \frac{15!}{2! \cdot 13!} = \frac{13! \cdot 14 \cdot 15}{2! \cdot 13!} = \frac{14 \cdot 15}{1 \cdot 2} = 105$.

Ответ: 105.

Пример 1.18. Сколько различных дробей можно составить из чисел 3, 5, 7, 11, 13, 17 так, чтобы в каждую дробь входили 2 различных числа? Сколько среди них будет правильных дробей?

Решение: Различных дробей из 6 чисел: 3, 5, 7, 11, 13, 17 можно составить $C_6^2 \cdot 2 = \frac{6!}{4! \cdot 2!} \cdot 2 = \frac{4! \cdot 5 \cdot 6}{4! \cdot 2!} \cdot 2 = 5 \cdot 6 = 30$ штук (C_6^2 способами выбираем два числа из 6, и двумя способами составляем из них дробь, сначала одно число – числитель, другое – знаменатель и наоборот). Из этих 30 дробей 15 будут правильные.

Ответ: 30; 15.

Пример 1.19. Боря, Дима и Володя сели играть в карты. Сколькими способами им можно сдать по одной карте? (колода содержит 36 карт).

Решение: $C_{36}^3 = \frac{36!}{33! \cdot 3!} = 7140$ – способами можно извлечь 3 карты из колоды. Теперь рассмотрим, какую-нибудь одну из семи тысяч ста сорока комбинаций, например: король пик, 9 червей, 7 червей. Эти 3 карты можно «переставить» между Борей, Димой и Володей $P_3 = 3! = 6$ способами. Тогда $C_{36}^3 \cdot P_3 = 7140 \cdot 6 = 42840$ способами можно сдать по одной карте 3-м игрокам.

Ответ: 42840.

Получили формулу: $C_n^k \cdot P_k = A_n^k$.

Д. Правило сложения комбинаций

!Знак «плюс» следует понимать и читать как союз «ИЛИ»

Задача 1. Студенческая группа состоит из 23 человек, среди которых 10 юношей и 13 девушек. Сколькими способами можно выбрать 2-х человек одного пола?

Решение: Условие «выбрать 2-х человек одного пола» подразумевает, что необходимо выбрать двух юношей или двух девушек:

$$C_{10}^2 = \frac{10!}{2! \cdot 8!} = \frac{8! \cdot 9 \cdot 10}{2! \cdot 8!} = \frac{9 \cdot 10}{2} = 45 \text{ — способами можно выбрать 2-х юношей;}$$

$C_{13}^2 = \frac{13!}{2! \cdot 11!} = \frac{11! \cdot 12 \cdot 13}{2! \cdot 11!} = \frac{12 \cdot 13}{2} = 78$ — способами можно выбрать 2-х девушек;

Таким образом, двух человек одного пола (без разницы — юношей или девушек) можно выбрать: $C_{10}^2 + C_{13}^2 = 45 + 78 = 123$ способами.

Ответ: 123.

Пример 1.20. В группе 9 человек. Сколько можно образовать разных подгрупп при условии, что в подгруппу входит не менее 2 человек?

Решение: Не менее 2-х человек, т.е. 2+7 или 3+6 или 4+5 человек (5+4, 6+3, 7+2 — те же самые комбинации). В каждой выборке важен только состав, т.е. члены подгруппы не различаются по ролям, т.е. выборки — сочетания из различных элементов по m элементов.

$$\text{Число выборов из 2-х человек: } C_9^2 = \frac{9!}{2! \cdot (9-2)!} = \frac{9!}{2! \cdot 7!} = \frac{8 \cdot 9}{1 \cdot 2} = 36.$$

$$\text{Число выборов из 3-х человек: } C_9^3 = \frac{9!}{3! \cdot (9-3)!} = \frac{9!}{3! \cdot 6!} = \frac{7 \cdot 8 \cdot 9}{1 \cdot 2 \cdot 3} = 84.$$

$$\text{Число выборов из 4-х человек: } C_9^4 = \frac{9!}{4! \cdot (9-4)!} = \frac{9!}{4! \cdot 5!} = \frac{6 \cdot 7 \cdot 8 \cdot 9}{1 \cdot 2 \cdot 3 \cdot 4} = 126.$$

Применяем правило сложения: $C_9^2 + C_9^3 + C_9^4 = 36 + 84 + 126 = 246$ способов.

Ответ: 246.

Е. Правило умножения комбинаций

! Знак «умножить» следует понимать и читать как союз «И»

Задача 2. Студенческая группа состоит из 23 человек, среди которых 10 юношей и 13 девушек. Сколькими способами можно составить пару из юноши и девушки?

Решение:

$$C_{10}^1 = \frac{10!}{1! \cdot 9!} = 10 \text{ — способами можно выбрать 1 юношу;}$$

$$C_{13}^1 = \frac{13!}{1! \cdot 12!} = 13 \text{ — способами можно выбрать 1 девушку.}$$

Таким образом, 1-го юношу и 1 девушку можно выбрать: $C_{10}^1 \cdot C_{13}^1 = 10 \cdot 13 = 130$ способами.

Ответ: 130.

Пример 1.21. Предприятие может предоставить работу по одной специальности 4 женщинами, по другой — 6 мужчинам, по третьей — 3

работникам независимо от пола. Сколькими способами можно заполнить вакантные места, если имеются 14 претендентов: 6 женщин и 8 мужчин?

Решение: Имеем 14 претендентов и 13 рабочих мест. Сначала выберем работников на первую специальность, то есть 4 женщин из 6:

$$C_6^4 = \frac{6!}{4! \cdot 2!} = \frac{5 \cdot 6}{2} = 15.$$

Далее выберем мужчин на вторую специальность: $C_8^6 = \frac{8!}{6! \cdot 2!} = \frac{7 \cdot 8}{2} = 28.$

Осталось 2 женщины, 2 мужчин и 3 вакантных места, которые, по условию, могут занять любые из четырех оставшихся человек.

Это может быть сделано 2 вариантами:

1. 1 женщина и 2 мужчин (выбираем женщину $C_2^1 = 2$ способами).
2. 1 мужчина и 2 женщины (выбираем мужчину $C_2^1 = 2$ способами).

В итоге получаем $15 \cdot 28 \cdot (2+2) = 1680.$

Ответ: 1680.

Пример 1.22. Группу из 20 студентов нужно разделить на 3 бригады, причем в первую бригаду должны входить 3 человека, во вторую – 5 и в третью – 12. Сколькими способами это можно сделать?

Решение: Создавая первую бригаду, отбирают 3 человека из 20, создавая вторую – 5 из оставшихся 17, создавая третью – 12 из оставшихся 12. Для выборок важен только состав (роли членов бригады не различаются).

Создавая сложную выборку (из 3-х бригад), воспользуемся правилом умножения:

$$C_{20}^3 \cdot C_{17}^5 \cdot C_{12}^{12} = \frac{20!}{3! \cdot 17!} \cdot \frac{17!}{5! \cdot 12!} \cdot \frac{12!}{12! \cdot 0!} = \frac{13 \cdot 14 \cdot 15 \cdot 16 \cdot 17 \cdot 18 \cdot 19 \cdot 20}{1 \cdot 2 \cdot 3 \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} = 7054320$$

способов.

Ответ: 7054320.

Пример 1.23. Сколькими способами может быть сдана выигрышная комбинация из 2-х карт при игре в «очко»?

Информационная поддержка: выигрывает комбинация 10 + ТУЗ (11 очков) = 21 очко и будем считать выигрышной комбинацию из 2-х тузов.

Решение: $C_4^1 \cdot C_4^1 = 4 \cdot 4 = 16$ способами может быть сдана десятка и туз («каждая десятка с каждым тузом»); $C_4^2 = \frac{4!}{2! \cdot 2!} = \frac{24}{4} = 6$ способами может быть

сдана пара тузов. Итого: $C_4^1 \cdot C_4^1 + C_4^2 = 16 + 6 = 22$ выигрышные комбинации.

Ответ: 22.

Пример 1.24. Сколько существует трёхзначных чисел, которые делятся на 5?

Решение:

В разряде сотен можно записать любую из $C_9^1 = 9$ цифр.

В разряде десятков можно выбрать любую из 10 цифр: $C_{10}^1 = 10$.

По условию, число должно делиться на 5. Число делится на 5, если оно заканчивается на 5 либо на 0. Таким образом, в младшем разряде нас устраивают 2 цифры.

Итого, существует: $C_9^1 \cdot C_9^1 \cdot 2 = 180$ трёхзначных чисел, которые делятся на 5.

Ответ: 180.

Е. Перестановки с повторениями

Типичная смысловая нагрузка: «Количество способов, которыми можно переставить n объектов, среди которых 1-й объект повторяется n_1 раз, 2-й объект повторяется n_2 раз, 3-й объект повторяется n_3 раз, ..., k -й объект повторяется n_k раз».

Формула: $P_{n(\text{повт.})} = \frac{n!}{n_1! \cdot n_2! \cdot n_3! \cdot \dots \cdot n_k!}$, где $n_1 + n_2 + n_3 + \dots + n_k = n$.

Задача 3. У мамы 2 яблока и 3 груши. Каждый день в течение 5 дней подряд она выдает по одному фрукту. Сколькими способами это может быть сделано?

Решение: Имеем набор $\{я, я, г, г, г\}$. Всего перестановок пятиэлементного множества $5!$, но мы не должны учитывать перестановки, в которых объекты одного типа меняются местами несколько раз, поэтому нужно поделить на возможное число таких перестановок: $2! \cdot 3!$. В итоге получаем $P_{11(\text{повт.})} = \frac{5!}{2! \cdot 3!} = \frac{4 \cdot 5}{2} = 10$.

Ответ: 10.

Пример 1.25. Сколько различных буквосочетаний можно получить перестановкой карточек со следующими буквами: К, О, Л, О, К, О, Л, Ъ, Ч, И, К?

Решение: Всего: 11 карточек, среди которых буква:

К – повторяется 3 раза;

О – повторяется 3 раза;

Л – повторяется 2 раза;

Ь – повторяется 1 раз;

Ч – повторяется 1 раз;

И – повторяется 1 раз.

По формуле количества перестановок с повторениями:

$$P_{11(\text{повт.})} = \frac{11!}{3! \cdot 3! \cdot 2! \cdot 1! \cdot 1!} = \frac{39916800}{6 \cdot 6 \cdot 2} = 554400.$$

Ответ: 554400.

Пример 1.26. Сколько слов можно получить, переставляя буквы в слове Институт?

Решение: В слове «институт» 8 букв, из них две буквы «и», три буквы «т» и по одной букве «н», «с» и «у». Поэтому всего можно получить перестановками букв $P_{11} = \frac{8!}{2! \cdot 3! \cdot 1! \cdot 1! \cdot 1!} = 3360$ различных слов.

Ответ: 3360.

Пример 1.27. Алексей занимается спортом, причём 4 дня в неделю – лёгкой атлетикой, 2 дня – силовыми упражнениями и 1 день отдыхает. Сколькими способами он может составить себе расписание занятий на неделю?

Решение: По формуле количества перестановок с повторениями:

$$P_{7(\text{повт.})} = \frac{7!}{4! \cdot 2!} = \frac{5040}{24 \cdot 2} = 105 \text{ способами можно составить расписание}$$

занятий на неделю.

Ответ: 105.

Пример 1.28. Сколько чисел, больших 3000000, можно составить из цифр 3, 2, 2, 1, 1, 1, 0.

Решение: На первом месте обязательно должна стоять тройка. Оставшиеся 6 цифр образуют перестановку с повторениями:

$$P_{6(\text{повт.})} = \frac{6!}{2! \cdot 3! \cdot 1!} = \frac{720}{2 \cdot 6} = 60.$$

Ответ: 60.

Г. Сочетания с повторениями

Типичная смысловая нагрузка: «Для выбора предложено n множеств, каждое из которых состоит из одинаковых объектов. Сколькими способами можно выбрать m объектов?»

Формула: $C_{n(\text{повт.})}^m = C_{n+m-1}^m = \frac{(n+m-1)!}{(n-1)! \cdot m!}.$

Пример 1.29. В студенческой столовой продают сосиски в тесте, ватрушки и пончики. Сколькими способами можно приобрести пять пирожков?

Решение: Используя формулу количества сочетаний с повторениями, получаем

$$C_{3(\text{повт.})}^5 = C_{3+5-1}^5 = C_7^5 = \frac{7!}{2! \cdot 5!} = \frac{6 \cdot 7}{2!} = 21 \text{ способом можно приобрести 5}$$

пирожков.

Ответ: 21.

Пример 1.30. В почтовом отделении продаются открытки 10 видов. Сколькими способами можно купить 12 открыток для поздравлений?

Решение.

$$C_{10(\text{повт.})}^{12} = C_{21}^{12} = \frac{21!}{12! \cdot 9!} = \frac{13 \cdot 14 \cdot 15 \cdot 16 \cdot 17 \cdot 18 \cdot 19 \cdot 20 \cdot 21}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9} = \frac{13 \cdot 5 \cdot 17 \cdot 2 \cdot 19 \cdot 7}{1} = 293930$$

Ответ: 293930.

Н. Размещения с повторениями

Типичная смысловая нагрузка: «Дано множество, состоящее из n объектов, при этом любой объект можно выбирать неоднократно. Сколькими способами можно выбрать m объектов, если важен порядок их расположения в выборке? В частности, возможен случай, когда из n имеющихся объектов m раз будет выбран какой-то один объект».

Формула: $A_{n(\text{повт.})}^m = n^m$.

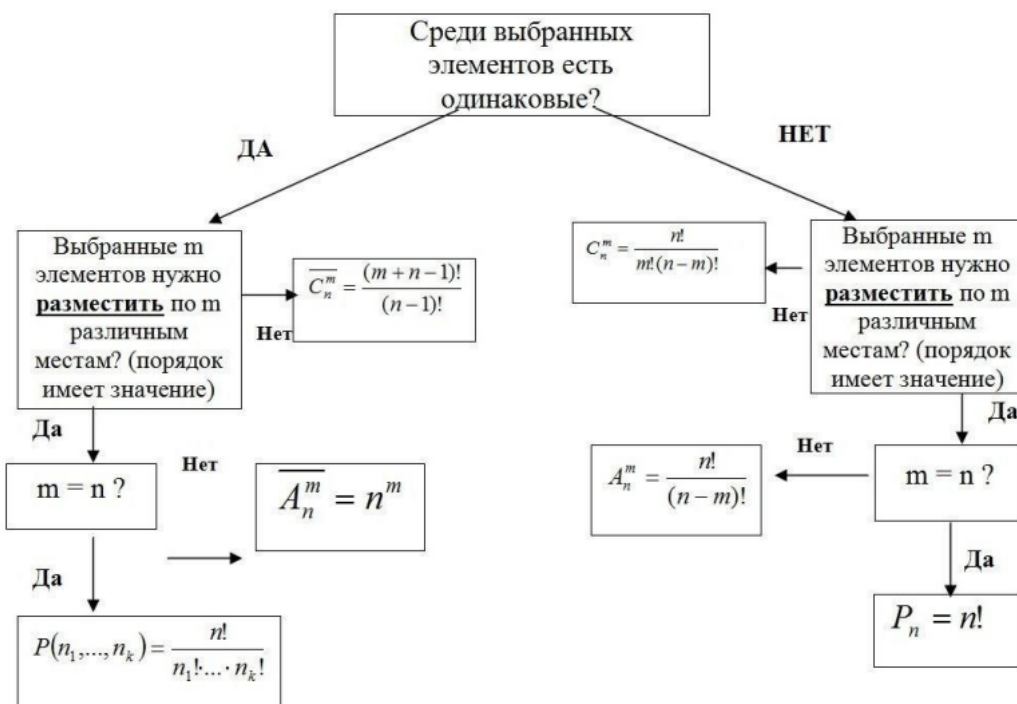
Пример 1.31. Человек, пришедший в гости, забыл код, открывающий дверь подъезда, но помнил, что он составлен из нулей и единиц и всего имеет четыре цифры. Сколько вариантов кода в худшем случае ему придётся перебрать, чтобы открыть дверь?

Решение: $A_{2(\text{повт.})}^4 = 2^4 = 16$. **Ответ:** 16.

Пример 1.32. Каких чисел от 1 до 1 000 000 больше: тех, в записи которых встречается единица, или тех, в которых она не встречается?

Решение: Подсчитаем количество чисел от 1 до 999999 в записи которых нет единиц. Каждую цифру можно выбрать 9 способами (любая цифра кроме 1), поэтому все 6 цифр можно выбрать 9^6 способами. При этом один вариант (000000) нужно убрать, так как число 0 не рассматривается. Получаем всего $9^6 - 1 = 531440$ чисел. Так как всего чисел 1 000 000, то видно, что чисел без единицы среди чисел от 1 до 1 000 000 больше, чем тех, в записи которых единица есть. **Ответ:** чисел без единицы больше.

Важное замечание! На практике при решении комбинаторных задач удобно руководствоваться схемой:



1.1.2 Множества и операции над ними

Определение 6: Под *множеством* понимается всякая совокупность объектов (предметов, понятий), рассматриваемая как единое целое; при этом объекты этой совокупности называют *элементами* множества.

Пример 1.33.

1. Множество студентов в данной аудитории.
2. Множество окружностей, имеющих центр в данной точке.
3. Множество всех букв русского алфавита.

Замечание: Множества обозначаются большими буквами $A, B, X, E, \dots$, элементы множества - малыми буквами $a, b, x, e, \dots$.

Замечание: Если объект a является элементом множества A (принадлежит A), то это обозначается $a \in A$. Если a не является элементом множества A (не принадлежит A), то пишут $a \notin A$.

Определение 7: Множества, элементами которых являются числа, называются *числовыми множествами*.

Определение 8: Множество, которое не содержит ни одного элемента, называется *пустым множеством* и обозначается символом $\emptyset$.

Пример 1.34.

1. Множество точек пересечения параллельных прямых – пустое множество.

2. Множество решений системы уравнений $\begin{cases} x + y = 1, \\ x + y = 2 \end{cases}$ - пустое множество.

Способы задания множества

Определение 9: Множество считается *заданным*, если известен способ, позволяющий для данного предмета решить, принадлежит ли он этому множеству.

Множества могут быть заданы несколькими способами:

1. Задание множества перечислением его элементов.

Пример 1.35. Множество из четырех букв $a, б, в, г$ записывается $\{a, б, в, г\}$.

Замечание: Перечислением можно задать только множество, содержащее *конечное* число элементов. Множества N натуральных чисел, Z целых чисел, множество Q рациональных чисел, R действительных чисел перечислением задать нельзя, так как они *бесконечные множества*.

2. Задание множества с помощью описания.

Пример 1.36.

1. Множество точек плоскости, равноудаленных от двух точек.
2. Множество решений уравнения $x^2 - 1 = 0$.

3. Задание множества с помощью характеристического свойства.

Замечание: Характеристических свойств, которыми обладают элементы множества, может быть несколько, тогда при перечислении они отделяются запятой или соединяются логическим союзом «и».

Пример 1.37. Число 2,5 принадлежит множеству $C = \{c \mid c \in R, -3 < c \leq 2,6\}$.

Отношения между множествами

Определение 10: Множество B называется *подмножеством* множества A (или частью множества A), если каждый элемент множества B является также элементом множества A . Говорят также, что множество B *включается во множество* A и обозначается как $B \subseteq A$.

Отношения между множествами наглядно изображаются с помощью множеств точек, ограниченных замкнутыми кривыми, - так называемых *диаграмм Эйлера-Венна*. Например, на рисунке 1.1. приведена диаграмма для отношения включения, когда $B \subseteq A$.

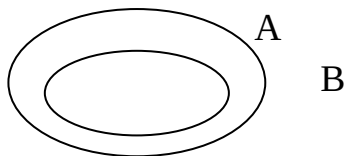


Рис. 1.1. Пример диаграммы Вена

Пример 1.38.

1. Множество параллелограммов является подмножеством множества четырехугольников.
2. Открытый отрезок (a, b) является частью замкнутого отрезка $[a, b]$, то есть $(a, b) \subset [a, b]$.

Из определения отношения включения следуют его *свойства*.

1. Любое множество является подмножеством самого себя (*рефлексивность*): $A \subseteq A$.
2. Для любых множеств A, B, C справедливо свойство *транзитивности*: если $A \subseteq B$ и $B \subseteq C$, то $A \subseteq C$.
3. Для всякого множества A пустое множество $\emptyset$ является его подмножеством: $\emptyset \subseteq A$.

Определение 11: Множества A и B называются *равными*, если они состоят из одних и тех же элементов, то есть если любой элемент одного из множеств принадлежит другому множеству.

Пример 1.39. $A = \{1, 3\}$, $B = \{3, 1\}$. Множества A и B состоят из одних и тех же элементов, поэтому они равные: $A = B$.

Операции над множествами

1. Пересечение множеств. *Пересечением множеств A и B* называется множество, обозначаемое $A \cap B$ и состоящее из всех тех и только тех элементов, которые принадлежат и множеству A , и множеству B

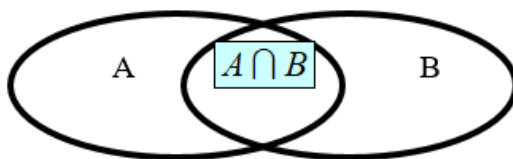


Рис. 1.2. Пересечение множеств A и B

Пример 1.40. Даны числовые множества $A = \{0, 1, 2, 3\}$, $B = \{2, 3, 4\}$. Найдем их пересечение: $A \cap B = \{x \mid x \in A \text{ и } x \in B\} = \{2, 3\}$.

Свойства операции пересечения множеств

1. $A \cap \emptyset = \emptyset$ - пересечение множества с пустым множеством есть пустое множество	3. $A \cap B = B \cap A$ - операция пересечения множеств коммутативна
2. $A \cap A = A$ - пересечение множества с самим собой есть само множество	4. $(A \cap B) \cap C = A \cap (B \cap C) = A \cap B \cap C$ - Операция пересечения множеств ассоциативна

2. Объединение множеств. *Объединением множеств A и B* называется множество $A \cup B$, состоящее только из тех элементов, которые принадлежат хотя бы одному из множеств A или B :
 $A \cup B = \{x \mid x \in A \text{ или } x \in B\}$.

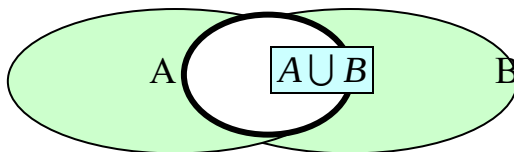


Рис. 1.3. Объединение множеств A и B

Объединение множеств $A \cup B$ содержит все элементы множества A и все элементы множества B .

Пример 1.41.

Пусть $A = \{0, 1, 2, 3\}$, $B = \{2, 3, 4\}$. При нахождении объединения множеств будем иметь в виду, что каждое объединение двух множеств содержит все элементы первого и второго множества, записанные по одному разу и в произвольном порядке:

$$A \cup B = \{x \mid x \in A \text{ или } x \in B\} = \{0, 1, 2, 3, 4\}.$$

Свойства операция объединения множеств

1. $A \cup \emptyset = A$, 2. $A \cup A = A$, 3. $A \cup B = B \cup A$ - коммутативность (переместительный закон) 4. $(A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C$ - ассоциативность (распределительный закон)	5. $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$ - дистрибутивность пересечения, 6. $(A \cup B) \cap C = (A \cap C) \cup (B \cap C)$ - дистрибутивность объединения
---	---

3. Вычитание множеств. *Разностью между множествами A и B* называется множество $A - B$, состоящее из тех элементов множества A , которые не принадлежат множеству B , то есть $A - B = \{x \mid x \in A \text{ и } x \notin B\}$.

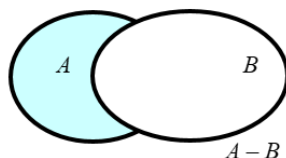


Рис. 1.4. Разность множеств A и B

Пример 1.42. Найдем разности множеств $A = \{0, 1, 2, 3\}$, $B = \{2, 3, 4\}$:
 $A - B = \{x \mid x \in A, x \notin B\} = \{0, 1\}$; $B - A = \{4\}$. Как видно, $A - B \neq B - A$.

Свойства операции вычитания множеств

1. $A - \emptyset = A$.
2. $\emptyset - A = \emptyset$.
3. $A - B = \emptyset$ тогда и только тогда, когда $A \subseteq B$.

4. Дополнение множества

Часто множества $A, B, C, \dots$ являются подмножествами некоторого более широкого множества U , принимаемого за универсальное. Для совокупности множеств $A, B, C, \dots$ **универсальным множеством** называют каждое множество U такое, что $A \subseteq U$, $B \subseteq U$, $C \subseteq U, \dots$.

Пример 1.43. Для совокупности множеств $A, B, C, \dots$ **универсальным множеством** называют каждое множество U такое, что $A \subseteq U$, $B \subseteq U$, $C \subseteq U, \dots$

Определение 12: Множество элементов универсального множества U , не принадлежащих множеству A , называется **дополнением множества A до множества U** или просто **дополнением** и обозначается $\bar{A}$. Таким образом, $\bar{A} = U - A$.

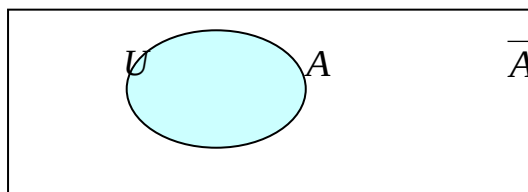


Рис. 1.5. Дополнение множеств A и B

Свойства операции дополнения

Для любых множеств A и B , принадлежащих универсальному множеству U , справедливы следующие свойства.

1. $\overline{\emptyset} = U$;	6. Дополнение объединения множеств равно пересечению дополнений этих множеств $\overline{A \cup B} = \overline{A} \cap \overline{B}$;
2. $\overline{U} = \emptyset$;	7. Дополнение пересечения множеств равно объединению дополнений этих множеств $\overline{A \cap B} = \overline{A} \cup \overline{B}$.
3. $A \cap \overline{A} = \emptyset$;	
4. $A \cup \overline{A} = U$;	
5. $\overline{\overline{A}} = A$;	

1.1.3 Упорядоченные множества. Отношения на множествах

Определение 13: Пусть A и B – произвольные множества. *Упорядоченная пара* на множествах A и B , обозначаемая записью $\langle a, b \rangle$, определяется не только самими элементами $a \in A$ и $b \in B$, но и порядком, в котором они записаны. И в этом состоит ее существенное отличие от неупорядоченной пары. Если $A = B$, то говорят об упорядоченной паре на множестве A .

Определение 14: Две упорядоченные пары $\langle a, b \rangle$ и $\langle c, d \rangle$ на множествах A и B называют *равными*, если $a = c$ и $b = d$.

Упорядоченную пару $\langle a, b \rangle$ не следует связывать с множеством $\{a, b\}$, так как упорядоченная пара характеризуется не только составом, но и порядком элементов в ней. Более того, определение этого объекта вообще не позволяет рассматривать его как множество. Но упорядоченную пару можно определить и как множество, полагая, что упорядоченная пара $\langle a, b \rangle$ есть неупорядоченная пара $\{\{a\}, \{a, b\}\}$, включающая в себя одноэлементное множество $\{a\}$ и неупорядоченную пару $\{a, b\}$. При $a = b$ получаем $\langle a, a \rangle = \{\{a\}\}$. Такое определение не изменит сути понятия, но тогда следует не определять явно равенство упорядоченных пар, а доказывать теорему о равенстве упорядоченных пар как определенного вида множеств.

Определение 15: Обобщением понятия упорядоченной пары является *упорядоченный n -набор*, или *кортеж*. В отличие от конечного множества $\{a_1, \dots, a_n\}$ кортеж $\langle a_1, \dots, a_n \rangle$ на множествах $A_1, \dots, A_n$ характеризуется не только входящими в него элементами $a_1 \in A_1, \dots, a_n \in A_n$, но и порядком, в котором они перечисляются.

Определение 16: Два кортежа $\alpha = \langle a_1, \dots, a_n \rangle$ и $\beta = \langle b_1, \dots, b_n \rangle$ на множествах $A_1, \dots, A_n$ равны, если $a_i = b_i, i = \overline{1, n}$.

Определение 17: Число n называется *длиной кортежа* (или размерностью кортежа), а элемент a_i — *i -й проекцией* (компонентой) кортежа.

Для двух кортежей одинаковой размерности их компоненты с одинаковыми номерами называют *одноименными компонентами*. Определение равенства кортежей можно переформулировать так: два кортежа одинаковой размерности равны тогда и только тогда, когда их одноименные компоненты совпадают. В отличие от множества, кортеж может иметь повторяющиеся элементы, но все эти элементы различны. Компоненты кортежа могут обозначать любые понятия, объекты, в том числе элементы множества или кортежа.

Простейшим примером кортежа является арифметический вектор.

Кортеж, который не содержит компонентов в своем составе, называется *пустым* кортежем и обозначается $\alpha = \langle \rangle$. Длина этого кортежа равна нулю.

Для любых кортежей α, β, γ справедливы утверждения:

- Если $\alpha = \beta$, то $\beta = \alpha$.
- Если $\alpha = \beta$ и $\beta = \gamma$, то $\alpha = \gamma$.

Пример 1.44. В программировании *кортежи (или tuples)* представляют собой упорядоченные наборы элементов, которые могут содержать данные разных типов. Они часто используются для хранения фиксированных наборов данных. Приведём несколько примеров кортежей, реализованных на языке программирования **Python**, которые могут быть полезны инженерам-программистам

1. Кортежи для представления координат:
`point = (3, 4)` # Кортеж, представляющий точку в двумерном пространстве.
2. Кортежи для представления множеств:
`set_example = (1, 2, 3)` # Кортеж, представляющий конечное множество.
3. Кортежи для хранения значений функции:
`function_values = ((0, 1), (1, 2), (2, 4), (3, 9))` # Кортежи для значений функции $y = x^2$.
4. Кортежи для представления отношений:
`relation = ((1, 2), (2, 3), (3, 4))` # Кортежи, представляющие бинарное отношение.
5. Кортежи для представления графов:
`edges = ((1, 2), (2, 3), (3, 4), (4, 1))` # Кортежи, представляющие ребра графа.
6. Кортежи для представления вероятностных распределений:
`probability_distribution = ((0, 0.1), (1, 0.5), (2, 0.4))` # Кортежи с вероятностями для различных исходов.
7. Кортежи для представления матриц:
`matrix = ((1, 2), (3, 4))` # Кортежи, представляющие матрицу 2x2.

Отношения на множествах

Определение 18: *Отношение* – это связь между любыми объектами в природе. На формальном языке *отношение* – это пара множеств, причем

упорядоченное, первая компонента которой является подмножеством квадрата второй компоненты.

Определение 19: *Бинарным отношением* на множестве A называется пара $\Phi = (A, G)$, где A – область задания отношения, G —график отношения, причём $G \in A^2$.

Замечание: Если $(x, y) \in G$, то будем писать $x\Phi y$ и говорить, что x и y *вступают в отношение* Φ . Если x и y не вступают в отношение Φ , будем писать $(x\Phi y)'$.

Определение 20: *Диагональю* множества A^2 называется график $\Delta_A = \{(x, x) | x \in A\}$.

Определение 21: Множество $D_R = \{x : (\exists y)xRy\}$ называется *областью определения* бинарного отношения R .

Определение 22: *Областью значений* бинарного отношения R называется множество $I_R = \{y : (\exists x)xRy\}$.

Замечание: Каждое бинарное отношение R есть подмножество прямого (декартова) произведения некоторых множеств X и Y , таких, что $D_R \subseteq X$ и $I_R \subseteq Y$.

Пример 1.45. Рассмотрим множество $\{(1, 2); (2, 4); (3, 3); (2, 1)\}$. Это бинарное отношение R для $X = \{1, 2, 3\}$; $Y = \{1, 2, 3, 4\}$. Область определения такого отношения D_R есть $\{1, 2, 3\}X$, а область значений I_R — множество $\{2, 4, 3, 1\} Y$.

Определение 23: *Обратным отношением* для отношения R называется отношение R^{-1} , такое, что $R^{-1} = \{(x, y) : (y, x) \in R\}$.

Определение 24: Множество упорядоченных n -к, т. е. $R \subseteq X_1 \times X_2 \times \dots \times X_n$, называется *n -местным отношением* Φ для $X_1, X_2, \dots, X_n$.

Замечание: Многочестные отношения удобно задавать с помощью реляционных таблиц. Такое задание соответствует перечислению множества n -к отношения Φ . Реляционные таблицы широко используют в компьютерной практике в реляционных базах данных. При этом имена множеств X_i называют *атрибутами* (свойствами), а элементы $x_i \in X_i$ называют *доменами* (значениями) атрибутов. Заметим, что реляционные таблицы широко используются в повседневной практике. Всевозможные производственные, финансовые, научные и другие отчеты часто имеют форму реляционных таблиц.

X_1	X_2	...	X_n
x_1	x_2	...	x_n
x'_1	x'_2	...	x'_n
...	...	...	...

Рис. 1.6. Реляционная таблица

Свойства отношений

1. Рефлексивность: если $\forall a \in A a \Phi a$;
2. Антирефлексивность: если $\forall a \in A a' \Phi a$;

3. Симметричность: если $\forall a, b \in A a \varphi b \rightarrow b \varphi a$;
4. Антисимметричность: если $\forall a, b \in A a \varphi b \& b \varphi a \rightarrow a = b$;
5. Транзитивность: если $\forall a, b, c \in A a \varphi b \& b \varphi c \rightarrow a \varphi c$;
6. Полнота, или линейность: если $\forall a, b \in A a \neq b \rightarrow a \varphi b$ или $b \varphi a$.

Определение 25: Отношение $\varphi = \langle \Phi, M \rangle$ называется *пустым*, если график Φ является пустым множеством. Т.е. $\varphi = \langle \emptyset, M \rangle$. Другими словами имеется область задания отношения, на котором не задан график отношения.

Определение 26: Отношение $\varphi = \langle \Phi, M \rangle$ называется *отношением равенства*, если $\Phi = \Delta M$. В теоретико-множественном плане можно записать, $(\forall a, b \in M)(a \varphi b \rightarrow a = b)$. Например задано $\varphi = \langle \Phi, M \rangle$, $M = \{1, 2, 3\}$, $\Phi = \{\langle 1, 1 \rangle, \langle 2, 2 \rangle, \langle 3, 3 \rangle\}$. Данное отношение является отношением *равенства*.

Определение 27: Отношение $\varphi = \langle \Phi, M \rangle$ называется *отношением неравенства*, если $\Phi = M^2 \setminus \Delta M$, т.е. $(\forall a, b \in M)(a \varphi b \rightarrow a \neq b)$. Например задано $\varphi = \langle \Phi, M \rangle$, $M = \{1, 2, 3\}$, $\Phi = \{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 1 \rangle\}$. Данное отношение является *отношением неравенства*.

Замечание: Отношения « $5 > 3$ » и « $3 < 10$ » также являются примерами отношения неравенства.

Определение 28: Отношение называется отношением *частичного порядка*, если оно рефлексивно, антисимметрично и транзитивно.

Определение 29: Отношение называется отношением *линейного порядка*, если оно является отношением частичного порядка и линейно.

Определение 30: Отношение называется отношением *строгого порядка*, если оно антирефлексивно, антисимметрично и транзитивно.

Определение 31: Отношение называется отношением *строгого линейного порядка*, если оно – линейное отношение строгого порядка.

Определение 32: Отношение называется отношением *эквивалентности*, если оно рефлексивно, симметрично и транзитивно.

Определение 33: *Классом эквивалентности*, порождённым элементом x , называется множество всех элементов из A , вступающих с x в отношение эквивалентности.

Определение 34: *Фактор-множеством* множества A по отношению эквивалентности φ называется множество всех различных классов эквивалентности, которое обозначается A/φ .

Определение 35: *Мощность* фактор-множества A/φ называется индексом разбиения, порождённого отношением φ .

1.1.4 Мультимножества

Мы рассмотрели конечные множества, в которых отсутствуют повторяющиеся элементы. В кортежах возможны повторяющиеся элементы, но при этом значение каждого элемента определяется его местоположением. В задачах искусственного интеллекта начинают использоваться объекты с повторяющимися элементами. Л.Б. Петровский такие элементы назвал мультимножествами и разработал основы их теории.

Определение 36: *Мультимножество* – это множество с

повторяющимися элементами, где один и тот же элемент может присутствовать многократно, особенностью мультимножества является понятие кратности вхождения элемента. Элементы мультимножеств будем обозначать строчными буквами с подстрочным индексом M : $a_M, b_M, \dots$; а мультимножества – прописными буквами с подстрочным индексом M : $A_M, B_M, \dots$.

Пример 1.46. Примером мультимножеств могут служить, например, следующие совокупности элементов a, b, c, d, e, f, g, h : $A_M = \{a, b, a, d, e, c, a, b, h, h\}$, $B_M = \{d, d, e, b, b, d, e, e, h\}$, $C_M = \{a, a, d, a, c, a, a, e, c, c, g, g, g\}$.

Порядок следования элементов в мультимножестве считается несущественным. Тогда приведенные мультимножества A, B, C можно переписать следующим образом: $A_M = \{3a, 2b, c, d, e, 2h\}$.

Отметим, что отсутствующие элементы не указываются в записи мультимножества.

Формальное определение мультимножества, данное А.Б Петровским:

Определение 37: *Мультимножеством A_M , определенном на множестве $A = \{x_1, x_2, \dots\}$, все элементы x_i , которого различны, называется совокупность групп одинаковых элементов $A_M = \{k_1x_1, k_2x_2, \dots\}$, $x_i \in A$.*

Определение 38: *Группу одинаковых элементов k_ix_i , называют компонентой мультимножества, элементы x_i , входящие в компоненту k_ix_i , – экземплярами элементов мультимножества. Функция k_i принимающая числовые значения, определяет число вхождений элемента $x_i \in A$ в мультимножество A_M . Ее также называют функцией кратности или функцией числа экземпляров мультимножества A_M .*

Говорят, что элемент x_i принадлежит мультимножеству A_M (обозначается $x_i \in A_M$) и в мультимножестве A_M имеется ровно k экземпляров элемента x_i , тогда и только тогда, когда кратность элемента x_i равна $k_ix_i > 0$. Когда кратность элемента x_i равна нулю $k_ix_i = 0$, тогда говорят, что элемент x_i не содержится в мультимножестве A_M (обозначается $x_i \notin A_M$). Тем самым принадлежность элемента x_i мультимножеству A_M определяется значением функции кратности.

Основными характеристиками мультимножества являются **мощность** и **размерность**.

Определение 39: *Мощность* мультимножества A_M определяется как общее число экземпляров всех его элементов $|A_M| = \text{card}A_M$, а *размерность* мультимножества A – как общее число различных элементов $|A_M| = \dim A_M$.

Размерность мультимножества не превосходит его мощности и мощности домена $|A_M| \leq |A|$, $|A_M| \leq |G|$. Мощность мультимножества $|A_M|$ в общем случае не связана с мощностью домена $|G|$. Конечные мультимножества, имеющие мощность t и состоящие из t элементов (считая повторения), называют ***t*-кардинальными мультимножествами** или ***t*-мультимножествами**, а имеющие размерность n и состоящие из n компонент – ***n*-мерными мультимножествами**.

Определение 40: *Высотой* или *пиковым значением* мультимножества

A_M называется максимальное значение его функции кратности k_i , а элемент x_{A^*} , для которого функция кратности k_A максимальна, - *пиком* или *пиковым* элементом мультимножества A_M .

Мультимножества A и B называют:

- *равномощными*, если $|A|=|B|$.
- *равноразмерными*, если $/A/=/B/$.
- *равными*, если они равномощны и равноразмерны.

Говорят, что мультимножество B_M содержится или включено в мультимножество A_M ($A_M \subseteq B_M$), если $k_j x_j \leq k_i x_i$, для каждого элемента x_i , $x_j \in G$, $k_i x_i \in A_M$, $k_j x_j \in B_M$. Мультимножество B_M называется тогда *подмультимножеством* мультимножества A_M , а мультимножество A_M – *надмультимножеством* мультимножества B_M .

Включение мультимножества обладает свойствами рефлексивности ($A_M \subseteq A_M$) и транзитивности ($A_M \subseteq B_M$, $B_M \subseteq C_M \rightarrow A_M \subseteq C_M$), а значит, является отношением предпорядка.

Операции над мультимножествами

Над мультимножествами определены следующие основные операции: *объединение, пересечение, арифметическое сложение, арифметическое вычитание, дополнение, симметрическая разность, умножение на число, арифметическое умножение и возведение в арифметическую степень, прямое произведение и возведение в прямую степень.*

Определение 41: *Объединением* мультимножеств A_M и B_M называется мультимножество, состоящее из всех элементов, которые присутствуют хотя бы в одном из мультимножеств, и кратность каждого элемента равна максимальной кратности соответствующих элементов в объединяемых мультимножествах:

$$C_M = A_M \cup B_M = \{\max(k_i x_i, k_j x_j)\}, k_i x_i \in A_M, k_j x_j \in B_M.$$

Другими словами, производится попарное сравнение каждого экземпляра мультимножеств и в каждой паре выбирается экземпляр с наибольшим значением функции кратности.

Определение 42: *Пересечением* мультимножеств A_M и B_M называется мультимножество, состоящее из всех элементов, которые одновременно присутствуют в каждом из мультимножеств, и кратность каждого элемента равна минимальной кратности соответствующих элементов в пересекаемых мультимножествах:

$$C_M = A_M \cap B_M = \{\min(k_i x_i, k_j x_j)\}, k_i x_i \in A_M, k_j x_j \in B_M.$$

Другими словами, производится попарное сравнение каждого экземпляра мультимножеств и в каждой паре выбирается экземпляр с наименьшим значением функции кратности.

