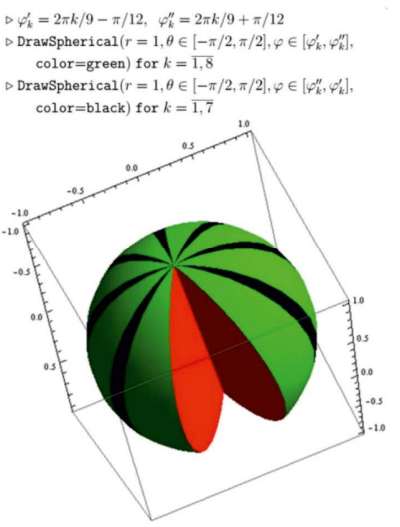
Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Владивостокский государственный университет»

ISBN 978-5-9736-0766-1

Г.В. Гренкин, О.И. Клочкова

ДИСКРЕТНАЯ МАТЕМАТИКА В АЛГОРИТМАХ

Электронное
учебное пособие



Владивосток
Издательство ВВГУ
2025

Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Владивостокский государственный университет»

Г.В. Гренкин
О.И. Клочкова

ДИСКРЕТНАЯ МАТЕМАТИКА В АЛГОРИТМАХ

Электронное учебное пособие

Рекомендовано решением учебно-методической комиссии ФГБОУ ВО «Владивостокского государственного университета»

Владивосток
Издательство ВВГУ
2025

УДК 519.1
ББК 22.176
Г80

Рецензенты:
Е.Д. Емцева, канд. физ.-мат. наук, доцент департамента математики ДВФУ;
А.А. Степанова, д-р физ.-мат. наук, профессор департамента математики ДВФУ

Гренкин Глеб Владимирович

Г80 **Дискретная математика в алгоритмах :** электронное учебное пособие / Г.В. Гренкин, О.И. Клочкова ; Владивостокский государственный университет ; Электрон. текст. дан. (1 файл: 1,76 МБ). – Владивосток: Изд-во ВВГУ, 2025. – 1 электрон., опт. диск (CD-ROM). – Систем. требования: Intel Pentium (или аналогичный процессор других производителей), 500 МГц; 512 Мб оперативной памяти; видеокарта SVGA, 1280×1024 High Color (32 bit); 5 Мб свободного дискового пространства; операц. система Windows XP и выше; Acrobat Reader, Foxit Reader либо любой другой их аналог.

ISBN 978-5-9736-0766-1

Содержит теоретический материал, задачи и лабораторные работы по курсу дискретной математики, а также разбор олимпиадных задач по информатике.

Для студентов, обучающихся по направлениям подготовки в области информатики и вычислительной техники.

УДК 519.1
ББК 22.176

Электронное учебное издание

Минимальные системные требования:
Компьютер: Pentium 3 и выше, 500 МГц; 512 Мб на жестком диске; видеокарта SVGA, 1280×1024 High Color (32 bit); привод CD-ROM. Операционная система: Windows XP/7/8. Программное обеспечение: Internet Explorer 8 и выше или другой браузер; Acrobat Reader, Foxit Reader либо любой другой их аналог.

ISBN 978-5-9736-0766-1

Редактор И.Г. Шабунина
Компьютерная верстка М.А. Портновой

690014, г. Владивосток, ул. Гоголя, 41 Объем 1,76 МБ. Усл.-печ. л. 5,0.
Подписано к использованию 15.10.2025 г. Уч.-изд. л. 3,0. Тираж 100 (I –50) экз.
Телефон: (423)240-40-54

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

В современной прикладной математике и информатике имеется огромный раздел, посвященный алгоритмам и структурам данных. Алгоритмические задачи пользуются популярностью у студентов и школьников, изучающих программирование. Существуют сайты для тренировок, на которых регулярно проходят соревнования по решению задач в режиме реального времени.

Решая задачи на тренировочных сайтах, можно многому научиться. Однако в реальной жизни недостаточно иметь здоровую смекалку. Нужны навыки и опыт, подкрепленные теоретическими знаниями в области дискретной математики. Без этого успешный олимпиадник рискует остаться «бегуном на короткие дистанции».

Данное пособие содержит лабораторный практикум по дискретной математике, снабженный разборами задач олимпиадного типа. В этих задачах требуется реализовать вычислительную процедуру по заданной спецификации. Показано, что при решении задач можно и нужно применять понятия и подходы, которые заложены в курсе дискретной математики.

Главной целью пособия не является подготовка студентов к олимпиадам по программированию, но оно могло бы в числе прочего этому поспособствовать.

Глава 1. МЕТОД МАТЕМАТИЧЕСКОЙ ИНДУКЦИИ

1.1. Теория

При построении алгоритмов возникает необходимость проверить выполнение некоторого свойства $P(n)$ на каждой итерации цикла, т.е. при любом натуральном n.

Метод математической индукции проверяет свойство $P(n)$ для $n = k + 1$, основываясь на *гипотезе*, что это свойство для $n = k$ уже доказано. При этом должно выполняться $P(1)$ – *база индукции*.

Данная схема рассуждений выражается логической формулой

$$P(1) \wedge \forall k\,(P(k) \rightarrow P(k+1)) \rightarrow \forall n\,P(n),$$

где переменные n, k принимают значения из множества натуральных чисел.

Пример 1. Алгоритм Евклида позволяет найти наибольший общий делитель двух натуральных чисел. Алгоритм основан на следующем логическом инварианте: НОД чисел, находящихся в переменных a и b, совпадает с НОД чисел, поданных алгоритму на вход:

```
# Вход: a, b > 0
# Выход: НОД(a, b)
def gcd(a, b):
   while a > 0 and b > 0:
#     НОД(a', b') = НОД(a, b)
      if a >= b:
         a %= b
      else:
         b %= a
#  НОД(x, 0) = x
   return a + b
```

Пример 2. Как разменять любую сумму денег, большую 6 рублей, только двухрублевыми и пятирублевыми монетами?

Применим *возвратную индукцию*, в которой утверждение предполагается доказанным не только для $n = k$, но и для всех $n \le k$.

База индукции: $n = 7 = 1 \cdot 2 + 1 \cdot 5$; $n = 8 = 4 \cdot 2$.

Гипотеза индукции: утверждение доказано для $n = 7, 8, ..., k - 1$, где $k \ge 9$.

Шаг индукции: докажем утверждение для $n = k$. Поскольку для $n = k - 2$ утверждение уже доказано, то $k - 2$ рублей можно разменять монетами по 2 и 5 рублей. Значит, добавив еще одну двухрублевую монету, получим k рублей, что и требовалось.

Пример 3. Найдем в упорядоченном числовом массиве наибольший элемент, не превосходящий заданного числа.

Вначале дадим четкую логическую формулировку задачи. Даны последовательность чисел $a_1, a_2, ..., a_n$ и число x. Выполняется условие $a_1 \le a_2 \le ... \le a_n$. Найти наибольший среди индексов i от 1 до n, для которых выполняется условие $a_i \le x$, либо определить, что таких индексов нет.

Для эффективного решения применим *двоичный поиск*, или метод деления пополам. Введем параметры l и r – границы рассматриваемого фрагмента массива. Будем поддерживать логический инвариант: слева от индекса l находятся числа, меньшие либо равные x, а справа от индекса r находятся числа, большие x:

$$\forall i((i < l \rightarrow a_i \le x) \wedge (i > r \rightarrow a_i > x)).$$

Инициализация. Первоначально (при $l = 1$, $r = n$) инвариант выполнен.

Шаг индукции. Докажем, что если на текущем шаге инвариант выполнен, то после очередной итерации инвариант сохранится.

Выберем на интервале $l..r$ средний элемент с индексом $m = \left\lfloor \frac{l+r}{2} \right\rfloor$ и сравним его с x. Если $a_m \le x$, то после присваивания `l = m + 1` инвариант сохранится. Если $a_m > x$, то после присваивания `r = m - 1` инвариант сохранится.

После окончания итераций будет выполнено $l = r + 1$. Возможен случай $r = 0$. Тогда искомого индекса не существует. В противном случае искомый индекс равен r.

После каждой итерации цикла интервал уменьшается примерно вдвое, поэтому время выполнения алгоритма составит $O(\log n)$.

Упражнение. Проверьте следующую реализацию на собственном наборе тестов:

```
# Вход: a - упорядоченный по возрастанию список,
#    x - число
# Выход: индекс наибольшего элемента i:
#    a[i] <= x или -1, если такого индекса нет
def upper_bound(a, x):
    n = len(a)
    l = 0
    r = n - 1
    while l <= r:
       m = (l + r) // 2
       if a[m] <= x:
           l = m + 1
       else:
           r = m - 1
    return r
```

1.2. Задачи

Задача 1. В компании из n человек ($n > 3$) каждый узнал по одному новому анекдоту (все анекдоты разные). За один телефонный разговор двое сообщают друг другу все известные им анекдоты. Докажите, что за $2n - 4$ звонков все смогут узнать все новые анекдоты.

Указание. Допустим, что для $n = k$ утверждение доказано. Укажите, как воспользоваться этой гипотезой, построив рекурсивный алгоритм. Рекурсия сводит задачу к аналогичной задаче, в которой n на единицу меньше.

Задача 2. Печенья продаются пачками по 6 и 13 штук. Докажите, что Вы всегда можете купить некоторое количество пачек первого и второго типов так, что в сумме Вы получите заданное число, большее или равное 60.

Указание. Рассмотрите начальные количества печеньев в качестве базы индукции. Затем укажите, как выбрать количество пачек с n печеньями, если для любого меньшего числа печеньев задача уже решена.

Задача*. Даны n монет одинакового достоинства, среди которых имеется одна фальшивая, отличающаяся весом. Доказать, что если $n \le 3^k$, то достаточно k взвешиваний на чашечных весах, чтобы обнаружить фальшивую монету.

Указание. Рассмотрите разные случаи, изобразив дерево рекурсии.

1.3. Лабораторные работы

Задача 1. Решите задачу про анекдоты с помощью рекурсивного алгоритма.

Заготовка для Вашего решения:

```
# Вход: количество человек n (n > 3)
# Выход: список пар (2-кортежей)
#   номеров людей (от 1 до n):
#   кто с кем разговаривает,
#   звонков должно быть не более 2n-4
def sol_anecdotes(n):
    # место для Вашего решения
    if n == 4:
        return [(1, 2), (3, 4), (1, 3), (2, 4)]
    else:
        return []
```

Ваше решение должно содержать поясняющие комментарии к исходному коду, а также Ваш собственный набор тестов. Для тестирования своего решения используйте автоматический тестер.

Задача 2. Решите задачу про печенья с помощью итеративного либо рекурсивного алгоритма.

Заготовка для Вашего решения:

```
# Вход: количество печеньев n (n >= 60)
# Выход: двухэлементный кортеж (a, b)
# a >= 0, b >= 0, 6a + 13b = n
# (a - количество пачек по 6 печеньев,
# b - количество пачек по 13 печеньев)
def sol_cookies(n):
     # место для Вашего решения
     # (решением должен быть алгоритм,
     # работающий на любом возможном тесте)
     if n == 60:
```

```
            return 10, 0
      elif n == 61:
            return 8, 1
      else:
            return 0, 0
```

Ваше решение должно содержать поясняющие комментарии к исходному коду, а также Ваш собственный набор тестов. Для тестирования своего решения используйте автоматический тестер.

Задача 3. Решите задачу про монеты.
Заготовка для Вашего решения:

```
# Вход: количество монет n (n >= 1)
# Программа должна за k взвешиваний
# на чашечных весах
# (вызывая функцию weigh с указанием множеств
# номеров взвешиваемых монет)
# обнаружить фальшивую монету,
# где k <= ceil(log_3(n)),
# ceil - округление вверх,
# log_3 - логарифм по основанию 3
# Выход: номер фальшивой монеты
def sol_coins(n):
    # место для Вашего решения
    if n == 1:
          return 1
    elif n == 2:
    w = weigh({1}, {2})
    if w == 1:
          return 1
    else:
          return 2
    elif n == 3:
    w = weigh({1}, {2})
    if w == 1:
          return 1
    elif w == 2:
          return 2
    else:
          return 3
```

```
    else:
        return 1
# coins1, coins2 – два множества монет
# (индексы от 1 до n)
# выход: 1, если первая чашка перевешивает,
# 2, если вторая чашка перевешивает,
# 0, если чашки уравновешивают друг друга
def weigh(coins1, coins2):
    global num_weigh
    num_weigh += 1
    if len(coins1) > len(coins2):
        return 1
    elif len(coins1) < len(coins2):
        return 2
    else:
    global correct_ans
    if correct_ans in coins1:
        return 1
    elif correct_ans in coins2:
        return 2
    else:
        return 0
```

Ваше решение должно содержать поясняющие комментарии к исходному коду, а также Ваш собственный набор тестов. Для тестирования своего решения используйте автоматический тестер.

Глава 2. БУЛЕВЫ ФУНКЦИИ

2.1. Теория

Булева переменная – это переменная, которая принимает одно из двух значений – 0, 1.

Булевой функцией от n переменных $x_1, x_2,..., x_n$ называют правило, по которому каждому набору значений булевых переменных $x_1, x_2,..., x_n$ ставится в соответствие значение 0 или 1.

В таблице 2.1 представлены все булевы функции от двух переменных x, y.

Таблица 2.1

Булевы функции от двух переменных

x	y	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Каждая булева функция имеет формульное выражение.

