Конспект курса: Основы дискретной математики

Е.В. Просолупов.

29th May 2009

Содержание

1 Элементы теории множеств. Комбинаторика.

1.1 Введение

Понятие "дискретная математика" объединяет все разделы математики, которые работают с дискретными структурами. Конечно, между ними есть взаимосвязи, местами они ссылаются друг на друга, но все вместе это является единой теорией в той же степени, в какой единой является и вся математика в целом. Это совокупность самостоятельных теорий и направлений.

Теория множеств и комбинаторика содержат базу, которые используются во многих других разделах математики (в том числе и дискретной).

Теория алгоритмов занимается формализацией понятия алгоритма, изучением его закономерностей и свойств. Много внимания уделяется оценке сложности алгоритмов и их сравнению.

Теория графов изучает один из видов дискретных структур — графы. В силу своей интуитивной понятности и наглядному визуальному представлению, графы очень часто используются в качестве моделей для структур самой разной природы.

Математическая логика изучает фундаметальные структуры, которые лежат в основе правильных математических рассуждений.

Дискретная математика содержит и много других подразделов, которые не вошли в рамки этого ознакомительного курса.

1.2 Примеры задач.

На первом занятии познакомимся с примерами задач, которые входят в рассмотрение дискретной математики.

1.2.1 Задача о расположении конвертов

Рассмотрим задачу, для решения которой не потребуется знания специальной теории. Многие достаточно подготовленные читатели могут еще в процессе рассуждений догадаться, каков ответ на задачу, но мы все-таки пройдем все стадии, чтобы лучше понять сам процесс решения.

Задача 1.2.1 . *Имеется n квадратных конвертов разного размера (нет двух одинаковых конвертов). Все конверты находятся в ящике, причем одни конверты можно вкладывать в другие. Сколько существует способов размещения конвертов в ящике без учета их пространственного положения?*

Пример 1.2.1 . *Пусть имеется три конверта размера l_1, l_2 и l_3, причем $0 < l_3 < l_2 < l_1$. На рисунке 1 изображены все возможные*

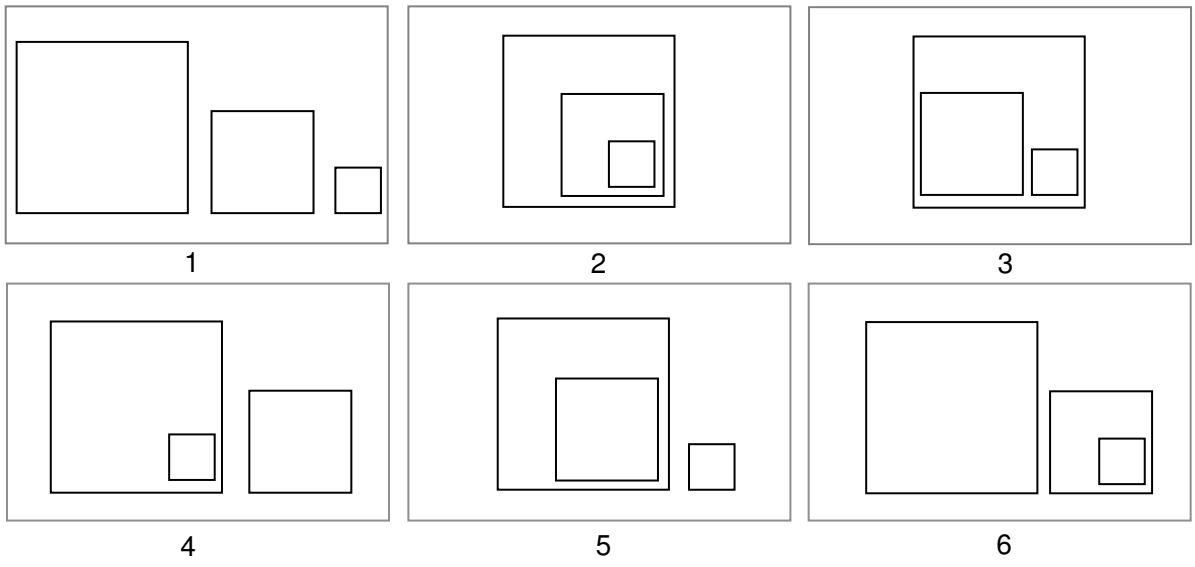

Рисунок 1: Все возможные относительные расположения трех конвертов в ящике

расположения трех конвертов. Всего их оказывается шесть.

Русский язык, как и любой другой разговорный язык, допускает различные понимания одной и той же последовательности слов (символов). Этим он плох для формулировки задачи. Необходимо поставить задачу формально (строго).

Пусть 0 — ящик. Упорядочим конверты по убыванию размера и сопоставим 1 самому большому конверту, 2 — следующему и так далее. Самый маленький конверт получит номер n. Тогда любому размещению конвертов в ящике можно сопоставить пару (n, A), где $n \in \mathbb{N}$ — число конвертов, а A некоторое множество пар (i, j), $i \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$, $j \in \mathbb{N}$, $i, j \leq n$. Если $(i, j) \in A$, то конверт j при нашем размещении непосредственно вложен в конверт i.

Пример 1.2.2 . *Например размещению 2 на рисунке 1 соответствует множество* $A = \{(0, 1), (1, 2), (2, 3)\}$*, а размещению 6 того же рисунка — множество* $A = \{(0, 1), (0, 2), (2, 3)\}$.

Не любому множеству пар A соответствует некоторое размещение конвертов. Введем дополнительные ограничения на A:
1) $i \in \mathbb{N},\ i \leq n \quad \Rightarrow \quad |\{(k, i) :\ (k, i) \in A\}| = 1$ — любой конверт непосредственно вложен ровно в один другой конверт или ящик.
2) $(i, j) \in A \ \Rightarrow \ i < j$ — меньший конверт вкладывается в больший.

Задача 1.2.2 . *Найти функцию натурального аргумента* $n \in \mathbb{N}$

$$f(n) = |\{A :\ A \subseteq \{0, 1, ..., n-1\} \times \{1, 2, ..., n\},$$

$$A \text{ удовлетворяет условиям 1) и 2)}\}|$$

Итак, мы сформулировали строгую постановку задачи и знаем, что нам нужно найти. Тем не менее ответить на поставленный вопрос сразу не так просто. Даже если мы угадаем ответ сейчас, его доказательство может оказаться достаточно сложным для понимания. Чтобы избежать необходимости непосредственно выяснять количество возможных пар (n, A), сопоставим каждой такой паре графическое изображение - *возрастающее дерево.*

Пример 1.2.3 . *Пусть у нас есть некоторое расположение* $(n, A) = (5, \{(0, 1), (0, 2), (0, 4), (1, 3), (1, 5)\})$. *Сопоставим ему возрастающее*

дерево (рис. 2): $\{0, 1, 2, 3, 4, 5\}$ *— множество вершин (узлов), 0 — корень, стрелки — ориентированные ребра (дуги), 1 — внутренний узел, 2, 3, 4, 5 — листья. Узел 0 называют родительским (или отцом) для узлов 1, 2 и 4, которые называют детьми (сыновьями) узла 0. Узел 1 в свою очередь является отцом узлов 3 и 5.*

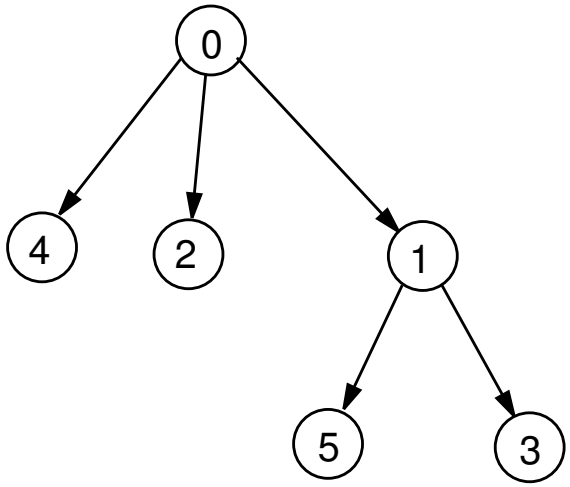

Рисунок 2: Пример возрастающего дерева

Особенности возрастающих деревьев: 1) В любой узел из корня можно попасть двигаясь по стрелкам и притом единственным образом; 2) при движении от корня по стрелкам номера узлов только возрастают.

Формально возрастающее дерево можно определить следующим образом:

Определение 1.2.1 . *1) Пусть x — некоторый узел, помеченный числом n_x. Тогда можно говорить, что x возрастающее дерево, состоящее из одного узла, и узел x является корнем этого дерева.*
2) Пусть x — некоторый узел, помеченный числом n_x, и T_1, T_2,..., T_k — возрастающие деревья с корнями соответственно x_1, x_2, ..., x_k; узел x_i помечен числом n_{x_i}, $i = \overline{1,k}$. Если $n_x < n_{x_i}$ для любого $i = \overline{1,k}$, то можно построить возрастающее дерево $x(T_1, T_2, ..., T_k)$ с корнем x и поддеревьями T_1, T_2,..., T_k.

Пример 1.2.4 . *Возрастающее дерево с рисунка 2 можно записать* $0(4, 2, 1(5, 3))$.

Нетрудно видеть, что любому возрастающему дереву T с узлами $\{0, 1, ..., n\}$ соответствует некоторое размещение n конвертов (n, A), где A состоит из пар (i, j) соединенных в дереве T стрелкой из i в j. Верно и обратное: по размещению конвертов можно построить возрастающее дерево, как это сделано в примере 1.2.3.

Таким образом, наша задача эквивалентна следующей:

Задача 1.2.3 . *Найти число возрастающих деревьев с n вершинами.*

Для определенности будем считать, что все дети одного родительского узла расположены в графическом изображении возрастающего дерева по убыванию слева направо. Так на рисунке 2 вершины 4, 2 и 1 могли бы быть изображены в другом порядке, структура дерева бы не изменилась. Чтобы не путать разные изображения одного дерева, вводим это ограничение.

Рассмотрим еще один способ описания возрастающего дерева — последовательность из $n + 1$ различных элементов из множества $\{0, 1, ..., n\}$.

Определение 1.2.2 . *Назовем перестановкой чисел* $\{1, 2, ..., n\}$ *произвольную упорядоченную последовательность* π *из* n *различных чисел* 1, 2, ..., n*:*

$$\pi \in \{(k_1, k_2, ..., k_n) \mid k_i \in \{1, 2, ..., n\}, i = \overline{1, n};\ k_i \neq k_j,\ \textit{при}\ i \neq j\}$$

Рассмотрим слово τ начинающееся с 0 и продолжающееся некоторой перестановкой чисел $\{1, 2, ..., n\}$: $\tau = (0, k_1, k_2, ..., k_n)$.

Пример 1.2.5 . *Пусть* $\tau = (0, 4, 2, 1, 5, 3)$. *Строим по* τ *возрастающее дерево* T. *Каждое число в* τ *— узел дерева* T*; 0 — корень.*

Пусть $i \neq 0$ *— элемент* τ. *Пусть* j *ближайший к* i *слева элемент* τ, *такой что* $j < i$. *Тогда* j *— отец узла* i. *Нетрудно убедиться, что полученное таким образом по* τ *дерево* T *совпадает с деревом на рисунке 2.*

Итак, чтобы получить по некоторому возрастающему дереву T соответствующее слово τ, нужно каждому листу i сопоставить запись $\tau_i = i$, а каждому родительскому узлу j с детьми i_1, i_2, ..., i_k, $i_1 \geq i_2 \geq ... \geq i_k$, сопоставить запись:

$$\tau_j = j, \tau_{i_1}, \tau_{i_2}, ..., \tau_{i_k},$$

где τ_{i_l} — запись для узла i_l.

Нетрудно видеть, что, поскольку номер любого сына больше номера родительского узла, а номера узлов i_1, i_2, ..., i_k, упорядочены по убыванию, то просматривая список влево от любого i_l мы правильно определим его родителя — j. Тогда $\tau = \tau_0$ и есть искомое представление дерева T.

Пример 1.2.6 . *Построим указанным способом слово τ для возрастающего дерева на рисунке 2. Считаем записи для листьев: $\tau_3 = 3$, $\tau_5 = 5$, $\tau_2 = 2$, $\tau_4 = 4$. Теперь можно получить записи для родительских узлов: $\tau_1 = 1, \tau_5, \tau_3 = 1, 5, 3$, $\tau_0 = 0, \tau_4, \tau_2, \tau_1 = 0, 4, 2, 1, 5, 3$. Мы получили, что $\tau = (0, 4, 2, 1, 5, 3)$, что соответствует слову τ из примера 1.2.5.*

Таким образом, по дереву T мы однозначно строим слово τ, состоящее из нуля с последующей перестоновкой чисел $\{1, 2, ..., n\}$. С другой стороны по слову τ однозначно восстанавливается дерево T способом, описанным в примере 1.2.5. Этим мы показали, что каждому возрастающему дереву T взаимнооднозначно соответствует некоторая перестановка чисел $\{1, 2, ..., n\}$, а наша задача сводится к следующей:

Задача 1.2.4 . *Найти число переставновок чисел $\{1, 2, ..., n\}$.*

Последнее число равно $n!$, что нетрудно доказать, например, по индукции.

Итак, на примере задачи о расположении конвертов мы разобрали следующие стадии решения задачи:
1) формулировка задачи из предметной области;
2) формализация условий задачи и построение математической модели;

3) последовательное сведение исходной задачи к вспомогательным задачам;
4) получение ответа путем решения вспомогательной задачи.

1.2.2 Задача о Ханойской башне

Рекомендуемая литература: [2]

Рассмотрим еще одну интересную задачу и отметим на ее примере некоторые другие этапы решения.

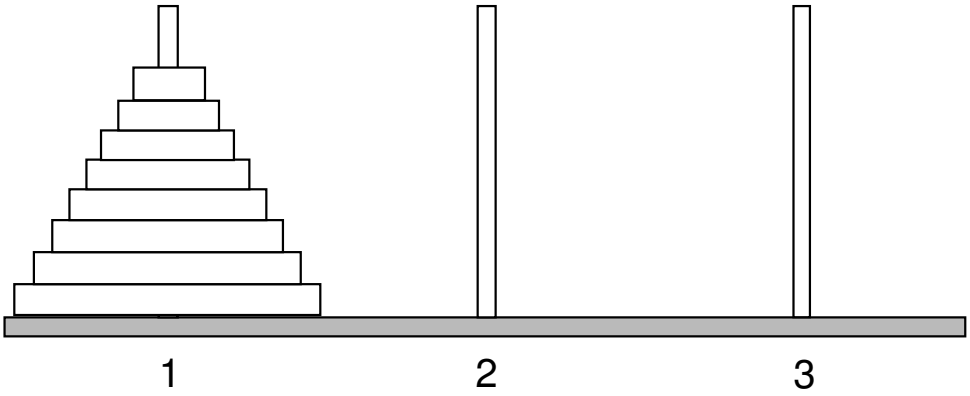

Рисунок 3: Ханойская башня

Известная головоломка *ханойская башня* изображена на рисунке 3. Головоломка состоит из трех колышков и некоторого числа дисков разного диаметра. Каждый диск имеет в середине отверстие для надевания на колышки.

В исходный момент все диски нанизаны на один из колышков в порядке уменьшения размера. Диски можно перемещать с колышка на колышек при соблюдении двух условий: за один раз можно перенести только один диск и больший диск нельзя помещать на меньший. Задача состоит в том, чтобы, при соблюдении правил перемещения, перенести все диски с исходного на какой-нибудь другой колышек.

Довольно быстро можно убедиться, что задача разрешима. Поставим вопрос, какой способ решения головоломки является оптимальным.

Задача 1.2.5 . *Каково наименьшее число перекладываний T_n, которое*

необходимо совершить, чтобы переместить в рамках задачи о ханойской башне пирамиду из n дисков с одного колышка на другой.

Рассмотрим *крайние случаи.* Довольно часто ответ на задачу легко находится в частных случаях, при малых значениях аргумента. Такие, казалось бы, очевидные случаи могут помочь лучше понять общие закономерности. Так, совершенно очевидно, что $T_1 = 1$, и не сложно убедиться, что $T_2 = 3$. Так же можно положить $T_0 = 0$, поскольку для перемещения пирамиды из 0 дисков не требуются перекладывания.

Теперь попытаемся понять, как же переместить пирамиду в общем случае. Если предположить, что изначально диски находятся на колышке с номером 1, для двух дисков нам требуется переложить меньший диск на колышек 2, затем больший на колышек 3 и наконец меньший диск с колышка 2 на колышек 3.

Для трех дисков задача усложняется, но идея остается той же. Мы можем переложить, как мы уже умеем, два диска на колышек номер 2, затем переложить большой диск на колышек 3 и в завершение переложить два меньших диска с колышка 2 на колышек 3.

Итак мы можем переместить пирамиду любой высоты n, если воспользуемся следующим алгоритмом: 1) переложить верхние $n-1$ дисков на второй колышек; 2) переложить диск n на третий колышек; 3) переложить $n-1$ меньший диск со второго колышка на третий. Таким образом, для перемещения n дисков нам достаточно $T_{n-1} + 1 + T_{n-1}$ перекладываний:

$$T_n \leq 2T_{n-1} + 1, \quad n > 0.$$

Кроме того, когда-нибудь нам несомненно потребуется переложить самый большой диск с первого колышка на то место, где мы снова соберем нашу пирамиду, например на третий колышек. В этот момент все меньшие диски должны лежать на колышке 2. Значит мы не сумеем переместить башню быстрее, чем за время равное $2T_{n-1} + 1$.

$$T_n \geq 2T_{n-1} + 1, \quad n > 0.$$

Таким образом мы получили рекурентное соотношение

$$\begin{aligned} &T_0 = 0, \\ &T_n = 2T_{n-1} + 1, \quad n > 0. \end{aligned} \tag{1}$$

Равенства (1) дают возможность посчитать значение T_n для любого натурального числа n. В частности уже известные нам $T_1 = 1$ и $T_2 = 3$ согласуются с этими формулами. С другой стороны использовать рекурентность для получения значений T_n не удобно при достаточно больших n. Мы бы хотели получить *замкнутую форму* выражения для T_n.

Как же решить рекурентное соотношение и получить его замкнутую форму? Для начала попробуем просто угадать. Здесь как и раньше нам помогут крайние случаи. Воспользуемся рекурентным соотношением (1) и вычислим T_n для нескольких первых натуральных чисел:

$$\begin{aligned} T_3 &= 2 \cdot 3 + 1 = 7, \\ T_4 &= 2 \cdot 7 + 1 = 15, \\ T_5 &= 2 \cdot 15 + 1 = 31, \\ T_6 &= 2 \cdot 31 + 1 = 63. \end{aligned}$$

Если хорошенько присмотреться, можно догадаться, что это напоминает: это похоже на степени двойки, уменьшенные на один. Можно предположить, что $T_n = 2^n - 1$ для любых $n \geq 0$.

Действительно, это выражение довольно просто доказать по индукции. Оно выполняется для T_0 и T_1. Предположим, что $T_{n-1} = 2^{n-1} - 1$. Тогда $T_n = 2T_{n-1} + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 1$, что и требовалось доказать.

Метод математической индукции очень часто помогает в решении проблем, но он потребовал от нас "угадать" ответ. Когда это возможно, хорошо получить результат напрямую, не полагаясь на везение. Продемонстрируем, как соотношение (1) может быть разрешено без использования индукции.

Интересно, как помогает "упростить" второе выражение в (1) добавление единицы с обоих сторон от знака равенства: $T_n + 1 = 2T_{n-1} + 2 = 2(T_{n-1} + 1)$. Введем вспомогательную величину $U_n = T_n + 1$. Тогда из (1) имеем

$$\begin{aligned} U_0 &= 1, \\ U_n &= 2U_{n-1}, \quad n > 0. \end{aligned}$$

Очевидно, что $U_n = 2^n$. Отсюда, по определению величины U_n следует, что $T_n = 2^n - 1$ для любых $n > 0$.

Итак, на примере задачи о ханойской башне мы разобрали следующие стадии решения задачи:
1) рассмотрение крайних случаев;
2) нахождение и доказательство математического выражения для интересующей величины (в нашем случае это рекурентное выражение);
3) нахождение и доказательство замкнутой формы для математического выражения.

1.2.3 Базовые обозначения

Введем некоторые общие обозначения, которые будут встречаться нам в процессе изучения всего материала курса.

Под записью $i = \overline{k,l}$ будем понимать, что величина i пробегает все целые значения от k до l.

$\lceil x \rceil$ - наименьшее большее целое для числа x.

$\lfloor x \rfloor$ - наибольшее меньшее целое для числа x.

Наряду со стандартными обозначениями для множеств чисел
$\mathbb{N}$ — множество натуральных чисел,
$\mathbb{Z}$ — множество целых чисел,
$\mathbb{R}$ — множество вещественных чисел,
будем использовать $\mathbb{N}_0$ для обозначения множества неотрицательных целых чисел:

$$\mathbb{N}_0 = \mathbb{N} \cup \{0\} = \{0, 1, 2, ...\}.$$

Для записи сумм с большим количеством подобных слагаемых используется символ Σ (сигма).

$$\sum_{i=k}^{l} f(i) = f(k) + f(k+1) + ... + f(l-1) + f(l).$$

Когда условие суммирования имеет более сложную форму, используют следующую запись

$$\sum_{P(x)} f(x)$$

которая значит, что суммирование происходит по всем возможным значениям параметра x для которых высказывание $P(x)$ является верным. Аналогично могут использоваться и другие знаки групповых операций.