Определение 43: *Арифметической суммой* мультимножеств A_M и B_M называется мультимножество, состоящее из всех элементов, которые присутствуют хотя бы в одном из мультимножеств, и кратность каждого элемента равна сумме кратностей соответствующих элементов в

складываемых мультимножествах:

$$C_M = A_M + B_M = \{kx \mid kx = k_i x_i + k_j x_j\}, k_i x_i \in A_M, k_j x_j \in B_M.$$

Операции объединение, пересечение и арифметическое сложение можно выполнять для произвольного числа мультимножеств.

Определение 44: *Арифметической разностью* мультимножеств A_M и B_M называется мультимножество, состоящее из тех элементов мультимножества A_M , кратность которых превышает кратность соответствующих элементов в мультимножестве B_M . Кратность каждого элемента результирующего множества равна разности кратностей соответствующих элементов в вычитаемых мультимножествах:

$$C_M = A_M - B_M = \{kx \mid kx = k_i x_i - k_j x_j, \text{ если } k_i > k_j; 0, \text{ в противном случае}\}, k_i x_i \in A_M, k_j x_j \in B_M.$$

Определение 45: *Симметрической разностью* мультимножеств A_M и B_M называется мультимножество, состоящее из тех элементов мультимножества A_M и B_M , кратности которых различны. Кратность каждого элемента результирующего множества равна модулю разности кратностей соответствующих элементов в вычитаемых мультимножествах:

$$C_M = A_M \setminus B_M = \{kx \mid kx = |k_i x_i - k_j x_j|\}, k_i x_i \in A_M, k_j x_j \in B_M.$$

Арифметическая и симметрическая разности мультимножеств применима только к двум мультимножествам.

Определение 46: *Дополнением* мультимножества A_M до универсума U называется мультимножество, состоящее из тех элементов, кратность которых равна разности кратностей соответствующих элементов в универсуме U и дополняемом мультимножестве A_M . Под *универсумом* в данном случае понимается некоторое мультимножество U , такое, что все остальные мультимножества являются подмультимножествами данного множества U . То есть, $A_M' = U - A_M = \{k_{A'} x \mid k_{Ux} - k_{Ax}, \forall x \in U\}$.

Из определений пустого мультимножества и дополнения мультимножества следует, что пустое мультимножество $\emptyset$ и универсум U взаимно дополняют друг друга: $\emptyset' = U$, $U' = \emptyset$.

Определение 47: *Арифметическим произведением* мультимножеств A_M и B_M называется мультимножество, состоящее из элементов, которые одновременно присутствуют в каждом из мультимножеств, и их кратность равна произведению кратностей соответствующих элементов в перемножаемых мультимножествах:

$$C_M = A_M \cdot B_M = \{kx \mid kx = |k_i x_i \cdot k_j x_j|\}, k_i x_i \in A_M, k_j x_j \in B_M.$$

Определение 48: *Прямое произведение* мультимножеств A_M и B_M называется мультимножество, состоящее из всех упорядоченных пар элементов $\langle x_i, x_j \rangle$, таких, что первый элемент каждой пары является элементом первого сомножителя $x_i \in A_M$, второй элемент пары - элементом второго сомножителя $x_j \in B_M$ и кратность каждой пары $\langle x_i, x_j \rangle$ равна произведению кратностей элементов x_i и x_j в перемножаемых мультимножествах:

$$C_M = A_M \times B_M = \{k_{A \times B} \langle x_i, x_j \rangle \mid k_{A \times B} = k_i x_i \cdot k_j x_j\}, x_i \in A_M, x_j \in B_M.$$

По аналогии с множествами, для мультимножеств также можно

сформулировать некоторые **правила выполнения операций**:

1. $(A \cup B)' = A' \cap B'$;
2. $(A \cap B)' = A' \cup B'$;
3. $(A + B)' = A' - B = B' - A$;
4. $(A - B)' = A' + B$;
5. $A' - B' = B - A$;
6. $A + B = (A \cup B) + (A \cap B)$;
7. $A \setminus B = (A \cup B) - (A \cap B) = A' \setminus B'$;
8. $(A - B) \cap (B - A) = \emptyset$;
9. $A \cup B = (A + B) - (A \cap B) = (A \cap B) + A \setminus B = A + (B - A) = B + (A - B)$;
10. $A \cap B = (A + B) - (A \cup B) = (A \cup B) - A \setminus B = A - (A - B) = B - (B - A)$;
11. $A - B = (A \cup B) - B = A - (A \cap B) = (A \cup B) \setminus B = A \setminus (A \cap B)$;
12. $A \setminus B = (A \cup B) - (A \cap B) = (A - B) + (B - A) = (A - B) \cup (B - A) = (A + B) - 2 \cdot (A \cap B)$;
13. $A + B = (A \cup B) + (A \cap B) = A \setminus B + 2 \cdot (A \cap B)$.

1.1.5 Кодирование

Вопросы кодирования издавна играли заметную роль в математике. Десятичная позиционная система счисления – это способ кодирования натуральных чисел. Римские цифры – другой способ кодирования натуральных чисел, причем гораздо более наглядный и естественный: палец – I, пятерня – V, две пятерни – X. Однако при этом способе кодирования труднее выполнять арифметические операции над большими числами, поэтому он был вытеснен позиционной десятичной системой. Любопытно, что у римлян не было символа для обозначения нуля.

Определение 49: *Декартовы координаты* – способ кодирования геометрических объектов числами.

Задача кодирования:

Пусть заданы алфавиты $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$ и функция $F: S \rightarrow B^*$, где S – некоторое множество слов в алфавите A , $S \subset A^*$. Тогда функция F называется *кодированием*, элементы множества S – *сообщениями*, а элементы $\beta = F(\alpha)$, $\alpha \in S$, $\beta \in B^*$ – *кодами* соответствующих сообщений.

Определение 50: Обратная функция F^{-1} (если она существует) называется *декодированием*.

Если $|B| = n$, то F называется ***n*-ичным кодированием**. Наиболее распространенный случай $B = \{0, 1\}$ – ***двоичное*** кодирование. Именно этот случай рассматривается далее.

I. Алфавитное кодирование

Определение 51: **Алфавитное кодирование** – кодирование, при котором элементарной частью сообщения является одна буква алфавита.

Пусть задано конечное множество $A = \{a_1, \dots, a_n\}$, которое называется **алфавитом**. Элементы алфавита называются **буквами**. Последовательность букв называется **словом**. Множество слов в алфавите A обозначается A^* . Если слово $\alpha = a_1 \dots a_k \in A^*$, то количество букв в слове называется **длиной** слова: $|\alpha| = |a_1 \dots a_k| = k$

Пустое слово обозначается Λ : $\Lambda \in A^*$, $|\Lambda| = 0$, $\Lambda \notin A$.

Замечание: Если $\alpha = \alpha_1 \alpha_2$, то α_1 называется началом, или *префиксом*, слова α , а α_2 – окончанием, или *постфиксом*, слова α . Если при этом $\alpha_1 \neq \Lambda$ (соответственно, $\alpha_2 \neq \Lambda$), то α_1 (соответственно, α_2) называется *собственным началом* (соответственно, *собственным окончанием*) слова α .

Алфавитное (или побуквенное) кодирование задается схемой (или таблицей кодов) σ :

$$\sigma = \langle a_1 \rightarrow \beta_1, \dots, a_n \rightarrow \beta_n \rangle, \quad a_i \in A, \quad \beta_i \in B^*.$$

Определение 52: Множество кодов букв $V = \{\beta_i\}$ называется множеством **элементарных кодов**. Алфавитное кодирование пригодно для любого множества сообщений S :

$$F: A^* \rightarrow B^*, \quad a_{i_1} \dots a_{i_k} = \alpha \in A^*, \quad F(\alpha) = \beta_{i_1} \dots \beta_{i_k}.$$

Пример 1.47. Рассмотрим алфавиты $A = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $B = \{1, 2\}$ и схему:

$$\sigma = \langle 0 \rightarrow 0, 1 \rightarrow 1, 2 \rightarrow 10, 3 \rightarrow 11, 4 \rightarrow 100, 5 \rightarrow 101, 6 \rightarrow 110, \\ 7 \rightarrow 111, 8 \rightarrow 1000, 9 \rightarrow 1001 \rangle.$$

Эта схема однозначна, но кодирование не является взаимно однозначным: $F_\sigma(333) = 111111 = F_\sigma(77)$, а значит, невозможно декодирование. С другой стороны, схема

$$\sigma = \langle 0 \rightarrow 0000, 1 \rightarrow 0001, 2 \rightarrow 0010, 3 \rightarrow 0011, 4 \rightarrow 0100, 5 \rightarrow 0101, 6 \rightarrow 0110, \\ 7 \rightarrow 0111, 8 \rightarrow 1000, 9 \rightarrow 1001 \rangle,$$

известная под названием «*двоично-десятичное кодирование*», допускает однозначное декодирование.

II. Разделимые схемы

Рассмотрим схему алфавитного кодирования σ и различные слова, составленные из элементарных кодов.

Определение 53: Схема σ называется *разделимой*, если $\beta_{i_1} \dots \beta_{i_k} = \beta_{j_1} \dots \beta_{j_l} \Rightarrow k = l \ \& \ \forall t \in 1..k \ i_t = j_t$, то есть любое слово, составленное из элементарных кодов, единственным образом разлагается на элементарные коды. Алфавитное кодирование с разделимой схемой допускает декодирование.

Определение 54: Схема σ называется *префиксной*, если элементарный код одной буквы не является префиксом элементарного кода другой буквы:

$$(\forall i \neq j \ \beta_i, \beta_j \in V) \Rightarrow (\forall \beta \in B^* \ \beta_i \neq \beta_j \beta).$$

Теорема 1.1. Префиксная схема является разделимой.

Замечание. Свойство быть префиксной является достаточным, но не является необходимым для разделимости схемы.

Пример 1.48. Разделимая, но не префиксная схема: $A = \{a, b\}$, $B = \{0, 1\}$, $\delta = \{a \rightarrow 0, b \rightarrow 01\}$.

Чтобы схема алфавитного кодирования была разделимой, необходимо, чтобы длины элементарных кодов удовлетворяли определенному соотношению, известному как *неравенство Макмиллана*.

Теорема 1.2. Если схема $\sigma = \langle a_i \rightarrow \beta_i \rangle_{i=1}^n$ разделима, то $\sum_{i=1}^n \frac{1}{2^{l_i}} \leq 1$, где $l_i = |\beta_i|$.

Пример 1.49 *Азбука Морзе* – это схема алфавитного кодирования $\langle A \rightarrow 01, B \rightarrow 1000, C \rightarrow 1010, D \rightarrow 100, E \rightarrow 0, F \rightarrow 0010, G \rightarrow 110, H \rightarrow 0000, I \rightarrow 00, J \rightarrow 0111, K \rightarrow 101, L \rightarrow 0100, M \rightarrow 11, N \rightarrow 10, O \rightarrow 111, P \rightarrow 0110, Q \rightarrow 1101, R \rightarrow 010, S \rightarrow 000, T \rightarrow 1, U \rightarrow 001, V \rightarrow 0001, W \rightarrow 011, X \rightarrow 1001, Y \rightarrow 1011, Z \rightarrow 1100 \rangle$,

где по историческим и техническим причинам 0 называется *точкой*, а 1 называется *тире*.

Имеем

$$\begin{aligned} & 1/4 + 1/16 + 1/16 + 1/8 + 1/2 + 1/16 + 1/8 + 1/16 + 1/4 + 1/16 + 1/8 + 1/16 + 1/4 + 1/4 + \\ & + 1/8 + 1/16 + 1/16 + 1/8 + 1/8 + 1/2 + 1/8 + 1/16 + 1/8 + 1/16 + 1/16 + 1/16 = \\ & = 2/2 + 4/4 + 7/8 + 12/16 = 3 + 5/8 > 1. \end{aligned}$$

Таким образом, неравенство Макмиллана для азбуки Морзе не выполнено, и эта схема не является разделимой. На самом деле в азбуке Морзе имеются дополнительные элементы – паузы между буквами (и словами), которые позволяют декодировать сообщения. Эти дополнительные элементы определены неформально, поэтому прием и передача сообщений с помощью азбуки Морзе, особенно с высокой скоростью, является некоторым искусством, а не простой технической процедурой.

III. Помехоустойчивое кодирование

Пусть имеется канал связи C , содержащий источник помех:

$$S \xrightarrow{C} S', \quad S \in A^*, S' \in B^*,$$

где S – множество переданных, а S' – соответствующее множество принятых по каналу сообщений.

Определение 55: Кодирование F , обладающее таким свойством, что

$$S \xrightarrow{F} K \xrightarrow{C} K' \xrightarrow{F^{-1}} S, \quad \forall s \in S, \quad F^{-1}(C(F(s))) = s,$$

называется *помехоустойчивым*.

Если $A=B=\{0,1\}$, то ошибки в канале могут быть следующих типов:

- $0 \rightarrow 1, 1 \rightarrow 0$ – ошибка типа замещения разряда;

- $0 \rightarrow \Lambda, 1 \rightarrow \Lambda$ – ошибка типа выпадения разряда;
- $\Lambda \rightarrow 1, \Lambda \rightarrow 0$ – ошибка типа вставки разряда.

Замечание: Пусть E_s^δ – множество слов, которые могут быть получены из слова s в результате всех возможных комбинаций допустимых в канале ошибок δ , то есть $s \in S \subset A^*, E_s^\delta \subset B^*$. Если $s' \in E_s^\delta$, то та конкретная последовательность ошибок, которая позволяет получить из слова s слово s' , обозначается $E_s^\delta \langle s, s' \rangle$. Если тип возможных ошибок в канале подразумевается, то индекс δ не указывается.

Теорема 1.3. Чтобы существовало помехоустойчивое кодирование с исправлением всех ошибок, необходимо и достаточно, чтобы $\forall s_1, s_2 \in S E_{s_1} \cap E_{s_2} = \emptyset$, то есть необходимо и достаточно, чтобы существовало разбиение множества B^* на множества B_s ($\cup B_s = B^*, \cap B_s = \emptyset$), такое что $\forall s \in S E_s \subset B_s$.

Определение 56: Расстояние Хэмминга $d(\vec{a}, \vec{b})$ между кодовыми словами определяется как число позиций i таких, что $\vec{a}_i \neq \vec{b}_i$. При построении вектора $\vec{a} + \vec{b}$ пары одинаковых двоичных символов $(a_i, b_i), a_i = b_i$ будут давать 0, а пары различных двоичных символов будут давать 1.

Пример 1.50 $(0, 1, 0, 1) + (1, 1, 1, 0) = (1, 0, 1, 1)$.

Поэтому расстояние Хэмминга $d(\vec{a}, \vec{b})$ можно определить как число компонент вектора $\vec{a} + \vec{b}$, которые равны 1.

Введенное таким образом расстояние Хэмминга удовлетворяет следующим условиям:

- $d(\vec{a}, \vec{b}) \geq 0$;
- $d(\vec{a}, \vec{b}) = d(\vec{b}, \vec{a})$;
- $d(\vec{a}, \vec{b}) + d(\vec{b}, \vec{c}) \geq d(\vec{a}, \vec{c})$ – «неравенство треугольника».

В общем случае расстоянием называют функцию, удовлетворяющую вышеприведенным условиям. Этим условиям, в частности, удовлетворяет геометрическое расстояние между двумя точками (x_1, y_1) и (x_2, y_2) на плоскости: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Замечание: При наличии l бит можно построить 2^l кодовых слов. Однако часто нужны не все из этих слов. Если использовать только часть кодовых слов, то с помощью проверок на четность и других приемов можно обнаружить ошибки и даже их исправлять.

Пример 1.51. Множество A_i называется областью декодирования для слова $\vec{a}^{(i)}$.

Замечание: Так как области декодирования не пересекаются, то при приеме слова из области декодирования A_i можно считать, что передавалось слово $\vec{a}^{(i)}$. Если при этом число ошибок не превосходит r , то при приеме

всегда будет приниматься правильное решение.

Так как число элементов $|A_i|$ в множестве A_i равно числу способов выбора некоторых k ($0 \leq k \leq r$) компонент слова $\vec{a}^{(i)}$, то

$$|A_i| = \binom{l}{0} + \binom{l}{1} + \dots + \binom{l}{r}.$$

Далее, так как $|V| = 2^l$, то: $p \cdot \sum_{k=0}^r \binom{l}{k} \leq 2^l$, то есть

$$p \leq \frac{2^l}{\sum_{k=0}^r \binom{l}{k}}. \quad (*)$$

Неравенство (*) называется *границей Хэмминга*. Оно дает оценку сверху для числа p кодовых слов, которые можно использовать для передачи, если их длина равна l бит и нужно исправлять все r -кратные ошибки.

Пример 1.52. Пусть $l=8$, тогда $|V| = 2^8 = 256$. Поскольку $\binom{l}{r} = \frac{l!}{r!(l-r)!}$, то $\binom{8}{0}=1; \binom{8}{1}=8; \binom{8}{2}=28; \binom{8}{3}=56; \binom{8}{4}=70$.

$$\text{Если } r=4, \text{ то } p=1 \leq \frac{256}{1+8+28+56+70} = \frac{256}{163}.$$

$$\text{Если } r=3, \text{ то } p=2 \leq \frac{256}{1+8+28+56} = \frac{256}{93}.$$

$$\text{Если } r=2, \text{ то } p=6 \leq \frac{256}{1+8+28} = \frac{256}{37}.$$

$$\text{Если } r=1, \text{ то } p=28 \leq \frac{256}{1+8} = \frac{256}{9}.$$

$$\text{Если } r=0, \text{ то } p=256 \leq \frac{256}{1}.$$

Таким образом, если встречаются только одиночные ошибки типа замещения, то при использовании 8 бит можно безошибочно передавать 28 слов. В общем случае, если имеется l бит и нужно исправлять одиночные ошибки типа замещения, то граница Хэмминга определяется следующим выражением, являющимся частным случаем формулы (*):

$$p \leq \frac{2^l}{1+l}. \quad (**)$$

Обозначим минимальное число бит, необходимое для передачи p слов, буквой m . Тогда: $p = 2^m$, и из формулы (**) следует

$$1+l \leq 2^{l-m}. \quad (***)$$

Таким образом, для помехоустойчивого кодирования сигналов при

одиночных ошибках типа замещения необходимо использовать дополнительные $l - m$ бит, количество которых должно удовлетворять условию (**). Рассматриваемый случай простейший, но одновременно практически очень важный. Таким свойством, как правило, обладают внутренние шины передачи данных в современных компьютерах.

Пример 1.53 Для сообщения длиной $m = 32$ потребуется $k = 6$ дополнительных разрядов, поскольку $64 = 2^6 > 32 + 6 + 1 = 39$.

Определение 57: Код, в котором используются дополнительные разряды для помехоустойчивой передачи сигналов, называется **кодом Хэмминга**.

1.1.6 Шифрование

Определение 58: **Шифрование** – это кодирование данных с целью защиты от несанкционированного доступа.

Определение 59: Область знаний о шифрах, методах их создания и раскрытия называется **криптографией (или тайнописью)**.

Криптография известна с глубокой древности и использует самые разнообразные шифры, как чисто информационные, так и механические. В настоящее время наибольшее практическое значение имеет защита данных в компьютере, поэтому далее рассматриваются программные шифры для сообщений в алфавите $\{0, 1\}$.

Начнем с определения операции: $a \bmod b$. Здесь операнды: a и b – целые числа, а результат данной операции – остаток, возникающий при целочисленном делении a на b .

Пример 1.54

$$15 \bmod 2 = 15 - 7 \cdot 2 = 1,$$

$$26 \bmod 3 = 26 - 8 \cdot 3 = 2,$$

$$32 \bmod 4 = 32 - 8 \cdot 4 = 0.$$

Пусть имеется генератор псевдослучайных чисел, работающий по определенному алгоритму. Часто используется следующий **алгоритм**:

$$T_{i+1} = (a \cdot T_i + b) \bmod c, \quad (1)$$

где T_i – предыдущее псевдослучайное число, T_{i+1} – последующее псевдослучайное число, а коэффициенты a, b, c постоянны и хорошо известны. Обычно $c = 2^n$, где n – разрядность процессора, $a \bmod 4 = 1$, a, b – нечетное. В этом случае последовательность псевдослучайных чисел имеет период c .

Пример 1.55 В табл. 1 приведены результаты расчетов по формуле (1) для различных ключей шифра T_0 и коэффициентов: $a=81, b=9, c=65536$.

Таблица 1

T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}
1	90	7299	1404	48197	37342	10055	28032	42377	24674	32523
10	819	812	245	19854	35319	42800	58937	55314	23995	43060

100	8109	1478	54191	64104	15089	42570	40307	53612	17205	17358
1000	15473	8138	3827	47852	9397	40270	50615	36592	14841	22482
10000	23577	9202	24475	16404	18013	17270	22623	63000	56737	8186

Процесс шифрования определяется следующим образом: **Шифруемое сообщение представляется в виде последовательности слов $S_0, S_1, \dots$, каждое длины n , которые складываются по модулю два со словами последовательности $T_0, T_1, \dots$, то есть $C_i = S_i \oplus T_i$.**

Определение 60: Последовательность $T_0, T_1, \dots$ называется *гаммой шифра*. Процесс расшифровывания заключается в том, чтобы еще раз сложить шифрованную последовательность с той же самой гаммой шифра: $S_i = C_i \oplus T_i$.

Определение 61: Шифры, в которых для зашифровки и расшифровки используется один и тот же ключ, называются *симметричными*.

Для повышения криптостойкости симметричных шифров применяются различные приемы.

- Вычисление гаммы шифра по ключу более сложным (или секретным) способом.
- Применение вместо $\oplus$ более сложной (но обратимой операции) для вычисления шифровки.
- Предварительное перемешивание битов исходного сообщения по фиксированному алгоритму.
-

I. Модулярная арифметика

В этом разделе все числа – целые.

Определение 62: Говорят, что число a *сравнимо по модулю n* с числом b (обозначение $a \equiv b \pmod{n}$), если a и b при делении на n дают один и тот же остаток:

$$a \equiv b \pmod{n} \Leftrightarrow a \bmod n = b \bmod n.$$

Отношение *сравнимости* рефлексивно, симметрично и транзитивно и является отношением эквивалентности.

Определение 63: Классы эквивалентности по отношению сравнимости (по модулю n) называются *вычетами* (по модулю n). Множество вычетов по модулю n обозначается Z_n .

Обычно из каждого вычета выбирают одного представителя – неотрицательное число, которое при делении на n дает частное 0. Это позволяет считать, что $Z_n = \{0, 1, 2, \dots, n-1\}$, и упростить обозначения.

Над вычетами (по модулю n) определены **операции сложения и умножения по модулю n** , обозначаемые соответственно $+_n$ и $\cdot_n$, и определяемые следующим образом:

$$a +_n b = (a + b) \bmod n, \quad a \cdot_n b = (a \cdot b) \bmod n.$$

Если из контекста ясно, что подразумевается операция по модулю n , то индекс n опускается.

Рассмотрим Z_n^* – подмножество Z_n чисел, *взаимно простых* с n . Числа называются взаимно простыми, если их наибольший общий делитель равен единице. Для чисел из множества Z_n^* существуют обратные числа по отношению к операции умножения по модулю n .

Пример 1.56

$$(2 \cdot 2) \bmod 3 = 1 \Rightarrow 2^{-1} \bmod 3 = 2,$$

$$(7 \cdot 3) \bmod 20 = 1 \Rightarrow 7^{-1} \bmod 20 = 3,$$

$$(6 \cdot 8) \bmod 47 = 1 \Rightarrow 6^{-1} \bmod 47 = 8.$$

Функция: $\varphi(n) = |Z_n^*|$ – называется **функцией Эйлера**. Если p – простое число, то $\varphi(p) = p - 1$, и вообще $\varphi(n) < n$.

Можно показать, что

$$\varphi(n) = n \left(1 - \frac{1}{p_1}\right) \dots \left(1 - \frac{1}{p_k}\right), \text{ где } p_1, \dots, p_k - \text{все простые делители } n.$$

Пример 1.57

$$\varphi(10) = 10 \cdot \left(1 - \frac{1}{2}\right) \cdot \left(1 - \frac{1}{5}\right) = \frac{10 \cdot 4}{2 \cdot 5} = 4,$$

$$Z_{10}^* = \{1, 3, 7, 9\}, \quad |Z_{10}^*| = 4.$$

$$\varphi(7) = 7 \cdot \left(1 - \frac{1}{7}\right) = 7 - 1 = 6,$$

$$Z_7^* = \{1, 2, 3, 4, 5, 6\}, \quad |Z_7^*| = 6.$$

Теорема 1.4. (Теорема Эйлера). Если $n > 1$, то $\forall a \in Z_n^* \quad a^{\varphi(n)} \equiv 1 \pmod{n}$

Пример 1.58 Пусть $n=10$. Тогда $Z_{10}^* = \{1, 3, 7, 9\}$ и $\varphi(10) = 4$.

$$1^4 \bmod 10 = 1 \bmod 10 = 1 - 0 \cdot 10 = 1,$$

$$3^4 \bmod 10 = 81 \bmod 10 = 1 - 8 \cdot 10 = 1,$$

$$7^4 \bmod 10 = 2401 \bmod 10 = 1 - 240 \cdot 10 = 1,$$

$$9^4 \bmod 10 = 6561 \bmod 10 = 1 - 656 \cdot 10 = 1.$$

Теорема 1.5. (Малая теорема Ферма). Если $p > 1$, то $\forall a \in Z_p^* \quad a^{p-1} \equiv 1 \pmod{p}$.

Имеет место следующее утверждение:

Теорема 1.6. Если числа $n_1, \dots, n_k$ попарно взаимно простые, число $n = n_1 n_2 \dots n_k$ – их произведение, x и a – целые числа, то $x \equiv a \pmod{n} \Leftrightarrow \forall i \in 1..k \ x \equiv a \pmod{n_i}$.

Последнее утверждение является следствием теоремы, которая известна как «*китайская теорема об остатках*».

II. Шифрование с открытым ключом

Шифрование с открытым ключом производится следующим образом.

1. Получателем сообщений производится генерация открытого ключа (пара чисел n и e) и закрытого ключа (число d). Для этого:

- выбираются два простых числа p и q ;
- определяется первая часть открытого ключа $n=pq$;
- определяется вторая часть открытого ключа – выбирается небольшое нечетное число e , взаимно простое с числом $(p-1)(q-1)$ (заметим, что $(p-1)(q-1) = pq(1-1/p)(1-1/q) = \phi(n)$);
- определяется закрытый ключ: $d = e^{-1} \pmod{(p-1)(q-1)}$.

После чего открытый ключ (числа n и e) сообщается всем отправителям сообщений.

2. Отправитель шифрует сообщение (разбивая его, если нужно, на слова S_i длиной менее $\log_2 n$ разрядов):

$$C_i = (S_i)^e \pmod{n},$$

и отправляет получателю.

3. Получатель расшифровывает сообщение с помощью закрытого ключа d :

$$P_i = (C_i)^d \pmod{n}.$$

Теорема 1.7. Шифрование с открытым ключом корректно, то есть в принятых обозначениях $P_i = S_i$.

Пример 1.59 Генерация ключей: $p=3, \quad q=11$;

$$n=pq=3 \cdot 11=33;$$

$$(p-1)(q-1)=2 \cdot 10=20, \quad e=7;$$

$$d=e^{-1} \pmod{20}=3, \quad (7 \cdot 3 \pmod{20}=1).$$

Пусть $S_1=3, S_2=1, S_3=2$ ($S_1, S_2, S_3 < n=33$). Тогда код определяется следующим образом:

$$C_1 = 3^7 \pmod{33} = 2187 \pmod{33} = 9;$$

$$C_2 = 1^7 \pmod{33} = 1 \pmod{33} = 1;$$

$$C_3 = 2^7 \pmod{33} = 128 \pmod{33} = 29.$$

При расшифровке имеем:

$$P_1 = 9^3 \bmod 33 = 729 \bmod 33 = 3,$$

$$P_2 = 1^3 \bmod 33 = 1 \bmod 33 = 1,$$

$$P_3 = 29^3 \bmod 33 = 24389 \bmod 33 = 2.$$

Шифры с открытым ключом сравнительно просты в реализации, очень практичны (поскольку нет необходимости пересылать по каналам связи закрытый ключ и можно безопасно хранить его в одном месте) и в то же время обладают высочайшей криптостойкостью.

Кажется, что дешифровать сообщение несложно: достаточно разложить открыто опубликованное число n на множители, восстановив числа p и q , и далее можно легко вычислить секретный ключ d .

1.2 Математическая логика

1.2.1 Соответствия и функции

Основные понятия соответствия

Определение 64: *Соответствием* между множествами X и Y будем называть тройку объектов: $\Gamma = (X, Y, G)$, где X — область отправления соответствия, Y — область прибытия соответствия, G — график соответствия, причём $G \subseteq X \times Y$.

Существует три способа задания соответствия:

- Теоретический
- Матричный
- Графический

Теоретический способ заключается в задании графика соответствия и множеств X и Y . Для графика соответствия справедливо: $G \subseteq X \times Y \Rightarrow G = X \times Y \vee G \subset X \times Y$.

При задании *матричным* способом соответствие представляется в виде матрицы R_Γ , размером $n \times m$, где строки представляют элементы множества X , столбцы — элементы множества Y , а элемент матрицы r_{ij} принимает значения:

$r_{ij} = 1$ — если существует кортеж $\langle x_i, y_j \rangle \in G$;

$r_{ij} = 0$, в противном случае.

Таким образом, соответствие можно представить в виде следующей матрицы:

$$R_\Gamma = \begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Соответствие, заданное в *графическом* виде, представляет собой граф, вершинами которого являются элементы, принадлежащие множествам X и Y соответствия $\Gamma = (X, Y, G)$, а кортежи вида $\langle x_i, y_j \rangle \in G$ представляются на графике соответствия в виде стрелок, направленных от x_i к y_j :

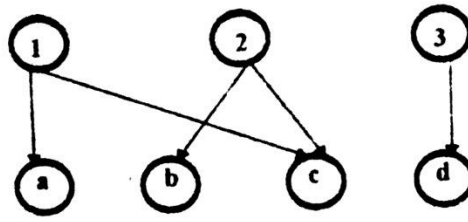


Рис. 1.7. Графическое представление соответствия

Определение 65: *Областью определения* соответствия будем называть $\text{пр}_1 G$.

Определение 66: *Областью значений* соответствия будем называть $\text{пр}_2 G$.

Определение 67: Соответствие называется *всюду определённым*, если $\text{пр}_1 G = X$.

Определение 68: Соответствие называется *сюръективным*, если $\text{пр}_2 G = Y$.

Определение 69: Соответствие будем называть *функциональным*, или функцией, если его график не содержит пар с одинаковыми первыми и различными вторыми координатами.

Определение 70: Соответствие называется *инъективным*, если его график не содержит пар с одинаковыми вторыми и различными первыми координатами.

Определение 71: Соответствие называется *отображением X в Y* , если оно всюду определено и функционально.

Определение 72: Соответствие называется *отображением X на Y* , если оно всюду определено, функционально и сюръективно.

Определение 73: Соответствие называется *взаимно однозначным*, если оно функционально и инъективно.

Определение 74: Соответствие называется *биекцией*, если оно всюду определено, сюръективно, функционально и инъективно.

Определение 75: *Образом* множества A при данном соответствии называется множество $\Gamma(B) = \{y | (x, y) \in G \text{ и } x \in A\}$.

Определение 76: *Прообразом* множества B при данном соответствии называется множество $\Gamma^{-1}(B) = \{x | (x, y) \in G \text{ и } y \in B\}$.

Определение 77: Множества называются *равномощными*, если между ними можно установить биекцию.

Определение 78: Множество называется *счётным*, если оно равномощно множеству натуральных чисел.

Определение 79: Множество называется *континуальным*, если оно равномощно множеству действительных чисел отрезка $[0, 1]$.

Операции над соответствиями

Поскольку соответствия можно считать множествами, то все операции над множествами (пересечение, объединение, разность, дополнение и т.д.) можно применить и к соответствиям. Заметим, что, говоря о дополнении

соответствия из A в B , мы имеем в виду дополнение до универсального соответствия из A в B , т.е. до декартова произведения $A \times B$. Естественно, что и равенство соответствий можно трактовать как равенство множеств.

Определение 80: Объединением соответствий $\Gamma_1 = \langle X, Y, F \rangle$ и $\Gamma_2 = \langle W, Z, P \rangle$ называют соответствие $\Gamma_1 \cup \Gamma_2 = \langle X \cup W, Y \cup Z, F \cup P \rangle$.

Определение 81: Пересечением соответствий $\Gamma_1 = \langle X, Y, F \rangle$ и $\Gamma_2 = \langle W, Z, P \rangle$ называют соответствие $\Gamma_1 \cap \Gamma_2 = \langle X \cap W, Y \cap Z, F \cap P \rangle$.

Определение 82: Разностью соответствий $\Gamma_1 = \langle X, Y, F \rangle$ и $\Gamma_2 = \langle W, Z, P \rangle$ называют соответствие $\Gamma_1 \setminus \Gamma_2 = \langle X \setminus W, Y \setminus Z, F \setminus P \rangle$.

Определение 83: Инверсией соответствия $\Gamma = \langle X, Y, F \rangle$ является соответствие Γ^{-1} , такое, что множество Y является областью отправления соответствия Γ^{-1} ; множество X является областью прибытия соответствия Γ^{-1} , а график соответствия F^{-1} является инверсией графика F соответствия Γ .

Определение 84: Композицией (произведением) соответствий $\Gamma_1 = \langle X, Y, F \rangle$ и $\Gamma_2 = \langle W, Z, P \rangle$ называют соответствие $\Gamma_1 \cdot \Gamma_2 = \langle X, Z, F \cdot P \rangle$. Поясним построение композиции двух соответствий. Областью отправления является область отправления Γ_1 , областью прибытия – область прибытия Γ_2 , а графиком – композиция графиков F и P .

В случае, если $Y \cap W = \emptyset$, то результатом композиции соответствий будет соответствие с *пустым графиком*.

Соответствие Ω называется инверсией соответствия Γ , если область отправления Γ равна области прибытия Ω и график Γ является инверсией графика Ω .

Четная инверсия оставляет соответствие самим собой, а нечетная – инвертирует. То есть $(\Gamma^{-1})^{-1} = \Gamma$, а $((\Gamma^{-1})^{-1})^{-1} = \Gamma^{-1}$. Соответствие $\Gamma^{-1} = \Gamma$ тогда и только тогда, когда график соответствия симметричен $G = G^{-1}$, а область отправления соответствия совпадает с областью прибытия.

Пример 1.60 $\Gamma = \langle G, X, X \rangle$, $X = \{1, 2, 3\}$, $G = \{ \langle 1, 1 \rangle, \langle 2, 2 \rangle \}$. Графическое представление этого соответствия:

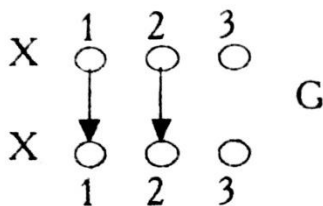


Рис. 1.8. Графическое представление соответствия примера 1.60

Для соответствия так же, как для отношений и множеств справедлива операция композиции. **Композиция соответствий** определяется через композицию их графиков. Композиция соответствий не является пустой, если существует хотя бы один элемент $y \in Y$ и $u \in Z$. Пусть заданы соответствия $\Gamma_1 = \langle G, X, Y \rangle$ и $\Gamma_2 = \langle H, Z, U \rangle$. Тогда $\Gamma_1 \cdot \Gamma_2 = \langle G \cdot H, X, U \rangle$ – определяет композицию двух соответствий.

Свойства соответствий

Определение 85: Соответствие Γ называется *функциональным*, если его график G функционален. График G называется *функциональным*, если в нем нет пар с одинаковыми первыми и разными вторыми элементами. Другими словами, из элементов области отправления может выходить не более одной стрелки.

Следовательно, соответствие $\Gamma = \langle G, X, Y \rangle$ функционально тогда, когда истинно $(\forall x \in X, \forall y_1, y_2 \in Y) [\langle x, y_1 \rangle \in G \ \& \ \langle x, y_2 \rangle \in G \Rightarrow y_1 = y_2]$.

Определение 86: Соответствие Γ называется *инъективным*, если его график инъективен. График G называется *инъективным*, если в нем нет пар с разными первыми и одинаковыми вторыми элементами.

Отметим, что в частном случае инъективные и функциональные, графики могут совпадать.

Соответствие инъективно, когда справедливо:

$$(\forall x_1, x_2 \in X, \forall y \in Y) [\langle x_1, y \rangle \in G \ \& \ \langle x_2, y \rangle \in G \Rightarrow x_1 = x_2].$$

Определение 87: Соответствие $\Gamma = \langle G, X, Y \rangle$ называется *всюду определенным*, если его область определения совпадает с его областью отправления.

Пример 1.61 $\Gamma = \langle G, X, Y \rangle = \langle \{ \langle 1, 2 \rangle, \langle 3, 2 \rangle, \langle 4, 5 \rangle \}; \{1, 3, 4\}; \{2, 5\} \rangle$. Здесь область отправления соответствия $X = \{1, 3, 4\}$ совпадает с областью определения.

Для всюду определенного соответствия справедливо выражение:

$$\text{пр}_1 G = X.$$

Аналогично можно записать: $(\forall x)(\exists y)[\langle x, y \rangle \in G]$.

Соответствие $\Gamma = \langle G, X, Y \rangle$ называется *сюръективным*, если его область значений совпадает с его областью прибытия.

Функция

Понятие «функции» является одним из основополагающих в математике. В данном случае подразумеваются прежде всего функции, отображающие одно конечное множество объектов в другое конечное множество. Мы избегаем использования термина «отображение» и предпочитаем слово «функция» в расчете на постоянное сопоставление читателем математического понятия функции с понятием функции в языках программирования.

Определение 88: Пусть f — отношение из A в B , такое что

$$\forall a (a, b) \in f \ \& \ (a, c) \in f \Rightarrow b = c.$$

Такое свойство отношения называется *однозначностью*, или *функциональностью*, а само отношение называется *функцией из A в B* и обозначается следующим образом: $f: A \rightarrow B$. Если $f: A \rightarrow B$, то обычно используется *префиксная* форма записи:

$$b = f(a) = (a, b) \in f.$$

Если $b = f(a)$, то a называют аргументом, а b — значением функции.

Пусть $f: A \rightarrow B$, тогда *область определения* функции: $f_A = \{a \in A \mid \exists b \in B \ b = f(a)\}$;

область значений функции: $f_B = \{b \in B \mid \exists a \in A \ b = f(a)\}$.

Если $f_A = A$, то функция называется **тотальной**, а если $f_A \neq A$ — **частичной**. **Сужением** функции $f: A \rightarrow B$ на множество $M \subset A$ называется функция $f|_M$, определяемая следующим образом:

$$f|_M = \{(a, b) \mid (a, b) \in f \text{ и } a \in M\}$$

Для тотальной функции $f = f|_A$.

Функция $f: A_1 \times \dots \times A_n \rightarrow B$ называется функцией **n аргументов**, или **n -местной функцией**.

Пусть $f: A \rightarrow B$. Тогда функция f называется:

инъективной, если $b = f(a_1) \text{ и } b = f(a_2) \Rightarrow a_1 = a_2$;

сюръективной, если $\forall b \in B \exists a \in A \text{ } b = f(a)$;

биективной, если она инъективная и сюръективная.

Биективную функцию также называют **взаимнооднозначной**.

Следующий рисунок иллюстрирует понятия отношения, функции, инъекции, сюръекции и биекции.

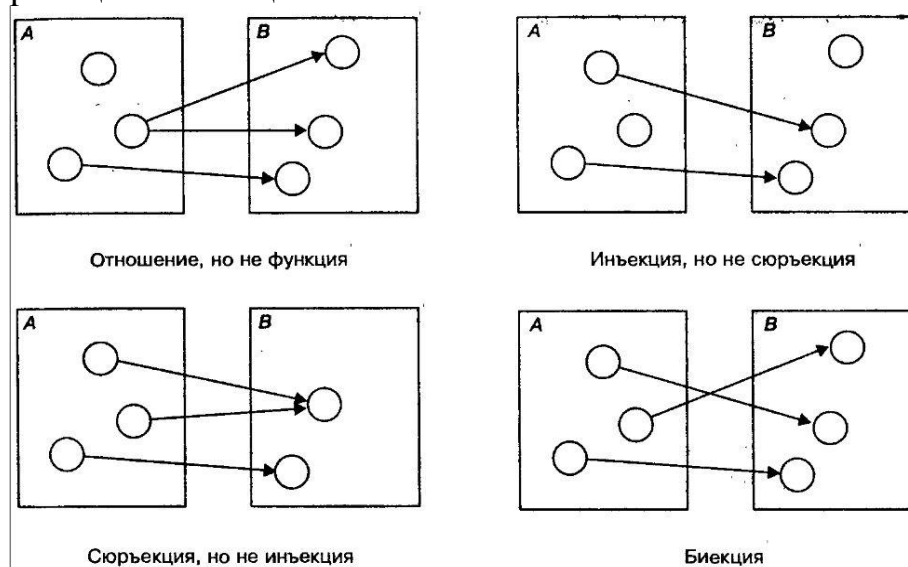


Рис. 1.9. Иллюстрация основных свойств функции

Принцип Дирихле. Пусть $f: A \rightarrow B$ — функция, причем X и Y — конечные множества. Если $|X| > |Y|$, то по крайней мере одно значение f встретится более одного раза. Неформально, принцип Дирихле можно например записать следующим образом: Если X — множество белок, а Y — множество клеток, и $|X| = 12$, а $|Y| = 11$, то 12 белок нельзя посадить в 11 клеток так, чтобы в каждой клетке находилась одна белка.