$f_1(x, y) = 0$ – тождественный ноль;

$f_2(x, y) = x \wedge y$ – конъюнкция (логическое И; умножение);

$f_3(x, y) = \neg(x \to y)$ – отрицание импликации;

$f_4(x, y) = x$ – первый аргумент;

$f_5(x, y) = \neg(y \to x)$ – отрицание обратной импликации;

$f_6(x, y) = y$ – второй аргумент;

$f_7(x, y) = x \oplus y$ – сложение по модулю 2 (исключающее ИЛИ);

$f_8(x, y) = x \vee y$ – дизъюнкция (логическое ИЛИ; сложение);

$f_9(x, y) = x \downarrow y$ – стрелка Пирса (отрицание дизъюнкции);

$f_{10}(x, y) = x \leftrightarrow y$ – эквиваленция;

$f_{11}(x, y) = \neg y$ – отрицание второго аргумента;

$f_{12}(x, y) = y \to x$ – обратная импликация;

$f_{13}(x, y) = \neg x$ – отрицание первого аргумента;

$f_{14}(x, y) = x \to y$ – импликация;

$f_{14}(x, y) = x \mid y$ – штрих Шеффера (отрицание конъюнкции);

$f_{16}(x, y) = 1$ – тождественная единица.

СДНФ – это дизъюнкция (неповторяющихся) конъюнкций всех переменных, некоторые из которых берутся с отрицанием. СКНФ – это конъюнкция (неповторяющихся) дизъюнкций всех переменных, некоторые из которых берутся с отрицанием.

Для всякой булевой функции, не равной тождественно 0, существует представление в совершенной дизъюнктивной нормальной форме (СДНФ). Для всякой булевой функции, не равной тождественно 1, существует представление в совершенной конъюнктивной нормальной форме (СКНФ).

СДНФ и СКНФ удобно строить с помощью *минтермов* и *макстермов* булевой функции. Минтерм – это элементарная конъюнкция, равная 1 ровно при одном наборе значений переменных. Макстерм – это элементарная дизъюнкция, равная 0 ровно при одном наборе значений переменных. В случае трех переменных X_1, X_2, X_3 и заданных значений функции Y минтермы m_i и макстермы h_i построены в таблице:

X_1	X_2	X_3	Y	Минтерм m_i	Макстерм h_i
0	0	0	0	$\overline{X_1} \wedge \overline{X_2} \wedge \overline{X_3}$	$X_1 \vee X_2 \vee X_3$
0	0	1	0	$\overline{X_1} \wedge \overline{X_2} \wedge X_3$	$X_1 \vee X_2 \vee \overline{X_3}$
0	1	0	0	$\overline{X_1} \wedge X_2 \wedge \overline{X_3}$	$X_1 \vee \overline{X_2} \vee X_3$
0	1	1	0	$\overline{X_1} \wedge X_2 \wedge X_3$	$X_1 \vee \overline{X_2} \vee \overline{X_3}$
1	0	0	0	$X_1 \wedge \overline{X_2} \wedge \overline{X_3}$	$\overline{X_1} \vee X_2 \vee X_3$
1	0	1	1	$X_1 \wedge \overline{X_2} \wedge X_3$	$\overline{X_1} \vee X_2 \vee \overline{X_3}$
1	1	0	1	$X_1 \wedge X_2 \wedge \overline{X_3}$	$\overline{X_1} \vee \overline{X_2} \vee X_3$
1	1	1	1	$X_1 \wedge X_2 \wedge X_3$	$\overline{X_1} \vee \overline{X_2} \vee \overline{X_3}$

Совершенная дизъюнктивная нормальная форма (СДНФ) определяется формулой

$$Y = \bigvee_{i=1}^{k} (Y_i \wedge m_i),$$

в которой Y_i – значение булевой функции в i-й строке таблицы истинности, m_i – минтерм i-й строки.

По данным таблицы функция принимает вид

$$Y = (X_1 \wedge \overline{X_2} \wedge X_3) \vee (X_1 \wedge X_2 \wedge \overline{X_3}) \vee (X_1 \wedge X_2 \wedge X_3).$$

Формула совершенной конъюнктивной нормальной формы (СКНФ) – произведение макстермов, где функция равна 0:

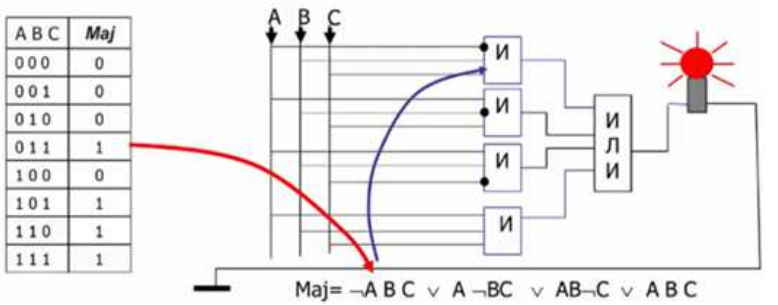
$$Y = \bigwedge_{i=1}^{k} (Y_i \vee h_i).$$

В примере

$$Y = (X_1 \vee X_2 \vee X_3) \wedge (X_1 \vee X_2 \vee \overline{X_3}) \wedge (X_1 \vee \overline{X_2} \vee X_3) \wedge (X_1 \vee \overline{X_2} \vee \overline{X_3}) \wedge (\overline{X_1} \vee X_2 \vee X_3).$$

Пример 1. Применение алгоритма представления мажоритарной функции в виде СДНФ. Мажоритарная функция используется в типичных спортивных задачах, когда, например, в боксерском поединке удар засчитывается, если хотя бы двое судей его засчитают и нажмут свою кнопку.

На рисунке 2.1 приведена таблица истинности функции и разложение ее в форму СДНФ по минтермам, построенным по строкам таблицы истинности, где функция равна 1, а также показана функциональная схема этой булевой функции.



A B C	*Maj*
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Рис. 2.1

Пример 2. Запишем СДНФ и СКНФ для функции $f_{10}(x,y) = x \longleftrightarrow y$.

Выберем все наборы значений переменных, при которых функция принимает значение 1. Для каждого такого набора запишем конъюнкцию, равную 1 только при этих значениях переменных:

$$x = 0, y = 0 \Longleftrightarrow \neg x \wedge \neg y = 1,$$

$$x = 1, y = 1 \Longleftrightarrow x \wedge y = 1.$$

Затем применим операцию дизъюнкции к полученным конъюнкциям. Получим СДНФ:

$$f_{10}(x,y) = (\neg x \wedge \neg y) \vee (x \wedge y).$$

Выберем все наборы значений переменных, при которых функция принимает значение 0. Для каждого такого набора запишем дизъюнкцию, равную 0 только при этих значениях переменных:

$$x = 0, y = 1 \Longleftrightarrow x \vee \neg y = 0,$$

$$x = 1, y = 0 \Longleftrightarrow \neg x \vee y = 0.$$

Затем применим операцию конъюнкции к полученным дизъюнкциям. Получим СКНФ:

$$f_{10}(x,y) = (x \vee \neg y) \wedge (\neg x \vee y).$$

Формулы называются *равносильными*, если им соответствует одна и та же булева функция.

Основные равносильности (логические законы):

1) $x \wedge 0 = 0$, $x \wedge 1 = x$, $x \vee 0 = x$, $x \vee 1 = 1$ (свойства относительно констант);
2) $x \wedge x = x$, $x \vee x = x$ (законы приведения подобных);
3) $x \wedge \neg x = 0$ (закон противоречия);
4) $x \vee \neg x = 1$ (закон исключенного третьего);
5) $\neg\neg x = x$ (закон двойного отрицания);
6) $x \to y = \neg x \vee y$ (формула замены импликации);
7) $x \wedge y = y \wedge x$, $x \vee y = y \vee x$ (законы коммутативности);
8) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$, $x \vee (y \vee z) = (x \vee y) \vee z$ (законы ассоциативности);

9) $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ (законы дистрибутивности);

10) $\neg(x \wedge y) = \neg x \vee \neg y$, $\neg(x \vee y) = \neg x \wedge \neg y$ (законы де Моргана);

11) $x \vee (x \wedge y) = x$, $x \wedge (x \vee y) = x$ (поглощение);

12) $(x \wedge y) \vee (x \wedge \neg y) = x$, $(x \vee y) \wedge (x \vee \neg y) = x$ (склеивание).

ДНФ (дизъюнктивная нормальная форма) – это дизъюнкция конъюнкций некоторых переменных и переменных, взятых с отрицанием.

Пример 3. Приведем формулу $((x \longleftrightarrow \neg y) \vee \neg z) \rightarrow y$ к ДНФ.

Применим формулу замены эквиваленции (из предыдущего примера):

$$x \longleftrightarrow \neg y = (x \vee y) \wedge (\neg x \vee \neg y).$$

Отсюда следует, что

$$((x \longleftrightarrow \neg y) \vee \neg z) \rightarrow y = (((x \vee y) \wedge (\neg x \vee \neg y)) \vee \neg z) \rightarrow y.$$

Применяя формулу замены импликации, получим

$$((x \longleftrightarrow \neg y) \vee \neg z) \rightarrow y = \neg(((x \vee y) \wedge (\neg x \vee \neg y)) \vee \neg z) \vee y.$$

Преобразуя отрицание дизъюнкции по закону де Моргана, получим равносильную формулу

$$(\neg((x \vee y) \wedge (\neg x \vee \neg y)) \wedge \neg\neg z) \vee y.$$

Применяя закон де Моргана к отрицанию конъюнкции и закон двойного отрицания, получим

$$((\neg(x \vee y) \vee \neg(\neg x \vee \neg y)) \wedge z) \vee y = (((\neg x \wedge \neg y) \vee (x \wedge y)) \wedge z) \vee y.$$

Согласно закону дистрибутивности, имеем

$$(((\neg x \wedge \neg y) \vee (x \wedge y)) \wedge z) \vee y = (\neg x \wedge \neg y \wedge z) \vee (x \wedge y \wedge z) \vee y.$$

Полученная формула представляет собой ДНФ – дизъюнкцию элементарных конъюнкций.

Сократим ДНФ. По закону поглощения исходная формула равносильна формуле

$$(\neg x \wedge \neg y \wedge z) \vee y.$$

Согласно второму закону дистрибутивности, имеем

$$(\neg x \wedge \neg y \wedge z) \vee y = ((\neg x \wedge z) \wedge \neg y) \vee y = ((\neg x \wedge z) \vee y) \wedge (\neg y \vee y)$$
$$== ((\neg x \wedge z) \vee y) \wedge 1 = (\neg x \wedge z) \vee y.$$

Пример 4. Построим сокращенную ДНФ для булевой функции, заданной следующей таблицей истинности:

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

Составим СДНФ по таблице истинности:

$$f(x,y,z) = (\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee\vee (x \wedge y \wedge \neg z) \vee (x \wedge y \wedge z).$$

Добавим дизъюнктивные слагаемые, полученные в результате всевозможных склеиваний: склеим 1-е и 2-е, 1-е и 3-е, 3-е и 4-е, 4-е и 5-е слагаемые. Например:

$$(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) = ((\neg x \wedge \neg y) \wedge \neg z) \vee ((\neg x \wedge \neg y) \wedge z) = \neg x \wedge \neg y.$$

Получим

$$f(x,y,z) = (\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee\vee (x \wedge y \wedge \neg z) \vee (x \wedge y \wedge z) \vee (\neg x \wedge \neg y) \vee (\neg y \wedge \neg z) \vee\vee (x \wedge \neg z) \vee (x \wedge y),$$

а затем произведем всевозможные поглощения:

$$f(x,y,z) = (\neg x \wedge \neg y) \vee (\neg y \wedge \neg z) \vee (x \wedge \neg z) \vee (x \wedge y).$$

Получили *сокращенную ДНФ*. Чтобы сократить ее до *минимальной ДНФ*, составим *таблицу Квайна*. По горизонтали перечислим все составляющие СДНФ, а по вертикали – составляющие сокращенной ДНФ. На пересечении составляющей СДНФ и составляющей ДНФ запишем отметку «плюс», если первая содержит вторую:

	$\neg x \wedge \neg y \wedge \neg z$	$\neg x \wedge \neg y \wedge z$	$x \wedge \neg y \wedge \neg z$	$x \wedge y \wedge \neg z$	$x \wedge y \wedge z$
$\neg x \wedge \neg y$	+	+			
$\neg y \wedge \neg z$	+		+		
$x \wedge \neg z$			+	+	
$x \wedge y$				+	+

Выберем минимальное число строк, плюсы в которых покрывают все столбцы. Получим два варианта минимальных ДНФ:

$$f(x,y,z) = (\neg x \wedge \neg y) \vee (\neg y \wedge \neg z) \vee (x \wedge y)$$

и

$$f(x,y,z) = (\neg x \wedge \neg y) \vee (x \wedge \neg z) \vee (x \wedge y).$$

Отметим, таким образом, что любую булеву функцию можно выразить через отрицание $\neg$, конъюнкцию $\wedge$ и дизъюнкцию $\vee$.

Упражнение. Покажите, что для представления произвольной булевой функции можно использовать всего две операции: отрицание $\neg$ и импликацию $\rightarrow$.

Покажем, что любую булеву функцию можно выразить через три булевы функции: тождественную 1, конъюнкцию и сложение по модулю 2. Указанное представление булевой функции носит название *многочлена Жегалкина*.

Пример 5. Приведем булеву функцию из предыдущего примера к многочлену Жегалкина.

Для этого рассмотрим СДНФ:

$$f(x,y,z) = (\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (x \wedge \neg y \wedge \neg z) \vee\vee (x \wedge y \wedge \neg z) \vee (x \wedge y \wedge z).$$

Поскольку все дизъюнктивные слагаемые не обращаются одновременно в 1 (взаимно исключают друг друга), то дизъюнкцию можно заменить на сложение по модулю 2:

$$f(x,y,z) = (\neg x \wedge \neg y \wedge \neg z) \oplus (\neg x \wedge \neg y \wedge z) \oplus (x \wedge \neg y \wedge \neg z) \oplus\oplus (x \wedge y \wedge \neg z) \oplus (x \wedge y \wedge z).$$

Выразим отрицание через сложение по модулю 2 и тождественную единицу: $\neg x = x \oplus 1$.