Пример 1.2.7 .

$$\sum_{i=1}^{10} i^2 = 1 + 4 + 9 + 16 + 25 + 36 + 49 + 64 + 81 + 100 = 385.$$

$$\sum_{\substack{x_1, x_2 \in \mathbb{N}, \\ x_1 + x_2 = 10}} x_1 \cdot x_2 = 9 + 16 + 21 + 24 + 25 + 24 + 21 + 16 + 9 = 155.$$

Символы $\forall$, $\exists$, $\vee$, $\wedge$, $\neg$ часто используются в краткой записи математических высказываний: $\forall$ — "для любых", $\exists$ — "существует", $\nexists$ — "не существует", $\exists!$ — "существует единственный", $\wedge$ — "и", $\vee$ — "или", $\neg$ — "не". В разделе нашего курса, посвященном математической логике мы подробнее изучим запись высказываний на формальном языке.

1.2.4 Правило суммы и правило произведения

При подсчете числа вариантов возникновения различных событий в задачах комбинаторики будут полезны следующие два правила.

Правило произведения: Если объект A может быть выбран m способами и для каждого из таких выборов объект B в свою очередь может быть выбран n способами, то выбор «A и B» может быть осуществлен $m \cdot n$ способами.

Пример 1.2.8 . *Бросают две игральные кости. Сколько существует различных вариантов их выпадения, если кости считаются различными?*

У каждой кости 6 граней. Для каждого варианта выпадения первой кости возможно 6 вариантов выпадения второй. Значит, по правилу произведения, искомое число $6 \cdot 6 = 36$.

Правило суммы: Если объект A может быть выбран m способами, а объект B — n способами, причем одновременный выбор A и B невозможен, то выбор «A или B» можно осуществить $m + n$ способами.

Пример 1.2.9 . *Бросают две различные игральные кости. Сколько существует вариантов выпадения, чтобы ровно на одной из двух выпала шестерка?*

Если шестерка выпадает на первой кости, то существует 5 вариантов допустимых значений для выпадения второй кости. Аналогично, если шестерка выпадает на второй кости, первая кость в подсчитываемых комбинациях может принимать значения 1,2, 3, 4 и 5. Поскольку одновременное выпадение шестерок не рассматривается, по правилу суммы искомое число $5 + 5 = 10$.

Пример 1.2.10 . *Бросают две игральные кости. Сколько существует вариантов выпадения, чтобы либо на каждой кости выпало четное число очков, либо на каждой — нечетное?*

Поскольку кости не могут одновременно дать и четный и нечетный результат, по правилу суммы нужно сложить число вариантов выпадения двух четных числе и число выпадения двух нечетных чисел.

Обычная игральная кость имеет шесть граней и ровно три из них четные. Так как для каждого варианта выпадения значения на первой кости возможны три варианта значения на второй, по правилу произведения число вариантов выпадения двух четных значений равно $3 \cdot 3 = 9$. *Аналогично, число выпадений двух нечетных чисел —* 9. *Тогда, искомое число равно* 18.

1.3 Основы теории множеств

Рекомендуемая литература: [1]

1.3.1 Понятие множества

Определение 1.3.1 . *Множество — любая совокупность определенных и различных между собой объектов, мыслимая как единое целое. Эти объекты называются элементами множества.*

Символ $\in$ обозначает отношение принадлежности. Запись $x \in S$ значит, что элемент x принадлежит множеству S (x является элементом S). Запись $x \notin S$ означает, что в множестве S нет элемента x.

Множество не содержащее элементов обозначают $\varnothing$. Такое множество называют *пустым множеством*

Обозначают $\{a_1, a_2, ..., a_k\}$ — множество, элементами которого являются $a_1, a_2, ..., a_k$ и только они.

Пусть $P(x)$ некоторая последовательность символов, задающая высказывание о x, которое будет принимать истинное или ложное значение при подстановке везде в нем вместо символа x одного и того же элемента.

Будем обозначать $\{x \mid P(x)\}$ — множество всех элементов, для которых высказывание $P(x)$ принимает истинное значение.

Пример 1.3.1 . *1)* $\{x \mid x$ *— положительное число меньше 6*$\} = \{1, 2, 3, 4, 5\}$*;*
2) $\{x \mid x = 2y,\ y \in \mathbb{N}\} = \{x \mid x$ *— четное число*$\}$.

Множества считаются равными, если состоят из одних и тех же элементов. Записывают $A = B$, если множества A и B равны, и $A \neq B$ — иначе.

Пример 1.3.2 . *Множество A всех положительных четных чисел равно множеству B положительных целых чисел, представимых в виде суммы двух нечетных чисел. (Показать)*

Пример 1.3.3 . *1)* $\{2,4,6\} = \{4,2,6\}$.
2) $\{x \mid x^2 + 2x = 0\} = \{0,-2\}$.
3) $\{a,b\} \neq \{\{a,b\}\}$.

Мощностью множества называется число его элементов. Мощность множества A обозначают $|A|$.

Пример 1.3.4 . *1)* $|\{1,2,3,4,5\}| = 5$;
2) $|\{1,2,3,2\}| = 3$;
3) $|\mathbb{N}| = \infty$.

1.3.2 Парадокс Рассела

Теория множеств в ее интуитивном изложении может приводить к парадоксам. Рассмотрим парадокс Б. Рассела.

Можно предположить существование множеств, которые принадлежат сами себе, и множества, которые не являются собственными элементами. Рассмотрим множество A всех таких множеств X которые не являются элементами X (сами себя):

$$A = \{X \mid X \notin X\}.$$

Принадлежит ли A само себе, как элемент. Если $A \in A$, то по определению этого множества получим $A \notin A$. Если предположить, что $A \notin A$, то оказывается $A \in A$. В любом случае получается, что $A \in A$ и $A \notin A$?! — противоречие.

1.3.3 Подмножества

Обозначают $A \subseteq B$ и говорят, что множество A является подмножеством множества B, если каждый элемент множества A является элементом множества B. Если также известно, что $A \neq B$, говорят, что A является собственным подмножеством множества B и пишут $A \subset B$.

Пример 1.3.5 . *1)* $\{1,3\} \subset \{5,3,1\}$;
2) $X \subseteq X$ *для любого множества* X *(рефлексивность);*
3) $X \subseteq Y$, $Y \subseteq Z \Rightarrow X \subseteq Z$ *(транзитивность);*
4) $\varnothing \subseteq X$ *для любого множества* X.

Множество всех подмножеств множества A будем обозначать 2^A. Мощность множества всех подмножеств множества с n элементами равна 2^n: $|2^A| = 2^{|A|}$.

Пример 1.3.6 . *Пусть* $A = \{a, b, c\}$. *Тогда* $2^A = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, A\}$. *Видно, что в этом множестве* $2^{|A|} = 2^3 = 8$ *элементов.*

1.3.4 Операции над множествами

Пусть U — универсальное множество — множество всех элементов рассматриваемой предметной области.

Объединение

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

Пересечение

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

Дополнение (абсолютное дополнение)

$$\overline{A} = \{x \mid x \in U \text{ и } x \notin A\}.$$

Разность (относительное дополнение)

$$A \setminus B = \{x \mid x \in A \text{ и } x \notin B\}.$$

Очевидными свойсвами объединения, пересечения и разности множеств являются то, что для любых двух множеств A и B выполняются включения:

$$(A \cap B) \subseteq A \subseteq (A \cup B),$$
$$A \setminus B = A \cap \overline{B} \subseteq A.$$

Симметрическая разность

$$A \dot{-} B = (A \setminus B) \cup (B \setminus A).$$

Утверждение 1.3.1 (Основные тождества алгебры множеств). *Для любых подмножеств* A, B *и* C *универсального множества* U *выполняются следующие тождества:*

1) Коммутативность: a) $A \cup B = B \cup A$; *b)* $A \cap B = B \cap A$;
2) Ассоциативность:
a) $A \cup (B \cup C) = (A \cup B) \cup C$; *b)* $A \cap (B \cap C) = (A \cap B) \cap C$;
3) a) Дистрибутивность $\cup$ *относительно* $\cap$*:*
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$;
b) Дистрибутивность $\cap$ *относительно* $\cup$*:*
$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$;
4) a) $A \cup \varnothing = A$; *b)* $A \cap \varnothing = \varnothing$;
5) a) $A \cup U = U$; *b)* $A \cap U = A$;
6) a) $A \cup \overline{A} = U$; *b)* $A \cap \overline{A} = \varnothing$;
7) Идемпотентность: a) $A \cup A = A$; *b)* $A \cap A = A$;
8) Законы де Моргана: a) $\overline{A \cup B} = \overline{A} \cap \overline{B}$; *b)* $\overline{A \cap B} = \overline{A} \cup \overline{B}$;
9) Законы поглощения: a) $A \cup (A \cap B) = A$; *b)* $A \cap (A \cup B) = A$.
10) Инволютивный закон: $\overline{\overline{A}} = A$.

Доказательство. Доказать утверждения самостоятельно.
□

Следствие 1.3.2 .

$$A \dot{-} B = (A \cup B) \setminus (B \cap A).$$

Доказательство.

$$\begin{aligned} A \dot{-} B &= (A \setminus B) \cup (B \setminus A) = (A \cap \overline{B}) \cup (B \cap \overline{A}) = \\ &= ((A \cap \overline{B}) \cup B) \cap ((A \cap \overline{B}) \cup \overline{A}) = \\ &= (A \cup B) \cap (\overline{B} \cup B) \cap (A \cup \overline{A}) \cap (\overline{B} \cup \overline{A}) = \\ &= (A \cup B) \cap (\overline{B} \cup \overline{A}) = (A \cup B) \cap (\overline{B \cap A}) = \\ &= (A \cup B) \setminus (B \cap A) \end{aligned}$$

□

Утверждение 1.3.3 . *Для любых множеств* A *и* B *следующие предположения попарно эквивалентны:*
1) $A \subseteq B$;
2) $A \cap B = A$;

3) $A \cup B = B$*;*
4) $\overline{A} \cup B = U$*;*
5) $A \setminus B = \varnothing$*.*

Доказательство. 1) $\Rightarrow$ 2) Очевидно $A \cap B \subseteq A$. Покажем $A \subseteq A \cap B$. Действительно $x \in A \Rightarrow x \in B$, так как $A \subseteq B$ и, следовательно, $x \in A \cap B$.

2) $\Rightarrow$ 3) $A \cap B = A$ следовательно $(A \cap B) \cup B = A \cup B$. По закону поглощения и коммутативности $(A \cap B) \cup B = B$. Таким образом $A \cup B = B$

3) $\Rightarrow$ 1) По предположению $A \cup B = B$. Тогда $A \subseteq A \cup B = B$, что и требовалось доказать.

3) $\Rightarrow$ 4) В объединении множеств $\overline{A}$ и B сделаем замену согласно свойству $A \cup B = B$:

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

4) $\Rightarrow$ 5) Возьмем дополнение от предположения пункта 4) и воспользуемся правилом де Моргана:

$$\varnothing = \overline{U} = \overline{\overline{A} \cup B} = A \cap \overline{B} = A \setminus B.$$

5) $\Rightarrow$ 3) Пусть $\varnothing = A \setminus B$. Объединим левую и правую части этого равенства с B:

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

□

1.3.5 Диаграммы Эйлера-Венна

Диаграммы Эйлера-Венна помогают наглядно проиллюстрировать многие соотношения между множествами. На рисунке 32 показано применение диаграмм Эйлера-Венна для наглядного изображения основных операций над множествами.

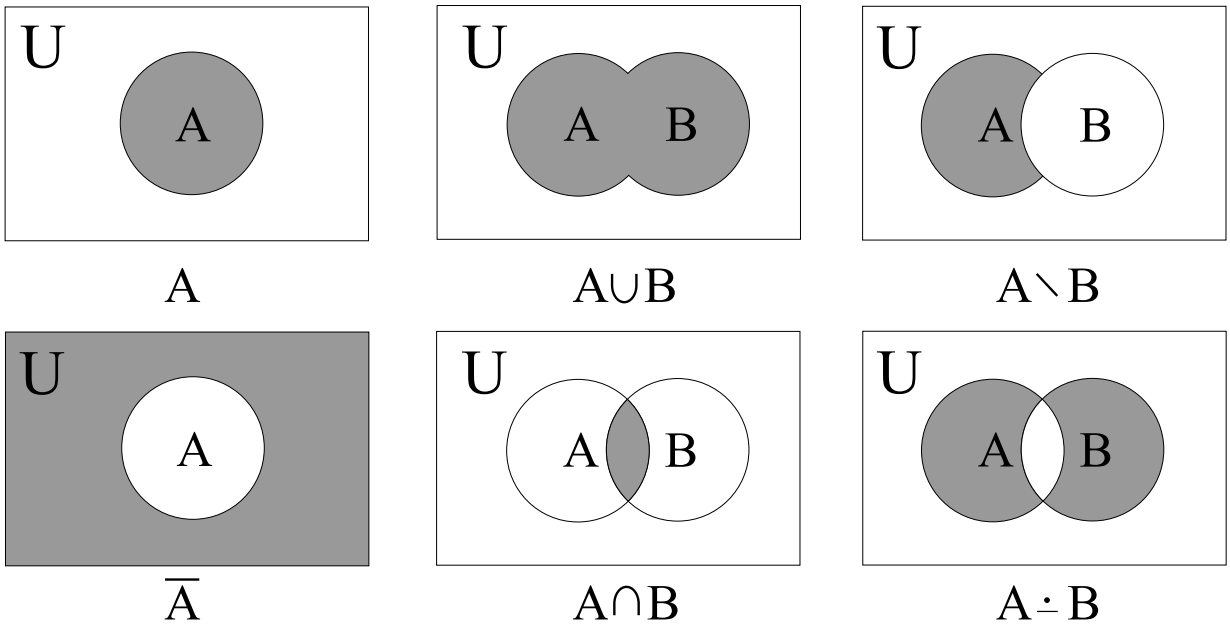

Рисунок 4: Использование диаграмм Эйлера-Венна для иллюстрации основных операций над множествами

1.3.6 Прямое произведение множеств

Пусть $A = \{a_1, a_2, ..., a_k\}$, $B = \{b_1, b_2, ..., b_l\}$

$$A \times B = \{(a_1, b_1), (a_1, b_2), ..., (a_1, b_l), (a_2, b_1), ..., (a_k, b_l)\} = \\ = \{(a, b) \mid a \in A,\ b \in B\}$$

— прямое (декартово) произведение множеств A и B.

Во многих случаях порядок проведения операций произведения не важен. Тогда считают, что

$$A \times (B \times C) = (A \times B) \times C = \{(a, b, c) \mid a \in A,\ b \in B,\ c \in C\}.$$

Аналогично определяется прямое произведение любого конечного числа множеств.

$$A_1 \times A_2 \times ... \times A_t = \{(a_1, a_2, ..., a_t) \mid a_i \in A_i,\ i = \overline{1, t}\}$$

Если речь будет идти о прямом произведении множества на себя, то будем заменять запись нескольких множеств, на знак степени:

$$\underbrace{A \times A \times ... \times A}_{t} = A^t.$$

1.4 Бинарные отношения и функции

1.4.1 Бинарные отношения

Определение 1.4.1 . *Бинарным (двуместным) отношением ρ называется множество упорядоченных пар. Если некоторая пара (x, y) принадлежит отношению ρ пишут $(x, y) \in \rho$ или $x\rho y$.*

Замечание 1.4.1 . *n-арным отношением называют множество упорядоченных n-ок. Любое множество можно назвать унарным отношением.*

Областью определения бинарного отношения ρ называется множество $D_\rho = \{x \mid$ существует такое y, что $x\rho y\}$.

Областью значений бинарного отношения ρ называется множество $R_\rho = \{y \mid$ существует такое x, что $x\rho y\}$

Пример 1.4.1 . *1)* $\rho = \{(1, 2), (2, 3), (1, 5), (3, 3)\}$. *Тогда* $D_\rho = \{1, 2, 3\}$, $R_\rho = \{2, 3, 5\}$.
2) $\{(x, x) \mid x$ *— вещественное число*$\}$. *Тогда* $D_\rho = R_\rho = \mathbb{R}$.
3) $\{(x, y) \mid x, y$ *— целые числа и найдется такое целое число* z, *что* $x + z = y\}$. *Тогда* $D_\rho = R_\rho = \mathbb{Z}$.

Каждое бинарное отношение является подмножеством прямого произведения множеств X и Y таких, что $D_\rho \subseteq X$ и $R_\rho \subseteq Y$.

Обратным отношением для отношения ρ называют отношение

$$\rho^{-1} = \{(x, y) \mid (y, x) \in \rho\}.$$

Композицией отношений ρ_1 и ρ_2 называется отношение

$$\rho_2 \circ \rho_1 = \{(x, z) \mid \text{существует такое } y, \text{ что } x\rho_1 y \text{ и } y\rho_2 z\}.$$

Утверждение 1.4.1 . *Для любых бинарных отношений ρ, ρ_1, ρ_2 выполняются следующие свойства:*
1) $(\rho^{-1})^{-1} = \rho$;
2) $(\rho_2 \circ \rho_1)^{-1} = \rho_1^{-1} \circ \rho_2^{-1}$

Доказательство. Пункт 1 непосредственно следует из определения обратного отношения. Покажем истинность пункта 2:

$$(\rho_2 \circ \rho_1)^{-1} = \{(z,x) \mid \exists y:\ x\rho_1 y, y\rho_2 z\} = \\ = \{(z,x) \mid \exists y:\ y\rho_1^{-1} x, z\rho_2^{-1} y\} = \rho_1^{-1} \circ \rho_2^{-1}$$

□

1.4.2 Функции

Определение 1.4.2 . *Бинарное отношение f называется функцией, если из $(x,y) \in f$ и $(x,z) \in f$ следует, что $y = z$.*

Две функции равны, если они состоят из одних и тех же элементов (как и любые множества).

Иногда приходится сталкиваться с трудностями в определении области значений функций. Тогда, если $D_f = X$ и $R_f \subseteq Y$, то говорят, что функция f задана на множестве X со значениями в множестве Y (осуществляет отображение множества X во множество Y), и пишут $f:\ X \to Y$.

Если f — функция, вместо $(x,y) \in f$ пишут $f(x) = y$ и говорят, что y — значение соответствующее аргументу x (образ элемента x). x называют прообразом элемента y.

Пример 1.4.2 . *1)* $\{(1,2),(2,3),(48,\star),(\square,\triangle)\}$ *— функция;*
2) $\{(1,2),(1,3),(2,4)\}$ *— не функция;*
3) $\rho\{(x, x^2+2x+1) \mid x$ *— вещественное число*$\}$ *— функция* $y = x^2 + 2x + 1$. *Обратное бинарное отношение* ρ^{-1} *в этом случае функцией не является.*

Пусть $f:\ X \to Y$.

Определение 1.4.3 . *Функцию (отображение) f назовем инъективной фукнцией (инъекцией), если для любых $x_1, x_2 \in X$ и любого $y \in Y$ из $y = f(x_1)$ и $y = f(x_2)$ следует, что $x_1 = x_2$.*

Определение 1.4.4 . *Функция f называется сюръективной функцией (сюръекцией), если для любого элемента $y \in Y$ существует элемент $x \in X$ такой, что $y = f(x)$.*

Определение 1.4.5 . *Функция f называется биективной функцией (биекцией), если она и сюръективна, и инъективна.*

Если указана биективная функция $f: X \to Y$, говорят. что осуществляется взаимнооднозначное соответствие между множествами X и Y.

Пример 1.4.3 . *Рассмотрим функции $f: \mathbb{R} \to \mathbb{R}$.*
1) $f(x) = e^x$ инъективна, но не сюръективна (рис. 5-a));
2) $f(x) = x^3 - x$ сюръективна, но не инъективна (рис. 5-b));
3) $f(x) = 2x + 1$ биективна.

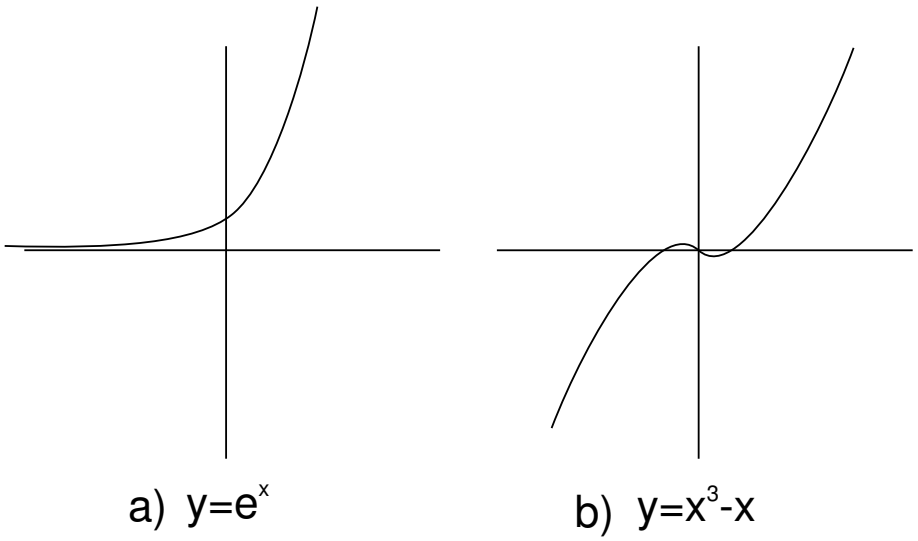

Рисунок 5: Примеры функций

Утверждение 1.4.2 . *Композиция двух функций есть функция. При этом, если $f: X \to Y$ и $g: Y \to Z$, то $g \circ f: X \to Z$.*

Доказательство. Пусть $(x, z_1) \in g \circ f$ и $(x, z_2) \in g \circ f$. Тогда существуют y_1 и y_2: $z_1 = g(y_1)$ и $z_2 = g(y_2)$. При этом $y_1 = f(x)$ и $y_2 = f(x)$. Поскольку, f — функция, то $y_1 = y_2$. Поскольку g — функция, то $z_1 = g(y_1) = g(y_2) = z_2$. Таким образом, $g \circ f$ — функция.