1.2.2 Элементарные булевы функции, канонические способы задания

Если x — логическая переменная, $\delta \in \{0,1\}$, то выражение

$$x^\delta \in \begin{cases} x, & \text{если } \delta = 1, \\ \bar{x}, & \text{если } \delta = 0 \end{cases}$$

называется **литерой**. Литеры x и $\bar{x}$ называются **контрарными**.

Элементарной конъюнкцией или **конъюнктом** называется конъюнкция литер.

Элементарной дизъюнкцией или дизъюнктом называется дизъюнкция литер.

Пример 1.62 Формулы $x \vee \bar{y} \vee \bar{z}$ и $x \vee y \vee x \vee \bar{z}$ - дизъюнкты, формулы $\bar{x}_1 x_2 x_3$ и $x_1 x_2 \bar{x}_3$ - конъюнкты, а $\bar{x}$ является одновременно и дизъюнктом, и конъюнктом.

Дизъюнкция конъюнктов называется *дизъюнктивной нормальной формой* (ДНФ); конъюнкция дизъюнктов называется *конъюнктивной нормальной формой* (КНФ).

Пример 1.63 Формула $x\bar{y} \vee yz$ — ДНФ, формула $(x \vee z \vee \bar{y})(x \vee z)y$ — КНФ, а формула $x\bar{y}$ является одновременно КНФ и ДНФ.

Теорема 1.8.

1. Любая формула эквивалентна некоторой ДНФ.
2. Любая формула эквивалентна некоторой КНФ.

Алгоритм приведения формулы к ДНФ.

1. Выразить все логические операции, участвующие в построении формулы, через дизъюнкции, конъюнкции и отрицания, используя эквивалентности

$$\begin{aligned}\varphi \rightarrow \psi &\sim \neg \varphi \vee \psi \\ \varphi \leftrightarrow \psi &\sim (\neg \varphi \vee \psi) \wedge (\neg \psi \vee \varphi)\end{aligned}$$

и определения операций:

штрих Шеффера (антиконъюнкция) $(X | Y) = \overline{X \wedge Y}$,

стрелки Пирса (антидизъюнкция) $X \downarrow Y = \overline{X \vee Y}$

сумма по модулю два (антиэквивалентность) $X \oplus Y = \overline{X \leftrightarrow Y}$.

2. Используя законы де Моргана, переносим все отрицания к переменным и сокращаем двойные отрицания по правилу

$$\neg \neg \varphi \sim \varphi$$

3. Используя закон дистрибутивности

$$(\varphi \vee (\psi \wedge \chi)) \sim ((\varphi \vee \psi) \wedge (\varphi \vee \chi)),$$

преобразуем формулу так, чтобы все конъюнкции выполнялись раньше дизъюнкций. В результате применения пп. 1-3 получается ДНФ данной формулы.

Пример 1.64. Привести к ДНФ формулу $\varphi = ((x \rightarrow y) \downarrow \neg(y \rightarrow z))$.

Решение. Выразим логические операции $\rightarrow$ и $\downarrow$ через $\wedge$, $\vee$ и $\neg$:

$$\varphi \sim ((\bar{x} \vee y) \downarrow \neg(\bar{y} \vee z)) \sim \neg((\bar{x} \vee y) \vee \neg(\bar{y} \vee z)).$$

В полученной формуле перенесем отрицание к переменным и сократим двойные отрицания:

$$\varphi \sim (\neg(\bar{x} \vee y) \wedge \neg \neg(\bar{y} \vee z)) \sim (\bar{x} \wedge \bar{y}) \wedge (\bar{y} \vee z) \sim (x \wedge \bar{y}) \wedge (\bar{y} \vee z).$$

Используя закон дистрибутивности, приводим формулу к ДНФ:

$$\varphi \sim (x \wedge \bar{y} \wedge \bar{y}) \vee (x \wedge \bar{y} \wedge z).$$

Приведение формулы к КНФ производится аналогично приведению ее к ДНФ, только вместо п. 3 применяется пункт

3'. Используя закон дистрибутивности $(\varphi \vee (\psi \wedge \chi)) \sim ((\varphi \vee \psi) \wedge (\varphi \vee \chi))$, преобразуем формулу так, чтобы все дизъюнкции выполнялись раньше, чем конъюнкции.

Пример 1.65 Привести к КНФ формулу $\varphi = (x \rightarrow y) \wedge ((\bar{y} \rightarrow z) \rightarrow \bar{x})$,

Решение. Преобразуем формулу φ к формуле, не содержащей $\rightarrow$:

$$\varphi \sim (\neg x \vee y) \wedge (\neg(\bar{y} \rightarrow z) \vee \bar{x}) \sim (\neg x \vee y) \wedge (\neg(\bar{y} \vee z) \vee \bar{x})$$

В полученной формуле перенесем отрицание к переменным и сократим двойные отрицания: $\varphi \sim (\bar{x} \vee y) \wedge ((\bar{\bar{y}} \wedge \bar{z}) \vee \bar{x}) \sim (\bar{x} \vee y) \wedge ((\bar{y} \wedge \bar{z}) \vee \bar{x})$

По закону дистрибутивности получаем, что формула φ эквивалентна формуле

$$\varphi \sim (\bar{x} \vee y) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{x} \vee \bar{z}),$$

являющейся КНФ.

Упростим полученную формулу:

1) используем закон дистрибутивности

$$(\bar{x} \vee y) \wedge (\bar{x} \wedge \bar{y}) \wedge (\bar{x} \vee \bar{z}) \sim (\bar{x} \vee (\bar{y} \wedge \bar{y})) \wedge (\bar{x} \vee \bar{z})$$

2) используем закон эквивалентности $\varphi \vee \varphi \sim \varphi$,

$$(\bar{x} \vee (\bar{y} \wedge \bar{y})) \wedge (\bar{x} \vee \bar{z}) \sim \bar{x} \wedge (\bar{x} \vee \bar{z})$$

3) используем закон поглощения

$$\bar{x} \wedge (\bar{x} \vee \bar{z}) \sim \bar{x}.$$

Таким образом, формула φ из примера 6.1.1 эквивалентными преобразованиями приводится к формуле $\bar{x}$ (являющейся одновременно ДНФ и КНФ формулы φ).

$$(\bar{x} \vee y) \wedge (\bar{y} \wedge \bar{y}) \wedge (\bar{x} \vee \bar{z}) \sim (\bar{x} \vee (\bar{y} \wedge \bar{y})) \wedge (\bar{x} \vee \bar{z}) \sim \bar{x} \wedge (\bar{x} \vee \bar{z}) \sim \bar{x}$$

Построить таблицу истинности $\varphi = (x \rightarrow y) \wedge ((\bar{y} \rightarrow z) \rightarrow \bar{x})$

Любая булева функция может иметь бесконечно много представлений в виде ДНФ и КНФ. Особое место среди этих представлений занимают совершенные ДНФ (СДНФ) и совершенные КНФ (СКНФ).

Совершенные ДНФ (СДНФ) и КНФ (СКНФ).

Пусть $(x_1, \dots, x_n)$ — набор логических переменных, $\Delta = (\delta_1, \dots, \delta_n)$ — набор нулей и единиц.

Конституентой единицы набора Δ называется конъюнкт $K^1(\delta_1 \dots \delta_n) = x_1^{\delta_1} x_2^{\delta_2} \dots x_n^{\delta_n}$.

Конституентой нуля набора Δ называется дизъюнкт $K^0(\delta_1 \dots \delta_n) = x_1^{1-\delta_1} \vee x_2^{1-\delta_2} \vee \dots \vee x_n^{1-\delta_n}$.

Отметим, что $K^1(\delta_1 \dots \delta_n) = 1$, а $K^0(\delta_1 \dots \delta_n) = 0$ тогда и только тогда, когда $x_1 = \delta_1, x_2 = \delta_2, x_n = \delta_n$.

Совершенной ДНФ называется дизъюнкция некоторых конституент единицы, среди которых нет одинаковых.

Совершенной КНФ называется конъюнкция некоторых конститuent нуля, среди которых нет одинаковых.

Таким образом, СДНФ (СКНФ) есть ДНФ (КНФ), в которой в каждый конъюнкт (дизъюнкт) каждая переменная x_i из набора $\{x_1, \dots, x_n\}$ входит ровно один раз, причем входит либо сама x_i , либо ее отрицание $\bar{x}_i$.

Пример 1.66 Формула $x_1 \wedge \bar{x}_2 \wedge x_3$ есть конститuent единицы $K^1(1,0,1)$.

Формула $x \vee y \vee \bar{z}$ есть конститuent нуля $K^0(0,0,1)$.

Формула $x_1 \wedge x_2 \wedge x_3 \vee \bar{x}_1 \wedge x_2 \wedge x_3$ — СДНФ.

Формула $(x \vee y \vee \bar{z}) \wedge (\bar{x} \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z)$ — СКНФ.

Формула $x_1 \wedge \bar{x}_2 \wedge x_3 \vee \bar{x}_1 \wedge x_2 \wedge x_3 \vee x_1 \wedge \bar{x}_2 \wedge x_3$ не является СДНФ.

Первая теорема Шеннона

Для решения задачи нахождения СДНФ и СКНФ, эквивалентных исходной формуле φ , предварительно рассмотрим разложения булевой функции $f(x_1, x_2, \dots, x_n)$ по k ; переменным (для определенности по $x_1, x_2, \dots, x_k$) — разложения Шеннона.

Теорема 1.9. (Первая теорема Шеннона). Любая булева функция $f(x_1, x_2, \dots, x_n)$ представима в виде разложения Шеннона:

$$f(x_1, x_2, \dots, x_k, x_{k+1}, \dots, x_n) = \bigvee_{\substack{\bar{1}1 \dots \bar{1} \bar{1} \\ \bar{1} \bar{1} \bar{1} \bar{1} \bar{1} \\ (\delta_1, \dots, \delta_k)}} \left(\bigwedge_{i=1}^k x_i^{\delta_i} \right) f(\delta_1, \delta_2, \dots, \delta_k, x_{k+1}, \dots, x_n)$$

Доказательство. Прежде всего, заметим, что $x_i^{\delta_i} = 1 \Leftrightarrow x_i = \delta_i$. Подставим произвольно вместо первых k переменных их значения: $x_1 = \delta_1^*, x_2 = \delta_2^*, \dots, x_k = \delta_k^*$. Тогда левая часть доказываемой формулы равна $f(\delta_1^*, \delta_2^*, \dots, \delta_k^*, x_{k+1}, \dots, x_n)$. Правая часть представляет собой дизъюнкцию 2^k конъюнкций вида $x_1^{\delta_1} x_2^{\delta_2} \dots x_k^{\delta_k} f(\delta_1, \delta_2, \dots, \delta_k, x_{k+1}, \dots, x_n)$, которые этой подстановкой разбиваются на два класса. К первому классу относится конъюнкция, у которой набор $(\delta_1, \delta_2, \dots, \delta_k)$ совпадает с набором $(\delta_1^*, \delta_2^*, \dots, \delta_k^*)$:

$$\begin{aligned} & (\delta_1^*)^{\delta_1^*} (\delta_2^*)^{\delta_2^*} \dots (\delta_k^*)^{\delta_k^*} f(\delta_1^*, \delta_2^*, \dots, \delta_k^*, x_{k+1}, \dots, x_n) = \\ & = 1 \ 1 \ \dots \ 1 \ f(\delta_1^*, \delta_2^*, \dots, \delta_k^*, x_{k+1}, \dots, x_n) = f(\delta_1^*, \delta_2^*, \dots, \delta_k^*, x_{k+1}, \dots, x_n) \end{aligned}$$

Эта конъюнкция равна левой части формулы. Ко второму классу относится $2^k - 1$ конъюнкций, у каждой из которых хотя бы в одной переменной x_i , $i \in \{1, 2, \dots, k\}$ выполнено условие $\delta_i^* \neq \delta_i$. Следовательно, каждая из них равна нулю. Используя закон $a \vee 0 = a$, получаем, что левая и правая части формул равны при любой подстановке переменных $x_1, x_2, \dots, x_n$.

Вторая теорема Шеннона

В силу принципа двойственности для булевых алгебр справедлива

Теорема 1.10. (Вторая теорема Шеннона). Любая булева функция $f(x_1, x_2, \dots, x_n)$ представима в виде разложения Шеннона:

$$f(x_1, x_2, \dots, x_k, x_{k+1}, \dots, x_n) = \bigwedge_{\substack{\bar{1} \cdot \bar{a} \bar{n} \bar{a} \bar{1} \cdot \bar{1} \bar{a} \bar{a} \bar{1} \bar{0} \bar{a} \bar{1} \\ (\delta_1, \dots, \delta_k)}} \left(\bigvee_{i=1}^k x_i^{1-\delta_i} \vee f(\delta_1, \delta_2, \dots, \delta_k, x_{k+1}, \dots, x_n) \right)$$

В предельном случае, когда $k = n$, для булевой функции $f(x_1, x_2, \dots, x_n)$, не равной нулю, получаем ее представление в виде совершенной ДНФ:

$$f(x_1, x_2, \dots, x_n) = \bigvee_{\substack{\bar{1} \cdot \bar{a} \bar{n} \bar{a} \bar{1} \cdot \bar{1} \bar{a} \bar{a} \bar{1} \bar{0} \bar{a} \bar{1} \\ (\delta_1, \delta_2, \dots, \delta_n) \\ \bar{1} \bar{a} \cdot \bar{0} \bar{1} \bar{0} \bar{1} \bar{0} \bar{0} \bar{0} \\ f(\delta_1, \dots, \delta_n)=1}}$$

Аналогично для булевой функции $f(x_1, x_2, \dots, x_n)$, не равной единице, получаем ее представление в виде совершенной КНФ:

$$f(x_1, x_2, \dots, x_n) = \bigwedge_{\substack{\bar{1} \cdot \bar{a} \bar{n} \bar{a} \bar{1} \cdot \bar{1} \bar{a} \bar{a} \bar{1} \bar{0} \bar{a} \bar{1} \\ (\delta_1, \delta_2, \dots, \delta_n) \\ \bar{1} \bar{a} \cdot \bar{0} \bar{1} \bar{0} \bar{1} \bar{0} \bar{0} \bar{0} \\ f(\delta_1, \dots, \delta_n)=0}}$$

Приведенные формулы позволяют сформулировать следующую теорему о функциональной полноте.

Функциональная полнота

Теорема 1.11. (О функциональной полноте). Для любой булевой функции f найдется формула φ , представляющая функцию f . Если $f \neq 0$, то существует представляющая ее формула φ , находящаяся в СДНФ:

$$\varphi = \bigvee_{f(\delta_1, \dots, \delta_n)=1} K^1(\delta_1, \dots, \delta_n)$$

и такое представление единственно с точностью до порядка следования конъюнкт единиц. Если $f \neq 1$, то существует представляющая ее формула φ , находящаяся в СКНФ:

$$\varphi = \bigvee_{f(\delta_1, \dots, \delta_n)=0} K^0(\delta_1, \dots, \delta_n),$$

и такое представление единственно с точностью до порядка следования конъюнкт нуля.

Пример 1.67

Найти СДНФ и СКНФ функции $f(x, y, z)$, заданной следующей таблицей истинности:

x	0	0	0	0	1	1	1	1
y	0	0	1	1	0	0	1	1
z	0	1	0	1	0	1	0	1
$f(x, y, z)$	1	0	0	1	0	1	0	1

Решение. По теореме о функциональной полноте СДНФ имеет вид $\varphi_1 = \bar{x}\bar{y}\bar{z} \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee xyz$, а СКНФ — $\varphi_2 = (x \vee y \vee \bar{z})(x \vee \bar{y} \vee z)(\bar{x} \vee y \vee z)(\bar{x} \vee \bar{y} \vee \bar{z})$. Для нахождения СДНФ и СКНФ исходной формулы φ составляется ее таблица истинности, а затем по ней строится требуемая совершенная нормальная форма. По СДНФ φ_1 для функции f , можно составить ее таблицу истинности и по ней найти СКНФ φ_2 .

Описанный способ нахождения СДНФ и СКНФ по таблице истинности бывает часто более трудоемким, чем следующий алгоритм.

Алгоритм нахождения СДНФ.

Для нахождения СДНФ данную формулу нужно привести сначала к ДНФ, а затем преобразовать ее конъюнкты в конституенты единицы с помощью следующих действий:

- а) если в конъюнкт входит некоторая переменная вместе со своим отрицанием, то нужно удалить этот конъюнкт из ДНФ;
- б) если в конъюнкт одна и та же литера x^δ входит несколько раз, то удалить все литеры x^δ , кроме одной;
- в) если в некоторый конъюнкт $x_1^{\delta_1} x_2^{\delta_2} \dots x_k^{\delta_k}$ не входит переменная y , то этот конъюнкт заменить на эквивалентную формулу $x_1^{\delta_1} x_2^{\delta_2} \dots x_n^{\delta_n} \wedge (y \vee \bar{y})$ применяя закон дистрибутивности, привести полученную формулу к ДНФ; если недостающих переменных несколько, то для каждой из них к конъюнкту добавляется соответствующую формулу вида $(y \vee \bar{y})$;
- г) если в полученной ДНФ имеется несколько одинаковых конституент единицы, то оставить только одну из них. В результате получается СДНФ.

Пример 1.68 Найдем СДНФ для ДНФ $\varphi = x\bar{x} \vee x \vee yz$.

Решение. Имеем $\varphi \sim x \vee yz \sim$

$$\begin{aligned} &\sim (x(y \vee \bar{y}) \vee yz(x \vee \bar{x})) \sim \\ &\sim (xy \vee x\bar{y} \vee xyz \vee \bar{x}yz) \sim \\ &\sim (xy(z \vee \bar{z}) \vee x\bar{y}(z \vee \bar{z}) \vee xyz \vee \bar{x}yz) \sim \\ &\sim (xyz \vee xy\bar{z} \vee x\bar{y}z \vee x\bar{y}\bar{z} \vee xyz \vee \bar{x}yz) \sim \\ &\sim (xyz \vee xy\bar{z} \vee x\bar{y}z \vee x\bar{y}\bar{z} \vee \bar{x}yz). \end{aligned}$$

Алгоритм приведения КНФ к СКНФ аналогичен вышеизложенному описанию алгоритма приведения ДНФ к СДНФ.

Минимизация булевых функций в классе ДНФ

Вхождение переменной – это место, которое переменная занимает в формуле. Каждая формула имеет конечное число вхождений переменных. Задача заключается в том, чтобы для данной булевой функции f найти ДНФ, представляющую эту функцию и имеющую наименьшее число вхождений переменных.

Элементарным произведением называется конъюнкт, в который любая переменная входит не более одного раза.

Пример 1.70 Формула $x_2 \bar{x}_3 x_4$ — элементарное произведение, а формула $x_1 x_2 x_1 \bar{x}_3$ элементарным произведением не является.

Формула $\varphi(x_1, x_2, \dots, x_n)$ называется *импликантой* формулы $\psi(x_1, x_2, \dots, x_n)$, если φ — элементарное произведение и $\varphi \wedge \psi \sim \varphi$, т. е. для соответствующих формулам φ и ψ функций f_φ и f_ψ справедливо неравенство $f_\varphi \leq f_\psi$. Формула $\varphi(x_1, x_2, \dots, x_n)$ называется *импликантой* функции f , если φ — импликанта совершенной ДНФ, представляющей функцию f .

Импликанта $\varphi(x_1, x_2, \dots, x_n) = x_{i_1}^{\delta_1} x_{i_2}^{\delta_2} \dots x_{i_k}^{\delta_k}$ формулы ψ называется *простой*, если после отбрасывания любой литеры из φ не получается формула, являющаяся импликантой формулы ψ .

Пример 1.71 Найти все импликанты и простые импликанты для формулы

$$\varphi(x, y) = x \rightarrow y.$$

Всего имеется 8 элементарных произведений с переменными x и y . Ниже приведены их таблицы истинности:

x	y	$\varphi = x \rightarrow y$	$\bar{x}\bar{y}$	$\bar{x}y$	$x\bar{y}$	xy	$\bar{x}$	$\bar{y}$	x	y
0	0	1	1	0	0	0	1	1	0	0
0	1	1	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1	1	0
1	1	1	0	0	0	1	0	0	1	1

Из таблиц истинности заключаем, что формулы $\bar{x}\bar{y}$, $\bar{x}y$, $x\bar{y}$, xy , $\bar{x}$, y — все импликанты формулы φ , а из этих импликант простыми являются формулы $\bar{x}$ и y . Например, формула $\bar{x}\bar{y}$ не является простой импликантой, поскольку отбрасывая литеру $\bar{y}$, получаем импликанту $\bar{x}$.

Дизъюнкция всех простых импликант данной формулы (функции) называется *сокращенной ДНФ*.

Теорема 1.12. Любая булева функция, не являющаяся константой 0, представима в виде сокращенной ДНФ.

В примере функция, соответствующая формуле $x \rightarrow y$, представима формулой $\bar{x} \vee y$, которая является ее сокращенной ДНФ.

Сокращенная ДНФ может содержать лишние импликанты, удаление которых не меняет таблицы истинности. Если из сокращенной ДНФ удалить все лишние импликанты, то получается ДНФ, называемая *тупиковой*. Представление функции в виде тупиковой ДНФ в общем случае неоднозначно.

Выбор из всех тупиковых форм формы с наименьшим числом вхождений переменных дает *минимальную ДНФ* (МДНФ).

Метод Квайна

Рассмотрим *метод Квайна* для нахождения МДНФ, представляющей данную булеву функцию. Определим следующие три операции:

- операция полного склеивания - $\varphi \cdot x \vee \varphi \cdot \bar{x} \sim \varphi \cdot (x \vee \bar{x}) \sim \varphi$;
- операция неполного склеивания - $\varphi \cdot x \vee \varphi \cdot \bar{x} \sim \varphi \cdot (x \vee \bar{x}) \vee \varphi \cdot x \vee \varphi \cdot \bar{x} \sim \varphi \vee \varphi x \vee \varphi \bar{x}$;
- операция элементарного поглощения - $\varphi \cdot x^\delta \vee \varphi \sim \varphi, \delta \in \{0, 1\}$.

Теорема 1.13. (Теорема Квайна). Если, исходя из совершенной ДНФ функции, произвести все возможные операции неполного склеивания, а затем элементарного поглощения, то в результате получится сокращенная ДНФ, т. е. дизъюнкция всех простых импликант.

Пример 1.72 Определить сокращенную ДНФ функции f . Пусть функция $f(x, y, z)$ задана совершенной ДНФ $\varphi = \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}\bar{z} \vee xyz$.

Решение. Произведем в два этапа все возможные операции неполного склеивания, а затем элементарного поглощения, имеем

$$\begin{aligned}\varphi &\sim \bar{x}y(\bar{z} \vee z) \vee y\bar{z}(x \vee \bar{x}) \vee yz(x \vee \bar{x}) \vee xz(y \vee \bar{y}) \vee xy(z \vee \bar{z}) \vee \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}z \vee xyz \sim \\ &\sim \bar{x}y \vee y\bar{z} \vee yz \vee xz \vee xy \vee \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}z \vee xyz \sim \\ &\sim y(\bar{x} \vee x) \vee y(\bar{z} \vee z) \vee \bar{x}y \vee y\bar{z} \vee yz \vee xz \vee xy \vee \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}z \vee xyz \sim \\ &\sim y \vee \bar{x}y \vee y\bar{z} \vee yz \vee xz \vee xy \vee \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}z \vee xyz \sim \\ &\sim y \vee xz.\end{aligned}$$

Таким образом, сокращенной ДНФ функции f является формула $y \vee xz$.

На практике при выполнении операций неполного склеивания на каждом этапе можно не писать члены, участвующие в этих операциях, а писать только результаты всевозможных полных склеиваний и конъюнкты, не участвующие ни в каком склеивании.

Пример 1.73 Пусть функция $f(x, y, z)$ задана совершенной ДНФ $\varphi = \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}z \vee xyz$. Тогда, производя операции склеивания, а затем элементарного поглощения, имеем

$$\begin{aligned}\varphi &\sim \bar{x}y(\bar{z} \vee z) \vee y\bar{z}(x \vee \bar{x}) \vee xz(\bar{y} \vee y) \sim \\ &\sim \bar{x}y \vee y\bar{z} \vee xz.\end{aligned}$$

Для получения минимальной ДНФ из сокращенной ДНФ используется *матрица Квайна*, которая строится следующим образом.

В заголовках столбцов таблицы записываются конституенты единицы совершенной ДНФ, а в заголовках строк — простые импликанты из полученной сокращенной ДНФ.

В таблице звездочками отмечаются те пересечения строк и столбцов, для которых конъюнкт, стоящий в заголовке строки, входит в конституенту единицы, являющейся заголовком столбца.

Пример 1.74 Пусть функция $f(x, y, z)$ задана совершенной ДНФ $\varphi = \bar{x}y\bar{z} \vee \bar{x}yz \vee x\bar{y}z \vee xyz$.

Матрица Квайна имеет вид

Импликанты	Конституенты единицы			
	$\bar{x}y\bar{z}$	$\bar{x}yz$	$x\bar{y}z$	xyz
$\bar{x}y$	*	*		
$y\bar{z}$		*	*	
xz			*	*

По матрице Квайна находим, что минимальная ДНФ заданной функции есть $\bar{x}y \vee xz$.

В тупиковую ДНФ выбирается минимальное число простых импликант, дизъюнкция которых сохраняет все конституенты единицы, т. е. каждый столбец матрицы Квайна содержит звездочку, стоящую на пересечении со строкой, соответствующей одной из выбранных импликант.

В качестве минимальной ДНФ выбирается тупиковая, имеющая наименьшее число вхождений переменных.

В силу принципа двойственности для булевых алгебр все приведенные понятия и рассуждения очевидным образом можно преобразовать для нахождения минимальных конъюнктивных нормальных форм (МКНФ).

Каноническое представление логических функций

Каноническими формами логических (формул) функций называются выражения, имеющие стандартную форму булевой формулы такой, которая однозначно представляет логическую функцию.

В алгебре логики каноническими формами логических функций является совершенная конъюнктивная нормальная форма (СКНФ) и совершенная дизъюнктивная нормальная форма (СДНФ).

1. СДНФ (совершенной ДНФ) называется ДНФ, в которой нет равных элементарных конъюнкций и все натуральные конъюнкции, содержащие одни и те же переменные (длина одинакова). Причем каждая переменная входит только один раз, включая вхождение под знак отрицания.

В этой формуле присутствуют элементарные конъюнкции второго ряда (в ВТ называемые минтермами).

Для любой логической функции, не являющейся тождественным нулем, можно построить реализующие ее однозначно СДНФ.

Для построения СДНФ, заданной таблицей соответствия необходимо по каждому двоичному набору (булеву вектору, кортежу), на котором функция принимает значение 1, записать конъюнкцию n -го порядка так, что не инвертируется те переменные логической функции, которые имеют в кортеже значение 1.

2. СКНФ – совершенная конъюнктивная нормальная форма, такая форма, в которой нет одинаковых сомножителей и все сомножители содержат одни и те же переменные, причем каждая переменная только 1 раз включает знак вхождения под знак отрицания.

Логическая функция n аргументов не равная тождественно 1 реализуется однозначно СКНФ.

Замечания.

1) Для построения СКНФ логической функции n аргументов, заданной таблицей соответствия, необходимо по каждому кортежу переменных, на которой логическая функция принимает значение 0, записать дизъюнкцию всех n переменных, инвертируя переменные, имеющие значения 1.

Сигнатура называется *функционально полной*, если всякая логическая функция может быть реализована формулой, содержащей лишь символы функций из сигнатуры.

Система булевых функций называется *полной*, если любая булева функция может быть

Эквивалентные преобразования логических выражений (т. е. способ получения равносильных формул) направлен на их упрощение и минимизацию

а) Упрощение формул означает получение равносильных формул с

меньшим числом символов их образующих. В булевой алгебре логики для упрощения формул используются следующие тождества (аксиомы, законы, тождественные константы):

Элементарные булевы функции (**дизъюнкция**, **конъюнкция** и т.д.) в булевых функциях выполняют задание аналогичное элементарным функциям ($x, x^2, \sin, \log$, и т.п.) в классической математике при изучении анализа.

Определение 89: булевы функции (функции алгебры логики)

Пусть $U = \{u_1, u_2, \dots, u_m, \dots\}$ - исходный алфавит переменных (аргументов).

Рассмотрим функции $f(u_{i_1}, u_{i_2}, \dots, u_{i_m})$ ($u_{i_\nu} \neq u_{i_\mu}$ при $\nu \neq \mu$), аргументы которых определены на множестве $E_2 = \{0, 1\}$ и такие, что $f(\alpha_1, \alpha_2, \dots, \alpha_n) \in E_2$, когда $\alpha_i \in E_2$ ($i = 1, 2, \dots, n$) эти функции называются **функциями алгебры логики** или **булевыми функциями** (Джордж Буль (1815-1864)).

Другое определение.

Определение 90: Функции $f: E_2^n \rightarrow E_2$ где $E_2 = \{0, 1\}$ называются **функциями алгебры логики** или **булевыми функциями**. Множество булевых функций от n переменных обозначаем P_n . Булеву функцию от n переменных можно задать *таблицей истинности*.

Системы булевых функций

Пусть даны булевы функции $f(g_1, g_2, \dots, g_m)$ и $g_1(x_1, x_2, \dots, x_n), g_2(x_1, x_2, \dots, x_n), \dots, g_m(x_1, x_2, \dots, x_n)$. Подставим функции g_i в функцию f . Получим новую булеву функцию

$f(g_1(x_1, x_2, \dots, x_n), g_2(x_1, x_2, \dots, x_n), \dots, g_m(x_1, x_2, \dots, x_n))$,

которую называют *суперпозицией функций f и g_i ($i = 1, 2, \dots, m$)*, т.е. при помощи операции суперпозиции получили новую булеву функцию.

Набор булевых функций $M = \{f_1, f_2, \dots, f_k\}$ называется *полной системой булевых функций* или *базисом*, если любая булева функция выражается через них при помощи операции суперпозиции в конечном числе раз.

Пример 1.75

Набор булевых функций:

$$M = \{\vee, \wedge, \neg\} = \{\vee, \cdot, \neg\} = \{f_1 = x_1 \wedge x_2 = x_1 \cdot x_2, f_2 = x_1 \vee x_2, f_3 = \bar{x}\}$$

является полной системой булевых функций, так как любая булева функция может быть аналитически представлена в форме СДНФ или СКНФ.

Эту полную систему называют *стандартным базисом*.

Пример 1.76

$$x_1 \Rightarrow x_2 = \bar{x}_1 \vee x_2$$

$$x_1 | x_2 = \bar{x}_1 \vee \bar{x}_2$$

$$x_1 \oplus x_2 = (\bar{x}_1 \wedge x_2) \vee (x_1 \wedge \bar{x}_2)$$

Теорема 1.14. (О двух системах) Пусть имеется полная система булевых функций $M = \{f_1, f_2, \dots, f_m\}$. Тогда для полноты некоторой другой системы булевых функций $N = \{g_1, g_2, \dots, g_n\}$ необходимо и достаточно, чтобы любая

функция $f_i \in M$ выражалась через функции системы N при помощи операции суперпозиции.

Доказательство.

Необходимость очевидна. (Почему?)

Достаточность. Рассмотрим произвольную булеву функцию h . Тогда она выражается через функции $f_i \in M$ при помощи операции суперпозиции в силу полноты системы M . Но, в свою очередь, любая функция $f_i \in M$ выражается через функции $g_j \in N$ при помощи операции суперпозиции. Следовательно, можно через функции g_j выразить любую булеву функцию, и, значит, система N полна.

Базис Жегалкина.

Пример 1.77

Рассмотрим систему $N = \{\oplus, 1\}$. Она является полной, так как любая функция из стандартного базиса выражается через функции из N :

$$x_1 \wedge x_2 = x_1 \cdot x_2$$

$$\bar{x} = x \oplus 1$$

$$x_1 \vee x_2 = \overline{\bar{x}_1 \cdot \bar{x}_2} = \bar{x}_1 \cdot \bar{x}_2 \oplus 1 = (x_1 \oplus 1) \cdot (x_2 \oplus 1) = x_1 \cdot x_2 \oplus x_1 \oplus x_2$$

Эта полная система называется *базисом Жегалкина*.

Любая булева функция может быть представлена в этом базисе в форме многочлена Жегалкина по степеням неизвестных.

ЗВ $\bar{x}_1 \cdot \bar{x}_2 = (x_1 \oplus 1) \cdot (x_2 \oplus 1) = x_1 \cdot x_2 \oplus x_1 \oplus x_2 \oplus 1$

Пример 1.78

Рассмотрим систему булевых функций, состоящую из одной функции Шеффера $N = \{\downarrow\}$. Она является тоже полной, так как любая функция из стандартного базиса выражается через функции из N :

$$\bar{x} = x \downarrow x$$

$$x_1 \wedge x_2 = (x_1 \downarrow x_2) \downarrow (x_1 \downarrow x_2)$$

$$x_1 \vee x_2 = (x_1 \downarrow x_1) \downarrow (x_2 \downarrow x_2)$$

Класс булевых функций называется *замкнутым*, если всякая суперпозиция функций этого класса будет функцией из этого класса.

Классы булевых функций

Множество всех булевых функций P_2 является замкнутым классом (Почему?). Общее их число всех булевых функций n переменных равно его мощности $|P_2| = 2^{2^n}$. Из множества этих функций выделяют некоторые классы (подмножества).

1. Булева функция f сохраняет константу 0 , если

$$f(0, 0, \dots, 0) = 0$$

Множество всех булевых функций, сохраняющих константу 0 , образует класс $K_0 \subset P_2$.

Теорема 1.15. Число всех булевых функций, сохраняющих константу 0 , равно 2^{2^n-1} .

Доказательство. Булева функция, принадлежащая классу K_0 , на наборе $\langle 0, 0, \dots, 0 \rangle$ принимает значение равное 0 , но таких наборов $2^n - 1$, следовательно общее число таких функций $|K_0| = 2^{2^n-1}$.

2. Булева функция f сохраняет константу 1 , если $f(1, 1, \dots, 1) = 1$

Множество всех булевых функций, сохраняющих константу 1 , образует класс $K_1 \subset P_2$.

Теорема 1.16. Число всех булевых функций, сохраняющих константу 1 , равно 2^{2^n-1} .

Доказательство. Булева функция, принадлежащая классу K_1 , на наборе $\langle 1, 1, \dots, 1 \rangle$ принимает значение равное 1 , но таких наборов $2^n - 1$, следовательно общее число таких функций $|K_1| = 2^{2^n-1}$.

3. Булева функция $g(x_1, x_2, \dots, x_n) = \overline{f(\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n)}$ называется двойственной к функции f и обозначается f^* .

ЗВ. По определению $(x \wedge y)^* = \overline{(x \wedge y)} = (x \vee y)$, т.е. функция $x \vee y$ является двойственной к функции $x \wedge y$.

Рассмотреть, что происходит с таблицей истинности.

Замечание. Замена кортежей соответствует переворачиванию таблицы.

Булева функция называется *самодвойственной*, если она совпадает с двойственной себе функцией $f = f^*$.

ЗВ. Класс самодвойственных булевых функций: 1. $f = \bar{x}$; 2. $f = xy \vee xz \vee yz$.

Множество всех самодвойственных булевых функций, образует класс $S \subset P_2$.

Теорема 1.17. Число всех самодвойственных булевых функций равно $2^{2^{n-1}}$.

Доказательство. Любую булеву самодвойственную функцию, принадлежащую классу S , можно определить на половине наборов, т.е. 2^{n-1} , следовательно, общее число таких функций $|S| = 2^{2^{n-1}}$.

4. Булева функция f называется *линейной* функцией, если она может быть записана в следующем виде $f(x_1, x_2, \dots, x_n) = c_0 \oplus c_1 x_1 \oplus c_2 x_2 \oplus \dots \oplus c_n x_n$, где $c_i \in \{0, 1\}$ $\{k = 0, 1, 2, \dots, n\}$.

Множество всех линейных булевых функций, образует класс $L \subset P_2$.

Теорема 1.18. Число всех линейных булевых функций равно 2^{n+1} .

Доказательство. Имеется всего 2^{n+1} различных наборов значений коэффициентов c_k . следовательно, общее число таких функций $|L| = 2^{n+1}$.

5. Набор значений переменных $\langle \sigma_1, \sigma_2, \dots, \sigma_n \rangle$ не меньше набора $\langle \sigma'_1, \sigma'_2, \dots, \sigma'_n \rangle$, если $\sigma_j \geq \sigma'_j$ для всех $j = 1, 2, \dots, n$. В этом случае пишут

$\langle \sigma_1, \sigma_2, \dots, \sigma_n \rangle \geq \langle \sigma'_1, \sigma'_2, \dots, \sigma'_n \rangle$, в противном случае наборы называются *несравнимыми*.

Булева функция f называется *монотонной* функцией, если для любых двух наборов, таких, что $\langle \sigma_1, \sigma_2, \dots, \sigma_n \rangle \geq \langle \sigma'_1, \sigma'_2, \dots, \sigma'_n \rangle$ имеет место неравенство

$$f(\sigma_1, \sigma_2, \dots, \sigma_n) \geq f(\sigma'_1, \sigma'_2, \dots, \sigma'_n).$$

Множество всех *монотонных* булевых функций, образует класс $M \subset P_2$.

Число всех *монотонных* булевых функций оценивается асимптотически:

$$2^{C_n^{\lfloor \frac{n}{2} \rfloor}} \leq |M| \leq 2^{AC_n^{\lfloor \frac{n}{2} \rfloor}}, \text{ где } A - \text{некоторая константа.}$$

Теорема 1.19. (Пост-Яблонского) Для того чтобы множество N булевых функций было полной системой, необходимо и достаточно найти для каждого из пяти замкнутых классов Поста K_0, K_1, S, M, L функцию из N , которая ему не принадлежит.

ЗВ. Рассмотрим множество из одной функции Шеффера $N = \{ \downarrow \}$. Это - полная система. Согласно теореме Поста эта единственная функция должна не принадлежать ни одному из классов Поста.

1. $x_1 | x_2 = \overline{(x_1 \cdot x_2)} = \bar{x}_1 \vee \bar{x}_2$; $0 | 0 = 1 \vee 1 = 1$ ВЫВОД $\{ \downarrow \} \notin K_0$
2. $1 | 1 = 0 \vee 0 = 0$ ВЫВОД $\{ \downarrow \} \notin K_1$
3. $\bar{x}_1 | \bar{x}_2 = \bar{x}_1 \cdot \bar{x}_2 \neq x_1 | x_2$ ВЫВОД $\{ \downarrow \} \notin S$
4. $x_1 | x_2 = x_1 \cdot x_2 \oplus 1$ ВЫВОД $\{ \downarrow \} \notin L$
5. $0 | 0 = 1, 1 | 1 = 0$ ВЫВОД $\{ \downarrow \} \notin M$

Теорема Поста

Теорема Поста устанавливает необходимые и достаточные условия полноты системы булевых функций.

(Post E.L. The two-valued interactive systems of mathematical logic. – Annals of Math. Studies, v. 5, Princeton Univ. Press. Princeton-London, 1941).

Требования : P_2 класс - замкнутый, то Базис - конечный.

Теорема 1.21. (Пост). Каждый замкнутый класс из P_2 имеет конечный базис.

Теорема 1.22. (Пост). Мощность множества замкнутых классов в P_2 - счетная.

Хотя вторая теорема следует из первой, тем не менее, в доказательстве Поста сначала устанавливается второй факт, а затем - первый.

Теорема 1.23. (Пост). Система булевых функций F полна тогда и только тогда, когда она содержит

- хотя бы одну функцию, не сохраняющую нуль,
- хотя бы одну функцию, не сохраняющую единицу,
- хотя бы одну несамоудовлетворяющую функцию,
- хотя бы одну немонотонную функцию,

хотя бы одну нелинейную функцию:

$$[F] = P_n \Leftrightarrow \neg(F \subset T_0 \vee F \subset T_1 \vee F \subset T_* \vee F \subset T_{\leq} \vee F \subset T_L)$$

Доказательство.

Необходимость. От противного. Пусть $[F] = P_n$ и

$$F \subset T_0 \vee F \subset T_1 \vee F \subset T_* \vee F \subset T_{\leq} \vee F \subset T_L.$$

Введем обозначение: i – один из индексов $0, 1, *, \leq, L$.

Тогда $T_i = [T_i] \Rightarrow [F] \subset T_i \Rightarrow P_n \subset T_i \Rightarrow P_n = T_i$, но $P_n \neq T_i$ по таблице

Таблица принадлежности некоторых булевых функций рассмотренным замкнутым классам:

	T_0	T_1	T_*	$T_{\leq}$	T_L
0	+	-	-	+	+
1	-	+	-	+	+
$\bar{x}$	-	-	+	-	+
$x_1 \wedge x_2$	+	+	-	+	-

Таким образом, рассмотренные классы $T_0, T_1, T_*, T_{\leq}, T_L$ попарно различны, не пусты и не совпадают с P_n ($P_n \neq T_i$).

Функции двух переменных $z = f(x, y)$.