Тогда

$$f(x,y,z) = (x \oplus 1)(y \oplus 1)(z \oplus 1) \oplus (x \oplus 1)(y \oplus 1)z \oplus\oplus x(y \oplus 1)(z \oplus 1) \oplus xy(z \oplus 1) \oplus xyz.$$

Раскроем скобки, пользуясь законом дистрибутивности:

$$x \wedge (y \oplus z) = (x \wedge y) \oplus (x \wedge z).$$

Получим

$$f(x,y,z) = (xyz \oplus xy \oplus xz \oplus x \oplus yz \oplus y \oplus z \oplus 1) \oplus\oplus (xyz \oplus xz \oplus yz \oplus z) \oplus (xyz \oplus xy \oplus xz \oplus x) \oplus\oplus (xyz \oplus xy) \oplus xyz.$$

Приведем подобные слагаемые по формуле $x \oplus x = 0$. Будем иметь

$$f(x,y,z) = xyz \oplus xy \oplus xz \oplus y \oplus 1.$$

Полученное выражение – многочлен Жегалкина.

Пример 6. Приведем булеву функцию, заданную своей таблицей истинности, к многочлену Жегалкина при помощи метода неопределенных коэффициентов:

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

Запишем многочлен Жегалкина в наиболее общем виде:

$$f(x,y,z) = a_0 \oplus a_1x \oplus a_2y \oplus a_3z \oplus a_{12}xy \oplus a_{23}yz \oplus a_{13}xz \oplus\oplus a_{123}xyz.$$

Подставим вместо *x*, *y*, *z* всевозможные наборы значений переменных и составим 8 уравнений относительно коэффициентов $a_0, a_1, a_2, a_3, a_{12}, a_{23}, a_{13}, a_{123}$:

$f(0,0,0) = a_0 = 0,$
$f(0,0,1) = a_0 \oplus a_3 = 1,$
$f(0,1,0) = a_0 \oplus a_2 = 1,$
$f(0,1,1) = a_0 \oplus a_2 \oplus a_3 \oplus a_{23} = 1,$
$f(1,0,0) = a_0 \oplus a_1 = 0,$
$f(1,0,1) = a_0 \oplus a_1 \oplus a_3 \oplus a_{13} = 0,$
$f(1,1,0) = a_0 \oplus a_1 \oplus a_2 \oplus a_{12} = 1,$
$f(1,1,1) = a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_{12} \oplus a_{23} \oplus a_{13} \oplus a_{123} = 1.$

Решаем систему:

$a_0 = 0,$
$0 \oplus a_3 = 1 \Rightarrow a_3 = 1,$
$0 \oplus a_2 = 1 \Rightarrow a_2 = 1,$
$0 \oplus 1 \oplus 1 \oplus a_{23} = 1 \Rightarrow a_{23} = 1,$
$0 \oplus a_1 = 0 \Rightarrow a_1 = 0,$
$0 \oplus 0 \oplus 1 \oplus a_{13} = 0 \Rightarrow a_{13} = 1,$
$0 \oplus 0 \oplus 1 \oplus a_{12} = 1 \Rightarrow a_{12} = 0,$
$0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 1 \oplus a_{123} = 1 \Rightarrow a_{123} = 1.$

Таким образом,

$$f(x,y,z) = y \oplus z \oplus yz \oplus xz \oplus xyz.$$

Это многочлен Жегалкина.

Пример 7. Построим многочлен Жегалкина для следующей функции методом треугольника:

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

В первой строке перепишем всю таблицу истинности. Следующую строку получим так: под двумя соседними цифрами запишем их сумму по модулю 2.

Многочлен Жегалкина составляется из тех слагаемых, в чьих строках в треугольнике первый символ – это единица.

В нашем случае это строки (0, 1, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0). Им соответствуют одночлены y, yz, xz, xy. Поэтому многочлен Жегалкина – это $xz \oplus xy \oplus yz \oplus y$.

Любая булева функция представима в виде многочлена Жегалкина, и притом единственным образом. Для доказательства этого факта пересчитаем все булевы функции от *n* переменных: их $2^{(2^n)}$, потому что булева функция задается столбцом своей таблицы истинности из 2^n двоичных значений. Пересчитаем все многочлены Жегалкина: так как многочлен Жегалкина содержит 2^n возможных слагаемых, причем некоторые из них присутствуют, а остальные отсутствуют, то он также задается упорядоченным набором из 2^n двоичных значений. Следовательно, различных булевых функций столько же, сколько различных многочленов Жегалкина. Учитывая, что разным булевым функциям соответствуют разные многочлены Жегалкина, приходим к такому выводу: если бы какая-нибудь булева функция представлялась двумя разными многочленами, тогда существовала бы другая булева функция, не представимая в виде многочлена. В силу существования алгоритма построения многочлена для любой булевой функции это невозможно. Значит, многочлен Жегалкина для всякой булевой функции существует и единствен.

Поставим общую задачу: для произвольной системы булевых функций выяснить, обладает ли она свойством *полноты*, т.е. любую ли булеву функцию можно представить в виде суперпозиции функций этой системы?

Для ответа на этот вопрос нам понадобится ввести *основные классы булевых функций*:

1. T_0 – класс функций, сохраняющих 0.

К этому классу относят функции $f(x_1,x_2,...,x_n)$, для которых $f(0,0,...,0) = 0$.

2. T_1 – класс функций, сохраняющих 1.

Сюда относят функции $f(x_1,x_2,...,x_n)$, для которых $f(1,1,...,1) = 1$.

3. S – класс самодвойственных функций.

Функция *самодвойственна*, если на противоположных наборах ее значения противоположны: $\neg f(\neg x_1,\neg x_2,...,\neg x_n) = f(x_1,x_2,...,x_n)$.

4. L – класс линейных функций.

Функция *линейна*, если ее многочлен Жегалкина не содержит конъюнкций переменных.

5. M – класс монотонных функций.

Функция *монотонна*, если при увеличении значений переменных значение функции увеличивается:

$$(\alpha_1 \le \beta_1) \wedge ... \wedge (\alpha_n \le \beta_n) \to (f(\alpha_1,...,\alpha_n) \le f(\beta_1,...,\beta_n)).$$

Теорема Поста. Система булевых функций полна тогда и только тогда, когда для каждого из классов T_0, T_1, S, L, M в этой системе найдется хотя бы одна функция, которая этому классу не принадлежит.

В таблице 2.2 показана принадлежность некоторых булевых функций основным классам.

Таблица 2.2

Принадлежность булевых функций основным классам

	0	1	x	¬	∧	∨	→	↔	⊕
T_0	+	–	+	–	+	+	–	–	+
T_1	–	+	+	–	+	+	+	+	–
S	–	–	+	+	–	–	–	–	–
L	+	+	+	+	–	–	–	+	+
M	+	+	+	–	+	+	–	–	–

Пример 8. Является ли полной система функций $\{x \to y, \neg x \wedge y\}$?

Составим таблицу, в которой укажем принадлежность функций этой системы основным классам:

	$x \to y$	$\neg x \wedge y$
T_0	–	+
T_1	+	–
S	–	–
L	–	–
M	–	–

Принадлежность импликации основным классам берем из табл. 2.2. Функция $\neg x \wedge y$ сохраняет константу 0 (поскольку $\neg 0 \wedge 0 = 0$), не сохраняет константу 1 (поскольку $\neg 1 \wedge 1 = 0$), не является самодвойственной (поскольку двойственной к $\neg x \wedge y$ является другая функция $\neg x \vee y$), не является линейной (поскольку $\neg x \wedge y = (x \oplus 1)y = xy \oplus y$), не является монотонной (поскольку $\neg 0 \wedge 1 = 1$ и $\neg 1 \wedge 1 = 0$).

Из таблицы видно, что в каждой строке присутствует хотя бы один «минус». Значит, по теореме Поста данная система булевых функций является полной.

2.2. Задачи

Задание 1. Приведите данную булеву функцию к СДНФ, а затем к сокращенной ДНФ. Постройте многочлен Жегалкина.

1. $f(0,0,1) = f(0,1,1) = f(1,1,0) = 0$
2. $f(0,0,0) = f(0,0,1) = f(1,0,0) = f(1,1,0) = 0$
3. $f(0,1,1) = f(1,0,0) = f(1,0,1) = 1$
4. $f(0,0,1) = f(1,0,0) = f(1,1,0) = 1$
5. $f(0,1,1) = f(0,1,0) = f(1,0,0) = f(1,0,1) = 0$
6. $f(0,1,1) = f(0,1,0) = f(1,0,1) = f(1,1,1) = 1$
7. $f(0,1,0) = f(1,0,0) = f(1,0,1) = 0$
8. $f(0,1,1) = f(1,0,0) = f(1,1,0) = 0$
9. $f(0,0,0) = f(0,0,1) = f(1,0,1) = f(1,1,1) = 1$
10. $f(0,0,1) = f(1,1,1) = f(1,1,0) = 0$

Задание 2. Является ли полной система булевых функций?

1. $\{\neg x \to y, x \wedge \neg y\}$
2. $\{\neg x \downarrow y, \neg x \leftrightarrow \neg y\}$
3. $\{\neg x \oplus \neg y, \neg x \vee y\}$
4. $\{\neg x \wedge \neg y, \neg x \to y\}$
5. $\{\neg x \downarrow \neg y, x \leftrightarrow y\}$
6. $\{x \vee \neg y, \neg x \leftrightarrow y\}$
7. $\{x \vee y, \neg x \oplus y\}$
8. $\{x \to y, \neg x \wedge \neg y\}$
9. $\{x \leftrightarrow y, \neg x | \neg y\}$
10. $\{x \oplus y, \neg x \vee y\}$

2.3. Лабораторные работы

Задача 1. Вам дана программа, в которую Вы можете ввести свою булеву функцию. Программа построит ее таблицу истинности и выведет СДНФ. Возьмите булеву функцию из своего варианта, постройте с помощью программы СДНФ. Сократите СДНФ. Введите полученную сокращенную ДНФ в программу и проверьте, что для новой формулы СДНФ и таблица истинности остались прежними.

Задача 2. Вам дана заготовка программы для построения многочлена Жегалкина по таблице истинности методом треугольника. Допишите заготовку. Постройте с помощью своей программы многочлен Жегалкина для булевой функции из своего варианта.

Задача 3. Вам дана программа, проверяющая для заданной булевой функции ее принадлежность к основным замкнутым классам. Напишите программу, проверяющую полноту введенной системы функций по критерию Поста. Примените свою программу для решения своего варианта.

Глава 3. МНОЖЕСТВА, ОТНОШЕНИЯ И ФУНКЦИИ

3.1. Теория

Под *множеством* интуитивно можно понимать совокупность различных объектов, которые объединены по какому-либо общему для них признаку.

Примеры множеств: студенты в аудитории, города Приморского края, учебные заведения Владивостока.

Задать множество можно, перечислив все его элементы либо указав характеристическое свойство, которым обладают элементы множества и только они.

Примеры:

$A = \{$Владивосток, Уссурийск, Спасск - Дальний$\}$,

$B = \{x \in A : x$ - на юге края$\}$,

$E = \{n \in Z : (\exists k \in Z : n = 2k)\}$ – множество всех четных чисел.

Если каждый элемент множества *B* является элементом множества *A*, то множество *B* называется *подмножеством* множества *A*, что записывается как $B \subset A$.

Равенство $A = B$ означает, что множества *A* и *B* совпадают, т.е. $A \subset B$ и $B \subset A$.

Под у*ниверсумом* будем понимать множество, такое, что все рассматриваемые множества являются его подмножествами.

Список элементов множества, заключенный в круглые скобки, обозначает перечисление элементов в определенном порядке. Такое перечисление в заданном порядке называется *перестановкой* элементов множества.

Пример:

(a,b,c) и (c, b,a) – различные перестановки элементов множества {a,b,c}.

Для изображения множеств используют *диаграммы Эйлера – Венна*, на которых элементы множества считаются точками плос-

кости, ограниченными окружностью. Тогда отношение включения означает расположение одного круга внутри другого.

Вводят *пустое множество* $\varnothing$ – это множество, не содержащее ни одного элемента. Для любого множества A справедливо утверждение $\varnothing \subset A$.

Если на множестве A с помощью некоторой перестановки $(a_1,...,a_n)$ зафиксирован порядок элементов, то любое подмножество $B \subset A$ можно задать с помощью двоичного набора $(\beta_1,...,\beta_n)$, где $\beta_i = 1$, если $a_i \in B$, и $\beta_i = 0$, если $a_i \notin B$. Такой вектор называется *характеристическим вектором* подмножества B.

Пример:

$A = \{1,2,3,4,5\}$, $B = \{1,3,5\}$.

Тогда характеристический вектор равен (10101).

Таким образом, между подмножествами множества A и двоичными наборами длины $|A|$ существует взаимно однозначное соответствие. Пересчитав число наборов длины $n = |A|$, находим полное число подмножеств множества A – их 2^n. Совокупность всех подмножеств множества A называется его *булеаном* и обозначается $P(A)$.

Пример 1. Перечисление булеана.

Допустим, что множество задано списком – это перестановка элементов множества. Чтобы перечислить все подмножества, переберем все характеристические векторы длины n. Составим рекурсивную процедуру. Она принимает на вход k – количество заполненных элементов двоичного вектора. На k-е место ставим 0 и вызываем ту же процедуру рекурсивно с параметром $k + 1$, а затем 1 и вызываем ту же процедуру рекурсивно. Условие окончания рекурсии: $k = n$.