Покажем, что $g \circ f: X \to Z$. Так как $f: X \to Y$, для любого $x \in X$ существует $y \in R_f \subseteq Y$: $f(x) = y$. Так как $g: Y \to Z$, существует

$z \in R_g \subseteq Z$: $z = g(y) = g(f(x)) = (g \circ f)(x)$. Таким образом, для любого $x \in X$ мы нашли такой $z \in Z$, что $(x, z) \in g \circ f$. Следовательно, $D_{g \circ f} = X$. С другой стороны, $R_{g \circ f} \subseteq R_g \subseteq Z$, что и требовалось доказать.
□

Утверждение 1.4.3 . *Композиция двух биективных функций есть биективная функция.*

Доказательство. Пусть, $f : X \to Y$ и $g : Y \to Z$ — биекции. Пусть, для некоторых $x_1, x_2 \in X$ и $z \in Z$ верно $z = g \circ f(x_1)$ и $z = g \circ f(x_2)$. Пусть, $y_1, y_2 \in Y$ те элементы, для которых $y_1 = f(x_1)$ и $y_2 = f(x_2)$. Тогда, $z = g(y_1) = g(y_2)$. Поскольку g инъективна, $y_1 = y_2$ и, поскольку f инъективна, $x_1 = x_2$. Следовательно, $g \circ f$ — инъективна.

Пусть $z \in Z$. Тогда, поскольку g сюръективна, существует $y \in Y$: $g(y) = z$. Поскольку f сюръективна, существует $x \in X$: $f(x) = y$. Следовательно, $g \circ f(x) = z$ и $g \circ f$ сюръективна. Таким образом, $g \circ f$ — биекция.
□

Пусть f^{-1} — отношение обратное к f. Если f^{-1} осуществляет отображение множества Y во множество X, говорят, что f^{-1} — обратное отображение.

Утверждение 1.4.4 . *Отображение $f : X \to Y$ имеет обратное отображение $f^{-1} : Y \to X$ тогда и только тогда, когда f — биекция. При этом f^{-1} тоже будет биекцией.*

Доказательство. Достаточность. Пусть, $(y, x_1) \in f^{-1}$ и $(y, x_2) \in f^{-1}$. Тогда, $f(x_1) = f(x_2) = y$. Из инъективности функции f следует, что $x_1 = x_2$ и, следовательно, f^{-1} — функция.

Из сюръективности функции f следует, что для любого $y \in Y$ существует $x \in X$: $f(x) = y$. Другими словами, для любого $y \in Y$ существует $x \in X$: $f^{-1}(y) = x$. Следовательно, $D_{f^{-1}} = Y$ и f^{-1} осуществляет отображение из Y в X.

Покажем, что f^{-1} — биекция. Поскольку f определена на всем множестве X, f^{-1} — сюръекция. Пусть существуют такие $y_1, y_2 \in Y$ и $x \in X$, что $x = f^{-1}(y_1) = f^{-1}(y_2)$. Тогда, $(x, y_1) \in f$ и $(x, y_2) \in f$ и, поскольку f функция, $y_1 = y_2$. Следовательно, f^{-1} — инъекция.

Необходимость доказывается из тех же соображений, рассмотренных в обратном порядке: если f^{-1} функция, то f — инъективна; если f^{-1} осуществляет отображение из Y в X, f — сюръективна.
□

Замечание 1.4.2 . *Чтобы обратное отношение f^{-1} было функцией, достаточно инъективности f.*

Тождественным отображением множества X на себя называется отображение $e_X : X \to X$ такое, что для любого $x \in X$ верно $e_X(x) = x$. Если $f : X \to Y$ тогда верно, что $e_Y \circ f = f$ и $f \circ e_X = f$.

Утверждение 1.4.5 . *Если $f : X \to Y$ — биекция, то*
1) $(f^{-1} \circ f) = e_X$*;*
2) $(f \circ f^{-1}) = e_Y$*.*

Доказательство. Для любого $x \in X$, $(f^{-1} \circ f)(x) = f^{-1}(f(x)) = x$. Для любого $y \in Y$, $(f \circ f^{-1})(y) = f(f^{-1}(y)) = y$.
□

1.4.3 Специальные бинарные отношения: Отношение эквивалентности

Говорят, что ρ — бинарное отношение на множестве X, если $\rho \subseteq X \times X$. Для произвольного отношения ρ имеет смысл выбирать $X = D_\rho \cup R_\rho$.

Отношение ρ на множестве X называется *рефлексивным*, если для любого элемента $x \in X$ выполняется $x\rho x$

Отношение ρ на множестве X называется *иррефлексивным*, если для любых $x \in X$ из $(x, x) \notin \rho$.

Отношение ρ на множестве X называется *симметричным*, если для любых $x, y \in X$ из $x\rho y$ следует $y\rho x$

Отношение ρ на множестве X называется *антисимметричным*, если для любых $x, y \in X$ из $x\rho y$ и $y\rho x$ следует $x = y$.

Отношение ρ на множестве X называется *транзитивным*, если для любых $x, y, z \in X$ из $x\rho y$ и $y\rho z$ следует $x\rho z$.

Определение 1.4.6 . *Бинарное отношение ρ на множестве X называется отношением эквивалентности, если оно рефлексивно, транзитивно и симметрично.*

Если ρ — отношение эквивалентности и $x\rho y$, говорят, что x и y эквивалентны.

Пример 1.4.4 . *1) $\rho_= = \{(x, y) \mid x, y \in \mathbb{R},\ x = y\}$ — отношение эквивалентности.*
2) Отношение подобия на множестве треугольников является отношением эквивалентности.
3) Отношение сравнимости по модулю n

$$x\rho y \ \Leftrightarrow\ x \equiv y (\mathrm{mod}\, n)$$

на множестве всех целых чисел $\mathbb{Z}$ является отношением эквивалентности.

Определение 1.4.7 . *Пусть на множестве X введено отношение эквивалентности ρ. Классом эквивалентности, порожденным элементом x, называется подмножество множества X, состоящее из всех элементов эквивалентных x:*

$$[x] = \{y \mid y \in X,\ x\rho y\}.$$

Пример 1.4.5 . *Продолжим пример 1.4.4.*
1) Классы эквивалентности по отношению равенства на множестве вещественных чисел состоят из единственного элемента: $[x] = x$.
2) Класс эквивалентности по отношению подобия треугольников состоит из всех треугольников, подобных порождающему класс.
3) Класс эквивалентности для отношения сравнимости по модулю n *на множестве целых чисел* $\mathbb{Z}$, *порожденный элементом* a, *имеет вид* $\{a + kn \mid k \in \mathbb{Z}\}$. *Очевидно, что числа* $0, 1, 2, ..., n-1$ *порождают различные классы. С другой стороны, для любого числа* $t \in \mathbb{Z}$ *оно представимо в виде* $t = a + kn$, *где* $k \in \mathbb{Z}$, $a \in \{0, 1, ..., n-1\}$. *Значит, для любого целого числа порожденный им класс эквивалентности совпадает с одним из указанных. Таким образом, отношение сравнимости по модулю* n *порождает* n *различных классов эквивалентности:* $[0], [1], ..., [n-1]$.

Утверждение 1.4.6 . *Пусть* ρ — *отношение эквивалентности на множестве* X. *Тогда 1) для любого* $x \in X$ *верно, что* $x \in [x]$;
2) для любых $x, y \in X$, *если* $x \rho y$, *то* $[x] = [y]$ *(класс эквивалентности порождается любым своим элементом).*

Доказательство. Доказательство пункта 1) следует из рефлексивности отношения ρ.

Докажем 2). Пусть $y \rho z$. Тогда в силу транзитивности отношения ρ имеем $x \rho z$ и $z \in [x]$. Следовательно $[y] \subseteq [x]$. В силу симметричности отношения ρ получим $[x] \subseteq [y]$, что и требовалось доказать.
□

Определение 1.4.8 . *Разбиением множества* A *называется совокупность его попарно непересекающихся непустых подмножеств* A_i *таких, что каждый элемент* $x \in A$ *принадлежит одному из этих подмножеств:*

$$\{A_1 \cup A_2 \cup ... \cup A_k\}, \quad A_i \neq \varnothing, i = \overline{1, k},$$

$$A_i \cap A_j = \varnothing, \quad i, j = \overline{1, k}, i \neq j,$$

$$A = A_1 \cup A_2 \cup ... \cup A_k.$$

Утверждение 1.4.7 . *Всякое разбиение множества X определяет на X отношение эквивалентности ρ: $x\rho y$ тогда и только тогда, когда x и y принадлежат одному подмножеству разбиения.*

Доказательство. Рефлексивность и симметричность очевидны. Покажем транзитивность. Пусть $x\rho y$ и $y\rho z$. Тогда $x, y \in X_1$ и $y, z \in X_2$, где X_1 и X_2 — подмножества разбиения X. Поскольку $y \in X_1$ и $y \in X_2$, то $X_1 = X_2$. Таким образом $x, z \in X_1$ и $x\rho z$.
□

Утверждение 1.4.8 . *Всякое отношение эквивалентности ρ определяет разбиение множества X на классы эквивалентности по этому отношению.*

Доказательство. Из утверждения 1.4.6 следует, что каждый элемент множества X принадлежит некоторому классу эквивалентности. В то же время, из того же утверждения следует, что любые два класса эквивалентности либо не пересекаются, либо совпадают, если имеют хоть один общий элемент: $z \in [x]$, $z \in [y] \Rightarrow x\rho z$, $z\rho y \Rightarrow x\rho y \Rightarrow [x] = [y]$.
□

Совокупность классов эквивалентности элементов множества X по отношению эквивалентности ρ называется *фактор-множеством* множества X по отношению ρ и обозначается X/ρ

1.4.4 Специальные бинарные отношения: Отношение порядка

Определение 1.4.9 . *Бинарное отношение ρ на множестве X называется отношением порядка, если оно транзитивно и антисимметрично. Множество, на котором введено отношение порядка, называют упорядоченным.*

Пример 1.4.6 . *Множество всех пар (x, y) людей, для которых x старше y, является отношением порядка.*

Определение 1.4.10 . *Отношение порядка $\preceq$ называется отношением нестрогого (частичного) порядка на множестве X, если оно рефлексивно.*

Отношение порядка $\prec$ называется отношением строгого порядка на множестве X, если оно иррефлексивно:

$$\forall x, y \in X : x \prec y \ \Rightarrow x \neq y.$$

Определение 1.4.11 . *Отношение порядка ρ на множестве X называется отношением линейного порядка, если любые $x, y \in X$, $x \neq y$, сравнимы в смысле отношения ρ (либо $x \rho y$, либо $y \rho x$ обязательно выполняется).*

Пример 1.4.7 . *1) Отношение родитель-ребенок (рисунок 6-a)) не является отношением порядка. Очевидно, что у такого отношения отсутствует транзитивность: дед не является родителем своего внука.*

С другой стороны, отношение предок-потомок является отношением порядка. На рисунке 6-b) представлено то же фамильное дерево, что и на рисунке 6-a), но с указанием всех связей от дедов к внукам.

Отношение предок-потомок является отношением строгого порядка и не является линейным порядком.

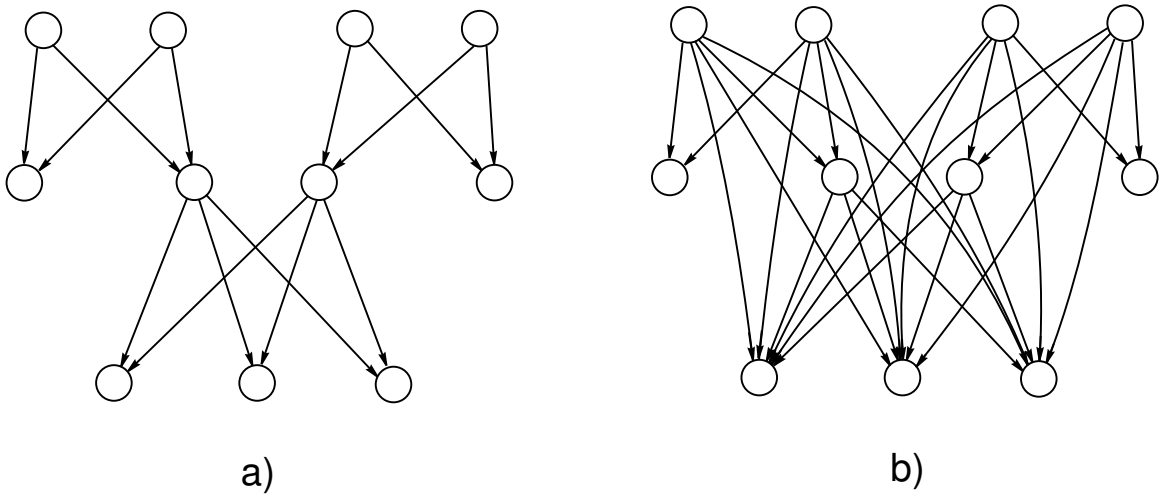

Рисунок 6: Генеалогическое древо

2) $\rho_> = \{(x, y) \mid x, y \in \mathbb{R},\ x > y\}$ — является отношением строгого

линейного порядка.
3) $\rho_{\geq} = \{(x, y) \mid x, y \in \mathbb{R},\ x \geq y\}$ *— является отношением нестрогого линейного порядка.*
4) Отношение $A \subseteq B$ *всех пар подмножеств* (A, B) *заданного универсального множества* U*, таких что* A *является подмножеством множества* B *называют отношением включения. Отношение включения является отношением нестрогого порядка и не является отношением линейного порядка.*

Определение 1.4.12 . *Пусть на множестве* X *введено отношение строгого порядка* $\prec$*. Пусть элемент* $x \in X$ *таков, что*

$$\forall y \in X,\ y \neq x \quad \Rightarrow \quad x \prec y.$$

Тогда элемент x *называют наименьшим.*

Лемма 1.4.9 . *Если на конечном непустом множестве* X *задан линейный строгий порядок, то существует наименьший элемент, и он единственен.*

Доказательство. Самостоятельно.
□

Теорема 1.4.10 . *Пусть на конечном непустом множестве* X *задано отношение линейного строгого порядка* $\prec$*. Тогда на* X *можно выбрать такую нумерацию элементов* $X = \{x_1, x_2, ..., x_n\}$*, что соотношение* $x_i \prec x_j$ *будет выполняться в том и только в том случае, когда* $i < j$*.*

Доказательство. Согласно лемме 1.4.9 для множества X существует наименьший элемент $x_1 \in X$, такой что $x_1 \prec y$ для любых $y \in X$. Удалим элемент x_1 из множества X.

Множество $X \setminus \{x_1\}$ также удовлетворяет условиям леммы 1.4.9 и, значит, в нем тоже существует наименьший элемент — x_2.

Мы можем повторять этот процесс до тех пор, пока в множестве X не закончатся элементы. Очевидно, по способу выбора элементов x_i, $\{x_1, x_2, ..., x_n\}$ и есть искомая нумерация.
□

Определение 1.4.13 . *Два нестрого упорядоченных множества X и Y называются изоморфными, если существует биекция $\varphi : X \to Y$, сохраняющая отношение нестрогого порядка. Иными словами, если $\preceq_x$ и $\preceq_y$ отношения нестрогого порядка соответственно множеств X и Y, то*

$$x_1 \preceq_x x_2 \quad \Longleftrightarrow \quad \varphi(x_1) \preceq_y \varphi(x_2).$$

Пример 1.4.8 . *Рассмотрим множество 2^A всех подмножеств множества $A = \{1, 2, 3\}$, упорядоченное отношением включения, и множество $X = \{1, 2, 3, 5, 6, 10, 15, 30\}$, упорядоченное отношением*

$$x \preceq y \Leftrightarrow y \text{ делится на } x.$$

Из рисунка 7 хорошо видно, что эти два упорядоченных множества изоморфны.

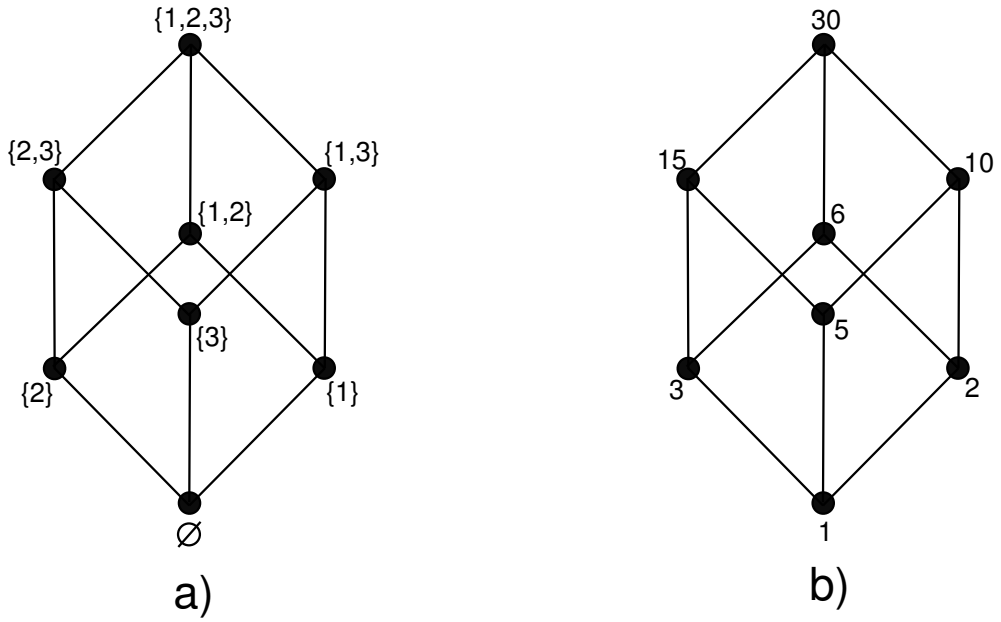

Рисунок 7: Изоморфные нестрого упорядоченные множества

Теорема 1.4.11 . *Всякое нестрого упорядоченное множество X изоморфно некоторой системе подмножеств множества X, нестрого упорядоченной отношением включения.*

Доказательство. Пусть $\preceq$ — отношение нестрогого порядка на множестве X. Для каждого элемента $a \in X$ рассмотрим множество $S_a = \{x \in X \mid x \preceq a\}$. Ясно, что $S_a \subseteq X$ для любого $a \in X$. Покажем, что система подмножеств $\{S_a \mid a \in X\}$, упорядоченная отношением включения, есть искомая система подмножеств. Рассмотрим отображение

$$\varphi : X \to \{S_a \mid a \in X\}$$

такое, что $\varphi(a) = S_a$. Если $S_a = S_b$, то, поскольку $a \in S_a$, то $b \in S_a$ и, следовательно, $b \preceq a$. Аналогично $a \preceq b$. Таким образом, в силу антисимметричности отношения $\preceq$, $a = b$. Значит, отображение φ инъективно. С другой стороны, у любого множества S_a есть прообраз a. Значит, φ сюрьективно. Следовательно φ — биекция.

Пусть $a \preceq b$. Тогда из $x \preceq a$ в силу транзитивности отношения следует $x \preceq b$, а значит $S_a \subseteq S_b$. Пусть $S_a \subseteq S_b$. Тогда, поскольку $a \in S_a$, то $a \in S_b$ и, следовательно, $a \preceq b$. Таким образом биекция φ сохраняет отношение нестрогого порядка.
□

1.4.5 Лексикографический порядок

Заслуживающим отдельного упоминания является *лексикографический порядок*. Лексикографический порядок - это порядок, в котором выстроены слова, например, в русско-английских словарях. Лексикографический порядок может быть введен на множестве слов над любым множеством, на котором уже введен линейный порядок.

Пусть на множестве X введен строгий линейный порядок $\prec$. Введем отношение $\prec_{lex}$ на множестве $X^* = \{(x_1, x_2, ..., x_k) \mid k \in \mathbb{N},\ x_i \in X\}$ всех слов над множеством X. Пусть $\widetilde{x}_k = (x_1, ..., x_k) \in X^*$, $\widetilde{y}_l = (y_1, ..., y_l) \in X^*$, $\widetilde{x}_k \neq \widetilde{y}_l$ и $k \leq l$.

Будем говорить, что $\widetilde{x}_k \prec_{lex} \widetilde{y}_l$, если 1) существует такой индекс t, $1 \leq t \leq k$, что $x_i = y_i$, $i = \overline{1, t-1}$ и $x_t \prec y_t$, или 2) $k < l$ и $x_i = y_i$, $i = \overline{1, k}$. В противном случае $\widetilde{y}_l \prec_{lex} \widetilde{x}_k$.

Замечание 1.4.3 . *На множестве слов одинаковой длины определение лексикографического порядка упростилось бы и приняло бы вид:*

$$\widetilde{x}_k \prec_{lex} \widetilde{y}_k \quad \Leftrightarrow \quad \exists t \in \{1, ..., k\} :\ x_i = y_i, i = \overline{1, t-1},\ \ x_t \prec y_t.$$

Пример 1.4.9 . *Выпишем все возможные перестановки чисел {1, 2, 3, 4} выстроенные в лексикографическом порядке:*

1 2 3 4	2 1 3 4	3 1 2 4	4 1 2 3
1 2 4 3	2 1 4 3	3 1 4 2	4 1 3 2
1 3 2 4	2 3 1 4	3 2 1 4	4 2 1 3
1 3 4 2	2 3 4 1	3 2 4 1	4 2 3 1
1 4 2 3	2 4 1 3	3 4 1 2	4 3 1 2
1 4 3 2	2 4 3 1	3 4 2 1	4 3 2 1

Заметим, что здесь мы привели только перестановки — последовательности из различных символов. Если мы захотим выписать все слова в данном алфавите, их окажется гораздо больше.

Подробнее о перестановках смотри раздел 1.6.