Различных функций двух переменных существует уже шестнадцать. Эти функции, их названия и обозначения приведены в табл. 4.1. Число этих функций равно $2^4 = 16$. Перенумеруем и расположим их тоже в естественном порядке.

Таблица принадлежности некоторых булевых функций рассмотренным замкнутым классам:

	K_0	K_1	S	M	L
0	+	-	-	+	+
1	-	+	-	+	+
$\bar{x}$	-	-	+	-	+
$x_1 \wedge x_2$	+	+	-	+	-

Таким образом, рассмотренные классы K_0, K_1, S, M, L попарно различны, не пусты и не совпадают с P_2 .

Алгебра Жегалкина

Сумма по модулю 2, конъюнкция и константы 0 и 1 образуют функционально полную систему, т.е. образуют алгебру - алгебру Жегалкина.

$$A = \langle F_B, \oplus, \wedge, 0, 1 \rangle$$

1.2.3 Высказывания, методы проверки логического следования

Математическая логика изучает базовые понятия синтаксиса (формы) и семантики (содержания) естественного языка. Рассмотрим три крупных направления исследований в математической логике — логику высказываний, логику предикатов и теорию доказательств. Эти направления потребуются

при изучении кибернетических и интеллектуальных систем.

Известно, что в исчислении высказываний теоремами являются общезначимые формулы, поэтому автоматизация доказательства осуществляется в виде проверки общезначимости формул с помощью таблиц истинности. Проверка истинности формул производится для любого конечного набора значений переменных. Таким образом, можно алгоритмизировать процесс доказательства теорем, но не процесс вывода теорем из аксиом.

Определения

Определение 91: *Высказыванием* называется повествовательное предложение (текст) естественного языка, о котором имеет смысл говорить истинно оно или ложно.

Например, «студент Петров присутствует на лекции», «эта стена белая», « $3^3 = 28$ » и т.д.

Предложение «Город x стоит на реке y » не является высказыванием, пока не заданы x и y , так как здесь нельзя определить истинно это или ложь.

Из данных высказываний можно составлять новые, сложные высказывания с помощью так называемых *логических операций*.

Истинность значений сложных высказываний определяется только истинностью значений составляющих высказываний, а не их смыслом. Простейшие высказывания, в которых не выделяются части, являющиеся высказываниями, будем обозначать прописными латинскими буквами $A, B, \dots, P, \dots$ и называть *атомарными*.

Определение 92: *Отрицанием* высказывания P называется высказывание, истинное тогда и только тогда, когда P ложно. Отрицание P обозначается $\neg P$ или $\bar{P}$ и читается «не P ».

Отрицание высказывания определяется табл. 1.

Таблица истинности отрицания

P	$\bar{P}$
И	Л
Л	И

В естественном языке отрицание соответствует составлению из высказывания P нового высказывания «неверно, что P », или «не P ».

Определение 93: *Конъюнкцией* двух высказываний P и Q называется высказывание, истинное тогда и только тогда, когда истинны оба высказывания. Конъюнкция P и Q обозначается $P \& Q$ или $P \wedge Q$ и читается « P и Q ».

Конъюнкция определяется

Таблица истинности конъюнкции

P	Q	$P \wedge Q$
И	И	И
И	Л	Л
Л	И	Л
Л	Л	Л

В естественном языке конъюнкция соответствует соединению высказываний союзом «и».

Определение 94: Дизъюнкцией двух высказываний P и Q называется высказывание ложное тогда и только тогда, когда ложны оба эти высказывания.

Дизъюнкция высказываний P и Q обозначается $P \vee Q$ и читается « P или Q ».

Дизъюнкция определяется таблицей:

Таблица истинности конъюнкции

P	Q	$P \wedge Q$
И	И	И
И	Л	И
Л	И	И
Л	Л	Л

В естественном языке дизъюнкция соответствует соединению высказываний союзом «или» в неразделительном смысле.

Рассмотрим теперь сложное высказывание, которое в естественном языке выражают, например, фразой «если P , то Q » (или «из P следует Q », или « P влечет Q »). Жизненный опыт подсказывает, что истинность этого высказывания должна обозначать следующее:

если P истинно, Q «обязано» быть истинным (следовать из P);

если же P ложно, то на Q нет ограничений, оно может быть как истинным, так и ложным (из ложного утверждения ничего не следует).

Значит, «из P следует Q » ложно в единственном случае: P истинно, Q ложно.

Следующие высказывания служат примерами:

1) если $0 = 0$, то $1 = 1$;

2) если $0 = 1$, то $0 = 0$;

3) если $0 = 0$, то $0 = 1$;

4) если $0 = 1$, то $1 = 2$.

Первое утверждение истинно, так как, используя равенство $0 = 0$ и другие свойства чисел, можно вывести $1 = 1$, прибавляя по единице к обеим частям равенства $0 = 0$.

Второе утверждение тоже естественно считать истинным, так как, умножая на нуль обе части равенства $0 = 1$, получим $0 = 0$.

Третье приходится считать ложным, так как исходя из истинного равенства $0 = 0$ с помощью умозаключений никогда не придем к ложному.

Четвертое естественно считать истинным. Прибавляя к обеим частям равенства $0 = 1$ по 1, получим $1 = 2$.

Используя оборот «если P , то Q » как логическую операцию, определим ее следующим образом.

Определение 95: Импликацией двух высказываний P и Q называется высказывание ложное тогда и только тогда, когда P истинно, а Q ложно.

Импликация высказываний обозначается $P \Rightarrow Q$, или $P \supset Q$ и читается: « P влечет Q », или «если P то Q », или «из P следует Q », или «пусть P , тогда Q », или « P является достаточным для Q », или « Q является необходимым для P ».

Высказывание P называется *посылкой* импликации, а Q - *заключением*.

Таблица истинности импликации и эквивалентности

P	Q	$P \Rightarrow Q$	$P \approx Q$
И	И	И	И
И	Л	Л	Л
Л	И	И	Л
Л	Л	И	И

Определение 96: Эквивалентностью двух высказываний P и Q называется высказывание, истинное тогда и только тогда, когда истинности P и Q совпадают. Эквивалентность обозначается $P \sim Q$ и читается: « P эквивалентно Q », или « P тогда и только тогда, когда Q », или « P является необходимым и достаточным для Q ».

Формулы логики высказываний

Теперь перейдем к понятию формулы логики высказываний. Среди способов задания множеств рассматривался процедурный, или рекурсивный. Формулы логики высказываний образуют перечислимое множество, которое зададим рекурсивным способом.

Определение 97: Алфавитом называется любое непустое множество. Элементы этого множества называются *символами* (буквами) данного алфавита.

Определение 98: Словом в данном алфавите называется произвольная конечная последовательность символов (возможно, пустая).

Слово a называется *подсловом* слова b , если $b = b_1 a b_2$ для некоторых слов b_1 и b_2 .

Всякое подмножество слов данного алфавита называется *языком* в этом алфавите.

Рассмотрим язык логики высказываний, т. е. подмножество слов в некотором выделенном алфавите. Слова в языке логики высказываний принято называть *формулами логики высказываний*. Эти формулы используются для моделирования высказываний.

Определение 99: Пусть известно.

1. Алфавит логики высказываний содержит следующие символы: высказывательные переменные X_1, X_2 , логические связки $\&$, $\vee$, $\neg$, $\Rightarrow$, $\sim$; символы скобок $(,)$, которые в дальнейшем будут играть разные роли.

2. Всякая высказывательная переменная есть формула, которая

называется *атомарной*.

3. Если a — формула, то $(\neg a)$ — формула. Если a и b — формулы, то $(a \& b)$, $(a \vee b)$, $(a \Rightarrow b)$, $(a \sim b)$ — формулы.

4. Других правил образования формул нет.

Замечание. Вместо высказывательных переменных $X_1, \dots, X_n, \dots$ иногда удобно пользоваться прописными латинскими буквами без индексов.

Пример 1.79

Слово $((\neg A) \& B) \Rightarrow (B \vee A)$ — формула.

Слово $(\neg A \& B) \Rightarrow B \vee A$ не является формулой (нет скобок).

Слово $((A \neg B) \Rightarrow (B \vee A))$ — не формула, так как знак $\neg$ не бинарная связка.

Замечание. Удобно при записи формул по умолчанию опускать внешние скобки и скобки при отрицании высказывательных переменных, например формула из примера 1:

$$(\neg A \& B) \Rightarrow (B \vee A).$$

Определение 100: Логические связки в логике высказываний задают алгебраические операции на множестве Φ всех высказываний.

Отрицание — унарная алгебраическая операция, остальные четыре логические связки: *конъюнкция*, *дизъюнкция*, *импликация*, *эквивалентность* — бинарные алгебраические операции.

В связи с этим логика высказываний — алгебра с пятью алгебраическими операциями ($\Phi, \&, \vee, \Rightarrow, \sim, \neg$), которая называется *алгеброй логики высказываний*.

Обсудим *синтаксис*, *семантику* и *прагматику* языка логики высказываний.

Синтаксис языка логики высказываний изучает правильность написания формул (слов языка), согласно определению формулы логики высказываний. Нетрудно предложить алгоритм отыскания ошибок в слове, например, указать на символ не из алфавита логики высказываний, на отсутствие необходимого или присутствие лишнего операнда логической операции, на отсутствие или несоответствие скобок в слове и т. д.

Семантика языка логики высказываний изучает смысл формул (слов языка) с точки зрения их оценивания на истинность. Вопросы семантики часто решаются с помощью таблиц истинности формул.

Прагматика языка логики высказываний изучает цели (задачи) использования тех или иных формул языка. Вопросы прагматики решаются в рамках функционирования (назначения) кибернетических и интеллектуальных систем, в которых используется логика высказываний.

Правила преобразования формул

Определение 101: Пусть a и b — две формулы, зависящие от одного и того же множества высказывательных переменных.

Эти формулы называются *равносильными*, если на любой оценке переменных они принимают одинаковые значения.

Равносильность формул a и b логики высказываний будем обозначать $a \equiv b$, так же как равносильность формул в элементарной алгебре, например $a^3 - b^3 \equiv (a - b)(a^2 + ab + b^2)$.

Пояснение. Оценка переменных — набор истинностных значений данных переменных.

Равносильность формул имеет свой алгебраический аналог в тождественном равенстве алгебраических выражений.

Замечание. Нужно различать символы « $\equiv$ » и « $\sim$ ».

Символ « $\sim$ » является символом логической операции формального языка (это необходимо и достаточно).

Символ « $\equiv$ » является метасимволом. Он не принадлежит алфавиту языка логики высказываний и говорит о равносильности слов с точки зрения их семантики.

Основные равносильности.

Для любых формул a, b, c справедливы следующие равносильности.

1. $a \& b = b \& a$ (коммутативность операции $\&$).
2. $a \& a = a$ (идемпотентность операции $\&$).
3. $a \& (b \& c) = (a \& b) \& c$ (ассоциативность операции $\&$).
4. $a \vee b = b \vee a$ (коммутативность операции $\vee$).
5. $a \vee a = a$ (идемпотентность операции $\vee$).
6. $a \vee (b \vee c) = (a \vee b) \vee c$ (ассоциативность операции $\vee$).
7. $a \vee (b \& c) = (a \vee b) \& (a \vee c)$ (дистрибутивность операции $\vee$ относительно операции $\&$).
8. $a \& (b \vee c) = (a \& b) \vee (a \& c)$ (дистрибутивность операции $\&$ относительно операции $\vee$).
9. $a \& (a \vee b) = a$ (первый закон поглощения).
10. $a \vee (a \& b) = a$ (второй закон поглощения).
11. $\neg \neg a = a$ (снятие двойного отрицания).
12. $\neg (a \& b) = \neg a \vee \neg b$ (первый закон де Моргана).
13. $\neg (a \vee b) = \neg a \& \neg b$ (второй закон де Моргана).
14. $a = (a \& b) \vee (a \& \neg b)$ (первая формула расщепления).
15. $a = (a \vee b) \& (a \vee \neg b)$ (вторая формула расщепления).

Любая из этих равносильностей легко может быть доказана с помощью таблиц истинности.

Еще одна группа равносильностей показывает, что одни связки могут быть выражены через другие.

16. $a \sim b \equiv (a \Rightarrow b) \& (b \Rightarrow a) \equiv (a \& b) \vee (\neg a \& \neg b)$.
17. $a \Rightarrow b \equiv \neg a \vee b \equiv \neg (a \& \neg b)$.
18. $a \vee b \equiv \neg a \Rightarrow b \equiv \neg (\neg a \& \neg b)$.
19. $a \& b \equiv \neg (a \Rightarrow \neg b) \equiv \neg (\neg a \& \neg b)$.

Рассмотрим доказательство приведенных равносильностей на примерах

тождеств 8, 10, 15. При этом в методических целях приведем три различных способа доказательства.

Пусть в тождестве 8 формулы a , b , c — атомарные, т. е. $a = A$, $b = B$, $c = C$.

Таблица истинности равносильных формул

A	B	C	$B \vee C$	$A \& (B \vee C)$	$A \& B$	$A \& C$	$(A \& B) \vee (A \& C)$
И	И	И	И	И	И	И	И
И	И	Л	И	И	Л	И	И
И	Л	И	И	И	Л	И	И
И	Л	Л	Л	Л	Л	Л	Л
Л	И	И	И	Л	Л	Л	Л
Л	И	Л	И	Л	Л	Л	Л
Л	Л	И	И	Л	Л	Л	Л
Л	Л	Л	Л	Л	Л	Л	Л

Пятый и восьмой столбцы в таблице совпадают, следовательно, тождество доказано. Этот способ доказательства равносильности назовем *табличным*.

Теперь докажем тождество 10 (второй закон поглощения) путем логических рассуждений. Этот способ называется *логическим*. Он имеет два хода рассуждения: «слева—направо» и «справа — налево».

Ход «слева — направо». Пусть в тождестве 10 слева будет *ложь*, т. е. $a \vee (a \& b) = Л$. Тогда по определению дизъюнкции будет: $a = Л$ и $(a \& b) = Л$. Тем самым получили, что справа в тождестве 10 имеем *ложь*: $a = Л$.

Ход «справа — налево». Пусть в тождестве 10 справа будет *ложь*, т. е. $a = Л$. Тогда для левой части тождества 10 имеем *ложь*. В самом деле $(a \& b) = Л$. Тем самым $a \vee (a \& b) = Л$.

Тождество 10 полностью доказано. Рассматривать отдельно случай, когда слева и справа в данном тождестве *истина* нет необходимости, так как его доказательство можно провести рассуждением от противного и использованием уже доказанного.

Например, пусть слева в тождестве — *истина*, тогда и справа будет *истина*, так как в противном случае, если справа будет *ложь*, то, по доказанному, слева будет *ложь*. А это противоречит предположению. Аналогично рассматривается случай «справа- налево».

Итак, в логическом способе доказательства равносильности достаточно рассматривать один случай — или для *истины*, или для *лжи*, но обязательно в два хода: «слева — направо» и «справа— налево».

Теперь докажем тождество 15 (вторая формула расщепления) третьим способом, который назовем *алгебраическим*.

Суть способа состоит в использовании уже доказанных равносильностей для преобразования левой и правой частей тождества к одному и тому же виду.

Рассмотрим правую часть тождества 15. Воспользуемся тождеством 7, которое предполагаем доказанным. Тогда вынесем a за скобки, будем иметь $(a \vee b) \& (a \vee \neg b) \equiv a \vee (b \& \neg b) \equiv a$, так как $(b \& \neg b) \equiv \perp$.

Тем самым правая часть тождества 15 преобразована в левую и равносильность доказана.

Рассмотрим правила, с помощью которых удобно переходить от одних равносильностей к другим.

Теорема 1.24. (правило равносильных добавлений и удалений формул).

Если a , b и c — произвольные формулы, то следующие равносильности выполняются и не выполняются одновременно:

$$\begin{aligned} a &\equiv b; \quad \neg a \equiv \neg b; \\ a \& c &\equiv b \& c; \quad c \& a \equiv c \& b; \\ a \vee c &\equiv b \vee c; \quad c \vee a \equiv c \vee b; \\ a \Rightarrow c &\equiv b \Rightarrow c; \quad c \Rightarrow a \equiv c \Rightarrow b; \\ a \sim c &\equiv b \sim c; \quad c \sim a \equiv c \sim b. \end{aligned}$$

Доказательство. Рассмотрим табличный метод перебора всех оценок формул. Надо для восьми (почему?) случаев оценок формул a , b , c

$$\begin{aligned} a = И, \quad b = И, \quad c = И; \\ a = И, \quad b = И, \quad c = Л; \end{aligned}$$

и т. д.

$$a = Л; \quad b = Л, \quad c = Л$$

убедиться в правильности каждой равносильности.

Теорема 1.25. (Правило равносильных замен переменных). Пусть $a = b$ и c — формула, в которой выделено одно вхождение переменной X . Пусть c_a получается из c заменой этого вхождения X на a , а c_b из c заменой того же вхождения X на b . Тогда $c_a \equiv c_b$.

Доказательство. Эту теорему докажем методом математической индукции по числу n логических связок в формуле c .

Для $n = 0$ формула $c = X$ (не имеет логических связок). Тогда $c_a = a$ и $c_b = b$. Следовательно, утверждение теоремы верно.

Пусть теорема верна для формулы с числом связок, не превосходящим натурального $n > 0$.

Рассмотрим случай, когда имеется $n + 1$ логическая связка. Тогда формула имеет один из пяти возможных видов $c = \neg d$, $c = f \vee g$, $c = f \& g$, $c = (f \Rightarrow g)$, $c = (f \sim g)$. Здесь d , f , g — формулы с числом логических связок не более n , для которых теорема справедлива. Заметим, что $c_a = \neg d_a$, $c_b = \neg d_b$ или $c_a = f_a \vee g_a$, $c_b = f_b \vee g_b$, или $c_a = f_a \& g_a$, $c_b = f_b \& g_b$ или и т. д. Так как для формул d , f , g теорема справедлива, имеем $d_a \equiv d_b$, $f_a \equiv f_b$, $g_a \equiv g_b$. Тогда в силу предыдущей теоремы получим то, что требовалось доказать.

Определение 102: Подформулой формулы c называется любое подслово c , само являющееся формулой.

Следующие правила приведем без доказательств [Нефедов В.Н. Курс дискретной математика. – М.: Изд-во МАИ, 1992].

Теорема 1.26. (Правило равносильных замен подформулы) Пусть c_a — формула, содержащая a в качестве своей подформулы. Пусть c_b получается из c_a заменой a в этом вхождении на b . Тогда, если $a \equiv b$, то $c_a \equiv c_b$.

Теорема 1.27. (Правило устранения импликаций и эквивалентностей) Для каждой формулы можно указать равносильную ей формулу, не содержащую логических символов $\Rightarrow$ и $\sim$.

Правило перехода к булевым функциям.

Нетрудно показать, что всякая формула логики высказываний может быть представлена булевой функцией и, наоборот, всякая булева функция соответствует некоторой формуле логики высказываний. Для этого достаточно перекодировать (переобозначить) логическое значение истинностной оценки языковых текстов следующим образом. Значение *истина* обозначить числом 1, а значение *ложь* числом 0. Далее, всякую атомарную формулу $X, Y, \dots$ логики высказываний, с помощью которой моделируем языковой текст, надо обозначить булевой переменной $x, y, \dots$. Теперь всякая оценка атомарной формулы будет соответствовать заданию значения булевой переменной. Всякая формула логики высказываний будет соответствовать своей булевой функции. При этом таблица истинности формулы будет соответствовать табличному заданию булевой функции, а аналитическое задание формулы аналитическому заданию булевой функции. Логические связи логики высказываний будут знаками алгебраических операций над булевыми переменными и константами.

Например, формула логики высказываний

$$(X \vee \bar{Y}) \Rightarrow (X \wedge \bar{Y})$$

будет соответствовать булевой функции

$$(x \vee \bar{y}) \Rightarrow (x \cdot \bar{y}) \equiv (\bar{x} \cdot y) \vee (x \cdot \bar{y}) \equiv x \oplus y.$$

Здесь X, Y обозначено через x, y , знак конъюнкции $\&$ — через $\cdot$, знак отрицания $\neg X$ — через $\bar{x}$.

Рассмотрим текстовую логическую задачу.

Нормальные формы формул логики высказываний

Определим теперь нормальные формы формул. Сначала заметим, что, в силу ассоциативности операций $\&$ и $\vee$, как бы не расставляли скобки в выражениях:

$$A_1 \& A_2 \& \dots \& A_k; \quad A_1 \vee A_2 \vee \dots \vee A_k \quad (k > 3),$$

всегда будем приходить к равносильным формулам.

Определение 103: Формулу называют *элементарной конъюнкцией*, или *конъюнктом*, если она является конъюнкцией переменных или их отрицаний.

Пример 1.79

Элементарные конъюнкции: $X_2 \& \neg X_3; \quad X_1 \& \neg X_2 \& X_4 \& X_2.$

Определение 104: Говорят, что формула находится в *дизъюнктивной нормальной форме* (ДНФ), если она является дизъюнкцией элементарных конъюнкций.

Пример 1.80

Рассмотрим формулу $(X_1 \& X_2 \& \neg X_3) \vee (X_1 \& \neg X_2 \& X_3)$. Эта формула находится в ДНФ.

Справедлива следующая теорема, которую приведем без доказательства.

Теорема 1.28. Для любой формулы a можно найти такую формулу b , находящуюся в *ДНФ*, что $a = b$. В этом случае формула b называется *дизъюнктивно нормальной формой* формулы a .

Определение 105: Пусть формула a не содержит символов $\Rightarrow, \sim$. Формула a^* называется *двойственной* формуле a , если она получена из a одновременной заменой всех символов $\&$, $\vee$ на им двойственные, т. е. $\&$ на $\vee$, $\vee$ на $\&$.

Определение 106: Говорят, что формула a находится в *конъюнктивной нормальной форме* (КНФ), если формула a^* находится в ДНФ.

Сформулируем без доказательства еще одну важную теорему.

Теорема 1.29. Для любой формулы a можно найти такую формулу b , что b находится в КНФ и $a = b$. В этом случае формула b называется *конъюнктивно нормальной формой* формулы a .

Замечание. ДНФ и КНФ формулы a , не являются однозначно определенными. Кроме ДНФ и КНФ вводятся еще другие нормальные формы, например *совершенные дизъюнктивные и конъюнктивные нормальные формы* (СДНФ и СКНФ) [4, 6, 10, 11], как и для булевых функций (см. подразд. 6.2).

Законы логики высказываний. Тавтологии

Пусть формула a зависит от множества переменных $X_1, \dots, X_n$.

Определение 107: Формула a называется *тавтологией* (или тождественно-истинной формулой), если на любых оценках списка переменных она принимает значение I , т. е. $a \equiv I$.

Здесь под *оценкой* списка переменных понимается сопоставление каждой переменной этого списка некоторого истинностного значения.

Формула a называется *выполнимой*, если на некоторой оценке списка переменных $X_1, \dots, X_n$ она принимает значение I . Формула a называется *опровержимой*, если на некоторой оценке списка переменных $X_1, \dots, X_n$ она принимает значение L .

Формула a называется *тождественно-ложной*, если на любых оценках списка переменных $X_1, \dots, X_n$ она принимает значение L , т. е. $a \equiv L$.

Рассмотрим некоторые утверждения, являющиеся очевидными следствиями данных определений:

• a — тавтология тогда и только тогда, когда, a не является опровержимой;

• a — тождественно-ложна тогда и только тогда, когда a не является выполнимой;

• a — тавтология тогда и только тогда, когда $\neg a$ тождественно-ложна;

• a — тождественно-ложна тогда и только тогда, когда $\neg a$ тавтология;

• $a \sim b$ — тавтология тогда и только тогда, когда a и b равносильны.

С точки зрения логики тавтология — не что иное, как *логические законы*, ибо при любой подстановке вместо переменных тавтологии конкретных высказываний получим истинное высказывание.

Основная проблема логики высказываний называется *проблемой разрешимости*. Она состоит в выяснении, является ли произвольная формула тавтологией. В логике высказываний всегда есть возможность (есть алгоритм) решить такую задачу. Например, решить ее можно:

использованием таблиц истинности (*табличный способ*);

приведением формулы к константе «И» с помощью правил равносильных преобразований (*алгебраический способ*);

имеются и другие способы.

Тем самым проблема разрешимости логики высказываний *алгоритмически разрешима*.

Пример 1.81

Доказать, что формула $(\neg a \Rightarrow \neg b) \Rightarrow (b \Rightarrow a)$ есть тавтология.

Используем алгебраический способ.

$$\begin{aligned} (\neg a \Rightarrow \neg b) \Rightarrow (b \Rightarrow a) &\equiv (\neg \neg a \vee \neg b) \Rightarrow (b \Rightarrow a) \equiv \\ &\equiv (a \vee \neg b) \Rightarrow (\neg b \vee a) \equiv \neg (a \vee \neg b) \vee (\neg b \vee a) \equiv \\ &\equiv (\neg a \& \neg \neg b) \vee (\neg b \vee a) \equiv (\neg a \& b) \vee \neg b \vee a \equiv \\ &\equiv ((\neg a \vee \neg b) \& (b \vee \neg b)) \vee a \equiv (\neg a \vee \neg b) \vee a \equiv \\ &\equiv \neg a \vee \neg b \vee a \equiv (\neg a \vee a) \vee \neg b \equiv I \vee \neg b \equiv I. \end{aligned}$$

Рассмотрим две теоремы в тавтологиях [4, 10].

Теорема 1.30. (О простейших логических законах). Пусть a, b, c — произвольные формулы. Тогда:

1) $a \vee \neg a$ — закон *исключенного третьего* (tertium non datur), его читают: « a или не a »;

2) $a \Rightarrow a$ (сравните с п. 1);

3) $a \Rightarrow (b \Rightarrow a)$;

4) $(a \Rightarrow b) \Rightarrow ((b \Rightarrow c) \Rightarrow (a \Rightarrow c))$ — цепное рассуждение (пусть из a следует b ; тогда, если из b , в свою очередь, следует c , то из a следует c);

5) $(a \Rightarrow (b \Rightarrow c)) \Rightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c))$ (если из a следует, что b влечет c , то из того, что a влечет b , следует, что a влечет c);

6) $(a \& b) \Rightarrow a$; $(a \& b) \Rightarrow b$;

7) $a \Rightarrow (b \Rightarrow (a \& b))$;

8) $a \Rightarrow (a \vee b)$; $b \Rightarrow (a \vee b)$;

9) $(\neg b \Rightarrow \neg a) \Rightarrow ((\neg b \Rightarrow a) \Rightarrow b)$;

10) $((a \Rightarrow b) \Rightarrow a) \Rightarrow a$ — закон Пирса.

Теорема 1.31. (О логических законах рассуждения от противного). Пусть a, b, c — произвольные формулы. Тогда тавтологиями будут:

11) $(a \Rightarrow b) \sim (\neg(a \Rightarrow b) \Rightarrow (c \& \neg c))$ (утверждение «из a следует b » эквивалентно утверждению о том, что из отрицания этого следует *ложь*);

12) $(a \Rightarrow b) \sim (\neg b \Rightarrow \neg a)$ (утверждение, что из a следует b эквивалентно утверждению, что из $\neg b$ следует $\neg a$);

13) $(a \Rightarrow b) \sim ((a \& \neg b) \Rightarrow \neg a)$ (из a следует b тогда и только тогда, когда из a и отрицания b следует отрицание a);

14) $(a \Rightarrow b) \sim ((a \& \neg b) \Rightarrow b)$ (из a следует b в том и только в том случае, когда из a и отрицания b следует b).

1.2.4. Высказывания, методы проверки логического следования

Предикаты — это отображения произвольных множеств во множество высказываний. Логика предикатов представляет собой развитие логики высказываний. Исторически понятие о предикатах явилось следствием логического анализа высказываний, т.е. выяснения их логической структуры.

Определение предиката

Определение 108: Пусть $X_1, X_2, \dots, X_n$ произвольные переменные. Эти переменные будем называть *предметными*. Пусть наборы переменных выбираются из множества X , которые будем называть *предметной областью*. *Предикатом* местности n (n - местным предикатом), определенным на предметной области X , называют отображение множества X во множество высказываний. Обозначение: $P()$ - n - местный предикат, определенный на $X := ()$.

Пример 1.82 « x простое число».

Это выражение не является высказыванием, но если в нем переменную x заменить на определенное число, то получим высказывание. Причем при замене x на число 3 получим истинное высказывание, тогда как при замене x на 8 получим ложное высказывание.

Таким образом, выражение: « x простое число» можно рассматривать как функцию $P(x)$, зависящую от переменной x . Область определения $P(x)$ — множество чисел, а область значения — высказывание.

Предикаты - отображения произвольных множеств во множество высказываний. Пусть $x_1, x_2, \dots, x_n$ - произвольные переменные. Эти переменные будем называть *предметными*. Пусть наборы переменных $x_1, x_2, \dots, x_n$ выбираются из множества X , которые будем называть *предметной областью*.

Предикатом местности n (n -местным предикатом), определенным на предметной области X , называют отображение множества X во множество

высказываний.

Обозначение: $P(x_1, x_2, \dots, x_n)$ - n -местный предикат, определенный на $X := \{x_1, x_2, \dots, x_n\}$.

Дадим другое определение предиката.

Определение 109: N -местный предикат - это связанное повествовательное предложение, содержащее n переменных и обладающее следующим свойством: при фиксации всех переменных о нем (предложении) можно сказать, истинно оно или ложно.

Примеры.

1) $P(x_1, x_2)$ "Натуральное число x_1 делится (без остатка) на натуральное число x_2 " - двуместный предикат, определенный на множестве пар натуральных чисел ($x_1, x_2 \in N$). Очевидно $P(4, 2) = 1$; $P(5, 3) = 0$

2) $P(x) = "x^2 < -1, x \in R"$ - одноместный предикат, определенный на R .

Ясно, что $P(-1) = 0$ и вообще предикат $P(x)$ - тождественно ложен, т.е. $P(x) = 0$

3) $P(x, y, z) = "x^2 + y^2 \leq z, x, y, z \in R"$ - трехместный предикат, определенный на R^3 . $P(1, 1, -2) = 0$, $P(1, 1, 2) = 1$

Предикат — это функция, значениями которой являются высказывания о n объектах, представляющих значения аргументов.

С помощью формальных теорий можно описать обширный класс высказываний, называемых предикатами. Дадим определение исчисления предикатов как формальной теории, а затем подробно остановимся на интерпретации.

Определение 110: (предиката). Функция $P(x_1, \dots, x_n)$, определенная на некотором множестве M и принимающая одно из двух значений: I (истина) или L (ложь):

$$P: M \rightarrow \{I, L\},$$

называется n -местным предикатом.

Произвольная функция $P: M^n \rightarrow B$, заданная на произвольном множестве M , называется n -местным предикатом $P(x_1, x_2, \dots, x_n)$, т.е. P задает семантическую характеристику.

Формальная теория $S = \langle A, F, P, R \rangle$ называется исчислением предикатов первого порядка, если заданы алфавит, формулы, аксиомы и правила вывода.

1. Алфавит A :

$x, y, z, \dots$ — предметные переменные, принимающие конкретные значения из некоего множества D . Тогда $x_0, y_0, z_0, \dots$ — значения предметных переменных, т.е. предметные постоянные (константы);

$p, q, r, \dots$ — переменные высказывания, принимающие два значения: 1 (истина) и 0 (ложь). Тогда $p_0, q_0, r_0, \dots$ — фиксированные значения;

P, Q, F — переменные, символизирующие само высказывание; $P_0, Q_0($) — постоянные предикаты;

$\rightarrow, \neg$ — символы логических операций; дополнительно используются символы $\wedge, \vee$;

$\forall, \exists$ — кванторы общности и существования;

служебные символы $), ($ — нужны для установления порядка выполнения связок и области действия кванторов;

можно использовать также знаки $!$ — единственность, $:$ — «такой, что...» и другие символы метаязыка. Например, $\forall x \in \{3, 4, \dots, 25\}, \exists! y \in (0, \infty): x = y^2$.

2. Формулы: F :

переменные есть формулы;

если A, B — формулы, x — переменная, то $A(x)$, $(\bar{A})$, $(A \rightarrow B)$, $\forall x A(x, \dots)$ и $\exists x A(x, \dots)$ — формулы.

3. Аксиомы:

исчисления высказываний:

$P_1: (A \rightarrow (B \rightarrow A));$

$P_2: ((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)));$

$P_3: (\bar{B} \rightarrow \bar{A}) \rightarrow ((\bar{B} \rightarrow A) \rightarrow B);$

кванторные:

$P_4: \forall x A(x \rightarrow A(y));$

$P_5: A(x) \rightarrow \exists y A(y).$

4. Правила вывода:

$R_1: \frac{A, A \rightarrow B}{B} \quad \text{modus ponens};$

$R_2: \frac{A \rightarrow B(x)}{A \rightarrow \forall x B(x)} \quad \text{введение квантора общности};$

$R_3: \frac{A(x) \rightarrow B}{\exists x A(x) \rightarrow B} \quad \text{введение квантора существования}.$

Построенная формальная теория S описывает весьма общие объекты, поэтому нужно ее интерпретировать в то, с чем можно работать.

Заметим, что предикаты дают возможность математически анализировать суждения. В классической логике *предикатом* называется *сказуемое* суждения, т.е. то, что утверждается или отрицается относительно субъекта этого суждения, имени предмета мысли, фиксирующее его определенные свойства. А в математической логике понятие предиката рассматривается как тождественное суждению, содержащему местоимения, т.е. пропозиционная функция, аргументами которой служат *имена*.

Пример 1.83

О высказывательной форме «Он получил специальность программиста» нельзя сказать, истинна она или ложна, пока не произведена замена местоимения «он» на существительное: «М. А. Иванов стал программистом» (истинно), «Дом стал программистом» (ложно).

Далее подробно опишем все составляющие формальной теории исчисления предикатов в такой интерпретации.

Чтобы задать n -местный предикат $P(x_1, x_2, \dots, x_n)$, следует указать множества $X_1, X_2, \dots, X_n$ — области изменения переменных $x_1, x_2, \dots, x_n$, причем чаще всего рассматривается случай, когда $X_1 = X_2 = \dots = X_n$.

С теоретико-множественной точки зрения предикат определяется заданием подмножества M в декартовом произведении $X_1 \times X_2 \times \dots \times X_n$.

Переменные $x_1, x_2, \dots, x_n$ называются *предметными переменными*. Элементы множеств $X_1, X_2, \dots, X_n$ называются *предметами*. Множество M — множество кортежей длины n $\langle x_1, x_2, \dots, x_n \rangle$ называется *полем предиката* $P(x_1, x_2, \dots, x_n)$.

Будем обозначать предметные переменные малыми буквами конца латинского алфавита (иногда будем снабжать эти буквы индексами) $x, y, z, \dots, x_1, x_2, \dots, x_n$.

Предметы из множеств $X_1, X_2, \dots, X_n$ — малыми буквами начала латинского алфавита $a, b, c, \dots, a_1, a_2, \dots$.

Предикаты — большими буквами латинского алфавита с приписанными предметными переменными или без них $A(x, x), B, F(x, y), P(x_1, x_2, \dots, x_n)$.

Число переменных будем указывать как верхний индекс у предиката: $P^k(x_1, x_2, \dots, x_k)$ — k -местный предикат, $Q^2(x, y)$ — двуместный предикат, $P(x)$ — одноместный предикат.

Итак, k -местный предикат — $P^k(x_1, x_2, \dots, x_k)$ есть функция, предметные переменные которой принимают значения из некоторого множества M_k , а сама она принимает только два значения: истина (**1**) или ложь (**0**), т.е.

$$P^k(x_1, x_2, \dots, x_k): M_k \rightarrow \{1, 0\}.$$

Например, если X — множество действительных чисел, то $x^2 > 1$ — одноместный предикат.

Если X, Y — множества действительных чисел, то $xy = 5$ — двуместный предикат.

Предикат называется *разрешимым*, если существуют такие кортежи, компоненты которых обращают предикат в истинное высказывание.

Если предикат при подстановке любых конкретных элементов из соответствующих множеств обращается в истинное высказывание, он называется *тождественно истинным*.

Если предикат при подстановке любых конкретных элементов из соответствующих множеств обращается в ложное высказывание, он называется *тождественно ложным*.

К предикатам, определенным на одном и том же множестве, можно применять операции алгебры высказываний: конъюнкцию, дизъюнкцию,

импликацию, эквивалентность, отрицание и получать новые предикаты.

Например, если к предикатам « $x = y$ » и « $x < y$ » — обозначим их соответственно $P(x, y)$ и $Q(x, y)$ — применить операцию конъюнкции, то получим новый предикат $P(x, y) \wedge Q(x, y)$.

Язык логики предикатов.

Символами $X, Y, Z, X_i, Y_i, \dots$ в логике предикатов принято обозначать *предметные переменные*, т.е. отдельные *предметы* — *имена*. Они могут быть простыми и сложными. Если такие предметы (имена) не содержат дополнительной информации о себе, то они называются *собственными* (простыми), например «земля», «студент» и др. Если такое имя содержит наряду с самим предметом его отдельные свойства, то оно является *сложным*, например «автор романа «Анна Каренина», «перпендикулярные прямые», «взаимно-однозначное соответствие» и др.

Символами $a, b, c, a_i, b_i, \dots$ принято обозначать *константы* или *предметные постоянные*, т.е. конкретные значения имен предметов из указанной предметной области. Высказывательные формы, входящие в предикаты, называют также *препозиционными функциями*, или *предикаторами*.

Любое непустое множество содержит два подмножества: само себя и пустое. Это свойство автоматически выделяет из области определения два случая.

Тождественно-истинным называется предикат, истинный всюду на области определения: $T(P) = D(P)$.

Тождественно-ложным называется предикат, множество истинности которого пусто: $T(P) = \emptyset$.

Два предиката в одной и той же области определения *различны* в том и только в том случае, если их множества истинности не совпадают. Это определение совпадает с отрицанием обычного определения равенства функций.

Логические операции (связки) над предикатами

Связки, аналогичные связкам булевой алгебры и исчисления высказываний, соответствуют логическим операциям над предикатами. Операции над n -местными предикатами вводятся аналогично одноместным.

Пусть, например, $P(x, \dots)$ и $Q(x, \dots)$ — предикаты, которые определены на множестве D , причем $T(P)$ и $T(Q)$ — их множества истинности.

Отрицанием предиката $P(x, \dots)$ называется предикат $\neg P(x, \dots)$, также определенный на множестве D и истинный при тех значениях переменной x , при которых $P(x, \dots)$ ложен, т.е. $T(\neg P) = D \setminus T(P)$ (1.10).

Рассмотрим примеры.

1. Для предикатов $P(x)$: « x — четное число» и $Q(x)$: « x кратно 7» конъюнкцией $P(x)$ и $Q(x)$ служит предикат « x — четное и кратно 7 число» или « x — число, кратно 14».

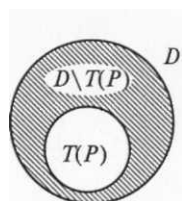


Рис. 1.10. Множество истинности предиката $P(x)$

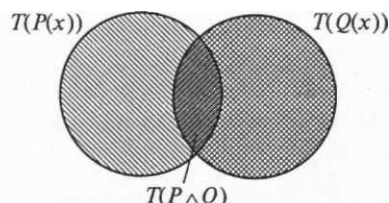


Рис. 1.11. Множество истинности конъюнкции предикатов

Пример 1.84

Предикат $P(x)$: « x — *простое число*» определен на множестве $D = \mathbb{Z}$ целых чисел, а его областью истинности являются только простые числа, т. е. числа, имеющие два делителя: x и 1 . Тогда предикат « x — *составное (целое) число*», также определенный на $\mathbb{Z}$, будет отрицанием предиката $P(x)$, т.е. $\overline{P(x)}$, а его областью истинности будет множество всех целых составных чисел (имеющих три и более делителей): $T(\overline{P(x)}) = D \setminus T(P)$.

Аналогично вводятся и остальные операции.

Конъюнкция предикатов $P(x, \dots)$ и $Q(x, \dots)$ есть новый предикат $P(x) \wedge Q(x)$, определенный на множестве D и истинный при тех значениях переменной x , при которых истинны одновременно *оба* предиката $P(x, \dots)$ и $Q(x, \dots)$, поэтому $T(P \wedge Q) = T(P) \cap T(Q)$ (рис. 1.11).

Пример 1.85

Для предикатов $P(x)$: « x — *четное число*» и $Q(x)$: « x — *кратно 7*» конъюнкцией $P(x) \wedge Q(x)$ служит предикат « x — *четное и кратно 7 число*» или « x — *число, кратно 14*»

Пример 1.86

Решить систему неравенств $\begin{cases} 2x \leq 16 \\ 3x > 15 \end{cases}$ означает: решить первое

неравенство, т.е. определить $T(P_1)$, решить второе неравенство — определить

$T(P_2)$: $\begin{cases} 2x \leq 16 \\ 3x > 15 \end{cases}, \Leftrightarrow \begin{cases} x \leq 8 \\ x > 5 \end{cases}$. Определить, при каких x верны *и* первое, *и* второе

неравенства. В данном случае система $\begin{cases} 2x \leq 16 \\ 3x > 15 \end{cases}$ означает конъюнкцию

высказываний $(x \leq 8) \wedge (x > 5) \Leftrightarrow 5 < x \leq 8$, а ответ является пересечением $T(P_1)$ и $T(P_2)$, т. е. интервалом $5 < x \leq 8$.

Обратите внимание, что в итоговый ответ вошла конъюнкция высказываний, эквивалентных данным в условии, а не самих исходных.