Рассмотрим операции на булеане $P(U)$. Если $A,B \in P(U)$, то *пересечение* $A \cap B$, *объединение* $A \cup B$ и *разность* $A \setminus B$ определяются равенствами:

$$A \cap B = \{x : x \in A \text{ и } x \in B\},$$
$$A \cup B = \{x : x \in A \text{ или } x \in B\},$$
$$A \setminus B = \{x : x \in A \text{ и } x \notin B\}.$$

Симметрическая разность $A \Delta B$ определяется формулой

$$A \Delta B = (A \setminus B) \cup (B \setminus A).$$

Дополнение множества *A* – это множество $\overline{A} = U \setminus A$.

Справедлива формула замены разности: $A \setminus B = A \cap \overline{B}$.

Основные тождества с множествами аналогичны основным равносильностям булевой алгебры:

1) $A \cup (B \cup C) = (A \cup B) \cup C$, $A \cap (B \cap C) = (A \cap B) \cap C$ (ассоциативность);
2) $A \cap B = B \cap A$, $A \cup B = B \cup A$ (коммутативность);
3) $A \cup A = A$, $A \cap A = A$ (идемпотентность);
4) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$,
   $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ (дистрибутивность);
5) $A \cup (A \cap B) = A$, $A \cap (A \cup B) = A$ (поглощение);
6) $\overline{A \cup B} = \overline{A} \cap \overline{B}$, $\overline{A \cap B} = \overline{A} \cup \overline{B}$ (законы де Моргана);
7) $A \cup \varnothing = A$, $A \cap \varnothing = \varnothing$, $A \cup U = U$, $A \cap U = A$, $A \cup \overline{A} = U$, $A \cap \overline{A} = \varnothing$ (законы нуля и единицы);
8) $\overline{\overline{A}} = A$ (двойное отрицание).

Еще один объект называется *упорядоченным набором* или *кортежем* $(a_1, a_2, \ldots, a_n)$, где $a_i \in A$. В отличие от перестановки элементов множества элементы кортежа могут повторяться. Кортеж длины 2 логично назвать *упорядоченной парой*.

Множество всевозможных упорядоченных пар (a,b), в которых $a \in A$ и $b \in B$, называется *прямым (декартовым) произведением* множеств *A* и *B*; обозначение $A \times B$.

Пример:

$$A = \{a,b,c,d,e,f,g,h\},\ B = \{1,2,3,4,5,6,7,8\}.$$

Тогда множеству пар $(x,y) \in A \times B$ соответствует множество клеток шахматной доски.

Если из декартова произведения выбрать какие-то пары $R \subset A \times B$, то получится *бинарное отношение*, или *соответствие*. Например, множество клеток, куда может пойти ферзь, находящийся в клетке a1, задается множеством пар, состоящим из диагонали, горизонтали и вертикали. Тем самым установлено отношение между множествами A и B.

Аналогичным образом можно установить отношение между множествами $A \times B$ и $A \times B$, если соотнести те клетки шахматной доски, между которыми может сделать ход ладья.

Дадим несколько определений, относящихся к произвольному соответствию $R \subset A \times B$ между множествами A и B.

Область определения отношения R – это множество

$\delta_R = \{x:(x,y) \in R$ для некоторого $y\}$.

Если $\delta_R = A$, то отношение называется *всюду определенным*.

Запишем логическую формулу, выражающую свойство всюду определенности отношения:

$$\forall x \in A\ \exists y \in B:(x,y) \in R.$$

Отношение *R* называется *функциональным*, если каждый элемент $x \in A$ связан отношением не более чем с одним элементом $y \in B$. Иначе говоря,

$$\forall x \in A\ \forall y_1 \in B\ \forall y_2 \in B:\ (x,y_1) \in R \wedge (x,y_2) \in R \rightarrow (y_1 = y_2).$$

Всюду определенное и функциональное соответствие задает *функцию* $f:A \rightarrow B$, т.е. правило, согласно которому всякому элементу $x \in A$ ставится в соответствие элемент $y = f(x) \in B$.

Область значений отношения *R* – это множество

$\rho_R = \{y:(x,y) \in R$ для некоторого $x\}$.

Если $\rho_R = B$, то отношение называется *сюръективным*.

Отношение *R* называется *инъективным*, если каждый элемент $y \in B$ связан отношением не более чем с одним элементом $x \in A$. Иначе говоря,

$$\forall y \in B\ \forall x_1 \in A\ \forall x_2 \in A:\ (x_1,y) \in R \wedge (x_2,y) \in R \rightarrow (x_1 = x_2).$$

Отношение *R биективно* (*взаимно однозначно*), если оно одновременно сюръективно и инъективно.

Рассмотрим два отношения: $R \subset A \times B$ и $S \subset B \times C$. Их *суперпозицией* $R \circ S$ называется отношение, которое связывает такие пары $(x,z) \in A \times C$, для которых существует элемент $y \in B$, такой, что $(x,y) \in R$ и $(y,z) \in S$:

$$(x,z) \in R \circ S \Leftrightarrow \exists y \in B:(x,y) \in R \wedge (y,z) \in S.$$

Дадим еще несколько определений, относящихся к отношениям $R \subset A \times A$.

Свойство *рефлексивности*:

$$\forall x \in A{:}(x,x) \in R.$$

Свойство *симметричности*:

$$\forall x,y \in A{:}(x,y) \in R \rightarrow (y,x) \in R.$$

Свойство *транзитивности*:

$$\forall x,y,z \in A{:}(x,y) \in R \wedge (y,z) \in R \rightarrow (x,z) \in R.$$

Для транзитивного отношения справедливо $R \circ R \subset R$.

Свойство *антисимметричности*:

$$\forall x,y \in A{:}(x,y) \in R \wedge (y,x) \in R \rightarrow (x = y).$$

Пример 2. Проверка антисимметричности отношения.

Допустим, что отношение задано своей булевой матрицей. Чтобы вычислить значение логической формулы с кванторами, запрограммируем циклы по каждой переменной, стоящей под знаком квантора:

```
antisymm = True
for x in range(n):
    for y in range(n):
        if R[x][y] and R[y][x]:
           if x!= y:
              antisymm = False
```

Рефлексивное, симметричное и транзитивное отношение называется *отношением эквивалентности*. Подмножества попарно эквивалентных элементов образуют классы эквивалентности. Пример отношения эквивалентности на множестве треугольников – отношение подобия.

Рефлексивное, антисимметричное и транзитивное отношение называется *отношением порядка*. Пример отношения порядка на множестве точек плоскости: две точки связаны отношением, если обе координаты первой из них не превосходят соответствующих координат второй из них. В заданном множестве точек плоскости может не быть наибольшего элемента, но всегда можно выделить множество максимальных элементов, не сравнимых между собой.

Пример 3. В вооруженных силах рота состоит из трех взводов, а взвод из трех отделений. В каждом отделении по 11 солдат и командир – сержант, взводом командует лейтенант, а ротой – капитан. Определить свойства бинарного отношения R = «быть командиром».

Отношение R:

1) не симметрично, антисимметрично, так как капитан может командовать лейтенантом, а лейтенант капитаном – нет;

2) не рефлексивно, так как капитан не может командовать сам собой;

3) транзитивно, так как капитан может командовать лейтенантом, а лейтенант сержантом, тогда и капитан может командовать сержантом.

Пример 4. В водоеме два пескаря, два карася и одна щука. Зная, что карась и щука – хищные рыбы, так как щука может съесть и пескаря, и карася, а карась может съесть пескаря, построить матрицу бинарного отношения $R = \{(x, y)$: рыба x может быть съедена рыбой $y\}$ с помощью матрицы.

Карась не является хищной рыбой, сама себя не ест ни одна рыба. Матрица бинарного отношения R имеет следующий вид:

	пескарь	пескарь	карась	карась	щука
пескарь	0	0	1	1	1
пескарь	0	0	1	1	1
карась	0	0	0	0	1
карась	0	0	0	0	1
щука	0	0	0	0	0

3.2. Задачи

Задание 1. Составьте формулы алгебры множеств и найдите соответствующие множества.

Пример. Универсум – все студенты Вашей группы. Множества: A – мужчины, $\overline{A}$ – женщины, B – студенты без академических задолженностей, $\overline{B}$ – студенты с задолженностями.

А. Какие существуют возможности составить упорядоченную пару из парня с задолженностями и девушки без задолженностей?

Ответ: все возможности образуют декартово произведение $(A \cap \overline{B}) \times (\overline{A} \cap B)$.

Б. Кто из девушек пойдет в библиотеку? Какие существуют возможности?

Ответ: все возможности образуют булеан множества девушек, т.е. $P(\overline{A})$.

1. Универсум – книги в библиотеке. Множество A – книги по математике, B – книги по информатике, C – книги по физике, D – новые книги, $\overline{D}$ – старые книги. Парень и девушка пришли в библиотеку, и парень решил взять старую книгу по математике или физике, а девушка – новую книгу по информатике. Какие существуют возможности им выбрать себе книги? Пусть E – множество книг, стоящих на полке. Некоторые из них студент читает, остальные – нет. Какие существуют возможности выбрать книги для чтения?

2. Универсум – населенные пункты. Множество A – города России, B – зарубежные города, C – деревни России, D – населенные пункты на востоке, $\overline{D}$ – населенные пункты на западе. Студент хочет поехать на зимние каникулы в какой-нибудь восточный город в России или за рубежом, а на летние каникулы – в деревню на западе. Какие существуют возможности? Пусть E – множество интересующих городов. В некоторых из них студент успел побывать, в остальных – нет. Какие существуют возможности?

3. Универсум – сотрудники предприятия, которые пришли на корпоратив. Множество A – сотрудники бухгалтерии, B – сотрудники отдела продаж, $C = \overline{A \cup B}$ – прочие сотрудники, D – мужчины, $\overline{D}$ – женщины. Какие существуют возможности выбрать для танца мужчину из бухгалтерии и женщину, работа которой не связана с финансами? Пусть E – множество сотрудников, которые не употребляют алкоголь. Какие существуют возможности?

4. Универсум – множество товаров в супермаркете. Множество A – продукты питания, B – парфюмерия, C – канцелярия, D –

дорогие товары, $\overline{D}$ – недорогие товары. Студентка хочет приобрести недорогие продукты питания, а ее соседка – дорогую парфюмерию или недорогие канцелярские принадлежности. Какие существуют возможности совершить покупки? Пусть E – множество вкусов йогуртов. Какие существуют возможности указать те из них, которые любят обе студентки?

5. Универсум – компьютерные игры. Множество A – стрелялки, B – бродилки, C – стратегии, D – игры, которые можно пройти за один вечер, $\overline{D}$ – все остальные игры. Двое студентов решили по очереди сыграть в игру. Первый из них хочет пройти за один вечер стрелялку, а второй – бродилку или стратегию, которую нужно проходить долго. Какие существуют возможности? Пусть E – игры, которые студенты установили себе на компьютер. В какие игры они могут сыграть сегодня, какие есть возможности?

Задание 2. В общежитии проживают студенты разных национальностей, и однажды они собрались и решили вместе приготовить ужин. Каждый студент сказал, какие блюда он любит. Получилось отношение *R* между множеством студентов и множеством блюд. Чтобы приготовить блюдо, нужен определенный набор продуктов. Таким образом, имеется отношение *S* между множеством блюд и множеством продуктов. Определите, являются ли отношения *R* и *S*: всюду определенными, функциональными, сюръективными, инъективными, биективными. Найдите суперпозицию отношений $R \circ S$, которая является отношением между множеством студентов и множеством продуктов.

Всюду определенность: у всех ли студентов есть любимые блюда?

Функциональность: правда ли, что у каждого студента не более одного любимого блюда?

Сюръективность: все ли из перечисленных блюд нравятся одному из студентов?

Инъективность: правда ли, что двум разным студентам не может нравиться одно и то же блюдо?

Биективность: является ли соответствие между студентами и блюдами взаимно однозначным?

Суперпозиция: какие продукты предпочитает каждый студент?

Задание 3. Придумайте любое множество и задайте на этом множестве любое отношение. Выясните, является ли оно: рефлексивным, симметричным, антисимметричным, транзитивным.

Задание 4. Задайте на любом множестве отношение эквивалентности (например, сравнимость по модулю на числовом множестве). Перечислите все его классы эквивалентности.

3.3. Лабораторные работы

Задача 1. Вам дана заготовка для реализации основных операций над множествами. Множества представляются в виде двоичных векторов, к которым поэлементно применяются логические операции. Допишите недостающие функции, не пользуясь стандартным контейнером set.

Задача 2. Найдите декартово произведение двух множеств и булеан множества для задачи из своего варианта.

Задача 3. Выясните свойства двух отношений между разными множествами (всюду определенность, функциональность, сюръективность, инъективность, биективность) и найдите их суперпозицию. Отношения можно представить сначала списком пар, а затем преобразовать в булеву матрицу.

Указание. Составьте логическую формулу для каждого свойства и запрограммируйте ее.

Задача 4. Выясните свойства отношения, заданного на одном множестве: рефлексивность, симметричность, антисимметричность, транзитивность.

Задача 5. Составьте алгоритм для перечисления всех классов эквивалентности введенного Вами отношения эквивалентности.

Глава 4. КОМБИНАТОРИКА

4.1. Теория

Комбинаторика изучает перечисление и подсчет элементов конечных множеств. Правило суммы: выбор элемента одного из двух непересекающихся множеств A и B осуществляется $|A| + |B|$ способами. Правило произведения: выбор упорядоченной пары из элемента множества A и элемента множества B осуществляется $|A|\cdot|B|$ способами.

Перестановка элементов множества $\{1,...,n\}$ иначе называется перестановкой длины n. Например, при $n = 3$ перестановок всего 6: (1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1).