Отметим также, что существует еще так называемый *антилексикографический порядок.* Он не является обратным лексикографическому порядку бинарным отношением. Список всех перестановок n чисел в антилексикографическом порядке может быть получен следующим образом: сначала нужно выстроить все такие перестановки в лексикографическом порядке, а затем развернуть список в обратном порядке и развернуть каждое слово, описывающее перестановку. Такой порядок может быть полезен, например, для словаря окончаний.

Используя те же обозначения и договоренности, что и при введении лексикографического порядка, антилексикографический порядок можно определить следующим образом:
Будем говорить, что $\widetilde{y}_l \prec_{alex} \widetilde{x}_k$, если 1) существует такой индекс t, $1 \le t \le k$, что $x_{k-i+1} = y_{l-i+1}$, $i = \overline{1, t-1}$ и $x_{k-t+1} \prec y_{l-t+1}$, или 2) $k < l$ и $x_{k-i+1} = y_{l-i+1}$, $i = \overline{1, k}$. В противном случае $\widetilde{y}_l \prec_{alex} \widetilde{x}_k$.

Замечание 1.4.4 . *На множестве слов одинаковой длины определение приняло бы следующий вид:*

$$\widetilde{x}_k \prec_{alex} \widetilde{y}_k \quad \Leftrightarrow \quad \exists t \in \{1, ..., k\} : \; x_i = y_i, i = \overline{t+1, k}, \quad y_t \prec x_t.$$

Пример 1.4.10 . *Приведем все возможные перестановки чисел {1, 2, 3, 4} выстроенные в антилексикографическом порядке:*

1 2 3 4	1 2 4 3	1 3 4 2	2 3 4 1
2 1 3 4	2 1 4 3	3 1 4 2	3 2 4 1
1 3 2 4	1 4 2 3	1 4 3 2	2 4 3 1
3 1 2 4	4 1 2 3	4 1 3 2	4 2 3 1
2 3 1 4	2 4 1 3	3 4 1 2	3 4 2 1
3 2 1 4	4 2 1 3	4 3 1 2	4 3 2 1

Лексикографический и антилексикографический порядки являются отношениями строгого линейного порядка.

1.5 Выборки с повторениями и без повторений

1.5.1 Размещения и сочетания

Пусть имеется множество $S = \{s_1, s_2, ..., s_n\}$. Набор элементов $s_{i_1}, s_{i_2}, ..., s_{i_k}$ из множества S называется выборкой объема k (k-элементной выборкой) из n элементов.

Выборка называется упорядоченной, если порядок элементов в ней задан. Иначе выборка называется неупорядоченной.

Так же различают выборки с повторениями и без повторений в зависимости от того, допускается или не допускается повторное вхождение в выборку одних и тех же элементов.

Пример 1.5.1 . *Пусть* $S = \{1, 2, 3\}$. *Тогда* $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 1)$, $(2, 2)$, $(2, 3)$, $(3, 1)$, $(3, 2)$, $(3, 3)$ — *все возможные упорядоченные выборки объема два из трех элементов.*

Определение 1.5.1 . *Размещением из* n *элементов по* k *называется упорядоченная выборка без повторений объема* k *из* n*-элементного множества.*

Поскольку элементы нашего множества S *пронумерованы некоторым образом, не умаляя общности можно называть размещением из* n *элементов по* k *упорядоченный набор из* k *различных чисел, принадлежащих множеству* $\{1, ..., n\}$.

Обозначим A_n^k количество различных размещений из n по k.

Пример 1.5.2 . *1) Пусть на экзамене у преподавателя* n *различных билетов и сдавать пришло* k *студентов. Тогда существует ровно* A_n^k *способов выдать всем студентам по одному билету для подготовки.*
2) Пусть $S = \{1, 2, 3\}$. *Тогда* $(1, 2)$, $(1, 3)$, $(2, 1)$, $(2, 3)$, $(3, 1)$, $(3, 2)$ — *все возможные размещения из трех элементов по два.*

Утверждение 1.5.1 . *Пусть* $k, n \in \mathbb{N}$ *и* $1 \leq k \leq n$. *Тогда*

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

Доказательство. Действительно, существует n различных способов выбрать первый элемент набора из элементов множества $\{1, ..., n\}$. Аналогично, существует $n-1$ способ выбора второго элемента и так далее.
□

Определение 1.5.2 . *Сочетанием из n элементов по k называется неупорядоченная выборка без повторений объема k из n-элементного множества.*

Как и раньше, можем считать, что сочетанием из n элементов по k называется неупорядоченный набор из k различных чисел, принадлежащих множеству $\{1, ..., n\}$.

Количество сочетаний из n по k обозначим C_n^k или $\binom{n}{k}$.

Пример 1.5.3 . *1) Предположим из n участников спортивного клуба на соревнования должны поехать какие-то k. Тогда имеется C_n^k различных возможности собрать комманду.*
2) Пусть $S = \{1, 2, 3\}$. Тогда $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$ — все возможные сочетания из трех элементов по два.

Определение 1.5.3 . *Множество всех подмножеств множества S мощности k будем обозначать $\binom{S}{k}$:*

$$\binom{S}{k} = \{A \mid A \subseteq S,\ |A| = k\}.$$

Пример 1.5.4 . *Пусть $S = \{1, 2, 3, 4, 5\}$ и $k = 3$. Тогда $\binom{S}{k} = \{$ $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 2, 5\}$, $\{1, 3, 4\}$, $\{1, 3, 5\}$, $\{1, 4, 5\}$, $\{2, 3, 4\}$, $\{2, 3, 5\}$, $\{2, 4, 5\}$, $\{3, 4, 5\}$ $\}$*

Нетрудно видеть, что если мощность множества S равна n, то

$$\left|\binom{S}{k}\right| = \binom{|S|}{k} = \binom{n}{k}. \tag{2}$$

Утверждение 1.5.2 . *Пусть $k, n \in \mathbb{N}$ и $1 \leq k \leq n$. Тогда*

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3}$$

Доказательство. Действительно, каждому сочетанию из n по k соответствует $k!$ различных размещений из n по k с различным порядком следования элементов. Тогда $\binom{n}{k} = \frac{A_n^k}{k!} = \frac{n!}{k!(n-k)!}$
□

Очевидным следствием из формулы (3) является равенство

$$\binom{n}{k} = \binom{n}{n-k}. \tag{4}$$

Интуитивно эту формулу можно было бы обосновать следующим рассуждением. Выбирая k элементов из n мы тем самым выбираем $n-k$ элементов из n, которые не попадают в нашу выборку. Проще говоря, мы разбиваем наше множество мощности n на два подмножества мощностей k и $n-k$ соответственно.

Также можно непосредственно подстановкой формулы (3) убедиться в правильности равенства

$$\binom{n}{k} \cdot \binom{k}{l} = \binom{n}{l} \cdot \binom{n-l}{k-l}. \tag{5}$$

Действительно,

$$\begin{aligned}\binom{n}{k} \cdot \binom{k}{l} &= \frac{n!}{k!(n-k)!} \cdot \frac{k!}{l!(k-l)!} \cdot \frac{(n-l)!}{(n-l)!} = \\ &= \frac{n!}{l!(n-l)!} \cdot \frac{(n-l)!}{(k-l)!(n-k)!} = \binom{n}{l} \cdot \binom{n-l}{k-l}.\end{aligned}$$

Интуитивно формулу (5) можно описать, как разбиение множества из n элементов на три подмножества мощностей l, $k-l$, $n-k$ соответственно. Левая часть равенства описывает выбор сначала k элементов из n, а затем l элементов из выбранных k. Получим все варианты разбиения исходного множества на три подмножества указанных мощностей.

Правая часть равенства соответствует выбору сначала l элементов из n, а затем $k-l$ элементов из оставшихся $n-l$. Получим те же варианты подмножеств.

1.5.2 Треугольник Паскаля

Утверждение 1.5.3 . *Пусть $k, n \in \mathbb{N}$ и $2 \leq k \leq n$. Тогда*

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}. \tag{6}$$

Во-первых, это равенство довольно легко доказать явно подставив выражение согласно формуле (3). Приведем другое доказательство этого факта.

Доказательство. Пусть множество S имеет вид: $S = \{s_1, s_2, ..., s_n\}$. Определим множества A и B следующим образом:

$$A = \left\{ \{s_{i_1}, s_{i_2}, ..., s_{i_{k-1}}, s_n\} \mid \{s_{i_1}, s_{i_1}, ..., s_{i_{k-1}}\} \in \binom{S \setminus \{s_n\}}{k-1} \right\}$$

$$B = \binom{S \setminus \{s_n\}}{k}$$

Очевидно $A \cap B = \varnothing$. Кроме того, можно видеть, что $\binom{S}{k} = A \cup B$, поскольку все k-элементные сочетания из S, содержащие s_n, находятся в A, а все такие k-элементые сочетания, которые не содержат s_n, лежат в B. Тогда $|\binom{S}{k}| = |A| + |B|$. Учитывая, что по формуле (2) $|\binom{S}{k}| = \binom{n}{k}$, $|A| = \binom{n-1}{k-1}$ и $|B| = \binom{n-1}{k}$, получаем искомое равенство (6).
□

Положим следующие естественные начальные условия для числа сочетаний из n по k: $\binom{0}{0} = 1$; $\binom{n}{0} = 1$ для любых $n \in \mathbb{N}$; $\binom{n}{k} = 0$ для любых $k > n$. Тогда, пользуясь рекурентным соотношением (6), можно построить следующую таблицу, которую называют треугольником Паскаля:

$n\backslash k$	0	1	2	3	4	5	6	7	...
0	1	0	0	0	0	0	0	0	...
1	1	1	0	0	0	0	0	0	...
2	1	2	1	0	0	0	0	0	...
3	1	3	3	1	0	0	0	0	...
4	1	4	6	4	1	0	0	0	...
5	1	5	10	10	5	1	0	0	...
6	1	6	15	20	15	6	1	0	...
7	1	7	21	35	35	21	7	1	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Здесь каждый элемент $\binom{n}{k}$, кроме первого столбца и ниже диагонали является суммой элемента слева сверху, который соответствует $\binom{n-1}{k-1}$, и верхнего — $\binom{n-1}{k}$.

На рисунке 8 приведен другой способ изображения треугольника Паскаля, который может оказаться более наглядным. На рисунке

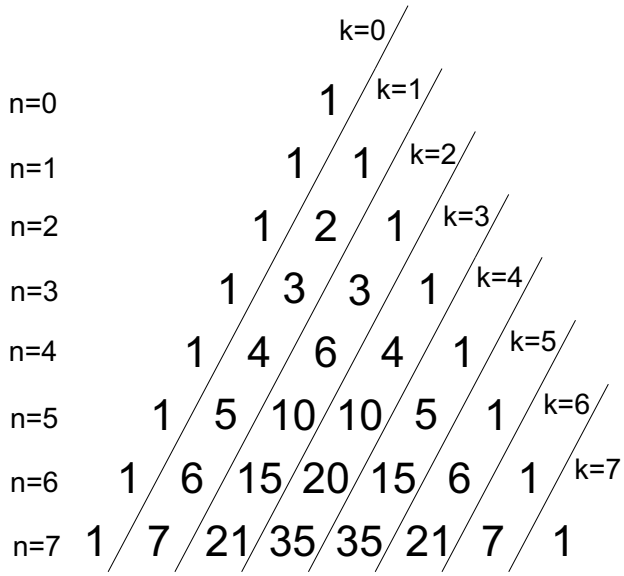

Рисунок 8: Треугольник Паскаля

каждый элемент, кроме крайних в каждой строке, является суммой вдух элементов, под которыми он стоит.

1.5.3 Связь сочетаний и (0,1)-векторов

С каждым сочетанием из n по k можно связать вектор из нулей и единиц, в котором число единиц равно k: позиции единиц указывают числа, которые должны войти в сочетание. Другими словами, установлено взаимооднозначное соответствие между множеством сочетаний из n по k и множеством (0,1)-векторов длины n с k единицами.

В свою очередь каждый (0,1)-вектор длины n с k единицами соответствует пути на прямоугольной решетке (рис. 9) длины $n-k$ и высоты k. Можно сопоставить каждому шагу вниз — единицу, а каждому

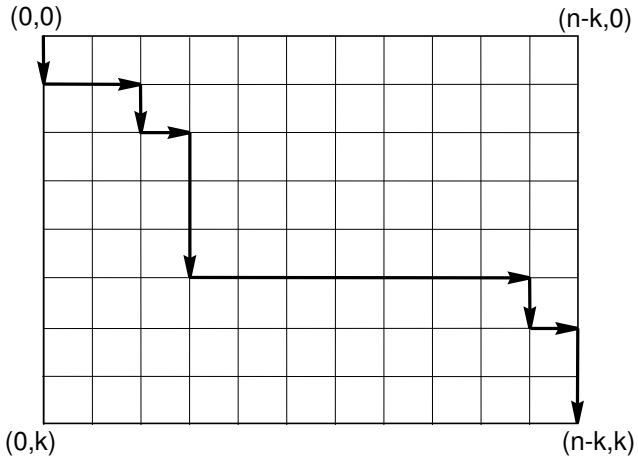

Рисунок 9: Выбор пути на прямоугольной решетке

шагу вправо — ноль. Тогда произвольному пути в n шагов из точки $(0,0)$ в точку $(n-k,k)$ взаимнооднозначно соответствует (0,1)-вектор длины n с k единицами. Таким образом, мы получили взаимнооднозначное соответствие между множеством сочетанием из n по k и множеством путей на прямоугольной решетке.

Заметим, что все множество путей из точки $(0,0)$ в точку $(n-k,k)$ складывается из множества путей из $(0,0)$ в $(n-k,k-1)$ с последним шагом из $(n-k,k-1)$ в $(n-k,k)$ и множества путей из $(0,0)$ в $(n-k-1,k)$ с последним шагом из $(n-k-1,k)$ в $(n-k,k)$ (рис. 10). Тогда количество путей из $(0,0)$ в $(n-k,k)$, равное $\binom{n}{k}$, совпадает с суммой количеств путей

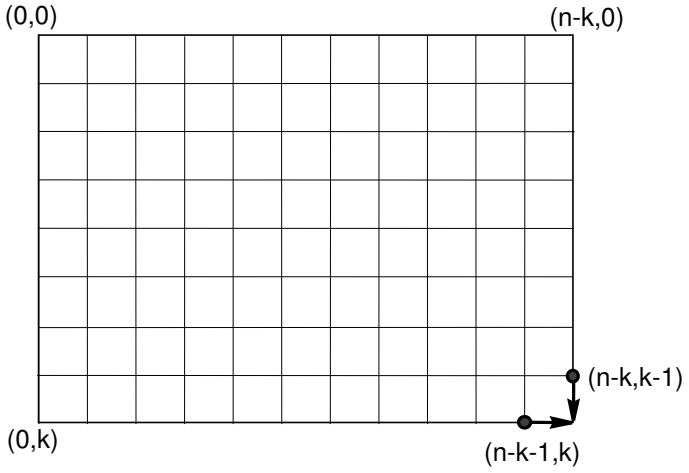

Рисунок 10: Все пути в точку $(n-k,k)$ складываются из двух групп

из $(0,0)$ в $(n-k,k-1)$ и из $(0,0)$ в $(n-k-1,k)$, что равно $\binom{n-1}{k-1}+\binom{n-1}{k}$. Таким образом, мы привели еще одно доказательство формулы (6).

1.5.4 Перебор сочетаний

В практических задачах может возникнуть необходимость рассмотреть все возможные сочетания из некоторого множества заданной мощности, чтобы сравнить их свойства, или проделать некоторую операцию для каждого сочетания. Рассмотрим алгоритм перебора сочетаний.

Пусть x_1, x_2, ..., x_k — числа из множества $\{1,2,...,n\}$, вошедшие в сочетание, причем $x_1 < x_2 < ... < x_k$. Пусть в начальный момент времени сочетание состоит из первых k чисел: $x_i = i$, $i = \overline{1,k}$.

Далее на каждом шаге будем просматривать вектор $(x_1, x_2, ..., x_k)$ начиная с x_k и искать первую такую компоненту x_i, которую можно увеличить (нельзя увеличить x_k, если он равен n; x_{k-1}, если он равен $n-1$ и так далее). Если такой компоненты не найдется, алгоритм завершает свою работу. В противном случае, пусть i наибольшее число, такое что $x_i < n-k+i$. Увеличим x_i на единицу, а для всех x_t, $t = \overline{i+1,k}$, присваимаем значения $x_t = x_i + (t-i)$. Повторяем процесс нужное число раз.

Пример 1.5.5 . *Рассмотрим, как работает алгоритм для* $n = 5$ *и* $k = 3$.
1) Сначала $x = (x_1, x_2, x_3) = (1, 2, 3)$.
2) Увеличиваем x_3*:* $x = (1, 2, 4)$.
3) Увеличиваем x_3*:* $x = (1, 2, 5)$.
4) x_3 *больше увеличить нельзя. Увеличиваем* x_2 *и переназначаем значение* x_3*:* $x = (1, 3, 4)$.
Далее аналогично
5) $x = (1, 3, 5)$
6) $x = (1, 4, 5)$
7) $x = (2, 3, 4)$
8) $x = (2, 3, 5)$
9) $x = (2, 4, 5)$
10) $x = (3, 4, 5)$
Таким обраом, мы перебрали все сочетания из 5 *по* 3.

1.5.5 Бином Ньютона

Утверждение 1.5.4 . *Пусть* x_1, x_2, ..., x_n *— независимые переменные. Обозначим* $X = \{x_1, x_2, ..., x_n\}$. *Тогда*

$$(1 + x_1)(1 + x_2)...(1 + x_n) = \sum_{Y \subseteq X} \prod_{x \in Y} x. \tag{7}$$

Доказательство. Проведем индукцию по n. Пусть $n = 1$. Тогда $X = \{x_1\}$.

$$\sum_{Y \subseteq X} \prod_{x \in Y} x = \prod_{x \in \varnothing} x + \prod_{x \in \{x_1\}} x = 1 + x_1.$$

Заметим, что здесь мы использовали $\prod_{x \in \varnothing} x = 1$, поскольку $x^0 = 1$. База индукции доказана.

Пусть утверждение верно для всех $n \leq \widehat{n}$.

Положим $n = \widehat{n} + 1$. $X = \{x_1, x_2, ..., x_{\widehat{n}}, x_{\widehat{n}+1}\}$. Тогда

$$\sum_{Y \subseteq X} \prod_{x \in Y} x = \sum_{Y \subseteq X \setminus \{x_{\widehat{n}+1}\}} \prod_{x \in Y} x + x_{\widehat{n}+1} \sum_{Y \subseteq X \setminus \{x_{\widehat{n}+1}\}} \prod_{x \in Y} x =$$

$$= (1 + x_1)(1 + x_2)...(1 + x_{\widehat{n}}) + x_{\widehat{n}+1}(1 + x_1)(1 + x_2)...(1 + x_{\widehat{n}}) =$$

$$= (1 + x_{\widehat{n}+1}) \cdot (1 + x_1)(1 + x_2)...(1 + x_{\widehat{n}}).$$

□

Следствие 1.5.5 (формула бинома Ньютона).

$$(1 + x)^n = \sum_{k=0}^{n} \binom{n}{k} x^k \tag{8}$$

Доказательство. Если в формулу (7) подставить $x_1 = x_2 = ... = x_n = x$, то получим

$$(1 + x)^n = \sum_{Y \subseteq X} \prod_{x \in Y} x = \sum_{Y \subseteq X} x^{|Y|} =$$

$$= \sum_{k=0}^{n} \sum_{\substack{Y \subseteq X, \\ |Y|=k}} x^k = \sum_{k=0}^{n} x^k \sum_{\substack{Y \subseteq X, \\ |Y|=k}} 1 = \sum_{k=0}^{n} \binom{n}{k} x^k$$

□

Значения $\binom{n}{k}$ получили благодоря этой формуле название *биномиальных коэффициентов*.

Более часто используемой формой формулы бинома Ньютона является следующее уравнение:

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} \tag{8'}$$

Действительно, если $b = 0$, формула очевидна. Пусть $b \neq 0$. Тогда обозначим за x значение $\frac{a}{b}$. Тогда

$$(a + b)^n = (b(\tfrac{a}{b} + 1))^n = b^n (x + 1)^n = b^n \sum_{k=0}^{n} \binom{n}{k} x^k =$$

$$= \sum_{k=0}^{n} \binom{n}{k} b^n \cdot x^k = \sum_{k=0}^{n} \binom{n}{k} b^n \cdot (\tfrac{a}{b})^k = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}.$$

Таким образом, формула (8′) эквивалентна формуле (8).

Следствие 1.5.6 .

$$\sum_{k=0}^{n} \binom{n}{k} = (1+1)^n = 2^n$$

Что словесно можно сформулировать как "число всех подмножеств n-элементного множества равно 2^n". Этот факт нам уже знаком.

Следствие 1.5.7 . *Пусть $n > 0$. Тогда*

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

Это равенство можно прочитать, например, как "суммарная мощность множества всех нечетных подмножеств множества равна суммарной мощности всех его четных подмножеств". Другими словами, число подмножеств четной и нечетной мощности совпадает.