Дизъюнкцией предикатов $P(x, \dots)$ и $Q(x, \dots)$ называется предикат $P(x) \vee Q(x)$, определенный на множестве D и истинный при тех значениях переменной x , при которых истинен хотя бы один из предикатов $P(x)$ или $Q(x)$.

Поэтому $T(P \vee Q) = T(P) \cup T(Q)$ (рис. 1.12).

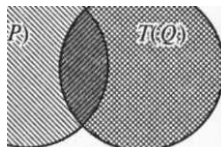


Рис. 1.12. Множество истинности дизъюнкции предикатов $T(P \vee Q)$

Пример 1.87

Для предикатов $P(x)$: « x — число, кратное 3» и $Q(x)$: « x — число, оканчивающееся на 3», определенных на N , дизъюнкцией $P(x) \vee Q(x)$ служит предикат: « x — число или кратное 3, или оканчивающееся цифрой 3».

Так, при решении уравнений (неравенств), левая часть которых есть произведение нескольких множителей, а правая — нуль, они разбиваются на совокупность уравнений (неравенств).

Пример 1.88

$x^2 - 8x - 20 = 0 \Leftrightarrow (x - 10)(x + 2) = 0 \Leftrightarrow x - 10 = 0 (P_1)$ или $x + 2 = 0 (P_2)$. Таким образом, нужно найти $T(P_1) = \{10\}$, $T(P_2) = \{-2\}$ и их объединение: $T(P) = \{-2, 10\}$.

Импликацией предиката $P(x, \dots)$ в $Q(x, \dots)$ называется предикат $P(x) \rightarrow Q(x)$, определенный на множестве D и ложный только при тех значениях переменной x , при которой предикат $P(x, \dots)$ истинен, а предикат $Q(x, \dots)$ ложен. В полном соответствии с формулой алгебры логики $a \rightarrow b = \bar{a} \vee b$ имеем: $P \rightarrow Q = \bar{P} \vee Q$ и $T(P \rightarrow Q) = (D \setminus T(P)) \cup T(Q)$ (рис. 1.13).

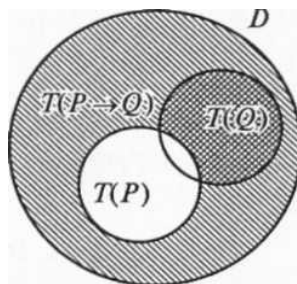


Рис. 1.13. Множество истинности импликации предикатов $T(P \rightarrow Q)$

Пример 1.89

Импликацией предикатов $P(x)$: « x — нечетное число» и $Q(x)$: « x кратно 5», определенных на $N \cup \{0\}$, служит предикат $P(x) \rightarrow Q(x)$: «Если x — нечетное число, то x кратно 5».

Здесь $T(P) = \{y | (y \bmod 2) = 1\} = \{1, 3, 5, \dots\}$;

$T(Q) = \{y | (y \bmod 5) = 0\} = \{0, 5, 10, \dots\}$.

Тогда $D/T(P) = \{y | (y \bmod 2) = 0\} = \{0, 2, 4, \dots\}$;

$T(P \rightarrow Q) = (D \setminus T(P)) \cup T(Q) = \{y | (y \bmod 2) = 0 \text{ или } (y \bmod 2) = 0\} = \{0, 2, 4, 5, 6, \dots\}$

Импликация верна, если число кратно двум или пяти.

Замечание. Поскольку в данном нами алфавите связка $\rightarrow$ является основной, а $\vee$ и $\wedge$ — дополнительными, то дадим введение конъюнкции и дизъюнкции через $\rightarrow$ и $\neg$:

$$P \vee Q \stackrel{\text{def}}{=} \overline{P} \rightarrow Q,$$

$$P \wedge Q \stackrel{\text{def}}{=} \overline{P \rightarrow Q}.$$

Эквиваленцией предикатов $P(x, \dots)$ и $Q(x, \dots)$ называется предикат $P(x) \leftrightarrow Q(x)$, определенный на множестве D и истинный при тех значениях переменной x , при которых либо оба предиката истинны, либо оба предиката ложны.

Поэтому $T(P \leftrightarrow Q) = (T(P) \cap T(Q)) \cup ((D \setminus T(P)) \cap (D \setminus T(Q)))$.

В силу законов Де Моргана $(T(P) \cap T(Q)) \cup (D \setminus (T(P) \cup T(Q))) = D \setminus (T(P) \Delta T(Q))$.

Если $T(P) = T(Q)$, то $T(P \leftrightarrow Q) = D$.

Пример 1.90

Эквивалентны предикаты $P(x)$: « x — натуральное число, кратное 3» и $Q(x)$: « x — натуральное число, сумма цифр которого делится на 3».

Кванторы

Помимо операций алгебры высказываний, в логике предикатов есть две операции, которые связаны с природой предикатов. Пусть дан предикат $P(x)$, зависящий от одной переменной и определенный на поле M .

а) Выражение $(\forall x)P(x)$ означает высказывание, истинное только в том случае, когда предикат $P(x)$ истинен для всех предметов из поля M . Выражение $(\forall x)P(x)$ читается «для всякого x , $P(x)$ », здесь символ $\forall$ — квантор общности.

б) Выражение $(\exists x)P(x)$ означает высказывание, истинное только в том случае, когда предикат $P(x)$ истинен хотя бы для одного предмета из поля M . Выражение $(\exists x)P(x)$ читается «существует x , что $P(x)$ », символ $\exists$ — квантор существования.

Рассмотрим примеры применения операций квантирования к предикатам. Пусть даны предикаты над полем натуральных чисел:

1) $x^2 = x$, тогда $(\forall x) (x^2 = x)$ — истинное высказывание;

2) $x + 2 = 7$, тогда $(\forall x) (x + 2 = 7)$ — ложное высказывание; а $(\exists x) (x + 2 = 7)$ — истинное высказывание;

3) $x + 2 = x$, тогда $(\forall x) (x + 2 = x)$ — ложное высказывание.

Название	Прочтение	Обозначение
Квантор общности	«все», «всякий», «каждый»,	$\forall$
Квантор существования	«Хотя бы один», «найдется», «существует»	$\exists$

Квантор общности — это оператор, приводящий в соответствии любому заданному предикату $y = P(x)$ такую двузначную логическую переменную z , которая принимает значение **1** тогда и только тогда, когда $y = 1$ при всех значениях x .

Квантор существования — это оператор, приводящий в соответствии любому одноместному предикату $y = P(x)$ такую двузначную логическую переменную z , которая принимает значение **0** тогда и только тогда, когда $y = 0$ при всех значениях x .

Рассмотрим некоторые общие свойства введенных операторов. В соответствии с определениями кванторов логическая переменная z в выражениях

$$z = (\forall x)P(x)$$

$$z = (\exists x)P(x)$$

уже не является функцией предметной переменной x .

Для того чтобы отметить отсутствие функциональной зависимости z от x , предметную переменную x в таких случаях называют *связанной*. Несвязанные переменные называют *свободными*.

Например, в предикате

$$(\forall x) A(x, y) \vee (\exists z) B(z, v)$$

переменные x и z — связанные, а y и v — свободные.

Если квантор общности или квантор существования применяется не к одноместному предикату, а к какому-нибудь k -местному предикату, то в результате этого получается снова предикат, но за счет связывания одной предметной переменной получаемый предикат будет $(k-1)$ -местным.

Кванторы.

Для количественных характеристик обычно используют понятия «все», «некоторые», «существуют» и др., которые называют *кванторами* (от лат. *quantum* — сколько). Мы часто пользовались символами $\forall$ и $\exists$, заменяющими слова «любой» и «существует». Покажем действие этих кванторов в высказывательных формах. Часть формулы, на которую распространяется действие квантора, называется *областью действия* этого квантора. Вхождение переменной в формулу может быть *связанным*, если переменная расположена либо непосредственно после знака квантора, либо в области действий квантора, после которого стоит переменная. Все прочие вхождения — свободные. Например, в выражении $\forall x P(x)$ переменная x связывает свойство предиката и квантор общности. Грубо говоря, от этой

переменной, ее конкретного вида и имени, ничего не зависит, т.е. $\forall xP(x)$ и $\forall yP(y)$ суть одно и то же. Так, можно произвольно называть индекс суммирования в рядах и переменную интегрирования в определенных интегралах. В частности, в определении множества как совокупности *всех* объектов, удовлетворяющих характеристическому свойству, использовалась запись $G = \{x \mid P(x)\}$. Очевидно, что в предикате со связанной переменной ее так же легко можно заменить на любую другую. При этом множество все равно будет совокупностью тех же элементов, удовлетворяющих свойству P . Переменная, не являющаяся связанной, называется *свободной*, если после подстановки вместо нее имени некоторых конкретных объектов предикат превращается в осмысленное предложение.

Между кванторами $\forall$ и $\exists$ и логическими операциями существует тесная связь. Пусть предикат $P(x)$ определен на конечном множестве $D = \{a_1, a_2, \dots, a_n\}$. Тогда высказывание $\forall x \in D(P(x))$ будет истинным только в том случае, если истинны одновременно все высказывания $P(a_1), P(a_2), \dots, P(a_n)$, т.е. если истинна их конъюнкция: $P(a_1) \wedge P(a_2) \wedge \dots \wedge P(a_n)$. Аналогично высказывание $\exists x P(x)$ означает, что оно истинно, когда истинно хотя бы одно из высказываний $P(a_1), P(a_2), \dots, P(a_n)$. Тогда должна быть истинной дизъюнкция высказываний $P(a_1) \vee P(a_2) \vee \dots \vee P(a_n)$. Поэтому для конечной области определения выполняются равносильности: $(\forall x \in D)P(x) \Leftrightarrow P(a_1) \wedge P(a_2) \wedge \dots \wedge P(a_n)$ и $(\exists x \in D)P(x) \Leftrightarrow P(a_1) \vee P(a_2) \vee \dots \vee P(a_n)$.

Таким образом, кванторы общности и существования являются дополнениями и аналогами соответственно логических операций конъюнкции и дизъюнкции.

Поскольку конъюнкцию можно выразить через отрицание и дизъюнкцию, то, вообще говоря, символ $\exists$ можно было не включать в число основных символов, так как квантор существования $\exists x A(x)$ по сути является сокращенной записью формулы $\overline{\forall x \overline{A(x)}}$, выражающей так называемую *двойственность*.

Пример 1.91

Записать с помощью формул логики предикатов следующее утверждение: «Для лечения любого известного компьютерного вируса имеются программы. Существуют новые (неизвестные) компьютерные вирусы, для лечения которых программы еще не разработаны»

Решение.

Введем обозначения элементарных формул:

$A(x)$ – известен компьютерный вирус x ;

$B(x)$ – для лечения вируса x существует программа;

Тогда с помощью логических связей и кванторов получим формулы:

$\overline{B}(x)$ - против вируса x нет программы;

$\forall x(A(x))$ - любой вирус известен;

$\exists x(\overline{A}(x))$ - существуют новые (неизвестные) вирусы;

$\forall x(A(x) \rightarrow B(x))$ - если вирус давно известен, то имеется программа для его лечения;

$\exists x(\bar{A}(x) \wedge \bar{B}(x))$ - существуют (появились) новые вирусы, для лечения которых программы еще не разработаны.

Тогда формализованное исходное утверждение примет вид:

$$(\forall x(A(x) \rightarrow B(x))) \wedge (\exists x(\bar{A}(x) \wedge \bar{B}(x)))$$

Отношение следования и равносильности между высказывательными формами связаны с тождественно-истинными импликацией и эквиваленцией, следовательно, их можно записать с помощью кванторов общности:

$$Q_1(x) \Rightarrow Q_2(x) \text{ тождественно } \forall x(Q_1(x) \rightarrow Q_2(x))$$

$$Q_1(x) \Rightarrow Q_2(x) \text{ тождественно } \forall x(Q_1(x) \equiv Q_2(x))$$

Пример 1.92

Запись $x^2 - 5x = 0 \Leftrightarrow x(x - 5) = 0$ является не формулой, а истинным высказыванием о равносильности двух формул, представленных в виде уравнений. В то же время справедлива запись $\forall x \in R (x^2 - 5x = 0) \equiv (x(x - 5) = 0)$, выражающая истинное высказывание, которое включает операцию эквиваленции в качестве составляющей.

Поэтому логическое следование можно определить через импликацию, а равносильность через эквиваленцию. Так, для эквиваленции справедливо: «Две высказывательные формы Q_1 и Q_2 истинны или ложны ($Q_1 \Leftrightarrow Q_2$) одновременно с высказыванием $\forall x(Q_1(x) \equiv Q_2(x))$ », что и было ранее введено.

Существует различие в употреблении знаков и « $\Rightarrow$ », « $\rightarrow$ », « $\Leftrightarrow$ » и « $\leftrightarrow$ ».

Знаки « $\rightarrow$ », « $\leftrightarrow$ » обозначают логические операции импликации и равносильности и входят составной частью в формулы.

Знаки « $\Rightarrow$ » и « $\Leftrightarrow$ » обозначают определенные отношения между высказывательными формами, не входя в них в качестве составной части.

Квантификация многоместных высказывательных форм

Пусть $Q(x_1, x_2, \dots, x_n)$ — n -местная высказывательная форма. Ее замену на высказывательную форму $\forall x_i Q(x_1, x_2, \dots, x_n)$ либо на $\exists x_i Q(x_1, x_2, \dots, x_n)$ называют *квантификацией* высказывательной формы $Q(x_1, x_2, \dots, x_n)$ по переменной x_i .

В процессе такой квантификации эта i -я переменная связывается одним из кванторов, а n -местная высказывательная форма превращается в $(n-1)$ -местную.

Это аналогично тому, что если функцию $f(x_1, x_2, \dots, x_n)$ проинтегрировать от a до b по переменной x_i , то полученный результат будет функцией от $n-1$ переменной и не будет зависеть от x_i : $I(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = \int_a^b f(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_n) dx_i$. Так, интеграл от функции одной ($n = 1$)

переменной является константой и вообще ни от чего не зависит.

Пусть дана двухместная высказывательная форма $x - y < 0$, определенная на $R \times (0, \infty)$. Произведем квантификацию по переменной y («навесим» квантор общности). Получим одноместную высказывательную форму $\forall y(x - y < 0)$ со свободной переменной x . Если для фиксированного $x = x_0$ будет выполнено $\forall y(x_0 - y < 0)$, то эта высказывательная форма превращается в истинное высказывание, например, при $x = -2$, а при $x = 3$ — в ложное.

Если в одноместной высказывательной форме связать квантором и вторую переменную x , то можно получить высказывание: либо $\exists x \forall y(x - y < 0)$ — истинное высказывание; $\forall x \forall y(x - y < 0)$ — ложное высказывание.

При «навешивании» кванторов на двухместную высказывательную форму $Q(x, y)$ можно получить одну из восьми комбинаций:

- 1) $\forall x \forall y Q(x, y)$ — «для любого x и любого y $Q(x, y)$ »;
- 2) $\forall y \forall x Q(x, y)$ — «для любого y и любого x $Q(x, y)$ »;
- 3) $\exists x \exists y Q(x, y)$ — «существует x и существует y , такие, что $Q(x, y)$ »;
- 4) $\exists y \exists x Q(x, y)$ — «существует y и существует x , такие, что $Q(x, y)$ »;
- 5) $\exists x \forall y Q(x, y)$ — «существует x , такой, что для любого y $Q(x, y)$ »;
- 6) $\forall x \exists y Q(x, y)$ — «для всякого x существует y , такой, что $Q(x, y)$ »;
- 7) $\exists y \forall x Q(x, y)$ — «существует y , такой, что для любого x $Q(x, y)$ »;
- 8) $\forall y \exists x Q(x, y)$ — «для всякого y существует x , такой, что $Q(x, y)$ ».

Очевидно, что первое и второе высказывания, а также третье и четвертое тождественны между собой, их значения истинности совпадают. Между остальными полученными высказываниями нельзя установить тождественности: если истинно высказывание 5, то истинным будет и высказывание 8, причем обратное неверно. Аналогично, если истинно высказывание 7, то истинно и высказывание 6, но не наоборот. То есть, если кванторы одноименны (1 — 4), то их порядок не играет роли и полученные высказывания эквивалентны. Если кванторы разноименные (5 — 8), то их порядок в полученном высказывании принципиально важен.

Например, для двухместного предиката «Город x находится в стране y » высказывание $\forall y \exists x P(x, y)$ имеет вид 0-местного предиката и читается «В каждой стране y есть некоторый город x ». Оно будет истинным, в то время как высказывание $\exists x \forall y P(x, y)$ читается «Существует город x , находящийся во всех странах y » будет ложным.

Пусть даны x, y — две различные переменные, $F(x)$, $\Phi(x)$ и $Q(x, y)$ — любые формулы логики предикатов и M — формула, не содержащая свободных вхождений x . Тогда справедливы равносильности, представленные с учетом двойственности кванторов $\exists$ и $\forall$ в табл. 5.5.

Таблица 5.5

Следствия и равносильности логики предикатов

Равносильности для $\exists$	Правила	Равносильности для $\forall$
$\exists x \exists y Q(x, y) \Leftrightarrow \exists y \exists x Q(x, y)$	Правила перестановки и кванторов	$\forall x \forall y Q(x, y) \Leftrightarrow \forall y \forall x Q(x, y)$
$\exists x \exists y Q(x, y) \Rightarrow \exists y \exists x Q(x, y)$		
$\bar{\exists} x F(x) \Leftrightarrow \forall x \bar{F}(x)$	Перенос отрицания с квантора на предикат	$\bar{\forall} x F(x) \Leftrightarrow \exists x \bar{F}(x)$
$\bar{\exists} x \bar{F}(x) \Leftrightarrow \forall x F(x)$		$\bar{\forall} x \bar{F}(x) \Leftrightarrow \exists x F(x)$
$\exists x (F(x) \vee \Phi(x)) \Leftrightarrow \exists x F(x) \vee \exists x \Phi(x)$		$\forall x (F(x) \wedge \Phi(x)) \Leftrightarrow \forall x F(x) \wedge \forall x \Phi(x)$
$\exists x (F(x) \wedge \Phi(x)) \Rightarrow \exists x F(x) \wedge \exists x \Phi(x)$		$\forall x (F(x) \vee \Phi(x)) \Rightarrow \forall x F(x) \vee \forall x \Phi(x)$
$\exists x (M \wedge F(x)) \Rightarrow M \wedge \exists x F(x)$		$\forall x (M \wedge F(x)) \Rightarrow M \wedge \forall x F(x)$
$\exists x (M \vee F(x)) \Rightarrow M \vee \exists x F(x)$		$\forall x (M \vee F(x)) \Rightarrow M \vee \forall x F(x)$

Применение предикатов в алгебре

Рассмотрим предикаты, в которых свободной является лишь одна переменная, которую обозначим через x , и обсудим применение предикатов в алгебре.

Типичным примером является уравнение, например, $x^2 - 3x + 2 = 0$. Свободная переменная может принимать здесь любое числовое значение. Для некоторых чисел x (а именно $x = 1$, $x = 2$) утверждение, содержащееся в этом уравнении, истинно, в остальных оно ложно. В подобных случаях, когда истинность или ложность предиката зависит только от значения, принимаемого свободной переменной x , множество допустимых значений x можно рассматривать как *множество логических возможностей* U , а множество всех значений этой переменной, при которых высказывание истинно — как его *множество истинности*.

В приведенном выше примере множество U состоит из всех действительных чисел, а множеством истинности является множество $\{1, 2\}$.

В результате введения понятия множества истинности для предикатов мы сможем сказать, что решить уравнение — значит найти один элемент или все элементы его множества истинности. При решении системы двух уравнений у нас имеется предикат, представляющий конъюнкцию двух уравнений. Поэтому мы ищем пересечение двух множеств истинности. Если это пересечение пусто, то система уравнений не имеет решений. Такие уравнения называются *несовместными*, поскольку их множества истинности не имеют общих элементов x .

Понятие множества истинности удобно не только в вопросах, связанных с решением уравнений, но и при рассмотрении неравенств.

Если U — множество действительных чисел, то множество истинности неравенства $x < 0$ состоит из всех отрицательных действительных чисел. Множество же истинности неравенства $x > -3$ состоит из всех действительных чисел, больших, чем -3 . Если мы потребуем, чтобы эти неравенства выполнялись одновременно, то множеством истинности будет множество,

являющееся пересечением двух исходных множеств, т.е. все действительные числа между -3 и 0 .

Понятие множества истинности предиката позволяет выяснить, чем разнятся между собой уравнения и тождества. Когда мы решаем уравнение, мы тем самым ищем один из элементов множества истинности этого уравнения или все его элементы. Если же мы доказываем тождество, то тем самым утверждаем, что оно справедливо для всех x . Таким образом, тождество представляет собой уравнение, множеством истинности которого является универсальное множество U , т. е. является логически истинным или тождественно истинным.

Предикаты P и Q , определенные на X , называются *равносильными*, если $P(x_1, x_2, \dots, x_n) \equiv Q(x_1, x_2, \dots, x_n)$ для любого набора $(x_1, x_2, \dots, x_n)$ предикатных переменных на X

Пусть P - предикат, определенный на X . *Отрицанием* предиката P называется предикат, обозначаемый определенным $\neg P$ (neP) на X следующим образом:

$$\neg P(x_1, x_2, \dots, x_n) = P(x_1, x_2, \dots, x_n)$$

Пример 1.93

$\neg P(x_1, x_2) = P(x_1, x_2) =$ "Натуральное число x_1 делится (без остатка) на натуральное число x_2 ". $\neg P(4, 2) = 0$, $\neg P(5, 3) = 1$,

Пусть P и Q предикаты, определенные на X .

Дизъюнкцией (конъюнкцией, импликацией, эквиваленцией) предикатов P и Q называется предикат, определенный на X обозначаемый $P \vee Q$, $P \wedge Q$, $P \rightarrow Q$, $P \sqcup Q$, и определяемый следующим образом:

$$P \vee Q(x_1, x_2, \dots, x_n) \equiv P(x_1, x_2, \dots, x_n) \vee Q(x_1, x_2, \dots, x_n)$$

$$P \wedge Q(x_1, x_2, \dots, x_n) \equiv P(x_1, x_2, \dots, x_n) \wedge Q(x_1, x_2, \dots, x_n)$$

$$P \rightarrow Q(x_1, x_2, \dots, x_n) \equiv P(x_1, x_2, \dots, x_n) \rightarrow Q(x_1, x_2, \dots, x_n)$$

$$P \sqcup Q(x_1, x_2, \dots, x_n) \equiv P(x_1, x_2, \dots, x_n) \sqcup Q(x_1, x_2, \dots, x_n)$$

Булева алгебра предикатов

Так как к предикатам можно применять логические операции, то для них справедливы основные законы булевой алгебры.

Теорема 1.32. (Свойства логических операций для предикатов).

Множество n -местных предикатов, определенных на X , образуют булеву алгебру предикатов, т.е. для них справедливы основные равносильности булевой алгебры.

1. $\overline{\overline{P}} = P$ - закон двойного отрицания
2. $P \vee Q = Q \vee P$ - коммутативность дизъюнкции;
3. $P \wedge Q = Q \wedge P$ - коммутативность конъюнкции;
4. $P \vee (Q \vee R) = (P \vee Q) \vee R$ - ассоциативность дизъюнкции;
5. $P \wedge (Q \wedge R) = (P \wedge Q) \wedge R$ - ассоциативность конъюнкции;

6. $P \vee (Q \wedge R) = (P \vee Q) \wedge (P \vee R)$ - дистрибутивность дизъюнкции относительно конъюнкции;
7. $P \wedge (Q \vee R) = (P \wedge Q) \vee (P \wedge R)$ - дистрибутивность конъюнкции относительно дизъюнкции;
8. $\overline{P \vee Q} = \overline{P} \wedge \overline{Q}$; $\overline{P \wedge Q} = \overline{P} \vee \overline{Q}$ - законы де Моргана;
9. $P \vee P = P$; $P \wedge P = P$ - идемпотентность;
10. $P \vee 1 = 1$; $P \wedge 0 = 0$; $P \vee 1 = 1$; - законы единицы и нуля
 $P \wedge P = P$ $P \wedge P = P$ $P \wedge P = P$ - идемпотентность;
11. $P \vee (P \wedge Q) = P$; - закон поглощения
12. $P \wedge (P \vee Q) = P$; - закон поглощения
13. $\overline{P} \vee P = 1$ - закон исключенного третьего
14. $\overline{P} \wedge P = 0$ - закон противоречия

Формулы логики предикатов

Наряду с *определенными предикатами* — для которых истинность или ложность известны для каждого набора значений свободных предметных переменных, будем рассматривать переменные предикаты, для которых не определены значения. Будем обозначать переменные предикаты большими буквами из конца латинского алфавита с приписанными предметными переменными или без них:

$W(x_1, x_2, \dots, x_n); U(x, y), \dots$

Применяя к переменным предикатам операции $\wedge$; $\vee$; $\rightarrow$; $\leftrightarrow$; $\neg$; $\forall$; $\exists$, получим формулы логики предикатов,

Формулой логики предикатов называется выражение, составленное из переменных предикатов с помощью логических операций и кванторов и обращающееся в конкретный предикат при подстановке вместо переменных конкретных предикатов.

Пример.

$((\forall x) W(x, y) \vee B) \rightarrow U(z)$ — формула логики предикатов. Формула логики предикатов называется *тавтологией*, если при подстановке любых конкретных предикатов она всегда обращается в тождественно истинный предикат.

Формулы

Определение формулы лежит в основе так называемой *логики предикатов первого порядка*, в которой разрешается квантифицировать (связывать кванторами) только предметные переменные. Логика предикатов первого порядка включает в себя все формулы логики высказываний, все равносильности исчисления высказываний, а также большинство правил вывода умозаключений из классической логики. Поэтому язык логики предикатов дает возможность анализировать рассуждения естественного языка и науки, делать выводы в различных формальных системах.

Так, высказывательная форма $\overline{p \vee Q(x)} \rightarrow q$ является формулой. В то же

время высказывательная форма $\overline{\exists xP(x) \rightarrow Q(x)}$ не будет формулой, поскольку в формуле $\exists xP(x)$ переменная x связана квантором существования, тогда как в формуле $Q(x)$ эта же переменная свободна.

В чем ценность формальных теорий?

Для описания каких объектов используется логика предикатов?

Вообще говоря, ценность любой формальной теории заключается в возможности описывать с ее помощью произвольные объекты и связи между ними.

Теоремы.

К числу основных равносильностей логики предикатов относят:

1. $\overline{\forall xP(x)} \Leftrightarrow \exists x\overline{P(x)}$.
2. $\overline{\exists xP(x)} \Leftrightarrow \forall x\overline{P(x)}$.
3. $\forall xP(x) \Leftrightarrow \overline{\exists x\overline{P(x)}}$.
4. $\exists xP(x) \Leftrightarrow \overline{\forall x\overline{P(x)}}$.
5. $\forall x(P(x) \wedge Q(x)) \Leftrightarrow \forall xP(x) \wedge \forall xQ(x)$.
6. $\exists x(P(x) \vee Q(x)) \Leftrightarrow \exists xP(x) \vee \exists xQ(x)$.

Сформулируем следующие правила.

(1) Формула логики предикатов называется *атомарной*, т.е. *элементарной*, если в ней нет связанных переменных.

(2) Пусть F — формула, тогда $\text{не}F$ — тоже формула. Свободные и связанные переменные формулы $\text{не}F$ — это соответственно свободные и связанные переменные формулы F .

(3) Пусть F и G — формулы, причем в них нет предметных переменных, которые были бы связаны в одной формуле и свободны в другой.

Тогда $F \wedge G$, $F \vee G$, $F \rightarrow G$, $F \leftrightarrow G$ — формулы, в которых свободные переменные формул F и G остаются свободными, а связанные — связанными.

(4) Пусть F — формула, содержащая свободную переменную x . Тогда $(\forall x)F$, $(\exists x)F$ — тоже формулы, в которых переменная x связана, а остальные свободные переменные, входящие в F , остаются свободными.

Заметим, что по определению формулы никакая переменная не может быть одновременно свободной и связанной.

Значение формулы определено лишь тогда, когда задана какая-то *интерпретация* входящих в нее символов.

Под *интерпретацией* понимают систему $M = \langle M, f \rangle$, состоящую из непустого множества M и соответствия f , которое сопоставляет каждой формуле определенный предикат. При заданной интерпретации предметные переменные пробегают множество M , а логические символы и символы кванторов имеют свой обычный смысл.

Равносильные формулы логики предикатов

Пусть формулы F и G имеют одно и то же множество свободных

переменных (в частности, пустое). Формулы F и G *равносильны в данной интерпретации*, если они принимают одинаковые значения на любом наборе свободных переменных, т. е. выражают в данной интерпретации один и тот же предикат.

Формулы F и G *равносильны на множестве M* , если они принимают одинаковые значения во всех интерпретациях заданных на множестве M .

Формулы F и G *равносильны* в логике предикатов, если они равносильны на всех множествах ($F = G$).

Рассмотрим правила перехода от одних формул к другим, им равносильным.

(1) *Перенос квантора через отрицание*. Пусть $W(x)$ — формула, содержащая свободную переменную x . Тогда справедливы равносильности:

$$\overline{(\forall x)W(x)} \equiv (\exists x)\overline{W(x)}, \quad \overline{(\exists x)W(x)} \equiv (\forall x)\overline{W(x)},$$

$$(\exists x)\overline{W(x)} \equiv (\forall x)\overline{W(x)}, \quad (\forall x)\overline{W(x)} \equiv (\exists x)\overline{W(x)}.$$

(2) *Вынос квантора за скобки*. Пусть формула $W(x)$ содержит свободную переменную x , а формула B не содержит переменной x . Формулы $W(x)$ и B удовлетворяют третьему правилу создания формул. Тогда справедливы равносильности:

$$(\exists x)(W(x) \wedge B) \equiv (\exists x)W(x) \wedge B, \quad (\forall x)(W(x) \wedge B) \equiv (\forall x)W(x) \wedge B,$$

$$(\exists x)(W(x) \vee B) \equiv (\exists x)W(x) \vee B, \quad (\forall x)(W(x) \vee B) \equiv (\forall x)W(x) \vee B.$$

(3) *Перестановка одноименных кванторов*. Имеем

$$(\exists x)(\exists y)W(x, y) \equiv (\exists y)(\exists x)W(x, y), \quad (\forall x)(\forall y)W(x, y) \equiv (\forall y)(\forall x)W(x, y).$$

(4) *Переименование связанных переменных*. Заменяя связанную переменную формулы W другой переменной, не входящей в эту формулу, в кванторе и всюду в области действия квантора, получим формулу, равносильную W .

Приведенные и нормальные формы в логике предикатов

Рассмотрим способ упрощения формул, опирающийся на приведенные равносильности.

Формулы, в которых из логических символов имеются только символы конъюнкция, дизъюнкция и отрицание, причем символ отрицания встречается над символами предикатов, будем называть *приведенными*.

Пример 1.94

Формула $(\forall x_1) A_1^{(1)}(x_2) \vee (\exists x_1) \text{не}(A_2^{(2)}(x_2, x_3))$ — приведенная;

Формула $\text{не}(\forall x_2) A_1^{(1)}(x_2) \rightarrow A_2^{(1)}(x_1)$ — неприведенная.

Для любой формулы существует равносильная ей приведенная формула, причем множества свободных и связанных переменных этих формул совпадают.

Такая приведенная формула называется *приведенной формой* данной формулы.

В логике высказываний мы ввели две нормальные формы — дизъюнктивную нормальную форму и конъюнктивную нормальную форму.

В логике предикатов также имеется *нормальная форма*, цель которой — упрощение процедуры доказательств.

Приведенная формула называется *нормальной*, если она не содержит символов кванторов или все символы кванторов стоят впереди (т.е. логические символы и символы предикатов стоят в области действия каждого квантора).

Для любой приведенной формулы существует равносильная ей нормальная формула той же длины (под длиной формулы будем понимать общее число входящих в нее символов предикатов, логических символов и символов кванторов).

Нормальная формула называется *нормальной формой* данной формулы.

Приведем несколько формул, находящихся в нормальной форме:

$$(\forall x)(\forall y)(P(x, y) \wedge Q(y)), (\forall x)(\forall y)(\overline{P(x, y)} \rightarrow Q(y)),$$

$$(\forall x)(\forall y)(\exists z)(Q(x, y) \rightarrow P(z)).$$

Алгоритм преобразования формул в нормальную форму

1. Исключить логические связки $\leftrightarrow$ и $\rightarrow$ с помощью формул

$$F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F), \quad F \rightarrow G = \neg F \vee G.$$

2. Использовать закон *двойного отрицания* $\neg \neg F = F$, законы де Моргана:

$$\neg(F \vee G) = \neg F \wedge \neg G, \quad \neg(F \wedge G) = \neg F \vee \neg G,$$

законы

$$\overline{(\forall x)F(x)} \equiv (\exists x)(\overline{F(x)}), \quad \overline{(\exists x)F(x)} \equiv (\forall x)(\overline{F(x)}),$$

чтобы пронести знак отрицания внутрь формулы.

3. Переименовать связанные переменные, если это необходимо.

4. Использовать равносильные формулы логики предикатов, чтобы вынести кванторы в самое начало формулы для приведения ее к нормальной форме. Например, приведем формулу $(\forall x)P(x) \rightarrow (\exists x)Q(x)$ к нормальной форме:

$$(\forall x)P(x) \rightarrow (\exists x)Q(x) = \overline{((\forall x)P(x))} \vee (\exists x)Q(x) =$$

$$= (\exists x)\overline{P(x)} \vee (\exists x)Q(x) = (\exists x)(\overline{P(x)} \vee Q(x))$$

Следовательно, нормальная форма формулы $(\forall x)P(x) \rightarrow (\exists x)Q(x)$ — это $(\exists x)(\overline{P(x)} \vee Q(x))$.

Исчисление предикатов

Исчисление предикатов называют еще *теорией первого порядка*.

В исчислении предикатов, так же как и в исчислении высказываний, на первом по важности месте стоит проблема *разрешимости*.

Но в исчислении высказываний проблема разрешимости состояла в решении вопроса является ли данная сложная функция тождественно истинной, выполнимой или тождественно ложной.

Теперь же вопрос следует поставить иначе. Принимает ли данная функция значение 1 при:

а) любых предметных переменных и любых предикатах,

б) на некотором множестве предметных переменных и любых

предикатах,

в) при некоторых значениях предметных переменных и любых предикатах,

г) является ли она тождественно ложной, т.е. невыполнимой?

Таким образом, в логике предикатов, в отличие от логики высказываний, нет эффективного способа для распознавания общезначимости функций.

Поэтому в исчислении предикатов указывается некоторая совокупность формул, которые называются *аксиомами* и составляют *аксиоматическую теорию*, и указывается конечное множество отношений между формулами, составляющее *правила вывода*.

Аксиоматическая теория и правила вывода и составляют *исчисления предикатов*.

Символами исчисления предикатов или алфавитом исчисления предикатов являются символы предметных переменных, символы предикатов, логические символы (отрицание и импликация), символы кванторов, а также скобки и запятая.

Сформулируем аксиомы исчисления предикатов и правила вывода исчисления предикатов.

Аксиомы исчисления предиката.

Пусть A , B и C - любые формулы.

Аксиома 1. $A \rightarrow (B \rightarrow C)$.

Аксиома 2. $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) (A \rightarrow C))$.

Аксиома 3. $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$.

Аксиома 4. $(\forall x_i) A(x_i) \rightarrow A(x_j)$, где формула $A(x_i)$ не содержит переменной x_j .

Аксиома 5. $A(x_i) \rightarrow (\exists x_j) A(x_j)$, где формула $A(x_i)$ не содержит переменной x_j .

Правила вывода исчисления предикатов.

(1) Пусть $(A(x) \rightarrow B)$ и B не содержит переменной x , тогда $((\exists x)A(x) \rightarrow B)$

Это правило связывания квантором существования.

(2) Пусть $B \rightarrow A(x)$ и B не содержит переменной x , тогда $(B \rightarrow ((\forall x)A(x)))$

Это правило связывания квантором общности.

(3) Связанную переменную формулы B можно заменить другой переменной, не являющейся свободной в B . Это правило переименования связанной переменной.

Следование и эквиваленция

Высказывательная форма Q_2 следует из высказывательной формы Q_1 , если импликация $Q_1 \rightarrow Q_2$ обращается в истинное высказывание при любых наборах значений переменных, входящих в нее. Для операции логического следования принято обозначение $Q_1 \Rightarrow Q_2$.

Пусть даны предикаты $Q_1(x_1, x_2, \dots, x_n)$ и $Q_2(x_1, x_2, \dots, x_n)$, а их

множества истинности соответственно $T(Q_1)$ и $T(Q_2)$. Поскольку $Q_1 \Rightarrow Q_2$, то если $(x_1, x_2, \dots, x_n) \in T(Q_1)$, т.е. Q_1 истинна, то должна быть истинна Q_2 , т.е. $(x_1, x_2, \dots, x_n) \in T(Q_2)$. Поскольку такое свойство должно быть у любого элемента из $T(Q_1)$, то это определение подмножества. Итак, $T(Q_1) \subset T(Q_2)$.

Пусть даны два предиката, определенные на одном множестве. Высказывательные формы Q_1 и Q_2 назовем *равносильными*, если при любом наборе значений переменных, входящих в них, высказывательные формы принимают одинаковые значения истинности: $Q_1 \Rightarrow Q_2$. Очевидно, что если $Q_1 \Rightarrow Q_2$, а $Q_2 \Rightarrow Q_1$ то $Q_1 \Leftrightarrow Q_2$. Тогда $T(Q_1) = T(Q_2)$. т.е. множества истинности равносильных предикатов также совпадают.

Пример 1.95

Пусть высказывательные формы заданы на множестве действительных чисел R .

$$\frac{x^2 - 5x + 6}{x - 2} = 0 \quad \text{и} \quad x^2 - 5x + 6 = 0 \quad \text{не являются равносильными.}$$

$$\frac{3x + 8}{x^2 + 1} = 0 \quad \text{и} \quad 3x + 8 = 0 \quad \text{являются равносильными.}$$

$$\ln(x - 1) + \ln(x + 1) = 2 \quad \text{и} \quad \ln(x^2 - 1) = 2 \quad \text{не являются равносильными.}$$

$$\sqrt{x+3} = x-1 \quad \text{и} \quad x+3 = (x-1)^2 \quad \text{не являются равносильными.}$$

$$\frac{4-8x}{2+x} \geq 0 \quad \text{и} \quad (4-8x)(2+x) > 0 \quad \text{не являются равносильными.}$$

$$\frac{4-8x}{2+x} > 0 \quad \text{и} \quad (4-8x)(2+x) > 0 \quad \text{являются равносильными.}$$

$$x^2 - x^4 \geq 0 \quad \text{и} \quad 1 - x^2 \geq 0 \quad \text{не являются равносильными.}$$

В математике нарушение цепочки тождественных преобразований при решении уравнений или неравенств влечет за собой потерю имеющих или приобретение посторонних корней, т.е. изменение множества истинности исследуемого предиката.

Можно доказать, что *отношение равносильности* высказывательных форм обладает известными свойствами, а именно, оно *рефлексивно* и *симметрично*. В том случае, когда одинаковые переменные в каждой из исследуемых форм принимают значения из одного множества, отношение равносильности будет обладать также и свойством *транзитивности*.

Тогда назовем *равносильным преобразованием* высказывательной формы Q_1 ее замену на равносильную форму Q_2 . Две равносильные

высказывательные формы с одинаковым набором переменных, для которых установлен одинаковый порядок, определяют один и тот же *предикат*.

Эти свойства предиката используются при решении уравнений и неравенств, которые тоже являются некоторыми высказывательными формами. Так, решение любого уравнения или неравенства предусматривает установление множества его истинности, т.е. множества истинности соответствующего ему предиката. В процессе поиска множества истинности производят замену одного предиката другим, равносильным данному, с целью упрощения имеющихся высказывательных форм.

Пример 1.96

$2x - 13 + x^2 - (6x^2 - 4x + 5 - 6x^2) = 0 \Leftrightarrow 6x = 18 \Leftrightarrow x = 3$, т. е. множество истинности каждого из этих уравнений состоит из одного числа 3.

Рассмотрим примеры, для которых областью определения является множество действительных чисел: $D = E$.

Пример 1.97

Для двух высказывательных форм — уравнений $(x - 2)(x - 3) = 0$ (Q_1) и $x - 3 = 0$ (Q_2) — из $x - 3 = 0$ следует, что $(x - 2)(x - 3) = 0$, т.е. верна запись $Q_2 \Rightarrow Q_1$. Однако из $(x - 2)(x - 3) = 0$ не следует $x - 3 = 0$. Например, $x = 2$ является корнем первого уравнения, но не второго.

Пример 1.98

Из уравнения $(x - 5)(x - 2) = 0$ следует неравенство $x > 0$, так как корни уравнения — числа 2 и 5 — удовлетворяют также и неравенству.

Пример 1.99

Тождественно-истинное высказывание $x^2 + 5 > 0$ может следовать из любой высказывательной формы Q , имеющей непустое множество истинности $T(Q) \subset R$, т.е. форма $Q \rightarrow (x^2 + 5 > 0)$ истинна при любых значениях x .

Отношения *следования* и *равносильности* для высказывательных форм, вообще говоря, зависят от того множества, на котором оно рассматривается.

Пример 1.100

Высказывательная форма $x > 9$ следует из неравенства $8 < x < 12$, если $D = \{2, 0, 4, 5, 7, 9, 10, 11, 13\}$, но не следует, если $D(Q) = N$. Действительно, при $D = \{2, 0, 4, 5, 7, 9, 10, 11\}$ $T(Q_1) = \{9, 10, 11\}$, а $T(Q_2) = \{9, 10, 11, 13\}$ и выполняется $T(Q_1) \subset T(Q_2)$, т.е. форма $Q_1 \rightarrow Q_2$ истинна.