Подсчитаем число различных перестановок длины n. После выбора первого элемента перестановки (n способов) выбор второго элемента можно осуществить $n - 1$ способами, затем выбор третьего элемента осуществляется $n - 2$ способами и т.д. Получается, что, согласно правилу произведения, выбрать порядок следования n элементов можно $n! = 1 \cdot 2 \cdot 3 \cdot ... \cdot n$ способами (читается «n-факториал»).

Упорядоченной выборкой k элементов из n, или *размещением*, называют кортеж, составленный из k различных элементов множества $\{1,...,n\}$. Например, при $n = 3$ и $k = 2$ существует 6 различных размещений: (1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2).

Подсчитаем число размещений из n элементов по k местам. На первое место элемент можно выбрать n способами, после этого число способов выбрать элемент на второе место равно $n - 1$, затем для выбора на третье место есть $n - 2$ способов и т.д. Итого получаем k множителей, идущих подряд по убыванию; начиная с n – такое произведение обозначают через $(n)_k$. Число размещений из n по k обозначают через A_n^k. Таким образом,

$$A_n^k = (n)_k = n \cdot (n - 1) \cdot (n - 2) \cdot ... \cdot (n - k + 1) = \frac{n!}{(n - k)!}.$$

Обратим внимание, что в полученной формуле мы вычитаем из n числа 0, 1, 2, …, $k - 1$, и получается k множителей. Поэтому последний множитель равен $n - (k - 1) = n - k + 1$.

Неупорядоченной выборкой k элементов из n, или *сочетанием*, называют k-элементное подмножество множества $\{1,...,n\}$. Например, при $n = 3$ и $k = 2$ существует 3 различных сочетания: $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$. Таким образом, сочетания отличаются от размещений тем, что два сочетания, отличающиеся только порядком следования элементов, считаются неразличимыми.

Чтобы найти число различных сочетаний, установим соответствие между сочетаниями и размещениями. Заметим, что каждому сочетанию из n элементов по k соответствует $k!$ различных размещений, получаемых из этого сочетания. Например, сочетанию $\{1, 2\}$ соответствуют размещения $(1, 2)$, $(2, 1)$, сочетанию $\{1, 3\}$ соответствуют размещения $(1, 3)$, $(3, 1)$ и сочетанию $\{2, 3\}$ соответствуют размещения $(2, 3)$, $(3, 2)$.

Итак, число сочетаний из n по k (обозначение C_n^k) в $k!$ раз меньше, чем число размещений из n по k. Поэтому

$$C_n^k = \frac{A_n^k}{k!} = \frac{n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-k+1)}{1 \cdot 2 \cdot 3 \cdot \ldots \cdot k} = \frac{n!}{k!(n-k)!}.$$

Для перечисления всех размещений из n по k можно использовать следующий рекурсивный алгоритм:

```
# выбрать из n элементов очередной элемент
#   на m-ю позицию, после чего рекурсивно
#   вызвать процедуру для m+1
# a[1..m-1] содержит заполненную часть кортежа
# u[1..n] - булев массив
#   (True - число использовано, False - нет)
Процедура ПеречислитьРазмещения (n, k, m, @a, u)
    Если m == n + 1:
        Выдать a
    Иначе:
      Для i = 1..n:
        Если not u[i]:
           u[i] = True
           a[m] = i
           ПеречислитьРазмещения(n, k,
                m + 1, @a, u)
           u[i] = False
Конец
Ввод(@n, @k)
ПеречислитьРазмещения(n, k, 1, @a, u)
```

Упражнение. Реализуйте перечисление перестановок, вызвав процедуру перечисления размещений.

Чтобы перечислить все сочетания, можно представить сочетание из *n* по *k* с помощью двоичного вектора длины *n*, в котором ровно *k* единиц:

```
# среди n элементов определиться, брать ли
  # число m в сочетание, после чего рекурсивно
  # вызвать процедуру для m+1
# b[1..n] - булев массив
 # (True - число задействовано, False - нет)
# s - число единиц в массиве b
Процедура ПеречислитьСочетания (n, k, m, @b, s)
   Если s == k:
     Выдать b
   Иначе Если m <= n:
     b[m] = False
     ПеречислитьСочетания(n, k,
        m + 1, @b, s)
     b[m] = True
     ПеречислитьСочетания(n, k,
        m + 1, @b, s + 1)
Конец

Ввод(@n, @k)
ПеречислитьСочетания(n, k, 1, @b, 0)
```

Пример 1. Число всевозможных *k*-элементных подмножеств множества {1,...,n} равно числу характеристических векторов, содержащих ровно *k* единиц. Это число равно количеству способов выбрать *k* позиций из *n*, где будет единица. Данное количество способов равняется числу сочетаний из *n* по *k*, или C_n^k.

Упражнение. Обоснуйте формулу $\sum_{k=0}^{n} C_n^k = 2^n$.

Упорядоченной повторной выборкой *k* элементов из *n*, или *размещением с повторениями*, называют кортеж, составленный из *k* элементов множества {1,...,n}, элементы которого могут повторяться. Например, при $n = 3$ и $k = 2$ существует 9 различных размещений с повторениями: (1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2), (1, 1), (2, 2), (3, 3).

Подсчитаем число размещений с повторениями из n элементов по k местам. На первое место элемент можно выбрать n способами, после этого число способов выбрать элемент на второе место равно n, затем для выбора на третье место есть n способов и т.д. Итого получаем k множителей, равных n. Поэтому число размещений с повторениями из n по k равно n^{k}.

Пример 2. Число различных функций из множества A в множество B, где $|\mathrm{A}| = \mathrm{n}$, $|\mathrm{B}| = \mathrm{m}$, равно числу кортежей, составленных из n чисел от 1 до m, или числу размещений с повторениями из m по n, или m^{n}. Число различных инъекций из множества A в множество B, где $|\mathrm{A}| = \mathrm{n}$, $|\mathrm{B}| = \mathrm{m}$, $\mathrm{n} \le \mathrm{m}$, равно числу кортежей, составленных из n различных чисел от 1 до m, или числу размещений из m по n, т.е. $\mathrm{A}_{\mathrm{m}}^{\mathrm{n}}$.

Неупорядоченной повторной выборкой k элементов из n, или *сочетанием с повторениями*, называют мультимножество, составленное из k элементов множества $\{1,...,\mathrm{n}\}$. Например, при $n = 3$ и $k = 2$ существует 6 различных сочетаний с повторениями: {1, 2}, {1, 3}, {2, 3}, {1, 1}, {2, 2}, {3, 3}.

Подсчитаем число сочетаний с повторениями из n по k. Закодируем сочетание с повторениями в виде двоичного кортежа из $k + n - 1$ элементов. В этом кортеже будет k единиц и $n - 1$ нулей. Число всех таких кортежей равно $\mathrm{C}_{\mathrm{k+n-1}}^{\mathrm{k}}$. Всякому кортежу соответствует сочетание с повторениями согласно следующему правилу. Нули разбивают блок из k единиц на n частей. Сколько единиц в i-й части, столько же и чисел, равных i, в сочетании с повторениями. Следовательно, число сочетаний с повторениями из n по k равно $\mathrm{C}_{\mathrm{k+n-1}}^{\mathrm{k}}$.

Пример 3. В магазине продаются японские сэндвичи с рисом трех видов: с лососем, крабом и курицей. Сколько возможных вариантов выбрать 6 сэндвичей?

Нас интересует число неупорядоченных повторных выборок из 3 по 6, или число сочетаний с повторениями, равное $\mathrm{C}_{6+3-1}^{6} = \mathrm{C}_{8}^{6} = \frac{8!}{6!\cdot(8-6)!} = \frac{7\cdot 8}{1\cdot 2} = 28$.

Пример 4. В группе из 40 студентов 30 умеют плавать, 27 умеют играть в шахматы, 5 не умеют ни плавать, ни играть в шахматы. Сколько студентов умеют и плавать, и играть в шахматы?

Пусть на множестве из N элементов заданы свойства $\alpha_1,\alpha_2,...,\alpha_m$. Число элементов множества, обладающих свойством α_i, обозначим через $N(\alpha_i)$. Число элементов множества, обладающих свойствами α_i и α_j, обозначим через $N(\alpha_i,\alpha_j)$. Для числа элементов множества, не обладающих ни одним из свойств α_i, справедлива *формула включений и исключений*:

$$\begin{aligned} N(\overline{\alpha_1},\overline{\alpha_2},...,\overline{\alpha_m}) = N - N(\alpha_1) - N(\alpha_2) - ... - N(\alpha_m) + N(\alpha_1,\alpha_2) + N(\alpha_1,\alpha_3) \\ + ... + N(\alpha_{m-1},\alpha_m) - N(\alpha_1,\alpha_2,\alpha_3) - ... - N(\alpha_{m-2},\alpha_{m-1},\alpha_m) \\ + ... + (-1)^m N(\alpha_1,\alpha_2,...,\alpha_m). \end{aligned}$$

В нашем примере $N = 40$, $N(\alpha_1) = 30$, $N(\alpha_2) = 27$, $N(\overline{\alpha_1},\overline{\alpha_2}) = 5$. Согласно формуле включений и исключений, имеем

$$N(\overline{\alpha_1},\overline{\alpha_2}) = N - N(\alpha_1) - N(\alpha_2) + N(\alpha_1,\alpha_2).$$

Поэтому

$$N(\alpha_1,\alpha_2) = N(\overline{\alpha_1},\overline{\alpha_2}) + N(\alpha_1) + N(\alpha_2) - N = 5 + 30 + 27 - 40 = 22.$$

Пример 5. В лифте находятся 8 человек, в здании – 5 этажей. Сколькими способами люди могут выбрать, на каком этаже выйти, если требуется, чтобы на каждом этаже выходил хотя бы один человек?

В задаче требуется найти число различных сюръекций из множества A в множество B, где $|A| = n = 8$, $|B| = m = 5$.

Число различных функций из A в B равно m^n. Определим свойства $\alpha_1,\alpha_2,...,\alpha_m$ со следующим содержанием: α_i = «i-й элемент множества B не задействован». Тогда число различных сюръекций из A в B равно количеству функций, не обладающих ни одним из свойств α_i. Чтобы вычислить это количество, воспользуемся формулой включений и исключений. В этой формуле каждое из C_m^k слагаемых с наличием k выбранных свойств равно числу различных функций из n-элементного множества в $(m - k)$-элементное множество. Отсюда получим формулу для числа сюръекций:

$$\sum_{k=0}^{m} (-1)^k C_m^k (m - k)^n.$$

Это число выражается через числа Стирлинга второго рода.

Упражнение. Вычислите ответ в задаче.

Пример 6. Согласно государственному стандарту, автомобильный номерной знак состоит из 3 цифр и 3 букв. При этом недопустим номер с тремя нулями, а буквы выбираются из набора А, В, Е, К, М, Н, О, Р, С, Т, У, Х (используются только те буквы кириллицы, написание которых совпадает с латинскими буквами). Сколько различных номерных знаков можно составить для региона?

Очевидно, что должна быть использована формула умножения размещений с повторениями для цифр и для букв, так как они в номере присутствуют одновременно. Необходимо убрать номер со всеми нулями:

$$(n^k - 1)m^k = (10^3 - 1) \cdot 12^3 = 999 \cdot 1728 = 1726272.$$

4.2. Задачи

Задание 1. У Вас на телефоне *N* контактов. Вы видите переписку с *K* из них, когда включаете мессенджер. Какие существуют возможности: а) с учетом порядка следования контактов; б) без учета порядка их следования?

Задание 2. Вам дана заготовка, которая принимает на вход координаты точек плоскости и рисует случайный маршрут, проходящий через все точки. Оптимизируйте маршрут, чтобы его суммарная длина была минимально возможной.

Указание. Переберите все возможные перестановки, показывающие, в каком порядке следует посетить точки.

Задание 3. У большой языковой модели есть *N* различных функций (например, досочинять стих, ответить на вопрос, решить задачу). Пробный период использования разрешает ввести *K* запросов. Какие существуют возможности для использования

функций: а) с учетом порядка следования функций; б) без учета порядка их следования?

Задание 4. В России автомобильные номера образца 1993 г. содержат три буквы и три цифры, но разрешается использовать лишь 12 букв, похожих на латинские, набор из трех нулей не допускается. Еще в правой части номера указывается код региона (обычно две цифры). Перечислите все различные номера и выведите их количество.

Задание 5. Решите числовой ребус перебором комбинаций:

1. МАГНИЙ + ТАНТАЛ = МЕТАЛЛЫ
2. ПОДАЙ – ВОДЫ = ПАША
3. КОРОВА + ДОЯРКА + ТРАВА = МОЛОКО
4. ДЕДКА + БАБКА + РЕПКА = СКАЗКА
5. СОЧИ + ЗИМА = СПОРТ

4.3. Лабораторные работы

Задача 1. Перечислите все различные размещения и сочетания из *N* по *K*. Найдите их количество двумя способами: по формуле и подсчетом.

Задача 2. Найдите минимальный гамильтонов контур в неориентированном взвешенном графе.

Задача 3. Перечислите все различные размещения и сочетания с повторениями из *N* по *K*. Найдите их количество двумя способами: по формуле и подсчетом.

Задача 4. Выведите все различные автомобильные номера, посчитайте их.

Задача 5. Решите числовой ребус.

Глава 5. ГРАФЫ

5.1. Теория

Ориентированным графом называется упорядоченная пара множеств $G = (V, E)$, где E – это подмножество множества всех упорядоченных пар элементов из V.

Элементы множества V называются *вершинами* графа, а элементы множества E – *ребрами* графа, соединяющими соответствующие вершины.

Если для всякого ребра $(u, v) \in E$ имеет место $(v, u) \in E$, то граф называется *неориентированным*, или просто *графом*. Если $(u, v) \in E$, то вершины u и v называются *смежными*.