Замечание 1.5.1 (Дельта-функция). *Отметим, что по следствию 1.5.7, если рассматривать выражение $\sum_{k=0}^{n}(-1)^k\binom{n}{k}$ как функцию от n на множестве целых чисел, мы получим дельта-функцию — функцию, которая принимает значение 1 только в одной точке и 0 во всех остальных:*

$$\delta_o(n) = \sum_{k=0}^{n} (-1)^k \binom{n}{k} = \begin{cases} 1, n = 0, \\ 0, n \neq 0. \end{cases} \tag{9}$$

Пользуясь $\delta_o(n)$ можно получить и другие δ-функции. Для любого целого числа a можно определить $\delta_a(n)$ на множестве целых чисел следующим образом:

$$\delta_a(n) = \delta_o(n-a) = \sum_{k=0}^{n-a} (-1)^k \binom{n-a}{k} = \begin{cases} 1, n = a, \\ 0, n \neq a. \end{cases} \tag{10}$$

Утверждение 1.5.8 (формула обращения). *Пусть даны две числовых последовательности a_0, a_1, a_2, ... и b_0, b_1, b_2, ... Тогда следующие два утверждения эквивалентны*

$$1) \qquad b_n = \sum_{k=0}^{n} \binom{n}{k} a_k, \quad n = 0, 1, 2... \tag{11}$$

$$2) \qquad a_n = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} b_k, \quad n = 0, 1, 2... \tag{12}$$

Доказательство. 1) Для начала докажем, что из формулы (11) следует (12). Пусть верно $b_n = \sum_{k=0}^{n} \binom{n}{k} a_k$, $n = 0, 1, 2....$ Тогда

$$\sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} b_k = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} \sum_{j=0}^{k} \binom{k}{j} a_j =$$

$$= \sum_{k=0}^{n} \sum_{j=0}^{k} (-1)^{n-k} \binom{n}{k} \binom{k}{j} a_j =$$

Прервем последовательность равенств, чтобы разъяснить следующее действие. Рассмотрим сумму $\sum_{k=0}^{n} \sum_{j=0}^{k} A(j,k)$. Сложение членов $A(j,k)$ идет по всем таким индексам j и k, что $0 \le k \le n$ и $0 \le j \le k$. Иначе это условие можно записать, как $0 \le j \le k \le n$.

Теперь рассмотрим условие $0 \le j \le n$ и $j \le k \le n$, которое описывает ограничение на индексы j и k для суммы $\sum_{j=0}^{n} \sum_{k=j}^{n} A(j,k)$. Можно видеть, что это условие тоже эквивалентно записи $0 \le j \le k \le n$. Таким образом, мы показали, что суммы $\sum_{k=0}^{n} \sum_{j=0}^{k} A(j,k)$ и $\sum_{j=0}^{n} \sum_{k=j}^{n} A(j,k)$ отличаются только порядком суммирования и значит

$$\sum_{k=0}^{n} \sum_{j=0}^{k} A(j,k) = \sum_{j=0}^{n} \sum_{k=j}^{n} A(j,k).$$

Теперь продолжим наши выкладки.

$$= \sum_{j=0}^{n} \sum_{k=j}^{n} (-1)^{n-k} \binom{n}{k} \binom{k}{j} a_j = \sum_{j=0}^{n} a_j \sum_{k=j}^{n} (-1)^{n-k} \binom{n}{k} \binom{k}{j} =$$

$$= \sum_{j=0}^{n} a_j \sum_{k=j}^{n} (-1)^{n-k} \binom{n}{j} \binom{n-j}{k-j} = \sum_{j=0}^{n} \binom{n}{j} a_j \sum_{k=j}^{n} (-1)^{n-k} \binom{n-j}{k-j} =$$

Для следующего перехода сделаем замену переменной индексирования во второй сумме. Положим $l = n - k$. Тогда $k = n - l$, $k - j = n - j - l$. В то время как индекс k пробегал все значения от j до

n, индекс l будет пробегать значения от $n-j$ до нуля. Поскольку операция сложения на множестве вещественных чисел комутативна (неважен порядок суммирования), будем считать, что переменная l проходит значения от нуля до $n-j$. Тогда получим:

$$= \sum_{j=0}^{n} \binom{n}{j} a_j \sum_{l=0}^{n-j} (-1)^l \binom{n-j}{n-j-l} = \sum_{j=0}^{n} \binom{n}{j} a_j \sum_{l=0}^{n-j} (-1)^l \binom{n-j}{l}.$$

Из формулы (10) следует, что $\sum_{l=0}^{n-j} (-1)^l \binom{n-j}{l} = \delta_j(n)$ и значит

$$\binom{n}{j} a_j \sum_{l=0}^{n-j} (-1)^l \binom{n-j}{l} = \begin{cases} \binom{j}{j} a_j = a_j, & n = j, \\ 0, & n \neq j. \end{cases}$$

Таким образом, из всей последней суммы останется только одно слагаемое

$$\sum_{j=0}^{n} \binom{n}{j} a_j \sum_{l=0}^{n-j} (-1)^l \binom{n-j}{l} = \sum_{j=0}^{n} \binom{n}{j} a_j \delta_j(n) = a_n,$$

что и требовалось доказать.

2) Доказательство, что из формулы (12) следует (11), проводится аналогично. Читатель может сделать это самостоятельно.
□

Пример 1.5.6 . *Пусть даны значения $b_0 = 2$, $b_1 = 1$, $b_2 = 5$, $b_3 = -2$, $b_4 = 3$ и верна формула (11). Воспользуемся формулой (12) для вычисления значений a_n:*

$$a_n = \sum_{k=0}^{n} (-1)^{n-k} \binom{n}{k} b_k, \quad n = 0, 1, 2, 3, 4.$$

Для подстановки правильных биномиальных коэффициентов в эту формулу будет удобно воспользоваться треугольником Паскаля, как он представлен на рисунке 8. В каждой строке для данного n там выписаны все необходимые нам коэффициенты.

$$a_0 = \binom{0}{0} b_0 = 2,$$

$$
\begin{aligned}
a_1 =& -\binom{1}{0}b_0 + \binom{1}{1}b_1 = -2 + 1 = -1,\\
a_2 =& \binom{2}{0}b_0 - \binom{2}{1}b_1 + \binom{2}{2}b_2 = 2 - 2\cdot 1 + 5 = 5,\\
a_3 =& -\binom{3}{0}b_0 + \binom{3}{1}b_1 - \binom{3}{2}b_2 + \binom{3}{3}b_3 =\\
=& -2 + 3\cdot 1 - 3\cdot 5 + (-2) = -16,\\
a_4 =& \binom{4}{0}b_0 - \binom{4}{1}b_1 + \binom{4}{2}b_2 - \binom{4}{3}b_3 + \binom{4}{4}b_4 =\\
=& 2 - 4\cdot 1 + 6\cdot 5 - 4\cdot(-2) + 3 = 39.
\end{aligned}
$$

Теперь проверим правильность нахождения значений a_n с помощью формулы (11)

$$
\begin{aligned}
b_0 =& \binom{0}{0}a_0 = 2,\\
b_1 =& \binom{1}{0}a_0 + \binom{1}{1}a_1 = 2 + (-1) = 1,\\
b_2 =& \binom{2}{0}a_0 + \binom{2}{1}a_1 + \binom{2}{2}a_2 = 2 + 2\cdot(-1) + 5 = 5,\\
b_3 =& \binom{3}{0}a_0 + \binom{3}{1}a_1 + \binom{3}{2}a_2 + \binom{3}{3}a_3 =\\
=& 2 + 3\cdot(-1) + 3\cdot 5 + (-16) = -2,\\
b_4 =& \binom{4}{0}a_0 + \binom{4}{1}a_1 + \binom{4}{2}a_2 + \binom{4}{3}a_3 + \binom{4}{4}a_4 =\\
=& 2 + 4\cdot(-1) + 6\cdot 5 + 4\cdot(-16) + 39 = 3.
\end{aligned}
$$

Как видно, значения b_n получены правильно.

1.5.6 Мультимножества

Пусть дано некоторое множество $S = \{s_1, s_2, ..., s_n\}$, $|S| = n$.

Определение 1.5.4 . *Мультимножеством M на множестве S назовем всюдуопределенную функцию*

$$\varphi : S \to \mathbb{N}_0,$$

где $\mathbb{N}_0$ — множество неотрицательных целых чисел.

Будем писать $M = \{s_1^{\varphi(s_1)}, s_2^{\varphi(s_2)}, ..., s_n^{\varphi(s_n)}\}$, имея в виду, что элемент s_i множества S встречается в мультимножестве M ровно $\varphi(s_i)$ раз. Мощность мультимножества M положим равной сумме количеств вхождений в M элементов s_i:

$$|M| = \sum_{i=1}^{n} \varphi(s_i).$$

Пример 1.5.7 . *Пусть $S = \{1, 2, 3, 4, 5\}$ и задана функция φ:*

x	1	2	3	4	5
$\varphi(x)$	1	0	2	1	3

.

Тогда мультимножество имеет вид $M = \{1^1, 2^0, 3^2, 4^1, 5^3\}$. То есть элементы 1 и 4 входят в мультимножество M по одному разу, элемент 3 входит в него 2 раза, элемент 5 — три раза и элемент 2 множества S не входит в мультимножество M. Мощность мультимножества равна $|M| = 1 + 0 + 2 + 1 + 3 = 7$.

Как можно видеть, мультимножество над множеством S — это неупорядоченная выборка с повторениями (сочетание с повторениями).

Определение 1.5.5 . *Множество всех мультимножеств на множестве S мощности k будем обозначать $\left(\binom{S}{k}\right)$.*

Пример 1.5.8 . *Пусть $S = \{1, 2, 3\}$. Тогда множество всех мультимножеств мощности 2 над множеством S будет иметь вид:* $\left(\binom{S}{2}\right) = \{\{1^2, 2^0, 3^0\}, \{1^1, 2^1, 3^0\}, \{1^1, 2^0, 3^1\}, \{1^0, 2^2, 3^0\}, \{1^0, 2^1, 3^1\}, \{1^0, 2^0, 3^2\}\}$.

Посчитаем, сколько возможно различных выборок с повторениями мощности k над множеством из n элементов. Положим

$$\left(\binom{n}{k}\right) = \left|\left(\binom{S}{k}\right)\right|.$$

Утверждение 1.5.9 . *Пусть* $k, n \in \mathbb{N}$.

$$\left(\binom{n}{k}\right) = \binom{n+k-1}{k} \tag{13}$$

Доказательство. Справа в формуле (13) стоит мощность множества всех подмножеств мощности k множества мощности $n+k-1$. Рассмотрим множество $\{1, 2, ..., n+k-1\}$ и его произвольное k-подмножество $\{a_1, a_2, ..., a_k\}$. Пусть, не умаляя общности, элементы a_i выстроены в порядке возрастания:

$$1 \leq a_1 < a_2 < ... < a_k \leq n+k-1. \tag{*}$$

Положим по определению $b_i = a_i - i + 1$, $i = \overline{1,k}$. Тогда

$$b_1 = a_1 \geq 1;$$
$$b_k = a_k - k + 1 \leq n + k - 1 - k + 1 = n;$$
$$b_{i+1} - b_i = a_{i+1} - (i+1) + 1 - (a_i - i + 1) = a_{i+1} - a_i - 1 \geq 0,$$
$$i = \overline{1, k-1}.$$

Следовательно,

$$1 \leq b_1 \leq b_2 \leq ... \leq b_k \leq n, \tag{**}$$

то есть b_1, b_2, ..., b_k задают некоторое мультимножество мощности k над множеством $\{1, 2, ..., n\}$.

Аналогично показывается обратное соответствие. Слева в формуле (13) стоит число мультимножеств мощности k множества мощности n. Рассмотрим произвольное мультимножество мощности k над множеством $\{1, 2, ..., n\}$. Не умаляя общности, считаем, что элементы мультимножества расположены в порядке неубывания и выполнена формула (**).

Положим по определению $a_i = b_i + i - 1$, $i = \overline{1,k}$. Тогда

$$a_1 = b_1 \geq 1;$$
$$a_k = b_k + k - 1 \leq n + k - 1;$$
$$a_{i+1} - a_i = b_{i+1} + (i+1) - 1 - (b_i + i - 1) = b_{i+1} - b_i + 1 > 0,$$
$$i = \overline{1, k-1}.$$

Следовательно, выполняется выражение (*).

Таким образом установлено взаимнооднозначное соответствие между множеством сочетаний из $n+k-1$ по k и множеством мультимножеств мощности k над множеством из n элементов. Следовательно мощности этих множеств равны.
□

1.5.7 Связь мультимножеств и (0,1)-векторов

Можно привести другое доказательство утверждения 1.5.9, используя связь мультимножеств с (0,1)-векторами. Как мы помним из пункта 1.5.3 каждому (0,1)-вектору из n элементов с ровно k единицами однозначно соответствует некоторое сочетание из n элементов по k.

Каждому мультимножеству мощности k на n-элементном множестве $S = \{s_1, s_2, ..., s_n\}$ можно поставить в соответствие (0,1)-вектор длины $n+k-1$ из k нулей и $n-1$ единицы, такой что число нулей, находящихся между $i-1$-й и i-й единицами, будет равно числу вхождений в мультимножество элемента s_i, $2 \leq i \leq n-1$; число нулей перед первой единицей равно числу вхождений в мультимножество элемента s_1; число нулей после $n-1$-й единицы равно числу вхождений в мультимножество элемента s_n. Это соответствие между множеством $\left(\binom{S}{k}\right)$ и множеством (0,1)-векторов длины $n+k-1$ с $n-1$-й единицей является взаимнооднозначным.

Таким образом, каждому (0,1)-вектору длины $n+k-1$ с $n-1$-й единицей однозначно соответствует сочетание из $n+k-1$ по $n-1$ и в тоже время ему соответствует мультимножество мощности k на n-элементном множестве. Следовательно, мы установили взаимнооднозначное соответствие между множеством всех сочетаний из $n+k-1$ по $n-1$ и множеством всех мультимножеств мощности k на n-элементном множестве. Значит, $\binom{n+k-1}{k} = \binom{n+k-1}{n-1} = \left(\binom{n}{k}\right)$, что и требовалось доказать.

1.5.8 Полином Ньютона

Утверждение 1.5.10 .

$$(x_1 + x_2 + ... + x_k)^m = \sum_{\substack{(\alpha_1,...,\alpha_k),\\ \alpha_1+\alpha_2+...+\alpha_k=m,\\ \alpha_i\in\mathbb{N}_0,\ i=\overline{1,k}}} \frac{m!}{\alpha_1!\alpha_2!...\alpha_k!} x_1^{\alpha_1} x_2^{\alpha_2} ... x_k^{\alpha_k}. \qquad (14)$$

Доказательство. Докажем по индукции. Для $k = 2$ утверждение верно в силу формулы бинома Ньютона $(8')$. Действительно,

$$\sum_{\substack{(\alpha_1,\alpha_2),\\ \alpha_1+\alpha_2=n,\\ \alpha_1,\alpha_2\in\mathbb{N}_0}} \frac{n!}{\alpha_1!\alpha_2!} a^{\alpha_1} b^{\alpha_2} = \sum_{\substack{k\in\{0,...,n\},\\ \alpha_1=k,\\ \alpha_2=n-k}} \binom{n}{\alpha_1} a^{\alpha_1} b^{\alpha_2} = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} = (a+b)^n.$$

Пусть утверждение верно для всех $k \le K$ для любых m.
Пусть теперь $k = K + 1$.

$$(x_1 + x_2 + ... + x_K + x_{K+1})^m = ((x_1 + ... + x_K) + x_{K+1})^m =$$

по индукционному предположению

$$= \sum_{\substack{\alpha+\beta=m,\\ \alpha,\beta\in\mathbb{N}_0}} \frac{m!}{\alpha!\beta!} (x_1 + ... + x_K)^{\alpha} x_{K+1}^{\beta} =$$

$$= \sum_{\substack{\alpha+\beta=m,\\ \alpha,\beta\in\mathbb{N}_0}} \frac{m!}{\alpha!\beta!} x_{K+1}^{\beta} \sum_{\substack{(\alpha_1,...,\alpha_K),\\ \alpha_1+\alpha_2+...+\alpha_K=\alpha,\\ \alpha_i\in\mathbb{N}_0,\ i=\overline{1,K}}} \frac{\alpha!}{\alpha_1!\alpha_2!...\alpha_K!} x_1^{\alpha_1} x_2^{\alpha_2} ... x_K^{\alpha_K} =$$

$$= \sum_{\substack{(\alpha_1,...,\alpha_K,\beta),\\ \alpha_1+\alpha_2+...+\alpha_K+\beta=m,\\ \alpha_i\in\mathbb{N}_0,\ i=\overline{1,K},\ \beta\in\mathbb{N}_0}} \frac{m!}{\alpha_1!\alpha_2!...\alpha_K!\beta!} x_1^{\alpha_1} x_2^{\alpha_2} ... x_K^{\alpha_K} x_{K+1}^{\beta},$$

Что и требовалось доказать.
□

Благодаря формуле (14) числа $\frac{m!}{\alpha_1!\alpha_2!...\alpha_k!}$ при $\alpha_1 + \alpha_2 + ... + \alpha_k = m$, $\alpha_i \in \mathbb{N}_0$, $i = \overline{1,k}$, $m \in \mathbb{N}_0$, получили название *мультиномиальных*

коэффициентов. Обычно их обозначают

$$\binom{m}{\alpha_1, \alpha_2, ..., \alpha_k}.$$

В этих обозначениях биномиальные коэффициенты имеют вид

$$\binom{n}{k} = \binom{n}{k, n-k}.$$

Легко доказать следующее свойство мультиномиальных коэффициентов:

Утверждение 1.5.11 . *Пусть* $m \in \mathbb{N}_0$.

$$\sum_{\substack{(\alpha_1,...,\alpha_k),\\ \alpha_1+\alpha_2+...+\alpha_k=m,\\ \alpha_i\in\mathbb{N}_0,\ i=\overline{1,k}}} \binom{m}{\alpha_1, \alpha_2, ..., \alpha_k} = k^m$$

Доказательство. Следует из утверждения 1.5.10 при подстановке $x_i = 1$, $i = \overline{1, k}$.
□

1.5.9 Разбиения множеств.

Рассмотрим комбинаторный смысл мультиномиальных коэффициентов. Рассмотрим множество $S = \{s_1, s_2, ..., s_m\}$. Пусть разбиение множества S на множеста S_1, S_2, ..., S_k таково, что каждое S_i содержит α_i элементов:

$$S = S_1 \cup S_2 \cup ... \cup S_k; \quad S_i \cap S_j = \varnothing,\ i \neq j; \quad |S_i| = \alpha_i,\ i = \overline{1, k}.$$

Отметим, что некоторые S_i могут быть пустыми и, очевидно, что $m = \alpha_1 + \alpha_2 + ... + \alpha_k$.

Утверждение 1.5.12 . *Число таких разбиений множества* S *равно мультиномиальному коэффициенту* $\binom{m}{\alpha_1,\alpha_2,...,\alpha_k}$.

Пример 1.5.9 . *Пусть имеется* m *шаров с различными номерами на них и имеется* k *коробок:* B_1, B_2, ..., B_k. *Сколькими способами можно*

разложить эти m шаров по k коробкам таким образом, чтобы в коробке B_i оказалось ровно α_i шаров, $i = \overline{1,k}$.

Для начала можем выбрать α_1 шаров, которые мы положим в коробку B_1. Это можно сделать столькими различными способами, сколько существует сочетаний из m по α_1 — $\binom{m}{\alpha_1}$.

Аналогично, α_2 шаров, которые мы положим в коробку B_2 можно выбрать $\binom{m-\alpha_1}{\alpha_2}$ способами.

Рассуждая далее подобным образом, мы получим, что число способов, которыми можно разложить все m шаров по k коробкам с соблюдением количеств шаров в коробках равно

$$\binom{m}{\alpha_1} \cdot \binom{m-\alpha_1}{\alpha_2} \cdot \binom{m-\alpha_1-\alpha_2}{\alpha_3} \cdot ... \cdot \binom{m-\alpha_1-...-\alpha_{k-1}}{\alpha_k} =$$
$$= \frac{m!}{\alpha_1!(m-\alpha_1)!} \cdot \frac{(m-\alpha_1)!}{\alpha_2!(m-\alpha_1-\alpha_2)!} \cdot \frac{(m-\alpha_1-\alpha_2)!}{\alpha_3!(m-\alpha_1-\alpha_2-\alpha_3)!} \cdot \ \dots$$
$$\dots \cdot \frac{(m-\alpha_1-\alpha_1-...-\alpha_{k-1})!}{\alpha_k!(m-\alpha_1-\alpha_1-...-\alpha_{k-1}-\alpha_k)!} =$$
$$= \frac{m!}{\alpha_1!\alpha_2!...\alpha_k!} = \binom{m}{\alpha_1, \alpha_2, ..., \alpha_k}.$$

Замечание 1.5.2 . *В примере 1.5.9 содержится доказательство утверждения 1.5.12. Действительно, выбрать α_1 элементов из m для первого подмножества можно $\binom{m}{\alpha_1}$ способами, α_2 из оставшихся элементов для второго подмножества — $\binom{m-\alpha_1}{\alpha_2}$ способами и так далее.*

Итак, в комбинаторном смысле мультиномиальный коэффициент $\binom{m}{\alpha_1,\alpha_2,...,\alpha_k}$ равен числу разбиений m-элементного множества на k подмножеств мощностей α_1, α_2, ..., α_k.

Пример 1.5.10 . *В студенческой группе, состоящей из 25 человек, при выборе профорга за выбранную кандидатуру проголосовало 12 человек, против — 10, воздержалось — 3. Сколькими способами могло быть проведено такое голосование?*

Пусть S — множество студентов группы, S_1 — множество студентов проголосовавших за выдвинутую кандидатуру, S_2 — множество проголосовавших против, S_3 — множество

воздержавшихся. Тогда $|S| = 25$, $|S_1| = 12$, $|S_2| = 10$, $|S_3| = 3$, $S = S_1 \cup S_2 \cup S_3$, $S_i \cap S_j = \varnothing$, $i \neq j$.

Следовательно искомое число равно

$$\binom{25}{12, 10, 3} = \frac{25!}{12!10!3!} = 1\ 487\ 285\ 800.$$

Тепеь рассмотрим, как можно подсчитать число неупорядоченных разбиений множества на подмножества заданной мощности.