Во втором случае ($D(Q) = N$), $T(Q_1) = \{8, 9, 10, 11\}$, а $T(Q_2) = \{9, 10, 11, 12, 13, 14 \dots\}$, но отношение $T(Q_1)$ с $T(Q_2)$ не выполняется, поскольку $8 \notin T(Q_2)$.

Правила вывода исчисления предикатов:

Правило заключения (modus ponens) — правило, аналогичное тому, которое введено в исчислении высказываний.

Правило обобщения ($\forall$ -введения, иг-правило) R_2 : $\frac{F \rightarrow G(x)}{F \rightarrow \forall x G(x)}$, где $G(x)$ содержит свободные вхождения x , тогда как F не содержит свободных x .

Правило $\exists$ -введения (eg-правило) R_3 : $\frac{G(x) \rightarrow F}{\exists x G(x) \rightarrow F}$

Нарушение этих требований может привести к ложным выводам, полученным из истинных высказываний.

Пример 1.101

Даны предикаты $P(x)$: «натуральное x делится на 15», $Q(x)$: « x делится на 5». Высказывание $P(x) \rightarrow Q(x)$ истинно для любых $x \in N$. Применим для него правило обобщения. Имеем $P(x) \rightarrow \forall x Q(x)$: «Если x делится на 15, то каждое число x делится на 5». Получили ложное утверждение, так как правило $\forall$ -введения применимо к 0-местным, а не к одноместным, как $P(x)$, предикатам.

Можно доказанные теоремы делать новыми правилами вывода. Так, помимо правил $\forall$ - и $\exists$ -введения можно ввести правила удаления кванторов.

Пусть выведена или дана формула $\exists x F(x)$, например «Существуют студенты, работающие по специальности». Из предметного множества всех студентов выберем такого, о котором действительно известно, что он работает по специальности, и для него введем константу a . Поэтому $\exists x F(x) \rightarrow F(a)$. Это так называемое правило $\exists$ -удаления, или *es-правило* (правило выбора).

Правило $\forall$ -удаления снимает квантор общности, осуществляя переход от $\forall x F(x)$ к произвольным формулам $F(a)$, $F(y)$ и др. с учетом того, что эти переменные свободны от x в Fx .

Пример 1.102

Из высказывания «Каждый студент колледжа владеет компьютером» будет следовать, что конкретный студент Максимов тоже владеет компьютером, и произвольно выбранный некоторый студент y владеет компьютером, и всякий студент z тоже владеет компьютером. При этом необходимо помнить, что предметные переменные y и z не должны быть связанными.

Правило $\forall$ -удаления называют *правилом универсальной конкретизации*, или *us-правилом*.

Пример 1.103

1. «Все металлы (М) — плавятся (П). Цинк (Ц) — металл. Значит, цинк плавится». Формализация в логике предикатов примет вид: $\forall x (M(x) \rightarrow P(x)) \wedge \forall x (C(x) \rightarrow M(x)) \vdash \forall x (C(x) \rightarrow P(x))$. Снятие квантора общности:

$(M(x) \rightarrow P(x)) \wedge (C(x) \rightarrow P(x))$; тогда на основании транзитивности импликации имеем $(C(x) \rightarrow M(x)), (M(x) \rightarrow P(x)) \vdash C(x) \rightarrow P(x)$.

Вывод $\forall x(C(x) \rightarrow P(x))$ — обобщение по R_2 — верен.

2. «Все студенты (C) проходят практику (P). Некоторые студенты работают в фирме (Φ), значит, некоторые работающие в фирме — проходят практику». Формализация примет вид: $\forall x(C(x) \rightarrow P(x)) \wedge \exists x(C(x) \wedge \Phi(x)) \vdash \exists x(\Phi(x) \wedge P(x))$. Уберем кванторы по правилам us и es . Имеем $(C(a) \rightarrow P(a)) \wedge (C(a) \wedge \Phi(a)) \Leftrightarrow (C(a) \rightarrow P(a)) \wedge C(a) \wedge \Phi(a) \Leftrightarrow P(a) \wedge \Phi(a)$.

Вывод: $\exists x(P(x) \wedge \Phi(x))$, т.е. существуют студенты, которые проходят практику в фирме.

Свойства отношения классификации

Рассмотрим непустое множество U . Пусть дана одноместная высказывательная форма Φ с переменной, которая принимает значения из U , проявляя свойство некоторых объектов из него и соответствуя некоторому предикату Q . Множество истинности $T(Q)$ таких объектов является подмножеством U как универсального множества.

Пример 1.104

Пусть дано $U_x = \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14 \dots\}$.

Высказывательной форме « $5 < x < 12$ » соответствует подмножество

$T_1(Q) = \{5, 6, 7, 8, 9, 10, 11\}$ ($T(Q_1) \subset U_1$).

Из множества $U_2 = \{1, 3, 5, 7, 9, 11, 13, 15\}$ та же высказывательная форма выделяет множество истинности $T_2(Q) = \{5, 7, 9, 11\}$,

из $U_3 = \{5, 6, 7, 8, 9, 10, 11\}$ - $T_3(Q) = U_3$,

из $U_4 = \{12, 13, 14, 15\}$ рассматриваемая высказывательная форма выделяет пустое подмножество истинности $T_4(Q) = \emptyset$.

Эта высказывательная форма выражает на множестве U единственное свойство, характерное для рассматриваемого предиката на заданном множестве U , т. е. одноместный предикат Q (в данном случае « $5 < x < 12$ ») задает свойство данного множества. Тогда множество элементов, обладающих таким свойством Q , будем называть *объемом* этого свойства.

Если на множестве U задан предикат, выражающий некоторое свойство P , то множество U можно разбить на два подмножества $T(P)$ и $U \setminus T(P)$. Такое разбиение на непересекающиеся подмножества мы называем *классификацией* множества U по основанию P .

Пример 1.105

Так, в предыдущем примере $T(P) = \{5, 6, 7, 8, 9, 10, 11\}$ множество истинности предиката P : « $5 < x < 12$ » из множества $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$. $U \setminus T(P) = \{1, 2, 3, 4, 12, 13, 14, 15\}$.

Пусть на множестве U задано еще одно свойство Q . Тогда все

множество U , разбиваясь на четыре подмножества, представляет новую классификацию. С помощью логических операций такую классификацию записывают в виде: $P(x) \cdot Q(x) \vee P(x) \cdot \overline{Q}(x) \vee \overline{P}(x) \cdot Q(x) \vee \overline{P}(x) \cdot \overline{Q}(x)$.

Замечание. Это аналогично разложению булевой функции по двум переменным (см. подразд. 4.7), с той разницей, что каждое слагаемое должно иметь вид $P(x) \cdot Q(x) \cdot 1$, так как мы разлагаем формулу исчисления предикатов, имеющую вид тавтологии.

Пример 1.106

Так, в предыдущем примере $T(P) = \{5, 6, 7, 8, 9, 10, 11\}$ множество истинности предиката P : « $5 < x < 12$ » из множества $U = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$. $U \setminus T(P) = \{1, 2, 3, 4, 12, 13, 14, 15\}$.

Пусть в нашем примере предикат Q выражает новое свойство — «*быть нечетным числом*». Тогда эти два свойства одновременно классифицируют множество U на подмножества:

$$T(P) \cap T(Q) = \{5, 7, 9, 11\}, \text{ выполнено } P(x) \cdot Q(x);$$

$$T(P) \cap \overline{T(Q)} = \{6, 8, 10\}, \text{ выполнено } P(x) \cdot \overline{Q}(x);$$

$$\overline{T(P)} \cap T(Q) = \{1, 3, 13, 15\}, \text{ выполнено } \overline{P}(x) \cdot Q(x),$$

$$\overline{T(P)} \cap \overline{T(Q)} = \{2, 4, 12, 14\}, \text{ выполнено } \overline{P}(x) \cdot \overline{Q}(x).$$

Уточним понятие «отношение» с помощью понятия «предикат». Во всех n -местных предикатах ($n > 2$) устанавливаются некоторые *отношения* между переменными.

Пример 1.107

1. Высказывательная форма « x — *друг* y » выделяет из всего множества людей пары x и y , которые связаны между собой отношением дружбы.

2. Высказывательная форма « $x \perp y$ » выделяет из множества пар прямых, например на плоскости, те пары, которые связаны отношением перпендикулярности.

3. Высказывательная форма « $x^2 + y^2 + z^2 = 16$ » выделяет из всего множества троек координат те, которые связаны отношением «*точка с координатами $(x; y; z)$ лежит на сфере с центром в начале координат и радиусом $R = 4$* ».

Отрицания в исчислении предикатов

В разговорной речи для построения отрицания обычно перед сказуемым ставят частицу *не*.

Пример 1.108

1а «Студент x учится на факультете программирования» имеет отрицание

1б «Студент x не учится на факультете программирования».

? Но всегда ли построенное таким образом отрицание истинно?

Утверждения 2а «Все выпускники колледжей продолжили образование в вузе» и 2б «Все выпускники колледжей не продолжили образование в вузе» не являются отрицанием друг друга, так как они оба ложны.

Пары утверждений 3а «Некоторые выпускники колледжей продолжили образование в вузе» и 3б «Некоторые выпускники колледжей не продолжили образование в вузе» тоже не служат отрицанием друг друга, так как они оба истинны.

Вторая и третья пары утверждений отличаются от первых тем, что содержат кванторные слова «все» и «некоторые». А при построении отрицаний для предложений, содержащих кванторы, прием введения отрицания *не* перед сказуемым не срабатывает.

Можно воспользоваться другим, универсальным, приемом построения отрицаний предложений, содержащих кванторы, добавив общее отрицание *неверно, что...* Тогда во втором примере «Неверно, что все выпускники колледжей продолжили образование в вузе» совпадает по смыслу с утверждением «Некоторые выпускники колледжей не продолжили образование в вузе». Таким образом, отрицанием предложения 2а служит 3б, а отрицанием 3а служит 2б.

Символически общее отрицание принято записывать с помощью либо общей черты, либо отрицания самого квантора.

Для отрицания предложения $\forall x(\Phi(x))$ возможны записи $\bar{\forall}x(\Phi(x))$, или $\overline{\forall x(\Phi(x))}$, или $(\exists x(\bar{\Phi}(x)))$: $\bar{\forall}x(\Phi(x)) \Leftrightarrow \exists x(\bar{\Phi}(x))$;

для отрицания $\exists x(\Phi(x))$ аналогично: $\bar{\exists}x(\Phi(x))$, или $\overline{\exists x(\Phi(x))}$, или $\exists x(\bar{\Phi}(x))$: $\bar{\exists}x(\Phi(x)) \Leftrightarrow \exists x(\bar{\Phi}(x))$.

Эти равносильности являются обоснованием метода построения отрицаний высказываний, содержащих кванторы. Для построения *отрицания* высказываний, содержащих квантор $(\forall | \exists)$, достаточно заменить его на другой квантор $(\exists | \forall)$ и взять отрицание выражения, на которое этот квантор был «навешен».

Для многоместных кванторов также применяется это правило: осуществляется последовательный перенос отрицания с кванторного слова на предложение, стоящее за квантором, а сам квантор заменяют на двойственный.

Например, для формулы $\forall x \forall y \forall z (R(x, y, z))$ построим отрицание: $\bar{\forall}x \forall y \exists z (R(x, y, z)) \Leftrightarrow \exists x (\bar{\forall}y \exists z (R(x, y, z))) \Leftrightarrow \exists x \exists y (\bar{\exists}z (R(x, y, z))) \Leftrightarrow \exists x \exists y \forall z (\bar{R}(x, y, z))$. В подразд. 4.8 была показана булева двойственность конъюнкции и дизъюнкции. Поэтому для сложных высказываний, состоящих из простых, разделенных операциями конъюнкции и дизъюнкции, *отрицание* строится следующим образом: нужно все кванторы $(\exists)$ заменить на $(\forall)$, и наоборот; все связи *и* $(\wedge)$ заменить на *или* $(\vee)$, и наоборот; и взять отрицание утверждения.

1.3 Теория графов

1.3.1 Основные понятия теории графов, типы графов, обходы графов, способы задания

Граф $G=(V, E)$ состоит из двух множеств: конечного множества элементов, называемых вершинами, и конечного множества элементов, называемых ребрами. Каждое ребро определяется парой вершин. Если ребра графа определяются упорядоченными парами вершин, то G называется направленным или ориентированным графом. В противном случае G называется ненаправленным или неориентированным графом. Для обозначения вершин графа будем использовать символы $v_1, v_2, v_3, \dots$, а для обозначения ребер - $e_1, e_2, e_3, \dots$. Вершины v_i и v_j , определяющие ребро e_i , называются концевыми вершинами ребра e_i . В этом случае ребро e_i обозначается как $e_i=(v_i, v_j)$. Заметим, что в множестве E допускается более чем одно ребро с одинаковыми концевыми вершинами. Все ребра с одинаковыми концевыми вершинами называются параллельными. Кроме того, концевые вершины ребра не обязательно различны. Если $e_i=(v_i, v_i)$, то ребро e_i называется петлей. Граф называется простым, если он не содержит петель и параллельных ребер. Граф G является графом порядка n , если множество его вершин состоит из n элементов.

Граф, не имеющий ребер, называется пустым. Граф, не имеющий вершин (и, следовательно, ребер), называется нуль-графом.

Графически граф может быть представлен диаграммой, в которой вершина изображена точкой или кружком, а ребро — отрезком линии, соединяющим точки или кружки, соответствующие концевым вершинам ребра. Например, если $V=\{v_1, v_2, v_3, v_4, v_5, v_6\}$ и $E=\{e_1, e_2, e_3, e_4, e_5\}$, такие, что $e_1=(v_1, v_2)$, $e_2=(v_1, v_4)$, $e_3=(v_5, v_6)$, $e_4=(v_1, v_2)$, $e_5=(v_5, v_5)$, тогда граф $G=(V, E)$ представляется так, как изображено на рис. 1.14. В этом графе e_1 и e_4 - параллельные ребра, e_5 - петля. Говорят, что ребро инцидентно своим концевым вершинам. Две вершины смежны, если они являются концевыми вершинами некоторого ребра. Если два ребра имеют общую концевую вершину, они называются смежными.

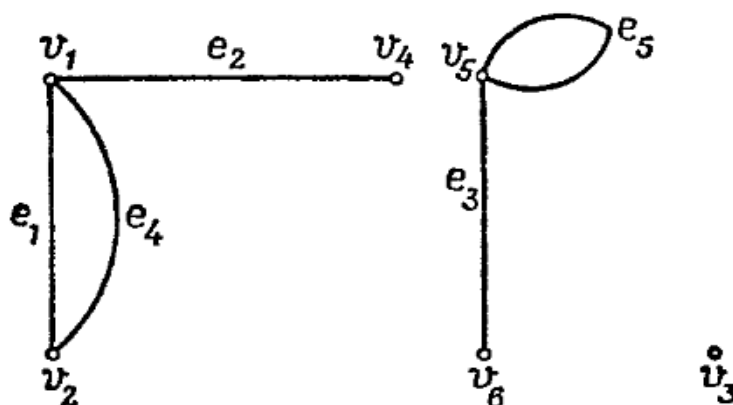


Рис. 1.14. Диаграмма

Например, в графе на рис. 1.14 ребро e_1 инцидентно вершинам v_1 и v_2 ; v_1 и v_4 являются смежными вершинами, а e_1 и e_2 - смежными ребрами.

Число инцидентных вершине v_i ребер называется степенью вершины и обозначается $d(v_i)$. Иногда степень вершины называется также ее валентностью. Вершина степени 1 называется висячей вершиной. Единственное ребро, инцидентное висячей вершине, называется висячим. Вершина степени 0 называется изолированной. По определению петля при вершине v_i добавляет 2 в степень соответствующей вершины. Величины $\delta(G)$ и $\Delta(G)$ обозначают минимальную и максимальную степени вершины в G соответственно.

В графе G на рис. 1.14 $d(v_1) = 3$, $d(v_2) = 2$, $d(v_3) = 0$, $d(v_4) = 1$, $d(v_5) = 3$, $d(v_6) = 1$.

Заметим, что v_3 - изолированная вершина, v_4 и v_6 - висячие вершины, e_2 — висячее ребро. Легко проверить, что сумма степеней вершин в данном графе G равна 10, тогда как число ребер равно 5. Таким образом, сумма степеней вершин графа G равна удвоенному числу ребер графа G и, следовательно, является четным числом. Более того, можно показать, что число вершин графа G нечетной степени также четно. Эти результаты свойственны не только графу на рис. 1.14.

Способы задания графов

Первое и на наш взгляд самое простое задание графа - это представление его с помощью картинки в соответствии с геометрическим определением графа. При этом, в соответствии с договоренностью выше, вершинам конкретного представления графа будут приписаны номера.

Так на рис.1.15 даны два представления одного и того же графа.

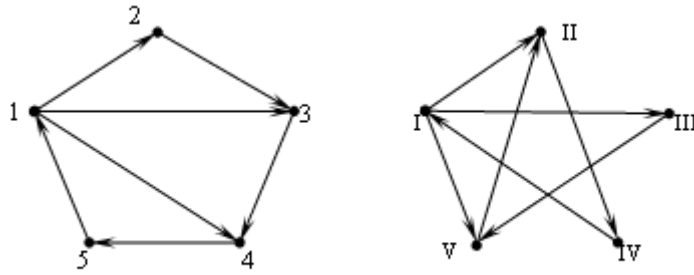


Рис. 1.15. Представления графов

Другое задание графа - списком. Можно считать, что в соответствии с теоретико-множественным определением графа все элементы множества $R \subseteq V \times V$, входящего в определение, т.е. упорядоченные пары, упорядочены сначала по первым элементам пар, а затем по вторым, в соответствии с нумерацией вершин (нумерацией элементов множества V). Тогда два представления графа с рис.1.15 будут заданы двумя списками (рис. 1.16):

1	2, 3, 4	I	II, III, V
2	3	II	IV
3	4	III	V
4	5	IV	I
5	1	V	II

Рис. 1.16. Представление графов списками

В первом столбце - первые элементы пар, затем по строкам, списком через запятую, идут вторые элементы.

Третье задание графа - матрицами. Ниже выписаны две матрицы - A и B , задающие два представления графа с рис. 1.15:

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Большинство задач информатики удобно решать при использовании матричного задания графов. Квадратная таблица $R = \|r_{i,j}\|_{n \times n}$ называется матрицей смежности, если ее элементы образуются по правилу:

$$r_{i,j} = \begin{cases} 1, & \text{если вершина } x_i \text{ смежна с вершиной } x_j \\ 0, & \text{в противном случае} \end{cases}$$

Представления графа в соответствии с различными определениями будем называть различными видами представлений. Между различными

видами представлений графа существует взаимнооднозначное соответствие. Действительно, поскольку речь идет о представлении графа, то множество вершин можно считать пронумерованным. Тогда дуге (ребро будем рассматривать как пару противоположно направленных дуг), идущей из вершины i в вершину j будет соответствовать упорядоченная пара (i, j) или, что то же самое, в списке вершины i будет присутствовать вершина j , а в матрице $A = (a_{ij})$, представляющей граф, элемент $a_{ij} = 1$. Отсутствию дуги, идущей из вершины i в вершину j , будет соответствовать отсутствие вершины j в списке вершины i , а $a_{ij} = 0$.

В силу указанного выше взаимнооднозначного соответствия между различными видами представлений мы и можем воспользоваться различными определениями одного и того же понятия - граф. При этом при изучении различных свойств графа мы стараемся каждый раз пользоваться тем языком, который наиболее удобен для описания выбранного свойства. Вероятно, не следует выбирать один из языков в качестве единственного языка теории графов. Иногда для описания того или иного свойства, атрибута графа, требуется конкретный язык. Так если мы говорим о плоском (планарном) графе, то нам по необходимости приходится использовать геометрический язык теории графов. Если же мы говорим о "спектре" графа, то мы формулируем это понятие на матричном языке.

Теория графов замечательна тем, что трудные задачи (проблемы) в ней появляются и легко формулируются сразу же после формулировки основных понятий теории графов. Так после формулировки понятия представления (представителя) графа сразу же появляется задача - если нам даны два представления, то это представления одного и того же графа или разных? Т.е. мы сразу же вышли на так называемую проблему изоморфизма графов.

Типы графов

Граф $G = (V, E)$ называют *полным*, если для любой пары вершин v_i и v_j в V существует ребро (v_i, v_j) в неориентированном графе $\bar{G} = (V, \bar{E})$ т. е. для каждой пары вершин графа G должна существовать, по крайней мере, одна дуга, соединяющая их (рис. 1.17,а).

Граф $G = (V, E)$ называется *симметрическим*, если в множестве дуг E для любой дуги (v_i, v_j) существует также противоположно ориентированная дуга (v_j, v_i) (рис. 1.17,б).

Антисимметрическим называется такой граф, для которого справедливо следующее условие: если дуга $(v_i, v_j) \in A$, то во множестве A нет противоположно ориентированной дуги, т. е. $(v_j, v_i) \notin A$ (рис. 1.17,в). Очевидно, что в антисимметрическом графе нет петель.

В качестве примера можно рассмотреть граф, являющийся моделью некоторой группы людей: вершины графа интерпретируют людей, а дуги — их взаимоотношения. Так, если в графе дуга, нарисованная от вершины v_i к вершине v_j , означает, что v_i является другом или родственником v_j , тогда данный граф должен быть симметрическим. Если дуга, направленная от v_i к v_j ,

означает, что вершина v_j подчинена вершине v_i , то такой граф должен быть антисимметрическим.

Комбинируя определения *полного* и *симметрического* графов и *полного* и *антисимметрического* графов, получили следующие определения:

- граф $G = (V, E)$, в котором любая пара вершин (v_i, v_j) соединена двунаправленными дугами, называется *полным симметрическим* (1.17,г);
- граф $G = (V, E)$, имеющий для каждой пары вершин (v_i, v_j) только одну дугу, называется *полным антисимметрическим* или *турниром*.

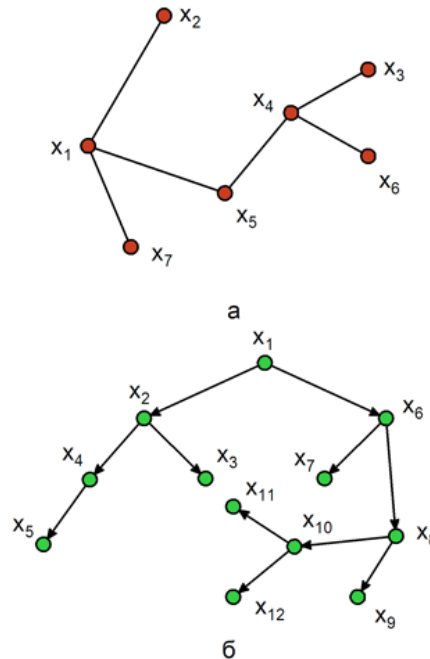


Рис. 1.17. Виды графов

Связный граф, не имеющий циклов, либо граф, в котором каждая пара вершин соединена одной и только одной простой цепью, называется *деревом* (рис 1.17). Граф типа “дерево”: а – *неориентированное дерево*, б – *ориентированное дерево*.

Ориентированное дерево представляет собой ориентированный граф без циклов, в котором полустепень захода каждой вершины, за исключением одной (например, вершины v_1), равна 1, а полустепень захода вершины v_1 (называют корнем этого дерева) равна 0 (рис 1.17,б)

Граф $G = (V, E)$, который может быть изображен на плоскости или сфере без пересечений называется *планарным* (рис 1.18).

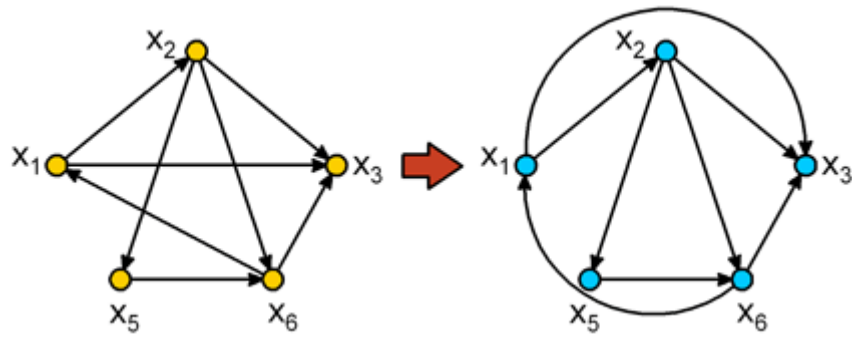


Рис. 1.18. Планарные графы

На рис. 1.19 показаны *непланарные* графы. Эти два графа играют важную роль в теории планарных графов и известны как графы Куратовского.

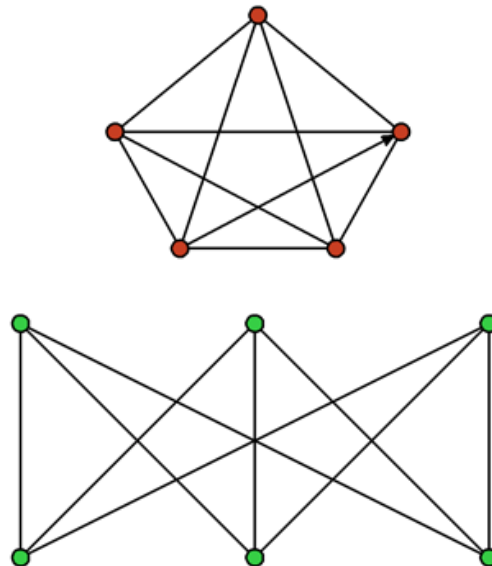


Рис. 1.19. Непланарные графы

1.3.2 Ориентированные ациклические графы и деревья Ориентированные ациклические графы

В этом разделе изучаются свойства важного класса ориентированных графов, а именно ациклических. Как мы знаем, ориентированный граф — ациклический, если он не содержит контуров. Очевидно, простейшим примером ациклического ориентированного графа является ориентированное дерево.

Основной результат, который мы получим в этом разделе, заключается в том, что вершины ациклического ориентированного графа G на n вершинах можно пометить таким образом целыми числами из множества $\{1, 2, \dots, n\}$, что если в графе G имеется дуга (i, j) , то $i < j$.

Определенное нами упорядочение вершин называется топологической сортировкой. Топологически отсортированы, например, вершины ациклического ориентированного графа на рисунке 1.20.

Справедливость основного результата этого раздела определяется следующей теоремой:

Теорема 1.32. В ациклическом ориентированном графе имеются по крайней мере одна вершина с нулевой полустепенью захода и одна вершина с нулевой полустепенью исхода.

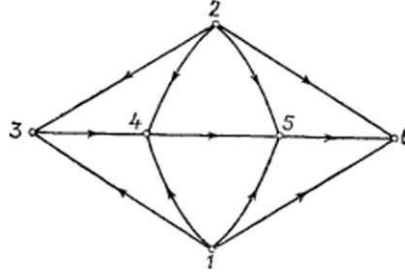


Рис. 1.20. Ациклический ориентированный граф

Доказательство. Пусть $P: v_1, v_2, \dots, v_p$ — максимальный ориентированный путь графа G . Покажем, что полустепень захода v_1 и полустепень исхода v_p равны нулю.

Если полустепень захода не равна нулю, то существует такая вершина w , что в графе G имеется дуга (w, v_1) . Возможны два случая.

Случай 1. Пусть $w \neq v_i, 1 \leq i \leq p$. Тогда существует ориентированный путь $P': w, v_1, v_2, \dots, v_p$, содержащий все дуги пути P . Это противоречит максимальной упомянутого пути.

Случай 2. Пусть для некоторого i $w = v_i$. Тогда в графе G имеется контур $C: v_1, v_2, \dots, v_i, v_1$. Это противоречит условиям теоремы.

Таким образом, не существует такой вершины w , что (w, v_1) — дуга графа G .

Другими словами, полустепень захода v_1 равна нулю. Рассуждая аналогичным образом, можно показать, что равна нулю полустепень исхода v_p . ■

Для осуществления топологической сортировки вершин ациклического ориентированного графа G на n вершинах сделаем следующее.

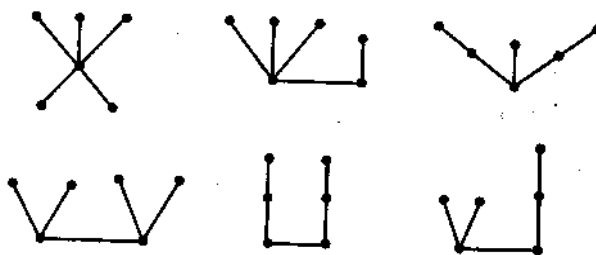
Выберем произвольную вершину с нулевой полустепенью исхода. Это возможно, поскольку граф G — ациклический, а по теореме 1 в нем должна иметься хотя бы одна такая вершина. Пометим выбранную вершину целым n . Теперь удалим из графа G эту вершину и инцидентные ему дуги. Обозначим получившийся граф через G' .

Поскольку граф G' также ациклический, можно выбрать в нем вершину с нулевой полустепенью исхода. Пометим эту вершину целым числом $n-1$. Повторим описанную процедуру до тех пор, пока не пометим все вершины. Легко видеть, что эта процедура порождает топологическую сортировку вершин графа G .

Деревья

Важным частным случаем ориентированных ациклических графов являются деревья. Класс деревьев занимает в теории графов особое положение. С одной стороны, это достаточно просто устроенные графы, и многие задачи, весьма сложные в общей ситуации, для деревьев решаются легко. С другой стороны, деревья часто встречаются в областях, на первый взгляд не имеющих отношения к теории графов.

Деревом называется связный граф, не содержащий циклов. Любой граф без циклов называется ациклическим (или лесом). Таким образом, компонентами леса являются деревья. На рисунке изображены все деревья шестого порядка.



В следующей теореме перечислены некоторые простые свойства деревьев.

Теорема 1.33. Пусть граф T имеет n вершин. Тогда следующие утверждения эквивалентны: (i) T является деревом; (ii) T не содержит циклов и имеет $n - 1$ ребер; (iii) T связен и имеет $n - 1$ ребер; (iv) T связен и каждое его ребро является мостом; (v) любые две вершины графа T соединены ровно одной простой цепью; (vi) T не содержит циклов, но, добавляя к нему любое новое ребро, мы получаем ровно один цикл.

Доказательство. Если $n = 1$, утверждение очевидно. Предположим поэтому, что $n > 2$.

(i) $\Rightarrow$ (ii). По определению T не содержит циклов; тогда удаление любого ребра разбивает T на два графа, каждый из которых является деревом. По индуктивному предположению число ребер в каждом из этих деревьев на единицу меньше числа вершин. Отсюда выводим, что полное число ребер графа T равно $n - 1$.

(ii) $\Rightarrow$ (iii). Если граф T несвязен, то каждая его компонента представляет собой связный граф без циклов. Из предыдущей части доказательства следует, что число вершин в каждой из компонент больше числа ее ребер на единицу. Значит, полное число вершин графа T больше полного числа его ребер по крайней мере на 2, а это противоречит тому, что T имеет $n - 1$ ребер.

(iii) $\Rightarrow$ (iv). Удаление любого ребра приводит к графу с n вершинами и $n - 2$ ребрами, который не может быть связным.

(iv) $\Rightarrow$ (v). Так как T связен, то каждая пара его вершин соединена по крайней мере одной простой цепью. Если же данная пара вершин соединена

двумя простыми цепями, то они замыкаются в цикл, а это противоречит тому, что каждое ребро в T является мостом.

(v) $\Rightarrow$ (vi). Если T содержит цикл, то любые две вершины этого цикла соединены по крайней мере двумя простыми цепями. Добавим теперь к графу T какое-то ребро e . Тогда мы получим цикл, поскольку инцидентные ребру e вершины уже соединены в T простой цепью.

(vi) $\Rightarrow$ (i). Предположим, что T несвязен; тогда добавление любого ребра, соединяющего вершину одной компоненты с вершиной другой компоненты, не приводит к образованию цикла. ■

Следствие. Пусть G — лес с n вершинами и k компонентами; тогда G имеет $n - k$ ребер.

Доказательство. Применим к каждой компоненте графа G предложение (ii) теоремы 1. ■

Заметим, что по лемме о рукопожатиях сумма степеней всех n вершин дерева равна удвоенному числу его ребер ($2n - 2$); отсюда следует, что при $n > 2$ дерево, имеющее n вершин, всегда содержит не менее двух висячих вершин. Известно, что в связном графе G удаление одного ребра, принадлежащего некоторому выбранному циклу, не нарушает связности оставшегося графа. Применим эту процедуру к одному из оставшихся циклов, и так до тех пор, пока не останется ни одного цикла. В результате получим дерево, связывающее все вершины графа G ; оно называется остовным деревом (или остовом, каркасом) графа G . Пример графа и одного из его остовных деревьев дан на рисунке 1.21.



Рис. 1.21. Пример графа и одного из его остовных деревьев

В общем случае обозначим через G произвольный граф с n вершинами, m ребрами и k компонентами. Применяя описанную выше процедуру к каждой компоненте G , получим в результате граф, называемый остовным лесом. Число удаленных в этой процедуре ребер называется циклическим рангом (или цикломатическим числом) графа G и обозначается через $\gamma(G)$. Очевидно, что $\gamma(G) = m - n + k$ и является неотрицательным целым числом.

Таким образом, циклический ранг дает меру связности графа: циклический ранг дерева равен нулю, а циклический ранг циклического графа равен единице, удобно также определить коциклический ранг (или ранг разреза) графа G как число ребер в его остовном лесе; коциклический ранг обозначается через $\chi(G)$ и равен $n - k$.

Докажем два простых результата, касающихся остовных лесов. В этой теореме дополнением остовного леса T некоторого (необязательно простого) графа G является граф, полученный из G удалением ребер T .

Теорема 1.34. Если T — остовный лес графа G , то (i) всякий разрез в G имеет общее ребро с T ; (ii) каждый цикл в G имеет общее ребро с дополнением T .

Доказательство. (i) Пусть C^* — разрез графа G , удаление которого разбивает одну из компонент G на два подграфа H и K . Поскольку T — остовный лес, в нем должно содержаться ребро, соединяющее вершину из H с вершиной из K ; это и есть требуемое ребро.

(ii) Пусть C — цикл в графе G , не имеющий ни одного общего ребра с дополнением T ; тогда C содержится в T , что невозможно. ■

С понятием остовного леса T графа G тесно связано понятие фундаментальной системы циклов, ассоциированной с T . Оно определяется следующим образом: если добавить к T любое не содержащееся в нем ребро графа G , то по предложению (vi) теоремы 1 получим единственный цикл; множество всех циклов, получаемых таким способом (т. е. путем добавления по отдельности каждого ребра из G , не содержащегося в T), называется фундаментальной системой циклов, ассоциированной с T . В том случае, когда нас не интересует, какой остовный лес рассматривается, мы говорим о фундаментальной системе циклов графа G . Ясно, что циклы данной фундаментальной системы должны быть различными и что их число должно равняться циклическому рангу графа G . На рисунке 1.22 показана фундаментальная система циклов графа ассоциированная с остовным деревом:

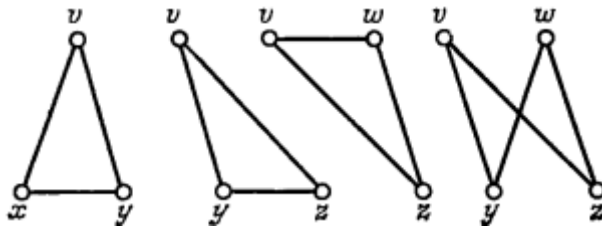


Рис. 1.22. Фундаментальная система циклов графа ассоциированная с остовным деревом

Можно определить фундаментальную систему разрезов графа G , ассоциированную с данным остовным лесом T . Покажем, что это действительно можно сделать. По предложению (iv) теоремы 1 удаление любого ребра из T разбивает множество вершин дерева T на два непересекающихся подмножества V_1 и V_2 . Множество всех ребер графа G , каждое из которых соединяет вершину из V_1 с вершиной из V_2 , является разрезом графа G . Множество всех разрезов, полученных таким способом (т. е. удалением по отдельности каждого ребра из T), называется фундаментальной системой разрезов, ассоциированной с T . Ясно, что разрезы

данной фундаментальной системы должны быть различными и что их число должно равняться коциклическому рангу графа G . Фундаментальной системой разрезов графа, изображенного на рис., ассоциированной с остовным деревом, изображенным на рис, является такая система: $\{e_1, e_5\}$, $\{e_2, e_5, e_7, e_8\}$, $\{e_3, e_6, e_7, e_8\}$ и $\{e_4, e_6, e_8\}$.

1.3.3 Маршруты, пути, циклы. Эйлеровы циклы

Маршрут в графе $G = (V, E)$ представляет собой конечную чередующуюся последовательность вершин и ребер $v_0, e_1, v_2, e_2, \dots, v_{k-1}, e_k, v_k$, начинающуюся и кончающуюся на вершинах, причем v_{i-1} и v_i являются концевыми вершинами ребра e_i , $1 \leq i \leq k$. С другой стороны, маршрут можно рассматривать как конечную последовательность таких вершин $v_0, v_1, v_2, \dots, v_k$, что (v_{i-1}, v_i) , $1 \leq i \leq k$ - ребро графа G . Такой маршрут обычно называется $v_0 - v_k$ -маршрутом, а v_0 и v_k - концевыми или терминальными вершинами маршрута. Все другие вершины маршрута называются внутренними. Заметим, что ребра и вершины в маршруте могут появляться более одного раза.

Маршрут называется открытым, если его концевые вершины различны, в противном случае он называется замкнутым. В графе на рис. 1.23 последовательность $v_1, e_1, v_2, e_2, v_3, e_8, v_6, e_9, v_5, e_7, v_3, e_{11}, v_6$ является открытым маршрутом, а последовательность $v_1, e_1, v_2, e_2, v_3, e_7, v_5, e_3, v_2, e_4, v_4, e_5, v_1$ -замкнутым.

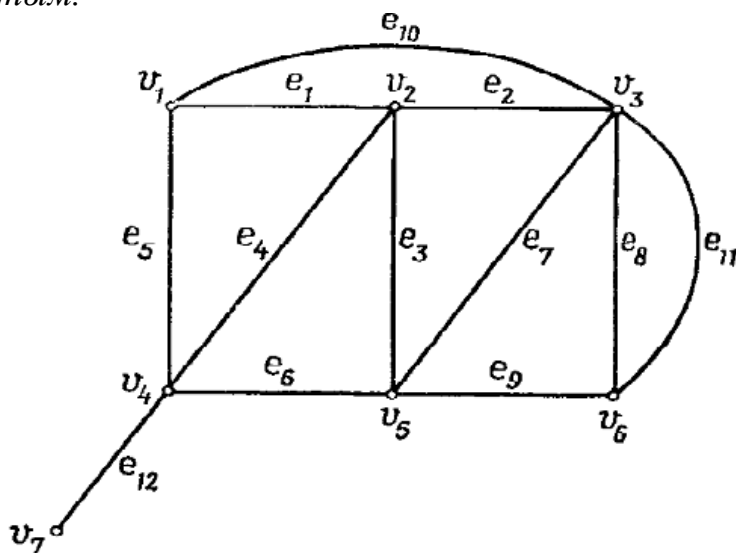


Рис. 1.23. Граф с маршрутом

Ребро графа G называется циклическим, если в графе G существует цикл, содержащий ребро. В противном случае ребро называется нециклическим. На рис. 1.23 все ребра, за исключением e_{12} , циклические.

Число ребер в пути называется длиной пути. Аналогично определяется длина цикла.

Необходимо указать следующие свойства путей и циклов.

1. Степень каждой не концевой вершины пути равна 2, концевые вершины имеют степень, равную 1.
2. Каждая вершина цикла имеет степень 2 или другую четную степень. Обращение этого утверждения, а именно то, что ребра подграфа, в котором каждая вершина имеет четную степень, образуют цикл,— неверно.
3. Число вершин в пути на единицу больше числа ребер, тогда как в цикле число ребер равно числу вершин.

1.3.4 Двудольные и планарные графы

Неориентированный граф $G = (V, E)$ называют **двудольным**, если множество его вершин V может быть разбито на такие два подмножества V_a и V_b , что каждое ребро имеет один конец в V_a , а другой в V_b .

Оrientированный граф G называется **двудольным**, если его неориентированный двойник – двудольный граф.

Двудольный граф $G=(V_a \cup V_b, E)$ называют **полным**, если для любых двух вершин $v_i \in V_a$ и $v_j \in V_b$ существует ребро (v_i, v_j) в $G=(V, E)$.