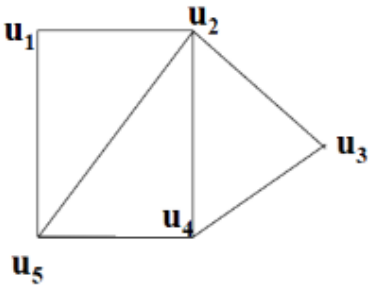
Вершины графа (множество V) обозначают точками на плоскости, а ребра графа (множество E) – линиями на плоскости, соединяющими соответствующие точки. Множество вершин и ребер графа G обозначают $V(G)$ и $E(G)$ соответственно.

В дальнейшем будем рассматривать только графы с конечным числом вершин, и это число называется *порядком* графа.

Матрицей смежности графа порядка n называется квадратная бинарная матрица A порядка n, определяемая следующим образом:

$$A = (a_{i,j})_{i,j=1}^{n}, \text{ где } a_{i,j} = \begin{cases} 1, & \text{если вершины i и j смежны,} \\ 0, & \text{если вершины i и j не смежны.} \end{cases}$$

Например, матрица смежности для неориентированного графа



имеет вид

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

Задание графа с помощью его матрицы смежности однозначно, если вершины графа занумерованы. Более того, любая квадратная бинарная матрица является матрицей смежности некоторого ориентированного графа, а произвольная бинарная симметрическая матрица – неориентированного графа.

Граф можно определить с помощью матрицы Кирхгофа:

$$K = (k_{i,j})_{i,j=1}^{n}, \text{ где } k_{i,j} = \begin{cases} -1, \text{ если вершины i и j смежны,} \\ 0, \text{ если вершины i и j не смежны,} \\ \text{степень вершины i, если i = j.} \end{cases}$$

Для графа, приведенного выше с матрицей смежности A, матрица Кирхгофа имеет вид

$$K = \begin{pmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ -1 & -1 & 0 & -1 & 3 \end{pmatrix}.$$

Очевидно, что сумма элементов каждой строки и столбца матрицы Кирхгофа равна нулю.

С помощью матриц можно вводить граф в компьютерную программу.

Последовательность вершин, в которой соседние вершины последовательности смежны, называется *маршрутом*. Если начальная и конечная вершины маршрута совпадают, то он называется *циклом*. Граф без циклов называют *ациклическим*. Если между любыми двумя вершинами в графе существует маршрут, граф называется *связным*. Связный ациклический граф называется *деревом*.

В графе определено отношение *достижимости*: две вершины *u*, *v* связаны отношением достижимости, если существует маршрут из *u* в *v*. Отношение достижимости является отношением эквивалентности. Классы эквивалентности называют *связными компонентами*.

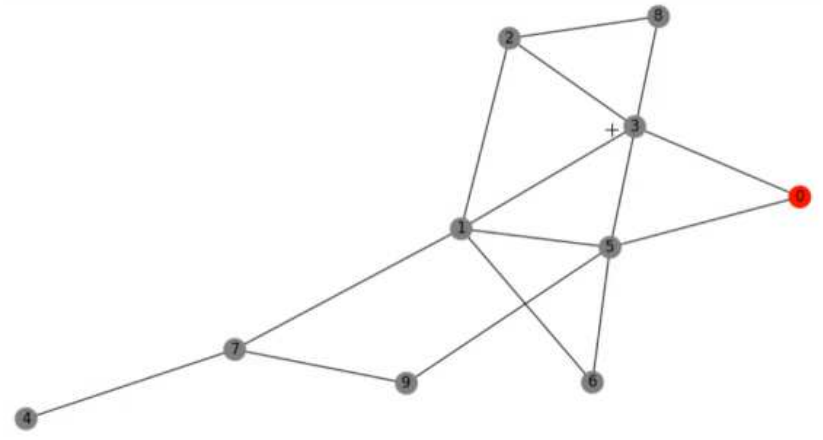
Маршрут можно задавать разными условиями. Известны несколько алгоритмов задания маршрута: поиск в ширину, поиск в глубину.

Поиск в ширину (BFS, breadth-first search) представляет собой волновой алгоритм, который можно сравнить с поджиганием вершин, начиная со стартовой.

Данный алгоритм использует структуру данных «очередь». Это последовательность данных с двумя методами: append – добавление данных в конец очереди, pop – извлечение данных из начала очереди.

Разберем алгоритм BFS на примере.

Дан невзвешенный неориентированный граф, который имеет 10 вершин, начиная от вершины 0 (красная). Самая дальняя вершина имеет номер 4. Алгоритм позволяет найти кратчайший маршрут из одной вершины графа до любой другой его вершины. Вершины, смежные с текущей вершиной, будут просматриваться в порядке возрастания номеров.



Граф в программе задается симметричной матрицей смежности.

	0	1	2	3	4	5	6	7	8	9
0				1		1				
1			1	1		1	1	1		
2		1		1					1	
3	1	1	1			1			1	
4								1		
5	1	1		1			1			1
6		1				1				
7		1			1					1
8			1	1						
9						1		1		

Шаг 0. Создать следующие объекты.

Множество посещенных вершин:
visited_nodes = {}
visited_nodes

Список расстояний от стартовой вершины:
distances = [0, None, None, None, None, None, None, None, None, None]

вершина	0	1	2	3	4	5	6	7	8	9
distances	0	None	None	None	None	None	None	None	None	None

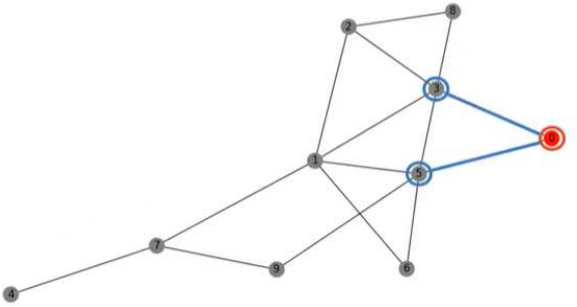
Список родителей вершин:
parents = [None, None, None, None, None, None, None, None, None, None]

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	None	None	None	None	None	None	None	None	None

Шаг 1. Снимаем из очереди вершину 0, помещаем в множество посещенных. Помещаем в очередь соседей текущей вершины. Расстояния до соседей полагаем равными 1. Запоминаем, что для соседей родителем будет вершина 0.

queue	3	5
visited_nodes	0	

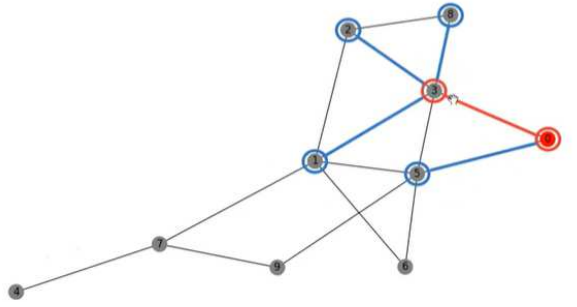
вершина	0	1	2	3	4	5	6	7	8	9
distances	0	None	None	1	None	1	None	None	None	None
parents	None	None	None	0	None	0	None	None	None	None



Шаг 2. Снимаем из очереди вершину 3, помещаем в множество посещенных. Помещаем в очередь соседей текущей вершины, кроме тех, что уже в очереди. Расстояния до этих вершин полагаем равными 2. Запоминаем родителя.

queue	~~3~~	5	1	2	8
visited_nodes	0	3			

вершина	0	1	2	3	4	5	6	7	8	9
distances	0	2	2	1	None	1	None	None	2	None
parents	None	3	3	0	None	0	None	None	3	None

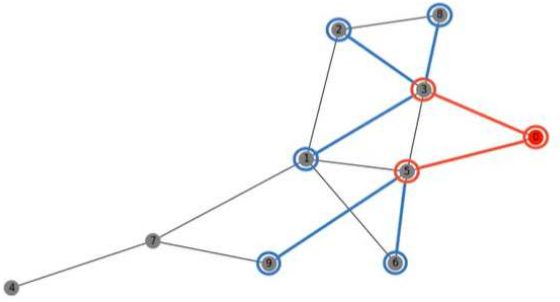


Аналогично производим остальные шаги. Ниже для сверки приведены шаг 3, шаг 9 и шаг 10.

Шаг 3.

queue	~~5~~	1	2	8	6	9
visited_nodes	0	3	5			

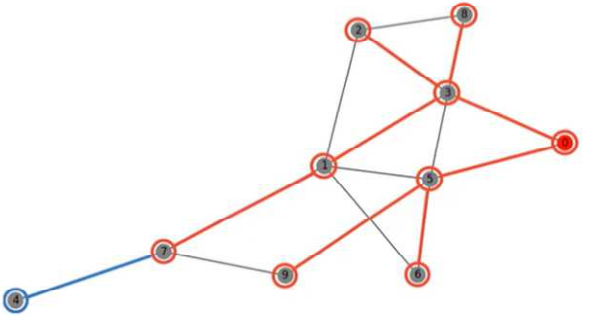
вершина	0	1	2	3	4	5	6	7	8	9
distances	0	2	2	1	None	1	2	None	2	2
parents	None	3	3	0	None	0	5	None	3	5



Шаг 9.

queue	~~5~~	~~6~~	~~8~~	~~1~~	~~2~~	~~9~~	4		
visited_nodes	0	3	5	1	2	8	6	9	7

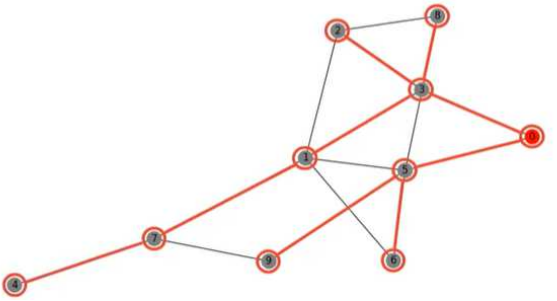
вершина	0	1	2	3	4	5	6	7	8	9
distances	0	2	2	1	4	1	2	3	2	2
parents	None	3	3	0	7	0	5	1	3	5



Шаг 10.

queue	~~4~~								
visited_nodes	0	3	5	1	2	8	6	9	7	4

вершина	0	1	2	3	4	5	6	7	8	9
distances	0	2	2	1	4	1	2	3	2	2
parents	None	3	3	0	7	0	5	1	3	5



Очередь пуста. Поиск окончен.

Шаг 11. Восстановление пути.

Восстановление пути выполняется по списку родителей. Например, конечной вершиной при вводе была вершина 4:

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	3	0	**7**	0	5	1	3	5

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	3	0	7	0	5	**1**	3	5

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	**3**	3	0	7	0	5	1	3	5

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	3	**0**	7	0	5	1	3	5

Таким образом, путь от конца к началу: 4-7-1-3-0; путь от начала к концу: 0-3-1-7-4 (ответ теста).

Алгоритм *поиска в глубину* (DFS, depth-first search) можно сравнить с продвижением вглубь, пока это возможно. Используется стек (магазин с патронами) – последний патрон берется первым, т.е. добавляем справа и снимаем справа.

Разберем алгоритм DFS на том же примере. Для удобства будем просматривать соседей вершины в порядке возрастания номеров. Если идти некуда, происходит возврат на уровень выше.

Шаг 0. Создать следующие объекты.

Создаем стек, в который будем помещать вершины по мере их обнаружения:

stack	0

Добавляем в стек **справа**, снимаем из стека **справа**.

Множество посещенных вершин:

visited_nodes	

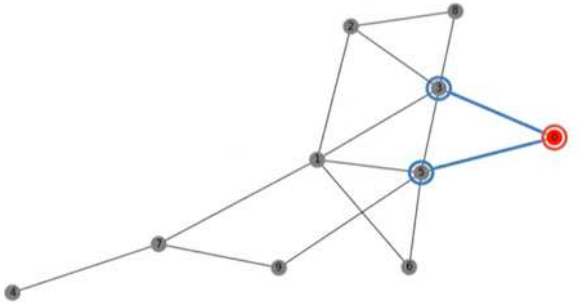
Список родителей вершин:

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	None	None	None	None	None	None	None	None	None

Шаг 1. Снимаем из стека вершину 0, помещаем в множество посещенных. Помещаем в стек соседей текущей вершины. Запоминаем родителя.

stack	**5**	**3**
visited_nodes	**0**	

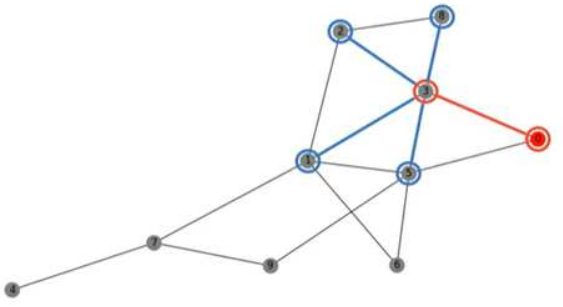
вершина	**0**	1	2	**3**	4	**5**	6	7	8	9
parents	None	None	None	**0**	None	**0**	None	None	None	None



Шаг 2. Снимаем из стека вершину 3, помещаем в множество посещенных. Смотрим соседей вершины 3. Если соседей не было в стеке, то добавляем в стек. Если сосед был в стеке, то передвигаем на вершину стека. Запоминаем нового родителя.

stack	5	~~8~~	8	5	2	1
visited_nodes	0	~~3~~				

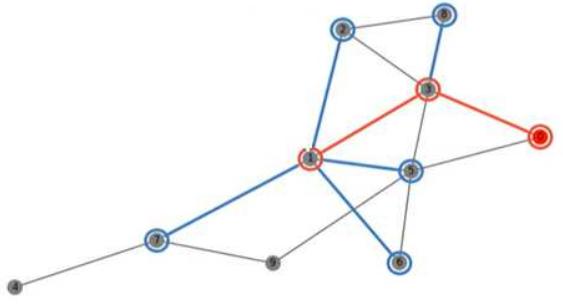
вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	3	0	None	3	None	None	3	None



Шаг 3.

stack	8	5	2	~~1~~	7	6	5	2
visited_nodes	0	3	1					

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	None	1	1	1	3	None

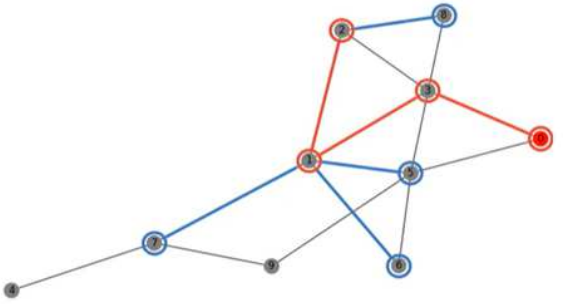


У всех соседей обновляются родительские вершины (см. вершину 5). Аналогично сделать остальные шаги. Ниже для сверки приведены шаги 4, 5, 9, 10.