Пусть дано множество S мощности m и числа n_1, n_2, ..., n_m таковы, что $\sum_{i=1}^{m} n_i \cdot i = m$. Рассмотрим такие разбиения множества S на подмножества, что среди множеств разбиения ровно n_i множеств мощности i, для каждого $i = \overline{1, m}$, причем порядок множеств разбиения не важен. Обозначим число таких разбиений за $N(n_1, n_2, ..., n_m)$.

Утверждение 1.5.13 . *Пусть* $m \in \mathbb{N}$ *и* $n_1, n_2, ..., n_m \in \mathbb{N}_0$ — *такие числа, что*

$$\sum_{i=1}^{m} n_i \cdot i = m.$$

Тогда

$$N(n_1, n_2, ..., n_m) = \frac{m!}{n_1!n_2! \cdots n_m!(1!)^{n_1}(2!)^{n_2} \cdots (m!)^{n_m}}.$$

Доказательство. Каждое из неупорядоченных разбиений, рассмотренных при определении величины $N(n_1, ..., n_m)$, можно, нумеруя множества в этом разбиении, привести $n_1! \cdots n_m!$ способами к упорядоченным разбиениям. Действительно, множества мощности 1 можно расставить $n_1!$ способами, множества мощности 2 — $n_2!$ способами, ..., множества мощности m — $n_m!$ способами. С другой стороны, число упорядоченных разбиений множества S можно подсчитать по утверждению 1.5.12. Тогда получим

$$\binom{m}{(1!)^{n_1}, (2!)^{n_2}, ..., (m!)^{n_m}} = N(n_1, n_2, ..., n_m) \cdot n_1!n_2! \cdots n_m!,$$

что и доказывает утверждение.

□

Пример 1.5.11 . *Сколькими способами можно разбить множество из десяти элементов на четыре подмножества (порядок подмножеств не важен), чтобы в одном из подмножеств был один элемент, в двух по два элемента и в одном — пять элементов? Ответ:*

$$N(1,2,0,0,1,0,0) = \frac{10!}{1!2!1!(1!)^1(2!)^2(5!)^1} = \frac{7!}{2 \cdot 4 \cdot 5!} = 3780$$

Пример 1.5.12 . *Сколькими способами из группы в 20 человек можно сформировать 4 каолиции по 5 человек?*
Пусть S множество людей в группе, n_i — число каолиций по i человек, $i = \overline{1,25}$. Тогда ответ:

$$N(0,0,0,0,4,0,...,0) = \frac{20!}{4!(5!)^4}$$

1.5.10 Приложение: программа перебора сочетаний

Приведем код программы, которая перебирает все (0,1)-вектора длины n с k единицами (а значит все сочетания из n по k).

```
#include<iostream.h>

const
n=7,
m=4;

void main()
{

cout << "Перебор сочетаний из " << n << " по " << m << ":"
<< endl << endl ;

int i,j,k;
int x[m];// Вектор номеров позиций единиц
int c[n];// (0,1)-вектор текущего сочетания.
 // Вспомогательный массив для вывода на печать.
```

```
// Исходная инициализация
for(i=0;i<m;i++)
x[i]=i;

////////////////////////////////
// Стандартный шаг алгоритма //
do {
////////////////////
// Вывод на экран //

// Инициализируем массив c[]
for(i=0; i<n; i++)
c[i]=0;
for(i=0; i<m; i++)
c[x[i]]=1;

// Выведем его на экран
for(i=0; i<n; i++)
cout << c[i];
cout << endl;


//////////////////////////////////////////
// Очередной шаг модификации сочетания //

k=m-1; // номер позиции самой последней единицы,
       // которую можно сдвинуть.
while( (k>=0) && (x[k]==n-m+k) )
k--;

////////////////////////////////
// Если позиция k существует //
```

```
if(k>=0)
{
// Увеличиваем номер позиции k-той единицы
// и выстраиваем все следующие единицы за ней

x[k]++;

for(i=k+1; i<m; i++)
x[i]=x[k]+(i-k);
}

}
while(k>=0);
}
```

Результатом работы будут все возможные (0,1)-вектора длины семь с четырьмя единицами:

1111000	1100011	1001011	0101101
1110100	1011100	1000111	0101011
1110010	1011010	0111100	0100111
1110001	1011001	0111010	0011110
1101100	1010110	0111001	0011101
1101010	1010101	0110110	0011011
1101001	1010011	0110101	0010111
1100110	1001110	0110011	0001111
1100101	1001101	0101110	

Замечание 1.5.3 . *Покскольку, как показано в параграфе 1.5.7, существует взаимнооднозначное соответствие между (0,1)-векторами и мультимножествами, приведенную программу легко представить и как программу перебора мультимножеств.*

1.6 Перестановки

1.6.1 Понятие перестановки

Определение 1.6.1 . *Перестановкой на множестве* $\{1, ..., n\}$ *называется инъективная функция*

$$\pi \ : \ \{1, ..., n\} \to \{1, ..., n\}.$$

Число n *называется порядком перестановки* π.

Замечание 1.6.1 . *Как любая инъективная на конечном множестве функция, перестановка является биективной.*

Замечание 1.6.2 . *В общем случае, перестановкой произвольного множества* X *называют биекцию*

$$\pi \ : \ X \to X.$$

Обозначим σ_n множество всех перестановок порядка n. Очевидно, что $|\sigma_n| = n!$

Существует несколько способов задания перестановки. Явное задание перестановки

$$\pi = \begin{pmatrix} 1 & 2 & ... & n \\ \pi(1) & \pi(2) & ... & \pi(n) \end{pmatrix}.$$

В записи выше первая строка всегда одинакова. Для задания перестановки достаточно второй строки

$$\pi = \pi(1)\pi(2)...\pi(n).$$

Также можно задать перестановку перечислением ее циклов, о чем мы скажем позже.

Определение 1.6.2 . *Пусть* $\pi_1, \pi_2 \in \sigma_n$. *Произведением перестановок* π_2 *и* π_1 *называют композицию этих функций:* $\pi_2 \circ \pi_1(i) = \pi_2(\pi_1(i))$, $i = \overline{1, n}$.

Докажем, что произведение перестановок есть перестановка. Очевидно, что $\pi_2 \circ \pi_1 \ : \ \{1, ..., n\} \to \{1, ..., n\}$. Покажем инъективность.

Пусть $i, j \in \{1, ..., n\}$ и $i \neq j$. Поскольку π_1 инъективна, $\pi_1(i) \neq \pi_1(j)$, а поскольку инъективна π_2, $\pi_2(\pi_(i)) \neq \pi_2(\pi_1(j))$. Следовательно,

$$\pi_2 \circ \pi_1(i) = \pi_2(\pi_1(i)) \neq \pi_2(\pi_1(j)) = \pi_2 \circ \pi_1(j).$$

То есть $\pi_2 \circ \pi_1$ — инъективна, а значит является перестановкой.

1.6.2 Группа перестановок

Определение 1.6.3 . *Группой называется непустое множество G с определенной на нем бинарной операцией $\diamond$, удовлетворяющей трем аксиомам:*

1 ассоциативность: для любых $a, b, c \in G$ верно, что

$$(a \diamond b) \diamond c = a \diamond (b \diamond c);$$

2 наличие нейтрального элемента: существует такой элемент $e \in G$, что для любого $a \in G$ справедливо

$$a \diamond e = e \diamond a = a;$$

3 наличие обратного элемента: для любого элемента $a \in G$ найдется такой элемент $a^{-1} \in G$, что

$$a^{-1} \diamond a = a \diamond a^{-1} = e.$$

Произведение перестановок ассоциативно:

$$((\pi_3 \circ \pi_2) \circ \pi_1)(i) = (\pi_3 \circ \pi_2)(\pi_1(i)) = \pi_3(\pi_2(\pi_1(i)) = \\ = \pi_3((\pi_2 \circ \pi_1)(i)) = (\pi_3 \circ (\pi_2 \circ \pi_1))(i)$$

Нейтральным элементом для множества σ_n будет служить тождественная перестановка $e = \begin{pmatrix} 1 & 2 & ... & n \\ 1 & 2 & ... & n \end{pmatrix}$. Легко заметить, что для любой перестановки $\pi \in \sigma_n$ верно $\pi \circ e = e \circ \pi = \pi$.

Для любой перестановки $\pi \in \sigma_n$, как для любой биективной функции, существует обратная функция π^{-1}: $\pi^{-1} \circ \pi = \pi \circ \pi^{-1} = e$. Для любых $i \in \{1, ..., n\}$ $\pi(i) = j \Rightarrow \pi^{-1}(j) = i$. Функция π^{-1} обратная для биективной функции π также будет биективной фукнцией: $\pi^{-1} : \{1, ..., n\} \to \{1, ..., n\}$. То есть, $\pi^{-1} \in \sigma_n$.

Замечание 1.6.3 . *Заметим, что обратная перестановка π^{-1} для данной перестановки π единственна. Действиетльно, если бы существовала еще одна такая перестановка π', что $\pi \circ \pi' = e$, то*

$$\pi' = e \circ \pi' = \pi^{-1} \circ \pi \circ \pi' = \pi^{-1} \circ e = \pi^{-1};$$

если перестановка π'' такова, что $\pi'' \circ \pi = e$, то

$$\pi^{-1} = e \circ \pi^{-1} = \pi'' \circ \pi \circ \pi^{-1} = \pi'' \circ e = \pi''.$$

Таким образом мы доказали, что множество перестановок σ_n с операцией произведения перестановок образуют группу. Эту группу называют группой перестановок или симметрической группой.

Замечание 1.6.4 . *Произведение перестановок в общем случае не коммутативно.*

Пример 1.6.1 . *Пусть*

$$\pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}, \qquad \pi_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}.$$

Тогда

$$\pi_2 \circ \pi_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \neq \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix} = \pi_1 \circ \pi_2.$$

1.6.3 Циклы перестановки

Будем обозначать

$$\pi^k = \underbrace{\pi \circ \pi \circ \cdots \circ \pi}_{k}; \qquad \pi^0 = e;$$
$$\pi^{-k} = \underbrace{\pi^{-1} \circ \pi^{-1} \circ \cdots \circ \pi^{-1}}_{k}.$$

Определение 1.6.4 . *Циклом длины l называется такая перестановка π которая тождественна на всём множестве $\{1, 2, ..., n\}$, кроме подмножества $\{x_1, ..., x_l\}$. Кроме того $\pi(x_l) = x_1$ и $\pi(x_i) = x_{i+1}$, $i = \overline{1, l-1}$.*

Цикл обычно обозначается

$$(x_1, x_2, ..., x_l) = (x_1, \pi(x_1), ..., \pi^{l-1}(x_1)).$$

Определение 1.6.5 . *Транспозиция - перестановка элементов множества* $\{1, 2, ..., n\}$, *которая меняет местами только два элемента.*

Замечание 1.6.5 . *Транспозиция — цикл длины 2.*

Утверждение 1.6.1 . *Пусть* $\pi \in \sigma_n$ *и* $i \in \{1, ..., n\}$. *Тогда существует такое число* $k \in \mathbb{N}$, *что* $\pi^k(i) = i$.

Доказательство. Пусть $i \in \{1, ..., n\}$. Пусть для любого $k \in \mathbb{N}$ верно $\pi^k(i) \neq i$. Так как множество $\{1, ..., n\}$ конечно, то элементы последовательности $i, \pi(i), \pi^2(i), ..., \pi^s(i), ...$ начинают повторяться начиная с некоторого момента. Тогда последовательность имеет вид:

$$a_0 = i, a_1, ..., a_l, b_1, b_2, ..., b_m, b_1, b_2, ..., b_m, b_1, ...$$

где $l \geq 1$, $i \notin \{a_1, ..., a_l, b_1, b_2, ..., b_m\}$ и $\{a_0, a_1, ..., a_l\} \cap \{b_1, b_2, ..., b_m\} = \varnothing$. Элемент b_1 — самый ранний повторяющийся элемент этой последовательности.

Но тогда $\pi(a_l) = \pi(b_m) = b_1$ и по инъективности перестановки π получаем $a_l = b_m$. Таким образом b_1 был не самым первым повторяющимся элементом. Противоречие доказывает утверждение.
□

Для произвольной перестановки $\pi \in \sigma_n$ введем бинарное отношение ρ_π на множестве $\{1, 2, ..., n\}$:

$$x \rho_\pi y \quad \Leftrightarrow \quad \exists k \in \mathbb{Z} : \ y = \pi^k(x).$$

Покажем, что $\rho_\pi \in \sigma_n$ — отношение эквивалентности.

1) Рефлексивность. Для любого $x \in \{1, ..., n\}$ $\quad x = \pi^0(x) \Rightarrow x \rho_\pi x$.

2) Симметричность. Для любых $x, y \in \{1, ..., n\}$, для которых $x \rho_\pi y$, существует $k \in \mathbb{Z}$: $y = \pi^k(x)$. Следовательно $x = \pi^{-k}(y)$ и $y \rho_\pi x$.

3) Транзитивность. Пусть $x, y, z \in \{1, 2, ..., n\}$, $x \rho_\pi y$ и $y \rho_\pi z$. Тогда существуют $k_1, k_2 \in \mathbb{Z}$: $y = \pi^{k_1}(x)$, $z = \pi^{k_2}(y)$. Следовательно

$$z = \pi^{k_2}(\pi^{k_1}(x)) = \pi^{k_1 + k_2}(x).$$

То есть $x \rho_\pi z$.

Пусть $\{1,2,...,n\} = B_1 \cup B_2 \cup ... \cup B_m$ — разбиение $\{1,2,...,n\}$ на классы эквивалентности относительно ρ_π. B_i называются орбитами перестановки.

Утверждение 1.6.2 . *Пусть $\pi \in \sigma_n$. Пусть $B_1 \cup B_2 \cup ... \cup B_m$ разбиение множества $\{1,2,...,n\}$ на классы эквивалентности, порожденное отношением ρ_π, и пусть $x_i \in B_i$. Тогда,*

1) для любого $i \in \{1,2,...,m\}$ существует $n_i \in \mathbb{N}$:

$$B_i = \{x_i, \pi(x_i), ..., \pi^{n_i-1}(x_i)\}.$$

2) перестановка π представима в виде произведения циклов:

$$\pi = (x_1, \pi(x_1), ..., \pi^{n_1-1}(x_1)) \circ (x_2, \pi(x_2), ..., \pi^{n_2-1}(x_2)) \circ \\ \circ \cdots \circ (x_m, \pi(x_m), ..., \pi^{n_m-1}(x_m)).$$

Доказательство. 1) По определению, B_i состоит из всех таких y для которых существует $s \in \mathbb{Z}$: $y = \pi^s(x_i)$. По утверждению 1.6.1, для x_i существует $k \in \mathbb{N}$: $\pi^k(x_i) = x_i$. Пусть n_i — наименьшее из таких чисел. Тогда $x_i, \pi(x_i), ..., \pi^{n_i-1}(x_i)$ — различные элементы из $\{1,2,...,n\}$ и $\{x_i, \pi(x_i), ..., \pi^{n_i-1}(x_i)\} \subseteq B_i$.

Пусть $y \in B_i \setminus \{x_i, \pi(x_i), ..., \pi^{n_i-1}(x_i)\}$. Тогда существует такое $s \in \mathbb{Z}$, что $y = \pi^s(x_i)$ и $s \notin \{1,2,...,n_i-1\}$. $s = p \cdot n_i + q$, $p \in \mathbb{Z} \setminus \{0\}$, $q \in \{0,1,...,n_i-1\}$. Следовательно,

$$y = \pi^s(x_i) = \pi^q(\pi^{p \cdot n_i}(x_i)).$$

Если $p > 0$, то

$$\pi^{p \cdot n_i}(x_i) = \underbrace{\pi^{n_i} \circ \pi^{n_i} \circ \cdots \circ \pi^{n_i}}_{p}(x_i) = x_i. \qquad (*)$$

Пусть теперь $p < 0$. Тогда, используя равенство $(*)$, получим

$$\pi^{p \cdot n_i}(x_i) = \underbrace{\pi^{-n_i} \circ \pi^{-n_i} \circ \cdots \circ \pi^{-n_i}}_{-p}(x_i) = \\ = \underbrace{\pi^{-n_i} \circ \pi^{-n_i} \circ \cdots \circ \pi^{-n_i}}_{-p}(\underbrace{\pi^{n_i} \circ \pi^{n_i} \circ \cdots \circ \pi^{n_i}}_{-p}(x_i)) = x_i,$$

поскольку

$$\pi^{-n_i} \circ \pi^{n_i}(i) = \underbrace{\pi^{-1} \circ \pi^{-1} \circ ... \circ \pi^{-1}}_{n_i} \circ \underbrace{\pi \circ \pi \circ ... \circ \pi}_{n_i}(i) = i.$$

Таким образом, $y = \pi^q(x_i)$, где $q \in \{0, 1, ..., n_i - 1\}$, что противоречит выбору y. Следовательно, $B_i = \{x_i, \pi(x_i), ..., \pi^{n_i-1}(x_i)\}$.

2) Согласно пункту 1) доказательства для цикла $(x_i, \pi(x_i), ..., \pi^{n_1-1}(x_i))$ выполняется формула

$$(x_i, \pi(x_i), ..., \pi^{n_1-1}(x_i))(x) = \begin{cases} x, & x \notin B_i, \\ \pi(x), & x \in B_i. \end{cases}$$

Обозначим $\pi_i = (x_i, \pi(x_i), ..., \pi^{n_1-1}(x_i))$, $i = \overline{1, m}$. Если $x \in B_j$ и $j \neq i$, то $\pi_i(x) = x$. То же верно и для $\pi(x)$, поскольку $\pi(x) \in B_j$.

Пусть $x \in \{1, 2, ..., n\}$ — произвольное значение. Пусть, не умаляя общности, $x \in B_i$. Тогда,

$$\begin{aligned} &(\pi_1 \circ \pi_2 \circ \cdots \circ \pi_i \circ \cdots \circ \pi_{m-1} \circ \pi_m)(x) = \\ &= (\pi_1 \circ \pi_2 \circ \cdots \circ \pi_i \circ \cdots \circ \pi_{m-1})(\pi_m(x)) = \\ &= (\pi_1 \circ \pi_2 \circ \cdots \circ \pi_i \circ \cdots \circ \pi_{m-1})(x) = \cdots = \\ &= (\pi_1 \circ \pi_2 \circ \cdots \circ \pi_i)(x) = (\pi_1 \circ \pi_2 \circ \cdots \circ \pi_{i-1})(\pi_i(x)) = \\ &= (\pi_1 \circ \pi_2 \circ \cdots \circ \pi_{i-1})(\pi(x)) = \cdots = \pi_1(\pi(x)) = \pi(x). \end{aligned}$$

Это верно для любого $x \in \{1, 2, ..., n\}$. Таким образом,

$$\pi = (\pi_1 \circ \pi_2 \circ \cdots \circ \pi_m),$$

что и требовалось доказать.

□

Сопоставим каждой орбите B_i перестановку π_i:

$$\pi_i(x) = \begin{cases} x, & x \notin B_i, \\ \pi(x), & x \in B_i. \end{cases}$$

Будем называть циклы π_i, $i = \overline{1, m}$, циклами перестановки π.

Замечание 1.6.6 . *Из утверждения 1.6.2 следует, что перестановку π можно представить в виде произведения всех ее циклов:*

$$\pi = \pi_1 \circ \pi_2 \circ \cdots \circ \pi_m.$$

Поскольку орбиты не пересекаются, перестановки в этой композиции можно располагать в любом порядке:

$$\pi = \pi_{i_1} \circ \pi_{i_2} \circ \cdots \circ \pi_{i_m},$$

где $i_1 i_2 ... i_m$ *— произвольная перестановка из* σ_m.

Пример 1.6.2 . *Пусть* $n = 7$ *и*

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 2 & 7 & 1 & 3 & 6 & 5 \end{pmatrix}.$$

Найдем орбиты π.

$$\begin{array}{ll} \pi(1) = 4, \pi^2(1) = \pi(4) = 1 & \Rightarrow B_1 = \{1, 4\}, \\ \pi(2) = 2, & \Rightarrow B_2 = \{2\}, \\ \pi(3) = 7, \pi^2(3) = \pi(7) = 5, \pi^3(3) = \pi(5) = 3 & \Rightarrow B_3 = \{3, 5, 7\}, \\ \pi(6) = 6, & \Rightarrow B_4 = \{6\}. \end{array}$$

Представление перестановки π *в виде произведения циклов имеет вид* $\pi = (1, 4)(2)(3, 7, 5)(6)$.

Пример 1.6.3 . *Пусть* $n = 7$ *и*

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 2 & 5 & 1 & 6 & 7 & 3 \end{pmatrix}.$$

Рассмотрим иллюстрацию к утверждению 1.6.2 (рисунок 11). Обозначим каждый элемент множества $\{1, 2, ..., n\}$ *вершиной графа. Будем рисовать дугу из вершины* i *в вершину* j, *если* $\pi(i) = j$. *Поскольку перестановка является биективной функцией, из каждой вершины выходит ровно одна дуга и в каждую вершину входит тоже ровно одна дуга. Получившийся граф оказывается гамильтоновым графом. Таким образом, все множество дуг графа разбивается на непересекающиеся контуры:* $((1, 4), (4, 1))$, $((2, 2))$, $((3, 5), (5, 6), (6, 7), (7, 3))$. *Каждому такому контуру соответствует цикл перестановки* π.