Планарные графы

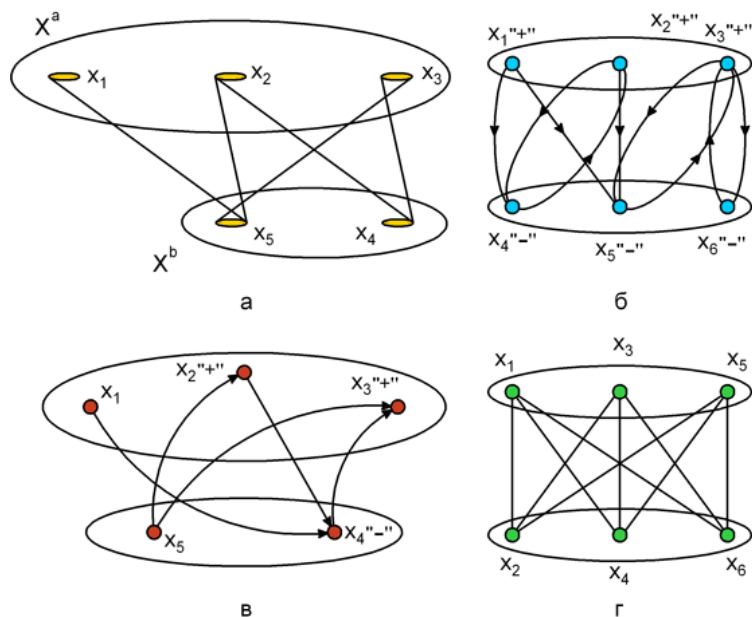


Рис. 1.24. Планарные графы

Будем говорить, что граф укладывается на поверхности S , если его можно так нарисовать на S , что никакие два его ребра не пересекаются. Граф называется **планарным**, если его можно уложить на плоскости; **плоский граф** — это граф, уже уложенный на плоскости. Например, кубический граф, показанный на рисунке, планарный, поскольку он изоморфен плоскому графу, изображенному на рисунке 1.25.

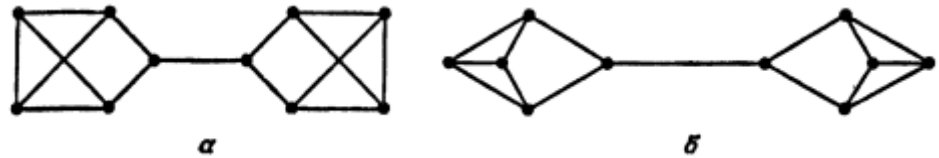


Рис. 1.25. Кубический граф

Очевидно, что если граф имеет петли или параллельные ребра, то ни в какой из его планарных укладок нельзя изобразить все ребра в виде отрезков прямых линий. Здесь, естественно, возникает вопрос: для каждого ли планарного графа G существует укладка, в которой все ребра могут быть изображены в виде отрезков прямых? Как устанавливается в следующей теореме, ответ на данный вопрос – положительный.

Теорема 1.35. Для каждого простого планарного графа существует планарная укладка, в которой все ребра графа можно изобразить в виде отрезков прямых линий. ■

Если граф не укладывается на плоскости, то его можно уложить на некоторой другой поверхности. Покажем, что укладываемость графа на плоскости и на сфере эквивалентны, т. е. если граф укладывается на плоскости, то он укладывается на сфере, и наоборот. В доказательстве этого утверждения используется понятие так называемой стереографической проекции сферы на плоскость, описываемое ниже.

Допустим, что мы положили сферу на плоскость. Назовем точку соприкосновения южным полюсом, а диаметрально противоположную точку на сфере — северным полюсом N . Пусть P — произвольная точка на сфере. Тогда точка P' , в которой прямая, соединяющая точки N и P пересекает плоскость, называется стереографической проекцией точки P на плоскости. Очевидно, что между точками на сфере и их стереографическими проекциями на плоскости существует взаимно-однозначное соответствие.

Теорема 1.36. Граф G укладывается на плоскости тогда и только тогда, когда он укладывается на сфере.

Доказательство. Пусть G' — укладка графа G на сфере. Положим сферу на плоскость так, чтобы северный полюс не был ни вершиной, ни точкой на ребре укладки G' .

Тогда образ G' в стереографической проекции — это укладка графа G на плоскости, поскольку ребра укладки G' пересекаются только в своих концевых вершинах, а между точками на сфере, и их образами в стереографической проекции существует взаимно однозначное соответствие. Обратное доказывается аналогично. ■

Два основных непланарных графа, называемые графами Куратовского, представлены на рисунке. Один из них K_5 — полный граф на пяти вершинах, а другой — $K_{3,3}$. Называем эти графы основными непланарными графами потому, что они играют основополагающую роль в характеристике планарности.

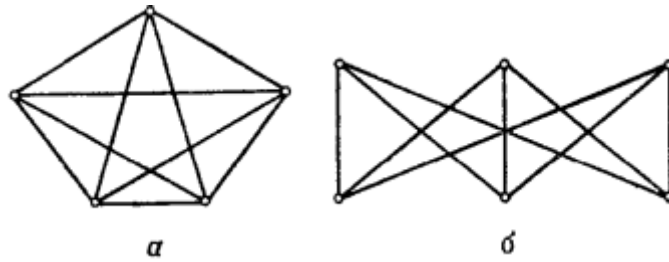


Рис. 1.26. Графы Куратовского

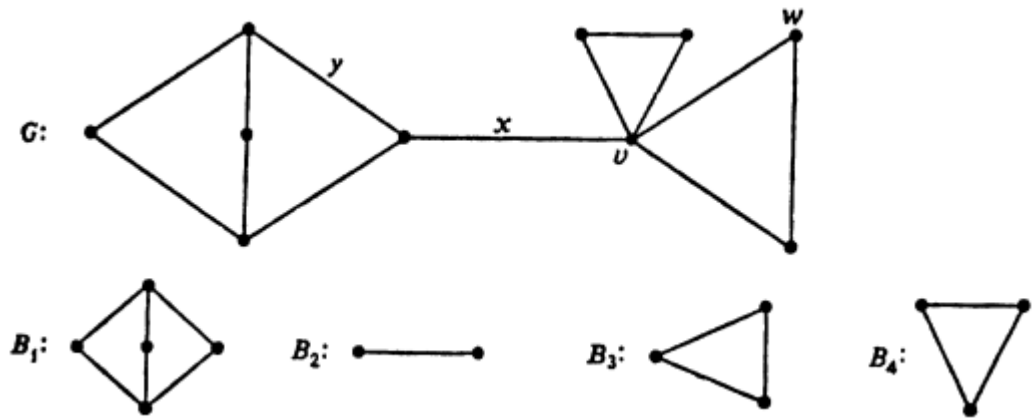
Точки сочленения, мосты и блоки

Точкой сочленения графа называется вершина, удаление которой увеличивает число компонент; ребро с таким же свойством называется мостом. Таким образом, если v — точка сочленения связного графа G , то граф $G - v$ не связен. Неразделимым графом называется связный, непустой, не имеющий точек сочленения граф. Блок графа — это его максимальный неразделимый подграф. Если G — неразделимый граф, то часто он сам называется блоком.

На рисунке v — точка сочленения, а w нет, x — мост, а y нет; отдельно приведены четыре блока графа G . Каждое ребро графа принадлежит точно одному из его блоков, так же как и каждая вершина, не являющаяся ни изолированной, ни точкой сочленения. Далее, ребра любого простого цикла графа G также принадлежат только одному блоку. Отсюда, в частности, следует, что блоки графа разбивают его ребра и простые циклы на множества, которые можно рассматривать как множества ребер. В первых трех теоремах этой главы устанавливаются несколько эквивалентных условий, обеспечивающих существование у графа точки сочленения и моста и неразделимость графа.

Теорема 1.37. Пусть v — вершина связного графа G . Следующие утверждения эквивалентны:

- (1) v — точка сочленения графа G ;
- (2) существуют такие вершины u и w , отличные от v , что v принадлежит любой простой $(u-w)$ -цепи;
- (3) существует разбиение множества вершин $V - \{v\}$ на такие два подмножества U и W , что для любых вершин $u \in U$ и $w \in W$ вершина v принадлежит любой простой $(u-w)$ -цепи.



Доказательство. (1) влечет (3). Так как v — точка сочленения графа G , то граф $G - v$ не связан и имеет по крайней мере две компоненты. Образует разбиение $V - \{v\}$, отнеся к U вершины одной из этих компонент, а к W — вершины всех остальных компонент. Тогда любые две вершины $u \in U$ и $w \in W$ лежат в разных компонентах графа $G - v$. Следовательно, любая простая $(u-w)$ -цепь графа G содержит v .

(3) влечет (2). Это немедленно следует из того, что (2) — частный случай утверждения (3).

(2) влечет (1). Если v принадлежит любой простой цепи в G , соединяющей u и w , то в G нет простой цепи, соединяющей эти вершины в $G - v$. Поскольку $G - v$ не связан, то v — точка сочленения графа G . ■

Теорема 1.38. Пусть x — ребро связного графа G . Следующие утверждения эквивалентны:

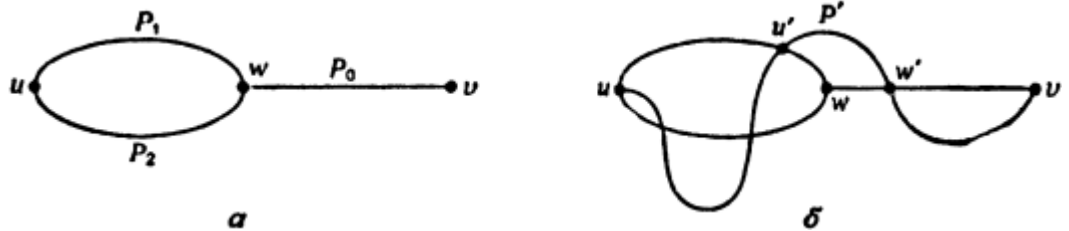
- (1) x — мост графа G ;
- (2) x не принадлежит ни одному простому циклу графа G ;
- (3) в G существуют такие вершины u и v , что ребро x принадлежит любой простой цепи, соединяющей u и v ;
- (4) существует разбиение множества V на такие подмножества U и W , что для любых вершин $u \in U$ и $w \in W$ ребро x принадлежит любой простой $(u-w)$ -цепи.

Теорема 1.39. Пусть G — связный граф с не менее чем тремя вершинами. Следующие утверждения эквивалентны:

- (1) G — блок;
- (2) любые две вершины графа G принадлежат некоторому общему простому циклу;
- (3) любая вершина и любое ребро графа G принадлежат некоторому общему простому циклу;
- (4) любые два ребра графа G принадлежат некоторому общему простому циклу;
- (5) для любых двух вершин и любого ребра графа G существует простая цепь, соединяющая эти вершины и включающая данное ребро;
- (6) для любых трех различных вершин графа G существует простая цепь, соединяющая две из них и проходящая через третью;

(7) для каждой трех различных вершин графа G существует простая цепь, соединяющая две из них и не проходящая через третью.

Доказательство. (1) влечет (2). Пусть u, v — различные вершины графа G , а U — множество вершин, отличных от u , которые лежат на простом цикле, содержащем u . Поскольку в G по крайней мере три вершины и нет точек сочленения, то в G нет также мостов. Значит, каждая вершина, смежная с u , принадлежит U , т. е. U не пусто.



Предположим, что v не принадлежит U . Пусть w — вершина в U , для которой расстояние $d(w, v)$ минимально. Пусть P_0 — кратчайшая простая $(w-v)$ -цепь, а P_1 и P_2 — две простые $(u-w)$ -цепи цикла, содержащего u и w (рис. 38, а). Так как w не является точкой сочленения, то существует простая $(u-v)$ -цепь P' , не содержащая w (рис. 38, б). Обозначим через w' ближайшую к u вершину, принадлежащую P' , которая также принадлежит P_0 , и через u' последнюю вершину $(u-w')$ -подцепи в P' , которая принадлежит или P_1 , или P_2 . Не теряя общности, предположим, что u' принадлежит P_1 .

Пусть Q_1 — простая $(u-w)$ -цепь, содержащая $(u-u')$ -подцепь цепи P_1 и $(u'-w')$ -подцепь цепи P' , а Q_2 — простая $(u-w')$ -подцепь, содержащая P_2 вслед за $(w-w')$ -подцепью цепи P_0 . Ясно, что Q_1 и Q_2 — непересекающиеся простые $(u-w')$ -цепи. Вместе они образуют простой цикл, так что w' принадлежит U . Поскольку w' принадлежит кратчайшей цепи, $d(w', v) < d(w, v)$. Это противоречит выбору w и, следовательно, доказывает, что uv лежат на одном простом цикле.

(2) влечет (3). Пусть u — вершина, vw — ребро графа G , а Z — простой цикл, содержащий u и v . Простой цикл Z' , содержащий u и vw , можно образовать следующим образом. Если w лежит на Z , то Z' содержит vw и $(v-w)$ -подцепь в Z , содержащую u . Если w не лежит на Z , то существует $(w-u)$ -цепь P , не содержащая v , поскольку иначе по теореме 6, v — точка сочленения. Пусть u' — первая вершина цепи P в Z . Тогда Z' содержит vw вслед за $(w-u')$ -подцепью цепи P и $(u'-v)$ -цепью в Z , включающей u .

(3) влечет (4). Доказательство, как в предыдущем случае.

(4) влечет (5). Каждая из двух вершин графа G инцидентна некоторому ребру; соответствующие ребра в силу утверждения (4) лежат на одном простом цикле. Следовательно, любые две вершины графа G принадлежат одному простому циклу, а отсюда следует (2) и, значит, (3). Пусть u и v — различные вершины, x — ребро графа G . Из утверждения (3) получаем, что существуют простой цикл Z_1 содержащий u и x , и простой цикл Z_2 ,

содержащий v и x . Таким образом, нужно рассмотреть только случай, когда v не лежит на Z_1 , а u не лежит на Z_2 . Начнем идти из u по Z_1 до тех пор, пока не достигнем первой вершины w цикла Z_2 , затем пойдём по цепи на Z_2 , которая соединяет w и v и содержит x . Такой обход образует простую цепь, соединяющую u и v и содержащую x .

(5) влечет (6). Пусть u , v и w — различные вершины графа G , а x — произвольное ребро, инцидентное w . Из утверждения (5) вытекает, что существует простая цепь, соединяющая u и v , которая содержит x и, следовательно, должна содержать w .

(6) влечет (7). Пусть u , v и w — различные вершины графа G . Из утверждения (6) вытекает, что существует простая $(u-w)$ -цепь P , содержащая v . Ясно, что $(u-v)$ -подцепь цепи P не содержит w .

(7) влечет (1). Используя (7), получаем, что для любых двух вершин u и v ни одна из остальных вершин не может принадлежать каждой $(u-v)$ -цепи. Следовательно, G должен быть блоком. ■

Теорема 1.40. В любом нетривиальном связном графе найдутся по крайней мере две вершины, не являющиеся точками сочленения.

Доказательство. Пусть u и v — вершины графа G , максимально удаленные друг от друга, т. е. такие, что $d(u, v) = d(G)$. Предположим, что v — точка сочленения. Тогда существует вершина w , принадлежащая той компоненте графа $G - v$, которая не содержит вершину u . Значит, v лежит на любой цепи, соединяющей u и w , и поэтому $d(u, w) > d(u, v)$, что невозможно. Следовательно, v , а также u не являются точками сочленения графа G . ■

1.3.5 Задача о кратчайшем пути. Алгоритм Дейкстры

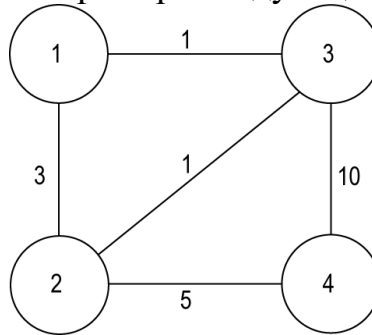
Алгоритм Дейкстры позволяет эффективно находить кратчайшие пути из заданной стартовой вершины графа до всех остальных его вершин.

Пусть дан некоторый граф с n вершинами и m ребрами. Граф может быть как ориентированным, так и неориентированным. Для каждого ребра графа указан вес (некоторое число). Также указана стартовая вершина s .

Задача — найти длину кратчайшего пути из s в каждую из остальных вершин графа. Под «кратчайшим» подразумевается путь, сумма весов ребер которого принимает минимально возможное значение.

Классический и самый наглядный вариант этой задачи — поиск оптимального маршрута из одного города в другой. Пусть нам известна карта дорог, связывающих между собой n городов; для каждой дороги известно, какие города она соединяет и сколько времени необходимо для ее преодоления. Требуется найти самый быстрый маршрут от одного заданного города до другого.

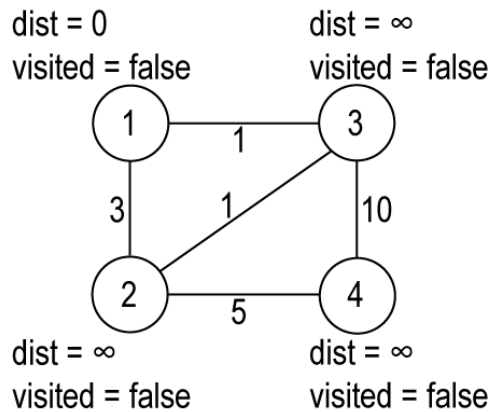
Рассмотрим в качестве примера следующий неориентированный граф



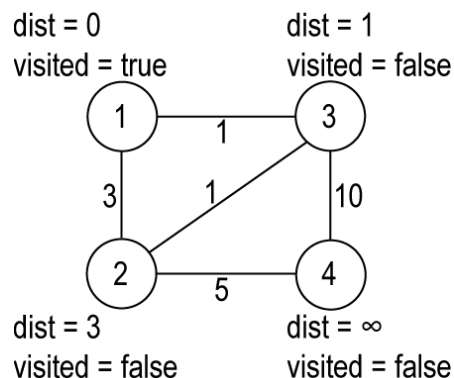
Если стоит задача найти кратчайший путь из города №1 в город №4, то оптимальным решением будет путь 1-3-2-4 с суммарной стоимостью 7.

Рассмотрим пошаговую работу алгоритма Дейкстра на приведенном выше графе со стартовой вершиной $s = 1$.

Изначально ни одна вершина не посещена, а текущие расстояния до всех вершин, кроме s , равны бесконечности.

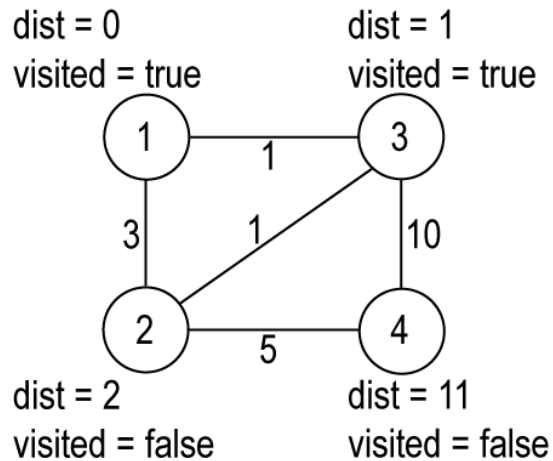


Алгоритм выбирает непосещенную вершину с минимальным расстоянием — это вершина $v = 1$. Эта вершина отмечается как посещенная, и производятся релаксации по исходящим из нее ребрам в вершины №2 и 3 (обе релаксации улучшают текущие расстояния до этих вершин).

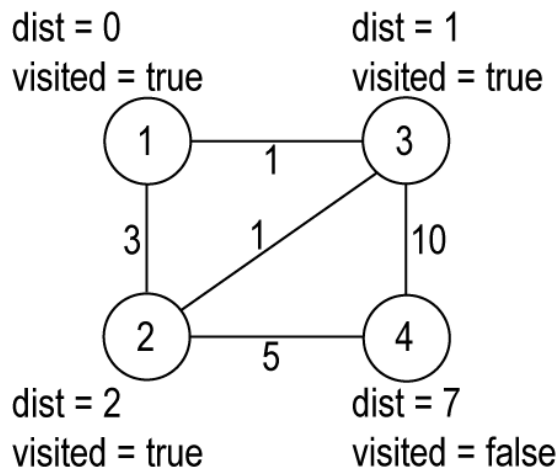


Алгоритм снова выбирает непосещенную вершину с минимальным расстоянием — теперь это вершина $v = 3$. Она помечается посещенной, и по ребрам из нее производятся релаксации. Ребро в вершину №1 не приводит к

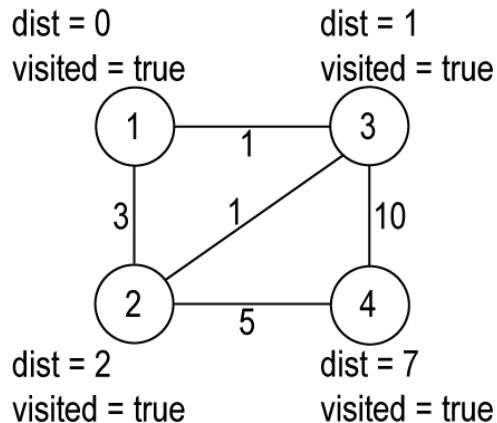
изменениям, но ребра в вершины №2 и 4 улучшают текущие расстояния до этих вершин.



Алгоритм снова выбирает вершину — теперь это $v = 2$. Она помечается посещенной, и производятся релаксации; при этом улучшается текущее расстояние до вершины №4.



Алгоритм снова выбирает вершину — это $v = 4$. Она помечается посещенной; ни одна релаксация из этой вершины больше не улучшает ответ.



Больше непосещенных вершин не осталось. Алгоритм завершает свою работу

2 Практический раздел

2.1. Комбинаторика. Теория множеств и отношения. Кодирование

Задание 1: Множества и операции над ними

Описание: Реализуйте класс `Set`, который будет представлять множество и поддерживать основные операции: объединение, пересечение, разность и симметричную разность.

Требования:

- Реализуйте методы для каждой операции.
- Добавьте возможность добавления и удаления элементов из множества.
- Напишите тесты для проверки корректности работы всех методов.

Пример использования:

```
set1 = Set([1, 2, 3])
set2 = Set([3, 4, 5])
print(set1.union(set2)) # {1, 2, 3, 4, 5}
print(set1.intersection(set2)) # {3}
```

Задание 2: Отношения между множествами

Описание: Реализуйте класс `Relation`, который будет представлять бинарное отношение между элементами двух множеств.

Требования:

- Реализуйте методы для проверки рефлексивности, симметричности и транзитивности отношения.
- Создайте несколько примеров отношений и проверьте их свойства.

Пример использования:

```
relation = Relation([(1, 2), (2, 3), (1, 3)])
print(relation.is_transitive()) # True
```

Задание 3: Кодирование строк

Описание: Реализуйте алгоритм Хаффмана для сжатия строк.

Требования:

- Создайте дерево Хаффмана на основе частоты символов в строке.
- Реализуйте кодирование и декодирование строки.
- Проверьте работу алгоритма на нескольких примерах.

Пример использования:

```
huffman = HuffmanCoding("hello world")
encoded = huffman.encode()
decoded = huffman.decode(encoded)
print(encoded) # Закодированная строка
print(decoded) # "hello world"
```

Задание 4: Генерация всех подмножеств

Описание: Напишите функцию, которая генерирует все возможные подмножества заданного множества.

Требования:

- Используйте рекурсию или итеративный подход.
- Убедитесь, что вы не дублируете подмножества.

Пример ввода: `set = {1, 2, 3}`

Пример вывода:

`{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}`

Задание 5: Кодирование с использованием базовых принципов комбинаторики

Описание: Реализуйте систему кодирования для представления множеств с использованием двоичных строк.

Требования:

- Каждому элементу множества сопоставьте бит (0 или 1).
- Реализуйте функции для кодирования и декодирования множества в двоичную строку.

Пример использования:

`set = {1, 3}`

`binary_string = encode_set(set) # "101" для множества {1, 2}`

`decoded_set = decode_set(binary_string) # {1, 3}`

Задание 6: Перекрывающиеся множества

Описание: Напишите программу, которая принимает два множества и определяет, сколько элементов у них пересекаются.

Требования:

- Используйте операции над множествами для нахождения пересечения.
- Выведите количество пересекающихся элементов и сами пересекающиеся элементы.

Пример ввода: `set1 = {1, 2, 3}, set2 = {2, 3, 4}`

Пример вывода:

Количество пересечений: 2

Пересекающиеся элементы: `{2, 3}`

Задание 7: Комбинаторные игры

Описание: Создайте простую игру на основе теории множеств и комбинаторики. Например, игра "Угадай число", где игроки по очереди выбирают числа из заданного множества.

Требования:

- Реализуйте логику игры с учетом правил и условий победы.
- Позвольте пользователю играть против компьютера или другого игрока.

2.2. Математическая логика

Задание 1: Логические высказывания

Описание: Определите истинность следующих логических

высказываний. Для каждого высказывания укажите, является ли оно истинным или ложным, и объясните свой ответ.

Выбор высказываний:

1. $p \wedge (q \vee r)$
2. $\neg (p \vee q)$
3. $p \rightarrow (q \wedge r)$
4. $(p \wedge q) \rightarrow r$

Требования:

- Задайте значения переменным p, q, r (истина или ложь).
- Объясните, как вы пришли к выводу о истинности или ложности каждого высказывания.

Задание 2: Таблицы истинности

Описание: Создайте таблицы истинности для следующих логических выражений:

1. $p \vee (q \wedge r)$
2. $(p \rightarrow q) \wedge (q \rightarrow r)$
3. $\neg (p \vee q) \vee r$

Требования:

- Заполните таблицы истинности для всех возможных комбинаций значений переменных.
- Объясните, как вы получили результаты.

Задание 3: Преобразование логических выражений

Описание: Преобразуйте следующие логические выражения в эквивалентные с использованием законов логики (например, закон де Моргана, дистрибутивный закон).

1. $\neg (p \wedge q)$
2. $p \vee (q \wedge r)$
3. $(p \vee q) \wedge (p \vee r)$

Требования:

- Укажите каждый шаг преобразования.
- Объясните, почему каждое преобразование эквивалентно исходному выражению.

Задание 4: Логические доказательства

Описание: Используя правила вывода (например, *modus ponens*, *modus tollens*), докажите следующие утверждения:

1. Если $p \rightarrow q$ и p , то q .
2. Если $p \rightarrow q$ и $\neg q$, то $\neg p$.
3. Если $p \wedge q$, то p .

Требования:

- Запишите формальные шаги вашего доказательства.
- Объясните каждый шаг и его логику.

Задание 5: Логические эквиваленты

Описание: Найдите логические эквиваленты для следующих выражений и объясните, почему они эквивалентны.

1. $p \rightarrow q$
2. $p \vee q$
3. $p \wedge (q \vee r)$

Требования:

- Используйте правила логики и законы эквивалентности.
- Объясните, какие законы вы использовали для нахождения эквивалентов.

Задание 6: Применение логики в программировании

Описание: Напишите программу на Python, которая принимает логическое выражение в виде строки и вычисляет его истинность для заданных значений переменных.

Требования:

- Используйте библиотеку `sympy` или `rueda` для работы с логическими выражениями.
- Программа должна принимать пользовательский ввод для переменных и выражения.
- Выведите результат вычисления.

Пример использования:

```
>>> evaluate_expression("p and (q or r)", p=True, q=False, r=True)
True
```

Задание 7: Сложные логические задачи

Описание: Решите несколько задач на применение логики в реальных сценариях. Например, задайте условия для выбора кандидата на работу, используя логические операторы.

Пример задачи:

Кандидат подходит, если он:

- Имеет опыт работы (E) и образование (D), или
- Имеет опыт работы (E) и хорошую рекомендацию (R).

Запишите логическое выражение для этой задачи и определите, подходит ли кандидат с заданными значениями.

Задание 8: Логические игры

Описание: Создайте простую игру на основе логических высказываний. Например, игра "Правда или ложь", где игроки должны определить истинность утверждений.

Требования:

- Реализуйте логику игры с учетом правил.
- Позвольте пользователю выбирать между несколькими

утверждениями и проверять свои ответы.

2.3. Теория графов

Задание 1: Основные понятия графов

Описание: Определите и проиллюстрируйте основные понятия графов: вершины, ребра, направленные и ненаправленные графы, взвешенные и невзвешенные графы.

Требования:

- Нарисуйте примеры направленного и ненаправленного графа.
- Укажите, какие из них являются взвешенными, а какие — невзвешенными.
- Объясните, как вы определили каждое из понятий.

Задание 2: Представление графов

Описание: Реализуйте различные способы представления графов в памяти: матрица смежности и список смежности.

Требования:

- Напишите код на Python, который принимает список рёбер графа и создает обе структуры.
- Приведите примеры и объясните, в каких случаях лучше использовать каждую из структур.

Задание 3: Поиск в глубину (DFS) и поиск в ширину (BFS)

Описание: Реализуйте алгоритмы поиска в глубину (DFS) и поиска в ширину (BFS) для обхода графа.

Требования:

- Напишите функции на Python для выполнения обоих алгоритмов.
- Проведите тестирование на примере простого графа и сравните результаты обхода.

Задание 4: Кратчайший путь

Описание: Реализуйте алгоритм Дейкстры для нахождения кратчайшего пути в взвешенном графе.

Требования:

- Напишите код на Python, который принимает граф в виде списка смежности и находит кратчайший путь от одной вершины до всех остальных.
- Проведите тестирование на примере взвешенного графа и выведите результаты.

Задание 5: Проверка связности графа

Описание: Напишите программу, которая проверяет, является ли заданный граф связным.

Требования:

- Используйте один из алгоритмов обхода (DFS или BFS) для проверки связности.
- Проведите тестирование на различных примерах (связные и несвязные графы).

Задание 6: Деревья

Описание: Определите, является ли данный граф деревом.

Требования:

- Напишите функцию на Python, которая проверяет, является ли граф деревом (граф без циклов, связный и с $n-1$ рёбрами).
- Проведите тестирование на нескольких примерах.

Задание 7: Поиск циклов в графе

Описание: Реализуйте алгоритм для поиска циклов в направленном графе.

Требования:

- Напишите код на Python, который использует DFS для обнаружения циклов.
- Проведите тестирование на примерах с циклами и без них.

Задание 8: Алгоритм Краскала

Описание: Реализуйте алгоритм Краскала для нахождения минимального остовного дерева.

Требования:

- Напишите код на Python, который принимает взвешенный граф и возвращает его минимальное остовное дерево.
- Проведите тестирование на примере взвешенного графа.

Задание 9: Алгоритм Флойда-Уоршелла

Описание: Реализуйте алгоритм Флойда-Уоршелла для нахождения кратчайших путей между всеми парами вершин.

Требования:

- Напишите код на Python и протестируйте его на примере взвешенного графа.
- Объясните временную сложность алгоритма и его применение.

Задание 10: Применение теории графов

Описание: Придумайте реальный сценарий, где можно применить теорию графов (например, маршрутизация, социальные сети, логистика).

Требования:

- Опишите задачу и представьте её в виде графа.
- Предложите алгоритмы для решения задачи и обоснуйте их выбор.

Задание 11: Графы в социальных сетях

Описание: Проанализируйте данные из социальной сети (например, Twitter или Facebook) и постройте граф взаимодействий пользователей.

Требования:

- Используйте API социальной сети для получения данных о пользователях и их взаимодействиях.
- Постройте граф и проанализируйте его свойства (например, степень вершин, наличие кластеров).

Задание 12: Визуализация графов

Описание: Используйте библиотеку для визуализации графов

(например, NetworkX или Graphviz) для отображения вашего графа.

Требования:

- Постройте визуализацию для одного из ваших предыдущих заданий.
- Добавьте аннотации к вершинам и рёбрам для лучшего понимания структуры графа.

Вопросы по курсу «Дискретная математика»

1. Высказывания. Сложные высказывания. Условные высказывания.
2. Таблицы истинности сложных высказываний.
3. Тавтологии и противоречия.
4. Логическая эквивалентность высказываний. Булева алгебра высказываний
5. Выполнимые и невыполнимые высказывания. Проблема выполнимости.
6. Предикаты. Кванторы. Ограничительные кванторы.
7. Логические эквивалентности выражений, содержащих кванторы.
8. Отрицания выражений, содержащих кванторы.
9. Правильные рассуждения в логике высказываний.
10. Правила вывода в логике высказываний.
11. Правила вывода для квантификаций.
12. Операции над множествами.
13. Булева алгебра множеств.
14. Компьютерное представление множеств.
15. Правило произведения.
16. Правило суммы.
17. Принцип Дирихле.
18. Формулы включений и исключений.
19. Бинарные отношения и их свойства.
20. Отношения эквивалентности.
21. Разбиения.
22. Отношения частичного порядка.
23. Диаграммы Хассе.
24. Алгоритм топологической сортировки.
25. Перестановки.
26. Сочетания.
27. Биномиальная теорема.
28. Тождество и треугольник Паскаля.
29. r -перестановки с повторениями.
30. r -сочетания с повторениями.
31. Перестановки с одинаковыми объектами.
32. Полиномиальная теорема.
33. Линейные однородные рекуррентные соотношения с

постоянными коэффициентами.

34. Характеристическое уравнение линейного однородного рекуррентного соотношения с постоянными коэффициентами.

35. Решение линейных однородных рекуррентных соотношений с постоянными коэффициентами степени 2 и степени 3.

36. Линейные неоднородные рекуррентные соотношения с постоянными коэффициентами и методы их решений.

37. Производящие функции, решение рекуррентных соотношений с помощью производящих функций.

38. Графы. Определения и примеры.

39. Пути и циклы.

40. Эйлеровы графы.

41. Гамильтоновы графы.

42. Изоморфизм графов.

43. Деревья.

44. Остовное дерево графа.

45. Планарные графы.

46. Формула Эйлера для связного планарного графа.

47. Непланарность графов $K_{3,3}$ и K_5 . Теорема Куратовского.

48. Деревья с корнем.

49. Коды Хаффмана.

50. Нагруженные графы.

51. Алгоритм Прима нахождения минимального остовного дерева.

52. Задача о поиске кратчайшего пути.

53. Алгоритм Дейкстры.

54. Задача о бродячем торговце.

55. Алгоритм ближайшего соседа.

56. Алгоритм ближайшей вставки.

57. Булевы функции.

58. Булева алгебра булевых функций.

59. Представления булевых функций: ДНФ и КНФ.

60. Полные системы булевых функций.

61. Минимальные ДНФ. Карты Карно.

62. Алгоритм Дейкстры и его реализация.

Программа курса

Раздел I. Комбинаторика. Теория множеств и отношения. Кодирование

Тема 1.1. Основные комбинаторные операции и принципы Основные комбинаторные соединения.

Тема 1.2. Множества и операции над ними

Множества и операции над ним и их свойства. Булева алгебра множеств. Отображения множеств. Композиция отображений. Виды отображений. Критерий обратимости. Декартово произведение множеств. Кодирование. Шифрование.

Тема 1.3. Упорядоченные множества. Отношения на множествах

Бинарные отношения. Отношения эквивалентности, фактор-множества. Отношения частичного порядка. Частично упорядоченные, линейно упорядоченные и вполне упорядоченные множества. Решетки.

Тема 1.4. Мультимножества

Мультимножества множества.

Тема 1.5. Кодирование**Тема 1.6. Шифрование****Раздел II. Математическая логика****Тема 2.1 Соответствия и функции**

Математическая логика. Высказывания. Логические союзы. Формулы. Тавтологии. Законы логики.

Тема 2.2 Элементарные булевы функции, канонические способы задания. Полные системы булевых функций, базисы

Булевы функции. Пропозиционные формулы. Многочлены Жегалкина. Полные системы булевых функций. Теоремы о полноте. Базисы. Нормальные формы булевых функций. СДНФ и СКНФ. Теорема Шеннона. Минимизация булевых функций в классе ДНФ.

Тема 2.3 . Высказывания, методы проверки логического следования

Алгоритм Квайна. Метод карт Карно. Реализации булевых функций. Логические схемы.

Тема 2.4 Предикаты, предикатные формулы

Предикаты. Кванторы. Формулы исчисления предикатов. Предваренная нормальная форма. Понятие об алгоритмических теориях. Полнота и непротиворечивость. Теоремы Геделя. Понятие о машине Тьюринга.

Раздел III. Теория графов

Тема 3.1 Основные понятия теории графов, типы графов, обходы графов, способы задания

Основные определения: графы, мультиграфы, псевдографы, оргграфы. Числовые характеристики и матрицы графов и оргграфов.

Тема 3.2 Ориентированные ациклические графы и деревья

Изоморфные графы. Маршруты, пути, циклы, цепи. Связные компоненты графа. Расстояние в графах. Радиус, диаметр, центры связного графа. Алгоритм поиска в ширину.

Тема 3.3 Маршруты, пути, циклы. Эйлеровы циклы

Теорема Куратовского-Понтрягина. Раскраски графа. Хроматическое число графа. Эйлеровы графы. Критерий эйлеровости графа

Тема 3.4 Двудольные и планарные графы

Алгоритмы нахождения остовов минимального веса, пути минимальной длины. Циклы и разрезы. Циклический и коциклический ранг графа. Двудольные графы. Критерий двудольности.

Тема 3.5 Задача о кратчайшем пути. Алгоритм Дейкстры

Задача о кратчайшем пути. Алгоритм Дейкстры

Список использованных источников

1. Горбатов В.А. Фундаментальные основы дискретной математики. Информационная математика. – М.: Наука. Физматлит, 2000. – 544 с.
2. Новиков Ф.А. Дискретная математика для программистов: учебник для вузов. 3-е изд. СПб.: Питер, 2009. – 384 с.: ил.
3. Кузнецов О. П. Дискретная математика для инженеров. — СПб.: Лань, 2004.
4. Гладков Л.А., Курейчик В.В., Курейчик В.М. Дискретная математика. Часть 1. Теория множеств. – Таганрог, 2005. – 160 с.
- 5.9. Гладков Л.А., Курейчик В.В., Курейчик В.М. Дискретная математика. Часть 2. Теория графов. – Таганрог, 2010. – 162 с.

Дополнительная литература

6. Белоусов А.И., Ткачев С.Б. Дискретная математика. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 744 с.

7. Берж К. Теория графов и ее применение. М.: Иностранная литература, 1962.8.
10. Информация в понятиях и терминах / Под ред. В.А. Извозчикова. М.: Просвещение, 1991.
11. Липский В. Комбинаторика для программистов. М.: Мир, 1988.
12. Мелихов А.Н. Ориентированные графы и конечные автоматы. М.: Наука, 1971.
13. Оре О. Теория графов. – М.: Наука, 1980. – 336 с.
14. Соболева Т.С., Чечкин А.В. Дискретная математика. – М.: Издательский центр «Академия», 2006. — 256 с.
15. Татт У. Теория графов. – М.: Мир, 1988. - 424 с.
16. Тишин В.В. Дискретная математика в примерах и задачах. – СПб.: БХВ-Петербург, 2008. — 352 с.
17. Уилсон Р. Введение в теорию графов. – М.: Мир, 1977. – 208 с.
18. Харари Ф. Теория графов / Пер. с англ. и предисл. В. П. Козырева. — М.: Едиториал УРСС, 2003. — 296 с.
19. Яблонский С.С. Введение в дискретную математику. М.: Наука, 1979

Приложение 1

1. Задача «Проверка наличия пути».

Дана карта авиаперелетов, которые связывают между собой аэропорты в различных городах; авиаперелеты однонаправленны, т. е. с помощью одного перелета можно добраться из одного города до другого, но не в обратную сторону. Требуется определить, можно ли добраться из города x в город y — с использованием одного или нескольких авиаперелетов.

Входные данные содержат четыре разделенных пробелом числа n , m , x , y : число городов, авиаперелетов, номер стартового города и номер конечного города (города нумеруются от 1 до n) соответственно. Ограничения: $2 \leq n \leq 105$, $0 \leq m \leq 105$, $x \neq y$. Последующие m строк содержат по два разделенных пробелом числа a_i и b_i ($a_i \neq b_i$), задающих город отправления и город прибытия каждого авиаперелета. Требуется вывести «YES» или «NO» в зависимости от того, существует ли требуемый путь.

Примеры

Входные данные	Требуемый результат
4 4 3 4 1 2 2 3 3 1 2 4	YES
Входные данные	Требуемый результат
4 2 1 3 3 2 2 1	NO

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере существует следующий путь: $3 \rightarrow 1 \rightarrow 2 \rightarrow 4$. Во втором примере пути из города №1 в город №3 нет (хотя есть путь в обратном направлении).

2.Задача «Конная прогулка».

Дано поле размером $n \times m$ клеток. В одной из его клеток находится конюшня, а остальные клетки либо пусты, либо содержат болото, т. е. непроходимы. Пустая клетка называется прогулочной, если ее можно достичь, стартовав из конюшни и двигаясь по правилам шахматного коня (т. е. прыгая на расстояние 1 по одной координате и 2 по другой), не выходя за пределы поля и не наступая в клетки с болотом. Подсчитайте количество прогулочных клеток. Входные данные состоят из пары натуральных чисел n и m в первой строке и описания поля в последующих n строках, с m символов в каждой. В описании поля используются следующие символы:

- точка («.»), обозначающая пустую клетку;
- решетка («#»), обозначающая болото,;
- заглавная буква икс («X»), обозначающая конюшню.

Гарантируется, что $1 \leq n, m \leq 100$ и описание поля содержит ровно один символ «X». Вывести требуется единственное число — количество прогулочных клеток.

Примеры

Входные данные	Требуемый результат
3 4 ..##	7

X...	
....	
Входные данные	Требуемый результат
3 3 ----- ----- #X# ###	0

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере можно, стартовав из клетки «X» и делая шаги шахматным конем, достичь всех клеток «.», кроме двух (кроме второй клетки второй строки и четвертой клетки третьей строки); поэтому ответ равен 7. Во втором примере нет ни одной пустой, а значит, и прогулочной, клетки

3.Задача «Проверка связности».