Шаг 4.

stack	8	7	6	5	~~2~~	8
visited_nodes	0	3	1	2		

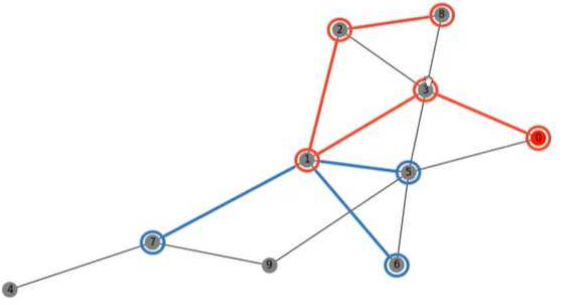
вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	None	1	1	1	2	None



Шаг 5.

stack	7	6	5	~~2~~		
visited_nodes	0	3	1	2	8	

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	None	1	1	1	2	None

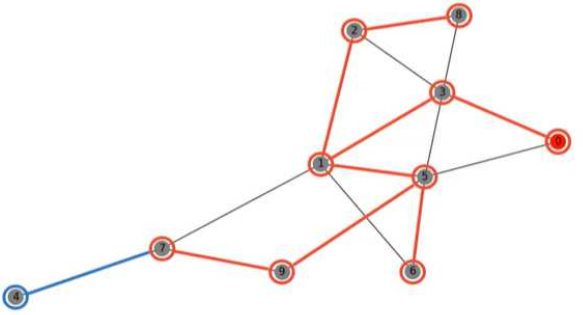


Если нет новых вершин, в стек ничего не помещаем.

Шаг 9.

stack	~~0~~	4							
visited_nodes	0	3	1	2	8	5	6	9	7

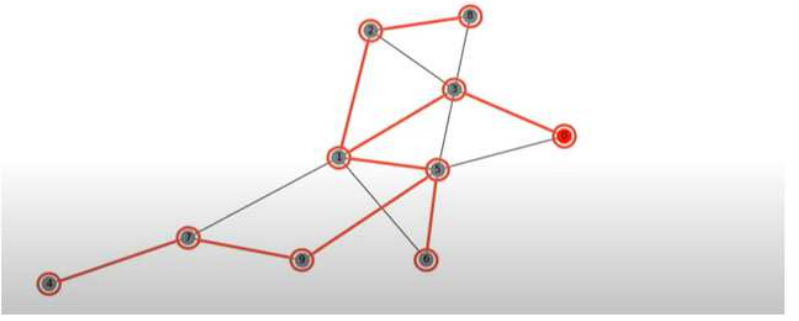
вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	7	1	5	9	2	5



Шаг 10.

stack	~~0~~									
visited_nodes	0	3	1	2	8	5	6	9	7	4

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	7	1	5	9	2	5



Стек пуст. Поиск окончен.

Шаг 11. Восстановление пути. Выполняется по списку родителей. Например, конечной вершиной при вводе была вершина 4:

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	**7**	1	5	9	2	5

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	7	1	5	**9**	2	5

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	7	1	5	9	2	**5**

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	0	7	**1**	5	9	2	5

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	**3**	1	0	7	1	5	9	2	5

вершина	0	1	2	3	4	5	6	7	8	9
parents	None	3	1	**0**	7	1	5	9	2	5

Таким образом, путь от конца к началу: 4-7-9-5-1-3-0; путь от начала к концу: 0-3-1-5-9-7-4 (ответ теста).

5.2. Задачи

Задание 1. Неориентированный граф задан своей матрицей смежности. Перечислите все его связные компоненты.

Решение. Заведем массив булева типа, в котором будем отмечать, посещена ли каждая вершина. Затем, просматривая по очереди все вершины графа, будем запускать из каждой непосещенной вершины поиск в ширину или поиск в глубину. Все посещенные при этом вершины отмечаем отдельной меткой, соответствующей номеру очередной связной компоненты.

Задание 2. Дан ориентированный ациклический граф. Требуется перечислить все его вершины в таком порядке, чтобы всякое ребро графа исходило из вершины в одну из последующих вершин (задача топологической сортировки). Например, вершины графа – это учебные дисциплины, и ребро идет из одной дисциплины в другую дисциплину, которая опирается на первую. В каком порядке требуется изучать дисциплины?

Решение. Будем хранить булев массив, обозначающий для каждой вершины графа, была ли вершина обработана. Заведем

счетчик меток, которому вначале присвоим число *n*, равное количеству вершин в графе. Перечисляем по очереди все вершины графа. Если вершина еще не обработана, вызываем из нее рекурсивную процедуру поиска в глубину. Данная процедура помечает текущую вершину обработанной, затем рекурсивно вызывается для всех смежных вершин, после чего присваивает текущей вершине метку, равную значению счетчика, а счетчик после этого уменьшается на единицу. После выполнения алгоритма метки вершин образуют топологическое упорядочение.

5.3. Лабораторные работы

Задача 1. Составьте программу для реализации алгоритма поиска в ширину. Выведите все данные в процессе работы алгоритма: список расстояний до стартовой вершины, список родителей вершин. Восстановите путь до выбранной вершины. Выведите изображение графа.

Задача 2. Составьте программу для реализации алгоритма поиска в глубину. Выведите все данные в процессе работы алгоритма. Восстановите путь до выбранной вершины. Выведите изображение графа.

Задача 3. Составьте программу, которая принимает на вход граф (в любом формате) и определяет, является ли он деревом.

Глава 6. РАЗБОР ЗАДАЧ

Две пешки и слон

Задача с олимпиады летних учебно-тренировочных сборов по информатике, 2011 г.

На шахматной доске на двух различных клетках расположены две пешки. На какую свободную клетку поставить слона, чтобы обе пешки оказались под боем?

Решение. Если доска имеет размер 8×8, задачу можно решить перебором. Введем на множестве номеров клеток 1..8 двухместный предикат $R(x, y)$, равный 1 в том и только том случае, когда слон, находящийся в клетке (x, y), бьет обе клетки (x_1, y_1) и (x_2, y_2).

Введем четырехместный предикат $D(x_1, y_1, x_2, y_2)$ = «клетки (x_1, y_1) и (x_2, y_2) находятся на одной диагонали». Тогда условие, что слон в клетке (x, y) бьет обе пешки, можно выразить так: клетка (x, y) не совпадает ни с одной из клеток (x_1, y_1) и (x_2, y_2), и при этом клетка (x, y) находится с каждой из них на одной диагонали:

$$\mathrm{R(x,y)} = (((\mathrm{x,y}) \neq (\mathrm{x_1,y_1})) \wedge ((\mathrm{x,y}) \neq (\mathrm{x_2,y_2})) \wedge \mathrm{D(x,y,x_1,y_1)} \wedge\wedge \mathrm{D(x,y,x_2,y_2)}).$$

Таким образом, достаточно перечислить все клетки (x, y) и выбрать ту из них, для которой $R(x, y) = 1$.

Упражнение. Дайте числовое выражение для предиката $D(x_1, y_1, x_2, y_2)$.

Если шахматная доска достаточно большая, то перебор займет слишком много времени. Поэтому для решения задачи потребуется пересечь два множества – множество клеток, стоящих с (x_1, y_1) на одной диагонали, и множество клеток, стоящих с (x_2, y_2) на одной диагонали: $\{(\mathrm{x,y}):\mathrm{D(x,y,x_1,y_1)}\} \cap \{(\mathrm{x,y}):\mathrm{D(x,y,x_2,y_2)}\}$. Каждое из этих множеств задается совокупностью из двух уравнений прямой.

Переворачивающиеся прямоугольники

Задача с XXIII испанской олимпиады по информатике

Дано прямоугольное игровое поле размера $n \times m$, разделенное на черные и белые клеточки. За один ход можно выбрать любую клеточку и поменять на противоположные цвета клеточек во всем прямоугольнике между выбранной клеточкой и правым нижним углом игрового поля. В начале игры цвета клеточек могут быть любыми; эти цвета подаются на вход Вашей программе. Цель игры – добиться того, что все клеточки станут белыми за минимальное число ходов. Вычислите это число ходов.

Решение. Обозначим начальное состояние игрового поля через a_{ij}, где $a_{ij} = 1$, если клетка (i, j) черная, и $a_{ij} = 1$, если клетка (i, j) белая. Определим булеву матрицу $\{x_{ij}\}$, в которой $x_{ij} = 1$, если осуществился ход по переворачиванию прямоугольника от клетки (i, j), и $x_{ij} = 0$, если такого хода не было. Тогда финальное состояние игрового поля задается булевой матрицей $\{b_{ij}\}$, в которой

$$b_{ij} = a_{ij} \oplus (x_{11} \oplus x_{12} \oplus \ldots \oplus x_{1j}) \oplus (x_{21} \oplus x_{22} \oplus \ldots \oplus x_{2j}) \oplus \ldots \oplus (x_{i1} \oplus x_{i2} \oplus \ldots \oplus x_{ij}).$$

Введем еще одну булеву матрицу $\{y_{ij}\}$, равную сумме по модулю 2 значений x_{ij} в прямоугольнике между клеткой (1, 1) и клеткой (i, j). Тогда $b_{ij} = a_{ij} \oplus y_{ij}$.

Поскольку нам нужно, чтобы в конце все клетки стали белыми, то $b_{ij} = 0$, поэтому $y_{ij} = a_{ij}$. Итак, нам известны значения сумм по модулю 2 элементов неизвестной булевой матрицы $\{x_{ij}\}$ по прямоугольникам с левым верхним углом (1, 1). Требуется найти x_{ij}.

Имеем систему уравнений

$$(x_{11} \oplus x_{12} \oplus \ldots \oplus x_{1j}) \oplus (x_{21} \oplus x_{22} \oplus \ldots \oplus x_{2j}) \oplus \ldots \oplus (x_{i1} \oplus x_{i2} \oplus \ldots \oplus x_{ij}) = a_{ij}$$

для всех $i = 1, 2, \ldots, n$ и $j = 1, 2, \ldots, m$.

Для сумм по прямоугольникам справедливо рекуррентное соотношение

$$y_{ij} = y_{i-1,j} \oplus y_{i,j-1} \oplus y_{i-1,j-1} \oplus x_{ij}.$$

Получим рекуррентную формулу для ответа:

$$x_{ij} = y_{i-1,j} \oplus y_{i,j-1} \oplus y_{i-1,j-1} \oplus y_{ij} = a_{i-1,j} \oplus a_{i,j-1} \oplus a_{i-1,j-1} \oplus a_{ij},$$

и искомое количество ходов равно сумме x_{ij} по всем i, j.

При решении мы использовали идею, аналогичную методу построения многочлена Жегалкина.

Океанские пары

В ВДЦ Океан отряды ходят в столовую океанскими парами (мальчик-девочка). Всего N отрядов, в которых a_i мальчиков и b_i девочек. Вожатые решили объединить некоторые отряды, чтобы увеличить количество океанских пар. Теперь отрядов стало M ($M < N$). Как нужно объединить отряды, чтобы максимизировать число океанских пар?

Решение. Предложим «жадный» алгоритм, обоснование которого остается открытым вопросом. Сформулируем задачу так: дан ряд чисел $d_i = a_i - b_i$, упорядоченный по возрастанию. За один шаг можно сложить какие-то два числа d_i и d_j и добавить их сумму в этот же ряд. Всего можно сделать $N - M$ шагов. Как достичь минимального значения функции $|d_1| + |d_2| + \ldots + |d_M|$?

Если следовать идее «жадного» алгоритма, то на каждом шаге следует выполнить действия, локально минимизирующие целевую функцию. Если все числа d_i одного знака, то суммирование любой пары чисел не изменит значение целевой функции. В противном случае возьмем самое маленькое отрицательное число и самое большое положительное число среди d_i. Сложим эти два числа и добавим сумму в ряд чисел.

При большом (порядка 10^5) числе N для реализации этой процедуры потребуется структура данных типа «динамическое множество» с эффективно реализованными операциями: извлечение минимума, извлечение максимума и вставка элемента в множество. Для реализации можно использовать структуру «сбалансированное дерево двоичного поиска».

Всё успеть

У студента Гены много дел: целых N. Каждому делу он назначил приоритет p_i и уровень сложности d_i. Гена – очень ответственный студент, поэтому он берется в первую очередь за наиболее важные дела, для которых p_i максимально. Да только он сейчас устал, поэтому может делать только те дела, для которых $d_i \le a$. Требуется найти среди таких дел наиболее важное. Вам дана последовательность чисел $a_1, a_2, ..., a_M$. Гена просматривает эти числа по очереди и находит среди всех дел с $d_i \le a_j$ дело с наибольшим приоритетом p_i, после чего удаляет его из множества дел. Реализуйте структуру данных для быстрого выполнения указанных операций.