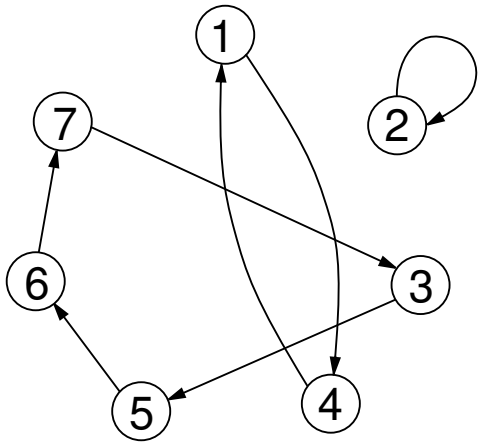

Рисунок 11: Разбиение перестановки на циклы

Определение 1.6.6 . *$p \in \mathbb{N}$ — степень перестановки π, если p — наименьшее из таких чисел, что $\pi^p = e$.*

Утверждение 1.6.3 . *Пусть $\pi \in \sigma_n$, $\pi = \pi_1 \circ \pi_2 \circ \cdots \circ \pi_m$ — разложение π на непересекающиеся циклы; n_i — длина цикла π_i. Степень перестановки определяется как наименьшее общее кратное длин ее циклов:*

$$Степень\ \pi = НОК(n_1, n_2, ..., n_m).$$

Доказательство. Пусть число $p \in \mathbb{N}$ таково, что $\pi^p = e$. Тогда $\pi_1^p \circ \pi_2^p \circ \cdots \circ \pi_m^p = e$. Поскольку все перестановки π_i воздействуют на разные элементы $\{1, 2, ..., n\}$, то $\pi_i^p = e$, $i = \overline{1, m}$. Следовательно, p является общим кратным для n_1, n_2, ..., n_m.

С другой стороны, пусть $q = \text{НОК}(n_1, n_2, ..., n_m)$. Тогда $\pi_i^q = \pi_i^{n_i \cdot t} = e$, $i = \overline{1, m}$, и $\pi^q = \pi_1^q \circ \pi_2^q \circ \cdots \circ \pi_m^q = e$.
□

Пример 1.6.4 . *Степень перестановки из примера 1.6.2 равна* 6.

1.6.4 Тип перестановки

Определение 1.6.7 . *Пусть $c_i = c_i(\pi)$ — число циклов длины i перестановки π. Тогда $(c_1, c_2, ..., c_n)$ — тип перестановки π.*

Обозначим

$$\sigma_n(c_1, c_2, ..., c_n) = \{\pi \mid \pi \in \sigma_n,\ (c_1, c_2, ..., c_n) \text{ — } \textit{тип } \pi\}.$$

Замечание 1.6.7 . $c(\pi) = c_1(\pi) + \cdots + c_n(\pi)$ — *число циклов перестановки и*

$$n = \sum_{i=1}^{n} i \cdot c_i(\pi).$$

Пример 1.6.5 . *Тип перестановки из примера 1.6.2* — $(2, 1, 1, 0, 0, 0, 0)$. *Тип перестановки из примера 1.6.3* — $(1, 1, 0, 1, 0, 0, 0)$.

Утверждение 1.6.4 .

$$|\sigma_n(c_1, c_2, ..., c_n)| = \frac{n!}{c_1!c_2! \cdots c_n! 1^{c_1} 2^{c_2} \cdots n^{c_n}}$$

Доказательство. По утверждению 1.5.13 множество $\{1, ..., n\}$ можно разбить на подмножества, среди которых ровоно c_i подмножеств имеют мощность i, $\frac{n!}{c_1!c_2! \cdots c_n! (1!)^{c_1} (2!)^{c_2} \cdots (n!)^{c_n}}$ способами. Чтобы получить перестановку из любого неупорядоченного разбиения, нужно расставить элеметы каждого подмножества в определенном порядке, чтобы определить циклы.

Сколько циклов можно получить из одного подмножества мощности i? Элементы множества можно расставить в различном порядке $i!$ способами. Это число надо разделить на количество вариантов выбора начальный точки цикла — i. Таким образом из данного подмножества можно составить $\frac{i!}{i}$ различных циклов. Домножим число неупорядоченных разбиений на число вариантов создания циклов из подмножеств. Получим:

$$\frac{n!}{c_1!c_2! \cdots c_n! (1!)^{c_1} (2!)^{c_2} \cdots (n!)^{c_n}} \cdot \frac{(1!)^{c_1} (2!)^{c_2} \cdots (n!)^{c_n}}{1^{c_1} 2^{c_2} \cdots n^{c_n}} =$$

$$= \frac{n!}{c_1!c_2! \cdots c_n! 1^{c_1} 2^{c_2} \cdots n^{c_n}},$$

что и требовалось доказать.
□

Пример 1.6.6 . *Пусть задан тип перестановки* $(0,2,1,0,0,0,0)$. *Согласно утверждению 1.6.4 число перестановок такого типа должно быть* $\frac{7!}{0!2!1!0!0!0!0!1^0 2^2 3^1 4^0 5^0 6^0 7^0} = \frac{7!}{24} = 7 \cdot 6 \cdot 5 = 210$, *где* 24 — *число перестановок, из которых можно получить одну и ту же перестановку типа* $(0,2,1,0,0,0,0)$.

Рассмотрим, например, перестановку $(1,2)(3,4)(5,6,7)$. *Из каких перестановок она может быть получена добавлением скобок? Перечислим такие перестановки:*

1) $(1,2,3,4,5,6,7)$, *13)* $(3,4,1,2,5,6,7)$,
2) $(2,1,3,4,5,6,7)$, *14)* $(3,4,2,1,5,6,7)$,
3) $(1,2,4,3,5,6,7)$, *15)* $(4,3,1,2,5,6,7)$,
4) $(2,1,4,3,5,6,7)$, *16)* $(4,3,2,1,5,6,7)$,
5) $(1,2,3,4,6,7,5)$, *17)* $(3,4,1,2,6,7,5)$,
6) $(2,1,3,4,6,7,5)$, *18)* $(3,4,2,1,6,7,5)$,
7) $(1,2,4,3,6,7,5)$, *19)* $(4,3,1,2,6,7,5)$,
8) $(2,1,4,3,6,7,5)$, *20)* $(4,3,2,1,6,7,5)$,
9) $(1,2,3,4,7,5,6)$, *21)* $(3,4,1,2,7,5,6)$,
10) $(2,1,3,4,7,5,6)$, *22)* $(3,4,2,1,7,5,6)$,
11) $(1,2,4,3,7,5,6)$, *23)* $(4,3,1,2,7,5,6)$,
12) $(2,1,4,3,7,5,6)$, *24)* $(4,3,2,1,7,5,6)$.

1.7 Разложения и разбиения натуральных чисел

1.7.1 Разложения натуральных чисел

Определение 1.7.1 . *Разложением числа* $n \in \mathbb{N}$ *называется представление* n *в виде упорядоченной суммы натуральных чисел*

$$(a_1, a_2, ..., a_k): \quad n = a_1 + a_2 + ... + a_k, \quad a_i \in \mathbb{N},\ k \in \mathbb{N}.$$

Пример 1.7.1 . *Существует всего* 8 *различных разложений числа* $n = 4$*:*

$$\begin{aligned} &1+1+1+1\\ &1+1+2\\ &1+2+1\\ &2+1+1\\ &1+3\\ &3+1\\ &2+2\\ &4 \end{aligned}$$

Обозначим $d_k(n)$ — число разложений n на k частей.

Утверждение 1.7.1 . *Число разложений* n *на* k *частей равно*

$$d_k(n) = \binom{n-1}{k-1}. \tag{15}$$

Доказательство. Пусть $(a_1, a_2, ..., a_k)$ — произвольное разложение n на k частей: $n = a_1 + a_2 + ... + a_k$. Тогда

$$1 \le a_1 < a_1 + a_2 < a_1 + a_2 + a_3 < ... < a_1 + a_2 + ... + a_{k-1} \le n - 1.$$

Обозначим $b_i = a_1 + a_2 + ... + a_i, \quad 1 \le i \le k-1$. Тогда $\{b_1, ..., b_{k-1}\} \subset \{1, ..., n-1\}$, причем никакие два b_i и b_j не совпадают. Значит $\{b_1, ..., b_{k-1}\}$ — некоторая выборка $k-1$ элемента из $n-1$. При этом, различным разложениям n на k частей будут соответствовать разные выборки из $n-1$ по $k-1$. Следовательно, $d_k(n) \le \binom{n-1}{k-1}$.

Пусть теперь, $\{b_1, ..., b_{k-1}\}$ произвольная выборка мощности $k-1$ из $n-1$. Пусть, для определенности, $b_1 < b_2 < ... < b_{k-1}$. Тогда,

$$b_1 + (b_2 - b_1) + (b_3 - b_2) + \cdots + (b_{k-1} - b_{k-2}) + (n - b_{k-1}) = n.$$

Обозначим $a_1 = b_1$, $a_k = n - b_{k-1}$ и $a_i = b_i - b_{i-1}$, $i = \overline{2, k-1}$. Тогда, $n = a_1 + a_2 + \cdots + a_k$ — разложение n на k частей. Причем разным выборкам будут соответствовать различные разложения и, следовательно, $d_k(n) \geq \binom{n-1}{k-1}$.

Таким образом, $d_k(n) = \binom{n-1}{k-1}$.

□

Следствие 1.7.2 . *Рассмотрим уравнение*

$$x_1 + x_2 + \cdots + x_k = n, \qquad (*)$$

где n — заданное число из $\mathbb{N}_0$. Пусть, $N_=(n, k)$ — число решений этого уравнения на неотрицательных целых числах ($x_i \in \mathbb{N}_0$). Тогда

$$N_=(n, k) = \binom{n + k - 1}{k - 1}. \qquad (16)$$

Доказательство. Пусть $n = c_1 + c_2 + \cdots + c_k$, $c_i \in \mathbb{N}_0$. Тогда $n + k = (c_1 + 1) + (c_2 + 1) + \cdots + (c_k + 1)$, $c_i \in \mathbb{N}$. Таким образом, любому решению уравнения (*) соответствует разложение $n + k$ на k частей. И наоборот, для любого разложения

$$n + k = a_1 + a_2 + ... + a_k, \quad a_i \in \mathbb{N}$$

набор $\{(a_1 - 1), (a_2 - 1), ..., (a_k - 1)\}$ является решением уравнения (*).

То есть мы построили взаимнооднозначное соответствие между множеством решений уравнеия и множеством разложений $n + k$ на k слогаемых. Следовательно, по утверждению 1.7.1

$$N_=(n, k) = d_k(n + k) = \binom{n + k - 1}{k - 1}.$$

□

Следствие 1.7.3 . *Рассмотрим неравенство*

$$x_1 + x_2 + \cdots + x_k \leq n, \qquad (**)$$

где n — заданное число из $\mathbb{N}_0$. Пусть, $N_{\leq}(n,k)$ — число решений этого неравенства на неотрицательных целых числах. Тогда

$$N_{\leq}(n,k) = \binom{n+k}{k}. \qquad (17)$$

Доказательство. Пусть $c_1 + c_2 + \cdots + c_k \leq n$, $c_i \in \mathbb{N}_0$. Тогда, обозначим $c_{k+1} = n - c_1 + c_2 + \cdots + c_k$. $\{c_1, c_2, ..., c_k, c_{k+1}\}$ — решение уравнения

$$x_1 + x_2 + \cdots + x_k + x_{k+1} = n.$$

Обратно, для любого решения $c_1, c_2, ..., c_k, c_{k+1}$ уравнения $x_1 + x_2 + \cdots + x_k + x_{k+1} = n$, очевидно, верно, что $c_1 + c_2 + \cdots + c_k \leq n$.

Таким образом, множеству решений неравенства (**) взаимооднозначно соответствует множество решений уравнения $x_1 + x_2 + \cdots + x_k + x_{k+1} = n$. Тогда по следствию 1.7.2

$$N_{\leq}(n,k) = N_{=}(n,k+1) = \binom{n+k}{k}.$$

□

1.7.2 Разбиения натуральных чисел

Определение 1.7.2 . *Набор $(\lambda_1, \lambda_2, ..., \lambda_k)$, $\lambda_i \in \mathbb{N}$, называется разбиением числа $n \in \mathbb{N}$, если $n = \lambda_1 + \lambda_2 + \cdots + \lambda_k$, $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0$*

Пример 1.7.2 . *Существует 5 различных разбиений числа $n = 4$:*

$$1+1+1+1$$
$$2+1+1$$
$$2+2$$
$$3+1$$
$$4$$

Замечание 1.7.1 . *Можно использовать и другое определение разбиения натурального числа:*

Определение 1.7.3 . *Разбиением числа $n \in \mathbb{N}$ называется представление n в виде неупорядоченной суммы натуральных чисел.*

Действительно, можно считать, что разбиение $\{\lambda_1, \lambda_2, ..., \lambda_k\}$ — произвольный неупорядоченный набор, такой что $n = \lambda_1 + \lambda_2 + \cdots + \lambda_k$. Тогда расставив элементы λ_i в другом порядке мы получим то же самое разбиение. Чтобы получить возможность определять, имеем ли мы дело с двумя различными разбиениями, или с одним и тем же набором, но в с элементами следующими в другом порядке, удобно расставить элементы λ_i по убыванию. В таком случае, любому разбиению будет соответствовать упорядоченная последовательность, удовлетворяющая определению 1.7.2.

Обозначим $p_k(n)$ — число разбиений n на k частей.

Утверждение 1.7.4 . *Пусть $k, n \in \mathbb{N}$, $k > 1$ и $n > k$. Тогда*

$$p_k(n) = p_{k-1}(n-1) + p_k(n-k). \tag{18}$$

Доказательство. Пусть

$$S_k(n) = \{(\lambda_1, \lambda_2, ..., \lambda_k) \mid n = \lambda_1 + \lambda_2 + \cdots + \lambda_k, \\ \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 0,\ \lambda_i \in \mathbb{N}, i = \overline{1,k}\},$$

$$A = \{(\lambda_1, \lambda_2, ..., \lambda_k) \mid (\lambda_1, \lambda_2, ..., \lambda_k) \in S_k(n),\ \lambda_k = 1\},$$
$$B = \{(\lambda_1, \lambda_2, ..., \lambda_k) \mid (\lambda_1, \lambda_2, ..., \lambda_k) \in S_k(n),\ \lambda_k \geq 2\}.$$

Тогда $S_k(n) = A \cup B$ и $A \cap B = \varnothing$; $|S_k(n)| = |A| + |B|$.

$$|A| = |\{(\lambda_1, \lambda_2, ..., \lambda_{k-1}) \mid \lambda_1 + \lambda_2 + \cdots + \lambda_{k-1} = n-1, \\ \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{k-1} > 0,\ \lambda_i \in \mathbb{N}\}| = |S_{k-1}(n-1)| = p_{k-1}(n-1).$$

$$|B| = |\{(\lambda_1, \lambda_2, ..., \lambda_k) \mid \lambda_1 + \lambda_2 + \cdots + \lambda_k = n, \\ \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k > 1,\ \lambda_i \in \mathbb{N}\}| =$$

Сделаем замену $\mu_i = \lambda_i - 1$, $i = \overline{1,k}$.

$$= |\{(\mu_1, \mu_2, ..., \mu_k) \mid \mu_1 + \mu_2 + \cdots + \mu_k = n - k,$$
$$\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k > 0,\ \mu_i \in \mathbb{N}\}| = |S_k(n-k)| = p_k(n-k).$$

$$|S_k(n)| = p_k(n) = p_{k-1}(n-1) + p_k(n-k).$$

□

Замечание 1.7.2 . *Несмотря на схожесть определений вычисление величины $p_k(n)$ оказывается гораздо более сложной, чем $d_k(n)$. Если для числа разложений мы имеем явную формулу* (15)*, то для числа разбиений мы получили только рекурентное выражение* (18)*, которое, тем не менее, позволяет нам вычислить $p_k(n)$, пользуясь начальными условиями: $\forall n \in \mathbb{N} \quad p_n(n) = 1$, $p_1(n) = 1$; если $k > n$, то $p_k(n) = 0$.*

Пример 1.7.3 . *Заполним таблицу первых значений $p_k(n)$, пользуясь начальными значениями и формулой* (18).

$n\backslash k$	1	2	3	4	5	6	7	$\cdots$
1	1	0	0	0	0	0	0	$\cdots$
2	1	1	0	0	0	0	0	$\cdots$
3	1	1	1	0	0	0	0	$\cdots$
4	1	2	1	1	0	0	0	$\cdots$
5	1	2	2	1	1	0	0	$\cdots$
6	1	3	3	2	1	1	0	$\cdots$
7	1	3	4	3	2	1	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

$$p_2(3) = p_1(2) + p_2(1) = p_1(2) = 1,$$
$$p_2(4) = p_1(3) + p_2(2) = 1 + 1 = 2,$$
$$p_3(4) = p_2(3) + p_3(1) = p_2(3) = 1,$$

$$
\begin{aligned}
p_2(5) &= p_1(4) + p_2(3) = 1 + 1 = 2,\\
p_3(5) &= p_2(4) + p_3(2) = p_2(4) = 2,\\
p_4(5) &= p_3(4) + p_4(1) = p_3(4) = 1,\\
p_2(6) &= p_1(5) + p_2(4) = 1 + 2 = 3,\\
p_3(6) &= p_2(5) + p_3(3) = 2 + 1 = 3,\\
p_4(6) &= p_3(5) + p_4(2) = 2,\\
p_5(6) &= p_4(5) + p_5(1) = 1,\\
p_2(7) &= p_1(6) + p_2(5) = 1 + 2 = 3,\\
p_3(7) &= p_2(6) + p_3(4) = 3 + 1 = 4,\\
p_4(7) &= p_3(6) + p_4(3) = 3,\\
p_5(7) &= p_4(6) + p_5(2) = 2,\\
p_6(7) &= p_5(6) + p_6(1) = 1.
\end{aligned}
$$

1.8 Принцип включения-исключения

1.8.1 Принцип включения-исключения

Пусть A — заданное множество и S — множество свойств. Каждый элемент из A может обладать некоторыми свойствами из S. Другими словами, каждому элементу a из множества A сопоставляется некоторое подмножество S_a множества S и говорят, что элемент a обладает свойствами из S_a.

Пусть $T \subseteq S$. Обозначим

$f_=(T)$ — число элементов из множества A, которые обладают всеми свойствами из T и не обладают свойствами из множества $S \setminus T$;

$f_\geq(T)$ — число элементов из A, которые обладают всеми свойствами из T и возможно какими-то еще. Тогда

$$f_\geq(T) = \sum_{T \subseteq Y \subseteq S} f_=(Y). \tag{19}$$

Возможно ли, если мы знаем значение $f_\geq(T)$ для любых $T \subseteq S$, найти функцию $f_=$?

Утверждение 1.8.1 . *Пусть $\varphi : 2^S \to \mathbb{R}$ — произвольная функция, сопоставляющая каждому подмножеству конечного множества S, некоторое вещественное число. Пусть ψ — такая функция из 2^S в $\mathbb{R}$, что*

$$\psi(T) = \sum_{T \subseteq Y \subseteq S} \varphi(Y), \quad \forall\, T \subseteq S.$$

Тогда

$$\varphi(T) = \sum_{T \subseteq Y \subseteq S} (-1)^{|Y \setminus T|} \psi(Y), \quad \forall\, T \subseteq S.$$

Доказательство. Пусть $T \subseteq S$. Тогда

$$\sum_{T \subseteq Y \subseteq S} (-1)^{|Y \setminus T|} \psi(Y) = \sum_{T \subseteq Y \subseteq S} (-1)^{|Y \setminus T|} \sum_{Y \subseteq Z \subseteq S} \varphi(Z) =$$
$$= \sum_{T \subseteq Y \subseteq Z \subseteq S} (-1)^{|Y \setminus T|} \varphi(Z) = \sum_{T \subseteq Z \subseteq S} \varphi(Z) \sum_{T \subseteq Y \subseteq Z} (-1)^{|Y \setminus T|} =$$

$$= \sum_{T\subseteq Z\subseteq S} \varphi(Z) \sum_{i=0}^{|Z\setminus T|} \sum_{\substack{T\subseteq Y\subseteq Z,\\ |Y|=|T|+i}} (-1)^{|Y\setminus T|} =$$

$$= \sum_{T\subseteq Z\subseteq S} \varphi(Z) \sum_{i=0}^{|Z\setminus T|} (-1)^i \binom{|Z\setminus T|}{i} = \sum_{T\subseteq Z\subseteq S} \varphi(Z)\delta_o(|Z\setminus T|) = \varphi(T).$$

Здесь $\delta_o(n)$ — дельта-функция, которую мы обсуждали в замечании 1.5.1 параграфа 1.5.5.
□

Следствие 1.8.2 . *Поскольку,* $f_{\geq}(T) = \sum_{T\subseteq Y\subseteq S} f_{=}(Y)$, *то для любого* $T \subseteq S$

$$f_{=}(T) = \sum_{T\subseteq Y\subseteq S} (-1)^{|Y\setminus T|} f_{\geq}(Y). \tag{20}$$

В частности,

$$f_{=}(\varnothing) = \sum_{\varnothing\subseteq Y\subseteq S} (-1)^{|Y|} f_{\geq}(Y). \tag{21}$$

Этот результат помогает решить многие задачи.

1.8.2 Задача о беспорядках

Интуитивная постановка задачи о беспорядках:

Задача 1.8.1 . *Пусть, в рамках чемпионата в n городах должно пройти ровно по одному матчу. Матчи судят n судей, каждый из которых из одного из городов, все из разных. Сколькими способами можно распределить судей по матчам, чтобы ни один судья не судил в своем городе?*

Можно привести другую формулировку задачи, математическая идея которой будет той же.

Задача 1.8.2 . *Пусть n человек пришли в театр и сдали шляпы в гардероб. Когда спектакль закончился и зрители получили шляпы назад, оказалось, что каждый из n получил чужую шляпу. Сколько существует вариантов такого события?*

Формально задача ставится следующим образом.

Определение 1.8.1 . *Беспорядком на множестве* $\{1, ..., n\}$ *называется перестановка, которая не сохраняет ни одного элемента:* $\pi(i) \neq i,\ i = \overline{1, n}$.