Для заданного неориентированного графа определить, является ли он связным или нет. Входные данные состоят из числа вершин n и ребер m в первой строке ($1 \leq n, m \leq 105$) и описаний ребер в последующих m строках, содержащих по паре чисел — номеров вершин (натуральных чисел, не превосходящих n). Все ребра различны, и никакое ребро не ведет из вершины в нее же. Вывести требуется «YES» или «NO» в зависимости от того, связан граф или нет

Примеры

Входные данные	Требуемый результат
3 2 1 3 2 3	YES
Входные данные	Требуемый результат
5 3 1 4 2 3 5 2	NO

ПОЯСНЕНИЯ К ПРИМЕРАМ:

В первом примере из любой вершины можно добраться до двух остальных. Во втором примере из вершин №1 и 4 нельзя добраться до вершин №2, 3, 5.

4.Задача «Проверка двудольного графа».

Дан неориентированный граф. Требуется проверить, является ли он двудольным. Входные данные состоят из числа вершин и ребер в первой строке и описаний ребер в последующих строках. Каждое ребро задается номерами вершин-концов в 1- индексации. Число вершин положительно, число вершин и ребер не превосходит 105, и петли отсутствуют (т. е. нет ребер, ведущих из вершины в нее же). Вывести требуется «0», если граф не двудольный; в противном случае — последовательность из разделенных пробелом чисел «1» или «2», указывающих номер доли для каждой вершины. Если решений несколько, вывести можно любое.

Примеры

Входные данные	Требуемый результат
5 5 1 2 1 3 4 5 2 4 3 4	1 2 2 1 2
Входные данные	Требуемый результат
4 3 2 3 3 4 2 4	0

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере вершины №1 и 4 должны принадлежать одной доле, в то время как все остальные - другой доле. Во втором примере решения не существует.

5.Задача «Диаметр дерева».

Дано дерево, т. е. связный неориентированный граф без циклов. Требуется найти его диаметр, т. е. расстояние между двумя наиболее удаленными друг от друга вершинами. Входные данные содержат число вершин n в первой строке ($1 \leq n \leq 5 \cdot 10^5$) и описание ребер в виде $n - 1$ строк, содержащих по два числа каждая: номера вершин концов ребра. Вершины нумеруются начиная с единицы. Вывести требуется три числа — искомый

диаметр и номера двух вершин, расстояние между которыми равно диаметру. При наличии нескольких решений вывести можно любое.

Примеры

Входные данные	Требуемый результат
6 1 2 1 3 1 6 2 4 3 5	4 4 5
Входные данные	Требуемый результат
1	0 1 1

6. Задача «Кластер компьютеров».

Серверный кластер состоит из n компьютеров, некоторые из них напрямую связаны сетевым кабелем. Говорят, что два компьютера соединены быстрой связью, если можно передать данные с одного на другой, используя не более чем k промежуточных компьютеров. Например, при $k = 0$ только связанные напрямую кабелем компьютеры считаются соединенными быстрой связью. Задача — посчитать, со сколькими компьютерами соединен быстрой связью компьютер №1. Входные данные состоят из числа n компьютеров в кластере, параметра k и числа m прямых сетевых соединений ($1 \leq n \leq 105$, $0 \leq k < n - 1$, $0 \leq m \leq 105$) в первой строке. Последующие m строк содержат по паре чисел a_i и b_i , обозначающих, что компьютеры a_i и b_i связаны прямым сетевым кабелем ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$). Вывести требуется одно число — искомое количество компьютеров.

Примеры

Входные данные	Требуемый результат
5 1 4 1 2 2 5 3 1 5 4	3
Входные данные	Требуемый результат
4 2 2 1 4 2 3	1

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере компьютер №1 соединен быстрой связью с компьютерами №2, 3, 5; компьютер №4 не подходит, поскольку для передачи данных с компьютера №1 требуется два других компьютера: №2 и 5. Во втором примере компьютер №1 соединен быстрой связью только с компьютером №4.

7.Задача «Подсчет простых чисел в отрезке».

Даны два числа L и R таких, что $1 \leq L \leq R \leq 1016$ и $R - L \leq 106$. Требуется найти количество простых чисел в отрезке $[L; R]$.

Примеры

Входные данные	Требуемый результат
10 20	4
Входные данные	Требуемый результат
1000000000000 10000000000100	4

8.Задача «Подсчет путей фиксированной длины».

Дан неориентированный граф и число k . Требуется посчитать для каждой пары вершин (i, j) количество различных путей, имеющих длину, равную k (т. е. содержащих k ребер). Путь может содержать какие-либо ребра более одного раза; при подсчете длины учитывается каждое вхождение каждого ребра. Пути считаются различными, если они отличаются набором использованных ребер или их порядком. Поскольку ответ может быть большим, количества надо искать по модулю заданного числа. Входные данные содержат в первой строке число вершин n , число ребер m , число k и модуль p и затем описания ребер в последующих m строках. Каждое ребро описывается в виде двух чисел от 1 до n , задающих номера вершин-концов ребра. Гарантируется, что все ребра различны и нет ребер-петель. Ограничения: $1 \leq n \leq 100$, $0 \leq k \leq 109$, $2 \leq p \leq 109$. Вывести требуется n строк по n чисел в каждой: j -е число в i -й строке должно быть равно числу путей из вершины i в вершину j по модулю p .

Примеры

Входные данные	Требуемый результат
3 3 2 100 1 2 2 3 3 1	2 1 1 1 2 1 1 1 2
Входные данные	Требуемый результат
2 1 3 2	0 1

2 1	1 0
-----	-----

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере из вершины №1 есть два пути длины 2 в себя же: 1-2-1 и 1-3-1 и по одному пути длины 2 до остальных вершин: 1-2-3, 1-3-2. Ввиду симметричности графа количества путей, начинающихся в вершинах №2 и №3, такие же.

9.Задача «Наидлиннейшая возрастающая подпоследовательность».

Дана последовательность a_i целых чисел длины n . Требуется найти ее наидлиннейшую возрастающую подпоследовательность.

Входные данные содержат число n в первой строке и n чисел a_i во второй. Ограничения: $1 \leq n \leq 5 \cdot 10^5$, $|a_i| < 10^9$. Вывести требуется одно число — длину искомой наидлиннейшей возрастающей последовательности.

Примеры

Входные данные	Требуемый результат
6 1 4 2 3 6 5	4

10.Задача «Наименьший общий предок».

Дано дерево из n вершин. Требуется ответить на m запросов вида «найти наименьшего общего предка вершин a_i и b_i ». Входные данные содержат числа n и m в первой строке ($1 \leq n, m \leq 5 \cdot 10^5$). Вторая строка содержит описание дерева в виде $n - 1$ чисел p_i ($i = 2 \dots n$), обозначающих, что для вершины с номером i родителем является вершина с номером p_i ($1 \leq p_i \leq n$). Следующие m строк содержат запросы: по паре чисел a_i и b_i в каждой.

Вывести требуется m строк по одной для каждого запроса: номер вершины искомого наименьшего общего предка.

Примеры

Входные данные	Требуемый результат
7 4 1 1 2 2 3 3 4 5	2 1 3

3 4 3 6 1 1	1
-------------------	---

11.Задача «Пересадки в метро».

Дана карта метро: k линий, где i -я линия соединяет станции $a_{i,1}, a_{i,2}, \dots, a_{i,b_i}$ в этой последовательности. Все линии двунаправленные. Если через какую-либо станцию проходят несколько линий метро, то можно совершить пересадку. Требуется найти маршрут от станции x до станции y с наименьшим числом пересадок. Входные данные содержат числа k, x, y в первой строке и описания линий метро в последующих k строках. Каждая строка содержит число b_i , за которым следуют числа $a_{i,1}, a_{i,2}, \dots, a_{i,b_i}$. Ограничения: $1 \leq k \leq 103$, номера станций являются натуральными числами, не превосходящими 105, и сумма всех b_i не превосходит 105. Вывести требуется наименьшее число пересадок на маршруте из x в y , либо «-1», если такого маршрута нет.

Примеры

Входные данные	Требуемый результат
3 1 5 3 1 2 3 3 4 2 5 4 5 4 3 2	1
Входные данные	Требуемый результат
1 10 20 4 1 2 3 4	-1

12.Задача «Основание столицы».

На определенном этапе развития протогосударства X одно из его поселений становится столицей. Конечно, из столицы должны существовать пути до всех остальных поселений протогосударства. Некоторые поселения уже связаны маршрутами; какое наименьшее число маршрутов требуется добавить, чтобы из столицы можно было добраться до всех поселений?

В первой строке входных данных содержатся число поселений n и число имеющихся маршрутов m . Последующие m строк содержат по паре чисел a_i и b_i , обозначающих наличие **однаправленного** маршрута из поселения a_i в поселение b_i (поселения нумеруются, начиная с единицы). Столица имеет номер «1». Ограничения: $1 \leq n \leq 105$, $0 \leq m \leq 105$.

Вывести требуется одно число — минимальное количество дополнительных маршрутов.

Примеры

Входные данные	Требуемый результат
4 3 2 3 3 4 4 2	1
Входные данные	Требуемый результат
3 0	2

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере достаточно добавить маршрут из столицы (поселения №1) в любое другое поселение, например №2. Во втором примере требуется добавить маршруты из столицы в каждое из остальных поселений.

13.Задача «Перемножение по большому модулю».

Даны два целых числа a и b и модуль p . Требуется вывести результат произведения a и b по модулю p . Все три числа неотрицательны и не превосходят 1018; модуль p не меньше 2.

Примеры

Входные данные	Требуемый результат
2 3 5	4
Входные данные	Требуемый результат
123 0 6	0

14.Задача «Подсчет числа путей с длиной, не превосходящей заданную».

По заданному неориентированному графу посчитать для каждой пары вершин количество различных путей между этими двумя вершинами, имеющих длину **не более чем** k . Количества требуется найти по заданному модулю p . В графе могут присутствовать петли и кратные ребра. Первая строка входных данных содержит число вершин, число ребер, длину k и модуль p . Последующие строки содержат описания ребер, где каждая строка содержит номера вершин-концов ребра (в 1-индексации). Гарантируется, что число вершин не превосходит 100, число ребер — 104, число k находится в отрезке $[0; 109]$, а модуль p — в отрезке $[2; 109]$. Вывести требуется матрицу из $n \times n$ чисел, содержащую искомые количества.

Примеры

Входные данные	Требуемый результат
3 2 3 10	2 3 1

1 2	3 3 3
2 3	1 3 2
Входные данные	Требуемый результат
2 2 2 100	1 0
2 2	0 7
2 2	

15.Задача «Поиск самого дешевого пути с заданным числом ребер».

Дан неориентированный граф, каждому ребру которого приписана стоимость, и число k . Требуется для каждой пары вершин (i, j) найти самый дешевый путь, ведущий из вершины i в вершину j и содержащий ровно k ребер. Путь может проходить по ребрам более одного раза; в этом случае учитывается каждое вхождение ребра. Первая строка входных данных содержит число вершин, число ребер, длину k . Последующие строки содержат описания ребер, где каждая строка содержит номера вершин-концов ребра (в 1-индексации) и число — стоимость прохождения по этому ребру. Гарантируется, что число вершин не превосходит 100, число ребер — 104, стоимости ребер и число k находятся в отрезке $[0; 109]$. Вывести требуется матрицу из $n \times n$ чисел, содержащую искомые длины. В случае когда между парой вершиной отсутствует требуемый путь, выводить в соответствующей ячейке следует число «-1».

Примеры

Входные данные	Требуемый результат
4 4 2	2 -1 3 -1
1 2 4	-1 6 -1 5
2 3 3	3 -1 4 -1
3 4 2	-1 5 -1 2
4 1 1	
Входные данные	Требуемый результат
2 2 2 100	1 0
2 2	0 7
2 2	

16.Задача «Оптимальный путь четной длины».

Дан взвешенный неориентированный граф, содержащий n вершин и m ребер. Требуется найти путь из вершины №1 в вершину № n , состоящий из четного числа ребер, и при этом сумма весов входящих в него ребер была бы минимально возможной. Входные данные содержат n и m в первой строке и описания ребер в виде троек a_i, b_i, c_i в последующих строках. Гарантируется, что $2 \leq n \leq 105$, $0 \leq m \leq 105$, $1 \leq a_i, b_i \leq n$, $0 \leq c_i \leq 1000$. В графе могут быть кратные ребра и петли. Вывести требуется суммарную стоимость искомого пути в первой строке и вершины пути во второй. Если ответов несколько, разрешается вывести любой. Если искомого пути не существует, следует вывести «-1».

Примеры

Входные данные	Требуемый результат
3 3 1 2 10 1 3 3 2 3 10	20 1 2 3
Входные данные	Требуемый результат
3 2 1 2 1 1 3 1	-1

17.Задача «Ребра кратчайших путей».

Органы автомобильной инспекции установили, что наибольшее число превышений скорости допускается водителями города X , и в особенности во время их поездок в город Y и обратно (точные причины этого не установлены, но, возможно, причина кроется в насыщенной ночной жизни Y и активности жителей X). Инспекторы хотят установить радары на некоторые из дорог, чтобы штрафовать нарушителей. С одной стороны, поскольку установка и обслуживание радара требует больших затрат, нужно обойтись как можно меньшим числом радаров. С другой стороны, радары должны стоять на каждом участке всех дорог, используемых «лихачами». Инспекция установила, что «лихачи» всегда ездят вдоль кратчайшего пути из X в Y (что логично из соображений экономии времени), однако если кратчайших путей несколько, разные водители могут использовать разные пути. Ваша задача — по данной карте дорог найти те дороги, на которые требуется установить радары. Формат карты дорог следующий: число перекрестков и число дорог в первой строке и описания дорог в последующих строках в виде троек целых чисел « $A B C$ », где A и B — номера перекрестков, C — время проезда по дороге. Перекрестки нумеруются, начиная с единицы, все дороги двунаправленные, между каждой парой перекрестков есть не более одной дороги, и никакая дорога не связывает перекресток с самим собой. Выезд из города X соответствует перекрестку №1, въезд в город Y — перекрестку с

наибольшим номером; гарантируется, что существует хотя бы один маршрут, связывающий X и Y . Ограничения: число перекрестков лежит в диапазоне $[2; 105]$, число дорог — в диапазоне $[1; 5 \cdot 105]$, время проезда по каждой дороге — в диапазоне $[1; 1000]$.

Вывести нужно число требуемых радаров в первой строке и номера дорог, на которые их нужно установить, в виде возрастающей последовательности чисел во второй строке. Дороги нумеруются в том порядке, в котором они идут во входных данных, начиная с единицы.

Примеры

Входные данные	Требуемый результат
3 3 1 2 1 1 3 3 2 3 1	2 1 3
Входные данные	Требуемый результат
3 2 1 2 100 2 3 100	2 1 2

18. Задача «Железнодорожный вокзал».

В городе N строится железнодорожный вокзал. Из-за нехватки места архи-

тектор пытается определить, какое наименьшее число железнодорожных путей требуется на вокзале. Всего на вокзал будут прибывать k поездов, однако не все из них должны находиться на вокзале одновременно. Известен набор ограничений вида (a_i, b_i) , которые обозначают, что поезд с номером a_i должен прибыть на вокзал не позднее поезда b_i (в частности, это делается для того, чтобы пассажиры одного поезда могли пересесть в другой). Архитектор понял, что, несмотря на все его попытки разнести прибытие поездов по времени, из-за этих ограничений возможны ситуации, когда сразу несколько поездов должны одновременно находиться на вокзале. Помогите архитектору определить минимально необходимое число железнодорожных путей на вокзале. Входные данные состоят из числа поездов k и числа ограничений m в первой строке ($1 \leq k, m \leq 105$) и пар чисел « $a_i b_i$ » в последующих m строках ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$). Вывести требуется единственное число — искомое минимальное число путей.

Примеры

Входные данные	Требуемый результат
4 4 1 2 2 3 3 4 4 2	3
Входные данные	Требуемый результат
4 4 1 2 1 3 2 4 3 4	1

19.Задача «Сбор дани».

Команда историков занимается исследованиями экономических отношений между княжеской администрацией и подконтрольными ей городами. Удалось восстановить карту возможных маршрутов, а также размер дани, которую должен был выплатить каждый город. Известно, что княжеская дружина отправилась из города S в город T , однако конкретный путь их следования неизвестен. Историки предполагают, что дружина следовала маршрутом, который максимизирует суммарный сбор дани — при этом, однако, если дружина посещала какой-либо город повторно, собирать дань второй раз они там уже не могли. Помогите историкам найти, каким мог быть максимально возможный сбор. Первая строка входных данных содержит число городов N ($2 \leq N \leq 105$), число маршрутов M ($1 \leq M \leq 105$), номера S и T ($1 \leq S, T \leq N, S \neq T$). Во второй строке записаны N целых чисел W_i — размеры дани для каждого из городов ($1 \leq W_i \leq 1000$). Последующие M строк описывают дороги: каждая строка содержит по два числа A_i и B_i ($1 \leq A_i, B_i \leq N, A_i \neq B_i$), обозначающих, что из города A_i можно добраться в город B_i . Отметим, что маршруты являются односторонними (например, потому что это могли быть водные маршруты по быстрым рекам). Вывести требуется единственное число — искомый максимально возможный сбор дани вдоль оптимального пути. Гарантируется, что из S в T можно добраться по указанным маршрутам.

Примеры

Входные данные	Требуемый результат
4 4 1 4 1 1 1 1 1 2 2 3 3 1 1 4	4

Входные данные	Требуемый результат
4 4 1 3 1 2 3 4 1 2 1 4 2 3 4 3	8

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере оптимальным путем является путь 1-2-3-1-4; дань при этом собирается со всех городов. Во втором примере оптимально двигаться вдоль пути 1-4-3, что позволяет собрать дань со всех городов, кроме города №2.

20. Задача «Прыжки вперед и назад».

Листая страницы различных книг рекордов, Петя обратил внимание на то, что человеку гораздо труднее прыгать спиной вперед: рекорды по длине прыжка с места отличаются более чем в полтора раза (рекорд прыжка спиной вперед, конечно, хуже). Размышляя над этим, Петя задумался: а насколько близко можно оказаться к стартовой точке, совершив некоторое число прыжков вперед и некоторое — назад? Петя исходит из предположений, что в одном направлении спортсмен прыгает всегда на одно и то же расстояние. Кроме того, запрещено возвращаться прямо в стартовую точку или оказываться до нее: во всякий момент времени после первого прыжка координата спортсмена должна оставаться строго положительной.

Входной файл содержит два целых числа F и B — длина одного прыжка вперед и длина одного прыжка назад ($1 \leq B < F \leq 109$). Вывести требуется три целых числа — искомое минимально возможное расстояние, количество совершенных для этого прыжков вперед и количество совершенных прыжков назад. Из всех вариантов ответа следует выбрать вариант с наименьшим числом прыжков вперед.

Примеры

Входные данные	Требуемый результат
30 20	10 1 1
Входные данные	Требуемый результат
99 5	1 4 79

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере можно оказаться на расстоянии 10 от стартовой точки, совершив один прыжок вперед и один назад. Во втором примере можно

оказаться на расстоянии 1 от стартовой точки, совершив четыре прыжка вперед и семьдесят девять прыжков назад: $99 \cdot 4 - 5 \cdot 79 = 1$.

21.Задача «Линейное диофантово уравнение с двумя переменными».

Для заданных целых a , b и c решить линейное диофантово уравнение:

$$ax + by = c.$$

Входные данные состоят из трех разделенных пробелами целых чисел a , b , c , не превосходящих по модулю 109. Вывести требуется два разделенных пробелами целых числа — искомое решение,

либо «No solution», если решения не существует. Если решений несколько, можно вывести любое решение, элементы которого не превосходят по модулю 1018.

Примеры

Входные данные	Требуемый результат
3 4 2	2 -1
Входные данные	Требуемый результат
2 4 1	N0 solution

22.Задача «Обратное по составному модулю».

Дано натуральное число a и модуль p . Требуется найти число, обратное к a

по модулю p , т. е. решение следующего уравнения:

$$a * x = 1 \bmod p.$$

Входные данные состоят из натуральных чисел a и p , разделенных пробелом. Гарантируется, что $a < p < 109$. Вывести требуется одно число — искомый обратный элемент в промежутке $[0; p)$. Если решения не существует, нужно вывести «-1».

Примеры

Входные данные	Требуемый результат
7 11	8
Входные данные	Требуемый результат
3 20	7

23.Задача «Цикл минимального веса».

Дан взвешенный неориентированный граф. Требуется найти в этом графе простой цикл, сумма весов ребер которого минимальна. Простым

циклом является такая последовательность вершин, в которой все элементы различны, кроме последнего элемента, который должен совпадать с первым элементом. Входные данные содержат описание графа. В первой строке указано число вершин и число ребер. Каждая из последующих строк описывает очередное ребро графа в виде тройки чисел: номеров вершин-концов ребра (вершины нумеруются, начиная с единицы) и веса ребра (все веса лежат в интервале от 1 до 100). Число вершин не превосходит 500. Все ребра различны, и петли отсутствуют. Вывести требуется «-1», если цикла нет, либо последовательность номеров вершин, входящих в искомый цикл. Если циклов с минимальным весом несколько, можно вывести любой.

Примеры

Входные данные	Требуемый результат
4 6 1 2 2 1 4 2 2 3 1 2 4 1 3 4 1	2 3 4 2
Входные данные	Требуемый результат
3 2 1 2 1 2 3 1	-1

24. Задача «Картельный сговор».

Антимонопольная служба поручила вам реализовать эвристический алгоритм, обнаруживающий потенциально нелегальные отношения между компаниями. В наличии имеется база данных контрактов, представляющая собой пары вида (a_i, b_i) , обозначающие, что компания под номером a_i оказала услугу компании b_i . Пара компаний (i, j) считается подозрительной, если обе этих компании оказывали друг другу услуги (возможно, через посредников). Первая строка входных данных содержит число компаний n ($2 \leq n \leq 105$), число контрактов m ($0 \leq m \leq 105$), количество запросов k ($1 \leq k \leq 105$). Следующие m строк содержат описания контрактов в виде двух чисел a_i и b_i ($1 \leq a_i, b_i \leq n$). Последние k строк содержат запросы: пары x_i и y_i ($1 \leq x_i, y_i \leq n$). Вывести требуется k строк — ответы на запросы, т. е. «YES», если пара (x_i, y_i) является подозрительной, и «NO» в противном случае

Примеры

Входные данные	Требуемый результат
4 3 3 1 2	YES NO

2 3 3 1 1 2 1 4 2 3	YES
Входные данные	Требуемый результат
4 0 1 1 2	NO

25.Задача «Превращение графа в сильно связный».

Дан ориентированный граф. Требуется определить наименьшее число ребер, которые можно добавить, чтобы получить сильно связный граф. Входные данные содержат описание графа. В первой строке указано число вершин и число ребер. Каждая из последующих строк описывает очередное ребро графа в виде пары чисел: номер вершины-начала и номер вершины-конца. Число вершин и ребер не превосходит 105.

Примеры

Входные данные	Требуемый результат
4 4 1 2 1 3 2 4 3 4	1
Входные данные	Требуемый результат
5 2 1 2 3 4	3

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере достаточно добавить ребро (4, 1). Во втором примере один из возможных вариантов — ребра (2, 3), (4, 5), (5, 1).

26.Задача «Циклы перестановки».

Дана перестановка. Требуется вывести количество циклов в ней.

Примеры

Входные данные	Требуемый результат
1 3 2	2
Входные данные	Требуемый результат

2 3 4 1	1
---------	---

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере есть два цикла: (1) и (32), поскольку единица переходит в единицу, тройка — в двойку, а двойка — в тройку. Во втором примере единственный цикл — (1234), поскольку единица переходит в двойку, двойка — в тройку, тройка — в четверку, четверка — в единицу.

27.Задача «Исключающее ИЛИ от 1 до n».

Дано число n . Требуется посчитать $1 + 2 + \dots + n$ (где знак «+» это XOR), т. е. результат приме-

нения битовой операции XOR ко всем числам от 1 до n . Известно, что n — натуральное число, не превосходящее 1018.

Примеры

Входные данные	Требуемый результат
2	3
Входные данные	Требуемый результат
5	1

28.Задача «Врачи и посетители».

В маленькой клинике работают всего два врача. Несколько пациентов записались на прием в клинику в определенные часы; изначально неизвестно, к которому из врачей попадет посетитель. Ваша задача — спланировать работу врачей, т. е. определить, кто должен принять первого пациента, кто — второго и т. д. Разумеется, каждый врач может работать только с одним пациентом в каждый момент времени (однако допускается принять нового пациента в тот же момент времени, когда посещение другим пациентом завершается). Входные данные начинаются с числа n — натурального числа, не превосходящего 100 и задающего количество посетителей. Затем следуют описания запланированных посещений: два целых числа l_i и r_i ($0 < l_i < r_i < 1000$). Вывести требуется строку из n символов, в которой i -й символ должен быть «А» или «В» в зависимости от того, к первому или ко второму врачу должен пойти i -й пациент. Если существует несколько возможных решений, разрешается вывести любое из них. Если решения нет, то следует вывести «No solution».

Примеры

Входные данные	Требуемый результат
3 1 2	ААВ

2 3	
1 3	
Входные данные	Требуемый результат
3	No solution
1 3	
1 4	
2 4	

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере приведенное решение «ААВ» обозначает, что в момент времени 1 пациент №1 заходит на прием к первому врачу, а пациент №3 — ко второму врачу. В момент времени 2 пациент №1 уходит и вместо него на прием к первому врачу заходит пациент №2. В момент времени 3 пациенты №2 и 3 уходят.

29.Задача «Стоимость интернет-связи».

Интернет-провайдер ввел тариф с переменной ценой. Например, в некоторые дни недели цена одного мегабайта интернет-трафика ночью ниже, чем днем.

Тариф полностью описывается списком записей вида «день недели, час, цена», обозначающих, что указанная цена действует с начала указанного часа и вплоть до следующей записи. Тариф, действующий на момент окончания воскресенья, по умолчанию продолжает действовать в утро понедельника. Требуется посчитать стоимость сеанса интернет-связи, описанного списком записей «день недели, час, число мегабайт». Первая строка входного файла содержит натуральное число n — число записей в описании тарифа. Следующие n строк содержат записи описания тарифа, каждая состоящая из названия дня недели («Mon», «Tue», «Wed», «Thu», «Fri», «Sat» или

«Sun»), часа (целого числа от 0 до 23) и цены (неотрицательного целого числа, не превосходящего 1000), разделенных пробелами. Характеристики тарифа даны в хронологическом порядке, т. е. в порядке увеличения дней недели или в порядке увеличения номера часа внутри одного дня. Гарантируется, что никакие две записи не совпадают по времени. В следующей строке записано число k — число записей в описании сеанса связи. Следующие k строк содержат записи описания сеанса связи,

каждая состоящая из дня недели, часа и числа мегабайт, разделенных пробелами. Число мегабайт в каждой записи — натуральное, не превосходящее 1000. Описания сеанса связи могут иметь совпадающие день и час и заданы в произвольном порядке. Вывести требуется единственное число — стоимость сеанса связи.

Примеры

Входные данные	Требуемый результат
3 Mon 0 10 Mon 21 5 Tue 0 10 2 Mon 20 1 Mon 21 1	15
Входные данные	Требуемый результат
1 Wed 12 10 1 Mon 0 1	10

30.Задача «Проверка скобочной последовательности».

Дана строка, состоящая только из круглых скобок, т. е. «(» и «)». Строка не пуста и не превосходит по длине 105. Требуется проверить ее на правильность, т. е. могла ли она получиться из корректного арифметического выражения после удаления из него цифр и знаков операций. Ответ на задачу требуется вывести в виде строки «YES» или строки «NO».

Формально все правильные скобочные последовательности образуются следующим образом: это либо пустая строка, либо строка вида «(X)», либо строка вида «XY», где X и Y — правильные скобочные последовательности.

Примеры

Входные данные	Требуемый результат
((()))()	YES
Входные данные	Требуемый результат
)(	No
Входные данные	Требуемый результат
((	NO

**31.Задача «Проверка скобочной последовательности
с двумя типами».**

Дана непустая строка длины до 105, состоящая только из символов «(», «)», «[», «]». Требуется вывести «YES» или «NO» в зависимости от того, правильная это скобочная последовательность или нет.

Примеры

Входные данные	Требуемый результат
([()])[([])]	YES
Входные данные	Требуемый результат
(]	No
Входные данные	Требуемый результат
[] (	NO

32.Задача «Троичная сбалансированная система счисления».

Дано целое неотрицательное число n , записанное в десятичной системе счисления. Требуется перевести его в троичную сбалансированную систему счисления. Входное число не превосходит 109. Выводить результат требуется с помощью символов «0», «+» и «-» без лишних лидирующих нулей.

Примеры

Входные данные	Требуемый результат
5	+--
Входные данные	Требуемый результат
0	0

33.Задача «Фибоначчилистник».

Растение Фибоначчилистник растет следующим образом. В первый день из семечки вырастает ветвь. На второй день на ветви появляется лист и почка. На третий и последующие дни появляется по одному новому листу, а из почки вырастает по одной новой ветви; эти новые ветви развиваются аналогичным образом. Требуется вывести общее число листьев на момент окончания n -го дня ($1 \leq n \leq 50$).

Примеры

Входные данные	Требуемый результат
4	4
Входные данные	Требуемый результат
1	0

--	--

ПОЯСНЕНИЯ К ПРИМЕРАМ

В первом примере в конце четвертого дня имеется 3 ветви, на первой из которых 2 листа, на второй — 1, на третьей — ни одного.

34.Задача «Задача. Картина на кафеле».

На стену, выложенную кафельной плиткой, повесили картину. Требуется посчитать, сколько плиток полностью покрыты картиной и сколько — частично. Известно, что все плитки имеют равный размер, ориентированы одинаковым образом и каждая плитка выложена целиком и без наложений с другими плитками. Начало координат находится в левом нижнем углу стены.

В первой строке входных данных содержатся два целых числа — размеры плитки (по осям абсцисс и ординат соответственно). Во второй и третьей строках даны координаты двух противоположных углов картины (в виде пары абсциссы и ординаты); гарантируется, что обе размерности картины ненулевые. Все числа во входном наборе — неотрицательные целые, не превосходящие 103. Вывести требуется два числа, разделенных пробелом: количество полностью покрытых плиток и количество частично покрытых плиток.

Примеры

Входные данные	Требуемый результат
3 5 0 0 10 10	6 2
Входные данные	Требуемый результат
50 50 40 50 50 40	0 1

35.Задача «Манхэттенский центр».

Даны координаты n точек, заданных своими координатами x_i и y_i . Это локации мест в Манхэттене, и таксист хочет найти такое расположение своей базы, которое бы минимизировало сумму расстояний от базы до всех этих точек. В идеализированной модели Манхэттена, используемой в этой задаче, таксист может передвигаться только по линиям целочисленной решетки, параллельной координатным осям, и скорость его движения постоянна и не зависит от направления движения. Таксист может выбирать произвольные улицы, а повороты совершает мгновенно. Входные данные состоят из натурального числа n в первой строке ($1 \leq n \leq 105$) и n последующих строк, содержащих по паре координат x_i и y_i , разделенных пробелом. Все координаты целые и по модулю не превосходят 10 000. Вывести требуется два разделенных

пробелом целых числа — координаты искомой базы. Если оптимальных решений несколько, разрешается вывести любое.

Примеры

Входные данные	Требуемый результат
3 0 0 1 1 2 2	1 1
Входные данные	Требуемый результат
2 1 1 -1 1	0 1

Приложение 2

Пример цифрового сторителлинга по теме «Основные понятия теории графов, типы графов, обходы графов, способы задания»

История о разработчике Данииле и его проекте

Жил-был разработчик по имени Даниил, который работал над проектом по оптимизации маршрутов доставки для крупной логистической компании. Даниил знал, что для успешного выполнения задачи ему нужно хорошо разбираться в теории графов, чтобы моделировать и анализировать маршруты.

Глава 1: Введение в теорию графов

Даниил начал с изучения основных понятий теории графов. Он понял, что графы можно представить в виде множеств вершин и рёбер, соединяющих эти вершины.

Пример задачи: Построить граф, представляющий сеть складов и транспортных маршрутов между ними.

Глава 2: Типы графов

Затем Даниил изучил различные типы графов, такие как ориентированные и неориентированные графы, взвешенные и невзвешенные графы, деревья и циклы. Он понял, что каждый тип графа имеет свои уникальные свойства и области применения.

Пример задачи: Использовать взвешенный граф для моделирования затрат на транспортировку между складами.

Глава 3: Обходы графов

Даниил приступил к изучению методов обхода графов, таких как обход в ширину (BFS) и обход в глубину (DFS). Он понял, что эти методы помогают эффективно искать и анализировать пути в графе.

Пример задачи: Найти кратчайший путь между двумя складами с использованием метода обхода в ширину (BFS).

Глава 4: Способы задания графов

Даниил узнал о различных способах задания графов, таких как матрицы смежности, списки смежности и списки рёбер. Он понял, что правильный выбор способа задания графа зависит от конкретной задачи и требуемой эффективности.

Пример задачи: Представить сеть складов и маршрутов с помощью матрицы смежности для быстрого поиска путей.

Применение в профессиональной деятельности

Знания о теории графов, типах графов, обходах графов и способах их задания будут полезны вам в будущем на следующих курсах и в профессиональной деятельности.

1. Сетевые технологии

На старших курсах вы будете изучать курсы по сетевым технологиям, где теория графов играет ключевую роль. Вы научитесь моделировать и анализировать сети передачи данных.

Пример: Проектирование и оптимизация маршрутов передачи данных в сети Интернет с использованием теории графов.

2. Алгоритмы и структуры данных

В курсах по алгоритмам и структурам данных графы используются для представления и решения различных задач, таких как поиск пути, маршрутизация и оптимизация.

Пример: Разработка алгоритмов для поиска кратчайшего пути в графе, представляющем транспортную сеть или карту.

3. Информационные системы и базы данных

В курсах по информационным системам и базам данных графы применяются для моделирования связей и зависимостей между данными.

Пример: Построение графовых баз данных для управления информацией о сложных взаимосвязях между объектами, такими как социальные сети или системы рекомендаций.

Пример сторителлинга по теме «Ориентированные ациклические графы и деревья»

Уважаемые студенты, сегодня мы рассмотрим одну из важных тем дискретной математики — «Ориентированные ациклические графы и деревья». Эти концепции являются фундаментальными для вашего будущего профессионального роста в области информационных систем и технологий. Мы обсудим основные понятия, их применение, а также установим связь с специализированными курсами, которые вы будете изучать на старших курсах.

Основные понятия

Ориентированные ациклические графы (DAG, Directed Acyclic Graphs) и деревья — это специальные виды графов, которые имеют уникальные свойства и широкий спектр применения в различных областях. DAG — это графы, в которых отсутствуют циклы, а деревья — это связные ациклические графы с единственным корнем.

Пример 1: Ориентированные ациклические графы

Рассмотрим DAG, который можно использовать для моделирования зависимостей задач в проекте. В таком графе вершины представляют задачи, а рёбра — зависимости между задачами. Отсутствие циклов гарантирует, что не будет бесконечных зависимостей.

Пример: Ваша будущая профессиональная деятельность может включать управление проектами, где необходимо моделировать зависимости задач с помощью DAG.

Сторителлинг по теме «Маршруты, пути, циклы. Эйлеровы циклы»

История о логистическом инженере Викторе

Жил-был логистический инженер по имени Виктор, который работал над оптимизацией транспортных маршрутов для крупной логистической компании. Виктору нужно было разрабатывать и анализировать маршруты, пути и циклы, чтобы минимизировать затраты и повысить эффективность доставки.

Глава 1: Исследование маршрутов и путей

Виктор начал с изучения маршрутов и путей в графах. Он построил граф, где вершины представляли пункты доставки, а рёбра — маршруты между ними. Виктор использовал алгоритмы поиска кратчайшего пути, чтобы определить наиболее эффективные маршруты.

Пример задачи: Найти кратчайший путь между складом и всеми пунктами доставки, чтобы минимизировать затраты на транспортировку.

Глава 2: Работа с циклами

Затем Виктор приступил к изучению циклов. Он понял, что использование циклов в графах помогает обеспечить резервные маршруты при отказе основного пути. Виктор разработал систему, которая автоматически переключалась на резервные маршруты в случае сбоев.

Пример задачи: Обеспечить резервное маршрутизирование для сети доставки, чтобы предотвратить задержки при отказе основного маршрута.

Глава 3: Применение Эйлеровых циклов

Виктору также нужно было решить задачу оптимального планирования маршрута курьера, чтобы он посетил каждую улицу в районе ровно один раз. Виктор изучил условия существования Эйлеровых циклов и применил их к своему проекту.

Пример задачи: Построить маршрут для курьера, который покрывает все улицы в районе, минимизируя общее расстояние и время.

Применение в профессиональной деятельности

Знания о маршрутах, путях, циклах и Эйлеровых циклах будут полезны вам в будущем на следующих курсах и в профессиональной деятельности.

1. Сетевые технологии

На старших курсах вы будете изучать курсы по сетевым технологиям, где графы используются для моделирования и оптимизации маршрутов передачи данных.

Пример: Проектирование и оптимизация маршрутов передачи данных в сети Интернет с использованием теории графов и Эйлеровых циклов для обеспечения устойчивости сети.

2. Алгоритмы и структуры данных

В курсах по алгоритмам и структурам данных графы используются для представления и решения различных задач, таких как поиск путей, маршрутизация и оптимизация.

Пример: Разработка алгоритмов для поиска кратчайшего пути и планирования маршрутов в графах, представляющих транспортные сети или карты.

3. Информационные системы и базы данных

В курсах по информационным системам и базам данных графы применяются для моделирования связей и зависимостей между данными.

Пример: Построение графовых баз данных для управления информацией о сложных взаимосвязях между объектами, такими как логистические сети или системы рекомендаций.

Сторителлинг по теме «Сети и потоки. Задача о максимальном потоке»

История об инженерере Андрее и его проекте

Жил-был инженер по имени Андрей, который работал над проектом по оптимизации потоков данных в корпоративной сети. Андрею нужно было разрабатывать и анализировать сети и потоки, чтобы минимизировать задержки и максимизировать пропускную способность.

Глава 1: Исследование потоков в сети

Андрей начал с изучения основных понятий потоков в сети. Он построил граф, где вершины представляли узлы сети, а рёбра — соединения между ними с определённой пропускной способностью. Андрей использовал алгоритмы для вычисления текущих потоков данных в сети.

Пример задачи: Определить максимальный поток данных, который можно пропустить через корпоративную сеть от серверного центра к пользователям.

Глава 2: Применение задачи о максимальном потоке

Затем Андрей приступил к изучению задачи о максимальном потоке. Он применил алгоритм Форда-Фалкерсона для нахождения максимального потока в сети. Андрей разработал систему, которая автоматически балансировала потоки данных и обеспечивала высокую пропускную способность.

Пример задачи: Оптимизировать маршруты данных в сети, чтобы максимизировать пропускную способность между серверами и клиентами.

Глава 3: Применение методов в корпоративной сети

Андрей применил свои знания для создания системы управления потоками данных, которая могла автоматически адаптироваться к изменяющимся условиям и обеспечивать надёжное и эффективное распределение ресурсов.

Пример задачи: Создать систему, которая автоматически перенаправляет потоки данных в случае перегрузки или отказа узла, используя алгоритмы максимального потока.

Применение в профессиональной деятельности

Знания о сетях, потоках и задаче о максимальном потоке будут полезны вам в будущем на следующих курсах и в профессиональной деятельности.

1. Сетевые технологии

На старших курсах вы будете изучать курсы по сетевым технологиям, где графы используются для моделирования и оптимизации потоков передачи данных.

Пример: Проектирование и оптимизация маршрутов передачи данных в сети Интернет с использованием теории графов и задач о максимальном потоке для обеспечения устойчивости сети.

2. Алгоритмы и структуры данных

В курсах по алгоритмам и структурам данных графы используются для представления и решения различных задач, таких как поиск путей, маршрутизация и оптимизация.

Пример: Разработка алгоритмов для нахождения максимального потока в графе, представляющем транспортные сети или карты.

3. Информационные системы и базы данных

В курсах по информационным системам и базам данных графы применяются для моделирования связей и зависимостей между данными.

Пример: Построение графовых баз данных для управления информацией о сложных взаимосвязях между объектами, такими как логистические сети или системы

Сторителлинг по теме «Экстремальные задачи на графах»

История об аналитике данных Сергее и его проекте

Жил-был аналитик данных по имени Сергей, который работал над проектом по оптимизации распределения задач на серверной ферме. Сергею нужно было разрабатывать и анализировать графы, чтобы минимизировать конфликты и максимизировать эффективность использования ресурсов.

Глава 1: Исследование максимального независимого множества

Сергей начал с изучения задачи нахождения максимального независимого множества. Он построил граф, где вершины представляли задачи, а рёбра — конфликты между ними. Сергей использовал алгоритмы для нахождения наибольшего подмножества задач, которые можно выполнять параллельно без конфликтов.

Пример задачи: Определить максимальное независимое множество задач, чтобы минимизировать конфликты на серверной ферме и оптимизировать использование ресурсов.

Глава 2: Минимальное покрытие рёбрами

Затем Сергей приступил к изучению задачи минимального покрытия рёбрами. Он понял, что для эффективного распределения задач нужно минимизировать количество ресурсов, необходимых для выполнения всех задач.

Пример задачи: Найти минимальное количество ресурсов, необходимых для выполнения всех задач на серверной ферме, используя минимальное покрытие рёбрами.

Глава 3: Минимальное остовное дерево

Сергей также изучил задачу минимального остовного дерева. Он использовал эту концепцию для оптимизации сетевой структуры, чтобы минимизировать затраты на подключение всех узлов сети.

Пример задачи: Построить минимальное остовное дерево для сети серверной фермы, чтобы минимизировать затраты на подключение всех серверов.