Решение. Для ускорения поиска в множестве и модификации множества можно воспользоваться корневой структурой данных, в которой все N дел сгруппированы в $\approx \sqrt{N}$ групп по $\approx \sqrt{N}$ элементов в каждой. Это позволяет ускорить алгоритм в $\sqrt{N}$ раз.

Барабанная почта

Задача с Дальневосточного полуфинала Всероссийской командной олимпиады школьников по программированию, 2011 г.

Когда-то давно члены одного африканского племени, жившие в разных деревнях, использовали для передачи информации звуковую почту. Чтобы передать сообщение, отправитель бил в барабан в промежутки времени $a_i \le t \le b_i$, а получатель слушал и рассказывал жителям своей деревни. Сила звука зависит от погоды, например: во время дождя и грозы звук барабана практически не слышен. Однажды у племени поменялся вождь, и необходимо было оповестить об этом всех жителей племени. Но, как назло, погода в этот день была очень неустойчивая: то дождь, то туман, то ветер, то солнце. Поэтому звуки барабана можно было слышать только в промежутки времени $c_i \le t \le d_i$. Требуется определить, в какие промежутки времени получатели услышат звук барабана.

Решение. В задаче требуется найти пересечение множеств

$$([a_1,b_1] \cup ... \cup [a_n,b_n]) \cap ([c_1,d_1] \cup ... \cup [c_m,d_m]).$$

Исходные данные таковы, что отрезки $[a_i,b_i]$ не пересекаются и перечислены по возрастанию, то же касается отрезков $[c_i,d_i]$.

Рассмотрим множество координат всех концов всех отрезков, добавив к каждому числу метку: 1, если это начало одного из отрезков $[a_i,b_i]$; –1, если это конец одного из этих отрезков; 2, если это начало одного из отрезков $[c_i,d_i]$; –2, если это конец одного из этих отрезков. Упорядочим эти данные по возрастанию координаты точки.

Представим себе вертикальную сканирующую прямую, которая движется слева направо. Прямая может находиться в одном из четырех состояний:

а) не пересекает ни один из отрезков $[a_i,b_i]$ и не пересекает ни один из отрезков $[c_i,d_i]$;

б) не пересекает ни один из отрезков $[a_i,b_i]$ и пересекает какой-нибудь отрезок $[c_i,d_i]$;

в) пересекает какой-нибудь отрезок $[a_i,b_i]$ и не пересекает ни один из отрезков $[c_i,d_i]$;

г) пересекает какой-нибудь отрезок $[a_i,b_i]$ и пересекает какой-нибудь отрезок $[c_i,d_i]$.

Когда прямая встречает очередную точку, ее состояние меняется. Например, если точка имеет метку 1, то прямая пересекает отрезок $[a_i,b_i]$; если метку –1, то прямая больше не пересекает отрезок $[a_i,b_i]$; если метку 2, то прямая пересекает отрезок $[c_i,d_i]$; если метку –2, то прямая больше не пересекает отрезок $[c_i,d_i]$.

Как только прямая пересекает оба отрезка, этот момент нужно запомнить, и до следующей точки, которую встретит прямая, она будет сканировать искомое пересечение множеств.

Марфа Геннадьевна в парикмахерской

Однажды Марфа Геннадьевна пришла в парикмахерскую, в которой K парикмахеров одновременно обслуживают N клиентов ($N \geq K$). Электронная очередь позволяет осуществить запись к парикмахерам. Она устроена так, что к каждому парикмахеру должен записаться хотя бы один человек из N. Порядок следования записавшихся к одному парикмахеру существенен. Сколько существует возможных состояний электронной очереди?

Решение. Закодируем электронную очередь в виде перестановки длины *N*. Перестановка – это взаимно однозначное соответствие (биекция) между множеством {1,2,...,N} и им самим. Каждому клиенту поставим в соответствие следующего за ним по очереди клиента. Если клиент последний в очереди, то ему будет соответствовать первый в очереди. Таким образом, вся перестановка разбита на *K* циклов.

Из комбинаторики известно, что число всех перестановок длины *N*, состоящих из *K* циклов, выражается через число Стирлинга первого рода и равняется $(-1)^{N-K} s_{N,K}$.

Для чисел Стирлинга первого рода имеет место следующее рекуррентное соотношение: $s_{n+1,k} = s_{n,k-1} - n s_{n,k}$, с помощью которого легко получить следующую таблицу:

$s_{n,k}$	k = 1	k = 2	k = 3	k = 4	k = 5	k = 6	k = 7
n = 1	1						
n = 2	–1	1					
n = 3	2	–3	1				
n = 4	–6	11	–6	1			
n = 5	24	–50	35	–10	1		
n = 6	–120	274	–225	85	–15	1	
n = 7	720	–1764	1624	–735	175	–21	1

Документы

Задача с Дальневосточного полуфинала Всероссийской командной олимпиады школьников по программированию, 2015 г.

В папке с файлами для бумаг лежали документы. Кто-то перетасовал документы, и теперь в *i*-м файле находится документ с номером a_i. Требуется переложить документы в правильном порядке и при этом минимизировать количество выкладываний и вкладываний документов. Нужно, чтобы при перекладывании

каждый раз вне папки находилось не более двух документов (чтобы их не потерять). Требуется вывести последовательность действий по вкладыванию и выкладыванию документов, т.е. три параметра: вложили/выложили, номер документа, номер файла.

Решение. Минимальное количество действий равно количеству документов, которые лежат не на своих местах, умноженному на 2. Начнем с любого i, для которого $a_i \neq i$, и обойдем соответствующий цикл перестановки. Сначала вынем документ с номером $k = a_i$, находящийся в i-м файле. Положим $a_i = 0$. Предположим, что документ с номером k вынут. Затем повторим следующие действия, пока $a_k \neq 0$. Вынем документ с номером a_k, находящийся в k-м файле, и положим в k-й файл документ с номером k, который мы ранее вынули из другого файла. Далее положим $k = a_k$. Описанные действия нужно проделать для каждого цикла перестановки, т.е. начиная со всех i, для которых $a_i \neq i$.

Макет города

Задача с городской олимпиады по информатике, г. Владивосток, 2009 г.

Дано множество прямоугольников, которые служат основаниями зданий разной высоты, расположенными на одной плоскости. Некоторые здания могут соприкасаться стенками, но не пересекаться. Требуется найти суммарную площадь видимой поверхности зданий. Даны координаты левого нижнего (x_i, y_i) и правого верхнего (u_i, v_i) угла i-го прямоугольника, высота i-го здания h_i для $i = 1, \ldots, N$.

Решение. Суммарная площадь видимой поверхности зданий равна сумме поверхностей параллелепипедов минус удвоенная сумма (по всем парам зданий) площадей соприкосновения зданий. При больших N нас не устроит алгоритм со сложностью $O(N^2)$. Чтобы найти суммарную площадь соприкосновения зданий, нужно перечислить пары прямоугольников, соприкасающихся вертикальными сторонами, и посчитать длину соприкасающихся частей, а затем умножить это число на $2\min\{h_i, h_j\}$ и вычесть из общей суммы видимой поверхности. Аналогично с горизонтальными сторонами.

Как перечислить все пары соприкасающихся прямоугольников? Требуется перечислить элементы множества $\{p_i = (x_i, y_i, u_i, v_i),\ q_j = (x_j, y_j, u_j, v_j)\colon u_i = x_j\}$, а затем перечислить все такие пары (p_i, q_j) из этого множества, для которых отрезки $[y_i, v_i]$ и $[y_j, v_j]$ пересекаются. Нетрудно получить условие пересечения отрезков $[y_i, v_i]$ и $[y_j, v_j]$: $y_j < v_i \wedge y_i < v_j$. Итак, требуется перечислить элементы множества $\{p_i = (x_i, y_i, u_i, v_i),\ q_j = (x_j, y_j, u_j, v_j)\colon u_i = x_j, y_j < v_i, y_i < v_j\}$.

Для эффективного перебора всех элементов указанного множества сформируем два списка точек p_i: один отсортируем по возрастанию x_i, а при равных x_i – по возрастанию y_i, а другой отсортируем по возрастанию u_i, а при равных u_i – по возрастанию v_i.

Список $p_i = (x_i, y_i, u_i, v_i)$ упорядочен по возрастанию (x_i, y_i). Список $p'_i = (x'_i, y'_i, u'_i, v'_i)$ упорядочен по возрастанию (u_i, v_i).

1. Легко построить перечислитель всех пар (p_i, p'_j): $x_i = u'_j$. Для этого заведем две переменные i, j, указывающие на элементы двух массивов.

Возможны случаи:

- $x_i = u'_j \Longrightarrow$ продолжаем перечислять все такие пары, двигаясь вертикально;
- $x_i < u'_j \Longrightarrow$ увеличить i на 1;
- $x_i > u'_j \Longrightarrow$ увеличить j на 1.

2. Пусть $x_i = u'_j$. Требуется перечислить такие i, j, что $y'_j < v_i$, $y_i < v'_j$. Пока условие $x_i = u'_j$ сохраняется, делать следующее:

- если $v_i < y'_j \Longrightarrow$ увеличить i на 1;
- иначе, если $v'_j < y_i \Longrightarrow$ увеличить j на 1;
- иначе, т.е. если $y'_j \le v_i, y_i \le v'_j$, вычисляем длину пересечения отрезков $[y_i, v_i]$ и $[y'_j, v'_j]$ и вычитаем удвоенную длину, умноженную на $\min\{h_i, h'_j\}$, из суммарной площади соприкосновения зданий.

Разбиение треугольников

Задача с отборочного тура студенческого командного чемпионата мира по программированию ICPC, Дальневосточный регион, 2022 г.

Задано множество треугольников в трехмерном пространстве. Требуется разбить это множество на классы, в каждом из которых все треугольники лежат в одной плоскости.

Решение. Для каждого треугольника найдем уравнение плоскости, нормированное на первый ненулевой коэффициент. Затем введем на множестве коэффициентов уравнений отношение порядка и упорядочим коэффициенты. Для выделения классов треугольников достаточно сравнить соседние коэффициенты в полученной последовательности.

Равноудаленные строки

Задача с отборочного тура студенческого командного чемпионата мира по программированию ICPC, Дальневосточный регион, 2022 г.

Расстоянием Хэмминга между двумя строками a и b длины n называется количество позиций i от 1 до n, для которых $a[i] \neq b[i]$. Найдите количество строк длины n, состоящих из строчных латинских букв и цифр, равноудаленных от двух заданных строк.

Решение. Найдем количество позиций i, в которых $a[i] = b[i]$, обозначим это количество через k. Для того чтобы строка c была равноудалена от строк a и b, нужно выбрать символы в этих k позициях произвольно, а в остальных позициях оставить без изменения p символов из строки a и столько же символов из строки b. Существует $C_{n-k}^{2p} C_{2p}^{p}$ способов выбрать при заданном p те символы обеих строк, которые останутся без изменения, а остальные символы можно заменить на любые другие. Таким образом, ответ в задаче равен

$$36^{k} \sum_{p=0}^{[n-k]/2} C_{n-k}^{2p} C_{2p}^{p} \cdot 34^{n-k-2p}.$$

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Зуев, Ю.А. По океану дискретной математики: От перечислительной комбинаторики до современной криптографии. Т. 1. Основные структуры. Методы перечисления. Булевы функции / Ю.А. Зуев. – Москва: Книжный дом «ЛИБРОКОМ», 2012.
2. Иванов, М.К. Алгоритмический тренинг. Решения практических задач на Python и C++ / М.К. Иванов. – Санкт-Петербург: БХВ-Петербург, 2023.
3. Лабораторные работы по дискретной математике: методические указания / О.А. Пихтилькова, Т.М. Отрыванкина, Л.Б. Усова, Д.У. Шакирова. – Оренбург: ОГУ, 2018.
4. Рафгарден, Т. Совершенный алгоритм. Графовые алгоритмы и структуры данных / Т. Рафгарден. – Санкт-Петербург: Питер, 2020.
5. Скиена, С.С. Алгоритмы. Руководство по разработке / С.С. Скиена. – Санкт-Петербург: БХВ-Петербург, 2022.
6. Судоплатов, С.В. Дискретная математика: учебник и практикум для вузов / С.В. Судоплатов, Е.В. Овчинникова. – Москва: Юрайт, 2024.
7. Шень, А. Программирование: теоремы и задачи / А. Шень. – Москва: МЦНМО, 2021.
8. Шишмарёв, Ю.Е. Дискретная математика: сборник задач / Ю.Е. Шишмарёв, Е.Д. Емцева, К.С. Солодухин. – Владивосток: Изд-во ВГУЭС, 2002. – Ч. 1.
9. Шишмарев, Ю.Е. Дискретная математика: конспект лекций / Ю.Е. Шишмарев. – Владивосток: Изд-во ВГУЭС, 2007. – Ч. 1.
10. CATS: система автоматизации обучения. – URL: https://imcs.dvfu.ru/cats/
11. Codeforces. – URL: https://codeforces.com/
12. Материалы к лабораторным работам. – URL: https://github.com/lapkin25/discr-alg

Учебное издание

Гренкин Глеб Владимирович
Клочкова Ольга Ивановна

ДИСКРЕТНАЯ МАТЕМАТИКА В АЛГОРИТМАХ

Электронное учебное пособие

Редактор И.Г. Шабунина
Компьютерная верстка М.А. Портнова

Подписано к использованию: 15.10.2025. Формат 60×84/16
Объем 1,76 МБ. Усл.-печ. л. 5,0. Уч.-изд. л. 3,0
Тираж 100 экз. (I–50).

Издательство Владивостокского государственного университета

690014, Владивосток, ул. Гоголя, 41