Задача 1.8.3 . *Найти число беспорядков на множестве* $\{1, ..., n\}$*:*

$$D(n) = |\{\pi \mid \pi \in \sigma_n,\ \pi(i) \neq i,\ i = \overline{1, n}\}|.$$

Решим эту задачу с помощью принципа включения-исключения.

Будем говорить, что $\pi \in \sigma_n$ обладает свойством i, если $\pi(i) = i$. Тогда $S = \{1, ..., n\}$ — множество всех свойств, которыми может обладать перестановка. Пусть $T \subseteq S$. Пусть

$f_=(T)$ — число перестановок, каждая из которых обладает всеми свойствами из T и не обладает свойствами из $S \setminus T$:

$$f_=(T) = |\{\pi \mid \pi \in \sigma_n,\ \pi(i) = i, i \in T;\ \pi(j) \neq j, j \in S \setminus T\}|.$$

$f_\geq(T)$ — число перестановок, каждая из которых обладает всеми свойствами из T и возможно какими-то еще:

$$f_\geq(T) = |\{\pi \mid \pi \in \sigma_n,\ \pi(i) = i, i \in T\}|.$$

Следовательно верна формула (19): $f_\geq(T) = \sum_{T \subseteq Y \subseteq S} f_=(Y)$. Нетрудно видеть, что $f_\geq(T) = |S \setminus T|!$ — число перестановок на множестве $S \setminus T$.

Очевидно, $D(n) = f_=(\varnothing)$. Следующее утверждение решает задачу.

Утверждение 1.8.3 . $D(n) = n!\ \sum_{i=0}^{n} \frac{(-1)^i}{i!}$.

Доказательство. Так как $f_\geq(T) = |S \setminus T|!$, то по формуле (21)

$$D(n) = f_=(\varnothing) = \sum_{Y \subseteq S} (-1)^{|Y|} f_\geq(Y) = \sum_{Y \subseteq S} (-1)^{|Y|} |S \setminus Y|! =$$

$$= \sum_{i=0}^{n} \sum_{\substack{Y \subseteq S, \\ |Y| = i}} (-1)^i (n-i)! = \sum_{i=0}^{n} (-1)^i (n-i)! \binom{n}{i} =$$

$$= \sum_{i=0}^{n} (-1)^i \frac{(n-i)!\ n!}{i!\ (n-i)!} = n!\ \sum_{i=0}^{n} \frac{(-1)^i}{i!}.$$

□

1.8.3 Мощность объединения множеств

Приведем еще одну задачу, которая решается с помощью принципа включения-исключения, — задачу нахождения мощности объединения пересекающихся множеств.

Задача 1.8.4 . *Пусть A_1, A_2, ..., A_k — конечные множества. Пусть $A = A_1 \cup A_2 \cup ... \cup A_k$. Требуется найти мощность A, если известны $|A_i|$, $i = \overline{1,k}$, и $|\bigcap_{i\in I} A_i|$, $I \subseteq \{1, 2, ..., k\}$.*

Пусть $S = \{1, 2, ..., k\}$ и $T \subseteq S$. Будем говорить, что $x \in A$ обладает свойством i, если $x \in A_i$. Обозначим

$f_=(T)$ — число элементов из A, обладающих всеми свойствами из T и не обладающих другими свойствами из S;

$f_\geq(T)$ — число элементов из A, обладающих всеми свойствами из T. Очевидно, $f_\geq(T) = |\bigcap_{i\in T} A_i|$, если T не пусто и $f_\geq(\varnothing) = |A|$

Тогда верна формула (19) и, следовательно, можно использовать формулу (21).

Утверждение 1.8.4 .

$$|A_1 \cup A_2 \cup ... \cup A_k| = \sum_{i=1}^{k} |A_i| - \sum_{1\leq i_1<i_2\leq k} |A_{i_1} \cap A_{i_2}| +$$
$$+ \sum_{1\leq i_1<i_2<i_3\leq k} |A_{i_1} \cap A_{i_2} \cap A_{i_3}| - \cdots + (-1)^{k+1}|A_1 \cap A_2 \cap ... \cap A_k|. \quad (22)$$

Доказательство. Поскольку каждый элемент из A обладает хотя бы одним свойством из S, то $f_=(\varnothing) = 0$. С другой стороны из формулы (21) получаем

$$0 = f_=(\varnothing) = \sum_{Y\subseteq S} (-1)^{|Y|} f_\geq(Y) = |A| + \sum_{\substack{Y\subseteq S,\\ |Y|\geq 1}} (-1)^{|Y|} |\bigcap_{i\in Y} A_i| =$$
$$= |A| - \sum_{i=1}^{k} |A_i| + \sum_{1\leq i_1<i_2\leq k} |A_{i_1} \cap A_{i_2}| -$$
$$- \sum_{1\leq i_1<i_2<i_3\leq k} |A_{i_1} \cap A_{i_2} \cap A_{i_3}| + \cdots + (-1)^k |A_1 \cap A_2 \cap ... \cap A_k|.$$

Перенесем все слагаемые кроме первого за знак равенства и получим искомое утверждение.
□

Часто, именно формулу (22) называют принципом включения-исключения. Тем не менее, как мы видели, с помощью принципа включения-исключения решаются и другие задачи.

1.8.4 Число целочисленных решений системы неравенств

В подразделе 1.7.1 мы показали, что число решений уравнения $x_1 + x_2 + \cdots + x_k = n$ в неотрицательных целых числах находится по формуле (16):

$$N_{=}(n,k) = \binom{n+k-1}{k-1},$$

а число решений неравенства $x_1 + x_2 + \cdots + x_k \le n$ в неотрицательных целых числах — по формуле (17):

$$N_{\le}(n,k) = \binom{n+k}{k}.$$

Здесь мы расширим множество систем, для которых можно будет найти число решений.

Для начала положим, что на значения переменных наложены ограничения снизу. Пусть $a_i \in \mathbb{N}_0$, $i = \overline{1,k}$, и задана система

$$\begin{gathered} x_1 + x_2 + \cdots + x_k = n, \\ a_i \le x_i, \quad i = \overline{1,k}. \end{gathered}$$

Чтобы найти число решений этой системы, достаточно сделать замену $y_i = x_i - a_i$. Тогда $y_i \ge 0$ и

$$\begin{gathered} (y_1 + a_1) + (y_2 + a_2) + \cdots + (y_k + a_k) = n, \\ y_1 + y_2 + \cdots + y_k = n - a_1 - a_2 - \cdots - a_k. \end{gathered}$$

Согласно формуле (16) число целочисленных решений данной системы

$$\binom{n+k-1-\sum_{i=1}^{k} a_i}{k-1}. \tag{23}$$

Аналогично, число решений системы

$$x_1 + x_2 + \cdots + x_k \le n,$$
$$a_i \le x_i, \quad i = \overline{1,k},$$

равно

$$\binom{n + k - \sum_{i=1}^{k} a_i}{k}. \tag{24}$$

Рассмотрим систему с ограничениями на переменные и сверху, и снизу. Пусть $a_i, b_i \in \mathbb{N}_0$, $a_i \le b_i$, $i = \overline{1,k}$. Требуется найти число целочисленных решений системы

$$x_1 + x_2 + \cdots + x_k = n, \tag{25}$$
$$a_i \le x_i \le b_i, \quad i = \overline{1,k}. \tag{26}$$

Пусть A — множество решений системы

$$x_1 + x_2 + \cdots + x_k = n, \tag{*}$$
$$a_i \le x_i, \quad i = \overline{1,k}:$$

$$A = \{(\alpha_1, \alpha_2, ..., \alpha_k) \mid \alpha_1 + \alpha_2 + \cdots \alpha_k = n,\ a_i \le \alpha_i,\ i = \overline{1,k}\}.$$

Согласно (23)

$$|A| = \binom{n + k - 1 - \sum_{i=1}^{k} a_i}{k-1}.$$

Обозначим A_s, $s = \overline{1,k}$, — множество решений системы (*), для которых не выполняется ограничение $x_s \le b_s$:

$$A_s = \{(\alpha_1, \alpha_2, ..., \alpha_k) \mid \alpha_1 + \alpha_2 + \cdots \alpha_k = n,$$
$$a_i \le \alpha_i,\ i = \overline{1,k},\ b_s + 1 \le x_s\}.$$

Таким образом, все решения системы (*), которые не удовлетворяют хотябы одной верхней границе из (26), лежат в множестве $A_1 \cup A_2 \cup ... \cup A_k$. Следовательно, число решений системы (25)-(26) равно

$$|A \setminus (A_1 \cup A_2 \cup ... \cup A_k)| = |A| - |A_1 \cup A_2 \cup ... \cup A_k|.$$

Использовав утверждение 1.8.4, получим

$$|A| - |A_1 \cup A_2 \cup ... \cup A_k| = |A| - \sum_{\substack{Y \subseteq \{1,...,k\}, \\ |Y| \geq 1}} (-1)^{|Y|-1} |\bigcap_{i \in Y} A_i|. \qquad (**)$$

Теперь, чтобы найти число решений системы (25)-(26), достаточно научиться находить мощность пересечения произвольного набора множеств A_j.

Пусть $Y \subseteq \{1, ..., k\}$. Тогда

$$\bigcap_{j \in Y} A_j = \{(\alpha_1, \alpha_2, ..., \alpha_k) \mid \alpha_1 + \alpha_2 + \cdots \alpha_k = n, \\ a_i \leq \alpha_i,\ i = \overline{1,k},\ b_s + 1 \leq x_s,\ s \in Y\}.$$

Другими словами, $\bigcap_{j \in Y} A_j$ есть множество решений системы

$$x_1 + x_2 + \cdots + x_k = n,$$
$$a_i \leq x_i,\ i \in \{1, ..., k\} \setminus Y, \quad b_s + 1 \leq x_s,\ s \in Y.$$

Тогда согласно (23)

$$|\bigcap_{i \in Y} A_i| = \binom{n + k - 1 - \sum_{i=1}^{k} a_i - \sum_{j \in Y} (b_j + 1 - a_j)}{k - 1}.$$

Подставив эти значения в формулу (**), получим число решений системы (25)-(26).

Рассуждения для системы

$$x_1 + x_2 + \cdots + x_k \leq n,$$
$$a_i \leq x_i \leq b_i, \quad i = \overline{1,k}.$$

аналогичны рассуждениям для системы (25)-(26) с точностью до замены величины (23) на величину (24).

Пример 1.8.1 . *Рассмотрим систему всего с двумя переменными:*

$$x_1 + x_2 \leq n, \quad a_1 \leq x_1 \leq b_1, \quad a_2 \leq x_2 \leq b_2, \qquad (\star)$$

и систему без верхних ограничений на переменные:

$$x_1 + x_2 \le n, \quad a_1 \le x_1, \quad a_2 \le x_2. \qquad (\star\star)$$

Сравним графическое изображение области, в которой лежат решения системы $(\star\star)$ *(рис. 12), и области, где находятся решения системы* $(\star)$ *(рис. 13). Как можно видеть, чтобы из области решений для*

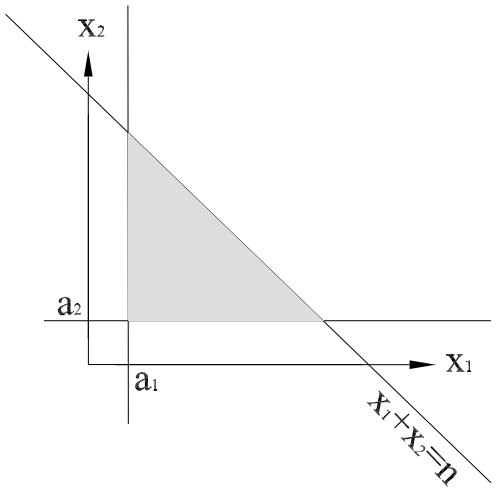

Рисунок 12: Множество решений неравенства $x_1 + x_2 \le n$ с нижними ограничениями на значения переменных.

системы $(\star\star)$ *получить область для системы* $(\star)$*, нужно удалить из нее объединение областей решений систем*

$$x_1 + x_2 \le n, \quad b_1 + 1 \le x_1, \quad a_2 \le x_2,$$

и

$$x_1 + x_2 \le n, \quad a_1 \le x_1, \quad b_2 + 1 \le x_2.$$

Пример 1.8.2 . *Пусть дана система неравенств*

$$x_1 + x_2 + x_3 \le 30,$$

$$0 \le x_1 \le 10, \qquad 5 \le x_2 \le 12, \qquad 1 \le x_3 \le 8.$$

Обозначим за A *множество целочисленных решений исходной системы без учета верхних границ на значение переменных:*

$$\begin{gathered} x_1 + x_2 + x_3 \le 30, \\ 0 \le x_1, \qquad 5 \le x_2, \qquad 1 \le x_3. \end{gathered} \qquad (*)$$

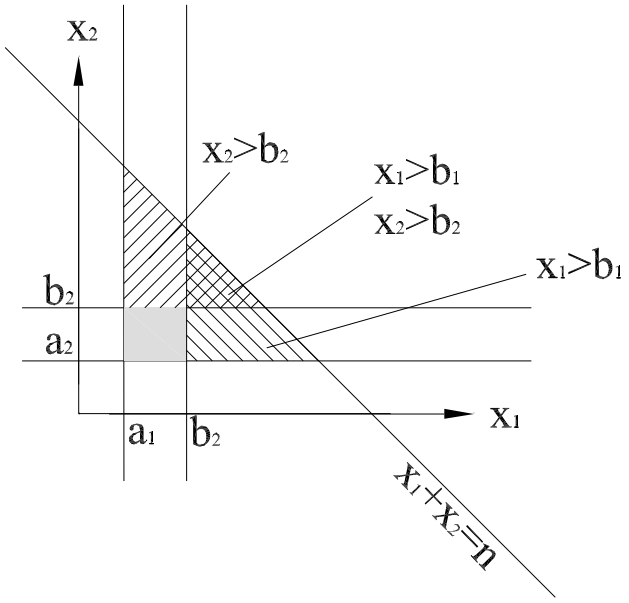

Рисунок 13: Множество решений неравенства $x_1+x_2 \leq n$ с ограничениями на значения переменных сверху и снизу.

Число целочисленных решений системы $(*)$ *вычисляем по известной формуле:*

$$|A| = \binom{n+k-\sum s_i}{k} = \binom{30+3-6}{3} = \binom{27}{3} = \frac{27!}{3! \cdot 24!} =$$
$$= \frac{27 \cdot 26 \cdot 25}{6} = 2925.$$

Обозначим за A_i *множество целочисленных решений исходной системы* $(*)$*, которые не удовлетворяют верхней границе исходной системы на значение* x_i*,* $i = \overline{1,3}$*.*

$$A_1: \quad \begin{array}{c} x_1 + x_2 + x_3 \leq 30, \\ 11 \leq x_1, \qquad 5 \leq x_2, \qquad 1 \leq x_3. \end{array}$$

$$A_2: \quad \begin{array}{c} x_1 + x_2 + x_3 \leq 30, \\ 0 \leq x_1, \qquad 13 \leq x_2, \qquad 1 \leq x_3. \end{array}$$

$$A_3: \quad \begin{array}{c} x_1 + x_2 + x_3 \leq 30, \\ 0 \leq x_1, \qquad 5 \leq x_2, \qquad 9 \leq x_3. \end{array}$$

Посчитаем мощности этих множеств.

$$|A_1| = \binom{30+3-17}{3} = \binom{16}{3} = \frac{16!}{3! \cdot 13!} = \frac{16 \cdot 15 \cdot 14}{6} = 560.$$

$$|A_2| = \binom{30+3-14}{3} = \binom{19}{3} = \frac{19!}{3! \cdot 16!} = \frac{19 \cdot 18 \cdot 17}{6} = 969.$$

$$|A_3| = \binom{30+3-14}{3} = 969.$$

Также рассмотрим все возможные пересечения этих множеств и посчитаем их мощности.

$$A_1 \cap A_2 : \qquad \begin{array}{c} x_1 + x_2 + x_3 \leq 30, \\ 11 \leq x_1, \qquad 13 \leq x_2, \qquad 1 \leq x_3. \end{array}$$

$$|A_1 \cap A_2| = \binom{30+3-25}{3} = \binom{8}{3} = \frac{8!}{3! \cdot 5!} = \frac{8 \cdot 7 \cdot 6}{6} = 56.$$

$$A_1 \cap A_3 : \qquad \begin{array}{c} x_1 + x_2 + x_3 \leq 30, \\ 11 \leq x_1, \qquad 5 \leq x_2, \qquad 9 \leq x_3. \end{array}$$

$$|A_1 \cap A_3| = \binom{30+3-25}{3} = 56.$$

$$A_2 \cap A_3 : \qquad \begin{array}{c} x_1 + x_2 + x_3 \leq 30, \\ 0 \leq x_1, \qquad 13 \leq x_2, \qquad 9 \leq x_3. \end{array}$$

$$|A_2 \cap A_3| = \binom{30+3-22}{3} = \binom{11}{3} = \frac{11!}{3! \cdot 8!} = \frac{11 \cdot 10 \cdot 9}{6} = 165.$$

$$A_1 \cap A_2 \cap A_3 : \qquad \begin{array}{c} x_1 + x_2 + x_3 \leq 30, \\ 11 \leq x_1, \qquad 13 \leq x_2, \qquad 9 \leq x_3. \end{array}$$

$$|A_1 \cap A_2 \cap A_3| = \binom{30+3-33}{3} = 0.$$

Теперь количество целочисленных решений исходной системы можно получить по формуле:

$$\begin{aligned} &|A| - |A_1 \cup A_2 \cup A_3| = |A| - (|A_1| + |A_2| + |A_3| - |A_1 \cap A_2| - \\ &- |A_1 \cap A_3| - |A_2 \cap A_3| + |A_1 \cap A_2 \cap A_3|) = 2925 - 560 - 969 - \\ &- 969 + 56 + 56 + 165 = 704. \end{aligned}$$

Рассмотрим теперь систему с ограничением на сумму переменных снизу:

$$\begin{gathered} x_1 + x_2 + \cdots + x_k \geq n, \\ a_i \leq x_i \leq b_i, \quad i = \overline{1,k}. \end{gathered}$$

Сделаем замену переменных $y_i = b_i - x_i$. Тогда ограничения на значения переменных трансформируются следующим образом:

$$a_i \leq b_i - y_i \leq b_i \quad \Rightarrow \quad 0 \leq y_i \leq b_i - a_i.$$

Подставим замену в неравенство для суммы переменных:

$$(b_1 - y_1) + (b_2 - y_2) + \cdots + (b_k - y_k) \geq n,$$

$$y_1 + y_2 + \cdots + y_k \leq b_1 + b_2 + \cdots + b_k - n.$$

Получили систему

$$\begin{gathered} y_1 + y_2 + \cdots + y_k \leq \sum_{i=1}^{k} b_i - n, \\ 0 \leq y_i \leq b_i - a_i, \quad i = \overline{1,k}, \end{gathered}$$

которая может быть решена указанным выше для системы (25)-(26) способом.

2 Математическая логика: Булева аллгебра

Математическая логика окончательно оформилась как самостоятельная математическая дисциплина к 30-м годам XX века. Основная причина ее появления — математические парадоксы, например парадокс Рассела. Сложно проводить математические рассуждения, не будучи уверенным в их непротиворечивости, а также в существовании объектов, которые они определяют. Возникла идея исследовать язык логики и математики, подойти к доказательству на основе понятий аксиом и правил вывода.

2.1 Булева алгебра. Функции алгебры логики.

2.1.1 Булевы функции

Будем рассматривать булевы функции (функции алгебры логики) — функции, аргументы и значения которых принимают значения истина и ложь. Истину и ложь будем обозначать соответственно 1 и 0. Положим $E_2 = \{0, 1\}$. Таким образом, функция n аргументов f есть

$$f \ : \ \underbrace{E_2 \times E_2 \times ... \times E_2}_{n} \ \rightarrow \ E_2.$$

Аргументы этих функций будем называть логическими переменными и обозначать буквами x, y и z, возможно с индексами. Множество всех булевых функций (функций алгебры логики) будем обозначать P_2.

Пример 2.1.1 . *Табличное задание функции f:*

	x	y	z	$f(x,y,z)$
2^3	0	0	0	1
	0	0	1	0
	0	1	0	1
	0	1	1	0
	1	0	0	0
	1	0	1	1
	1	1	0	0
	1	1	1	1

Всего существует 2^3 различных наборов значений трех переменных. Если их нумеровать от 0 до $2^3 - 1$, то набор с номером i оказывается представлением числа i в двоичной системе счисления. Всего различных функций от 3-х аргументов — 2^{2^3}

В общем случае число строк в таблице для функции от n аргументов равно 2^n. Число различных булевых функций от n аргуменов — 2^{2^n}.

Пример 2.1.2 . *Рассмотрим, как зависит функция f из примера 2.1.1 от переменной y. Пусть $\gamma' = (\alpha_1, 0, \alpha_3)$ и $\gamma'' = (\alpha_1, 1, \alpha_3)$ — два набора с произвольными значениями α_1 и α_2. Тогда по таблице выше можно убедиться, что $f(\gamma') = f(\gamma'')$: $f(0,0,0) = f(0,1,0)$, $f(0,0,1) = f(0,1,1)$ и так далее. В таком случае, можно сказать, что функция f не зависит существенно от переменной y, или, что y — несущественная переменная функции f.*

Замечание 2.1.1 . *Среди 2^{2^n} различных функций от n переменных далеко не все зависят от аргументов существенно. В это число войдут и все функции от $n-1$, $n-2$, и т. д. аргументов.*

Определение 2.1.1 . *Будем говорить, что функция $f(x_1, x_2, ..., x_n)$ не зависит существенно от x_n (x_n — фиктивная (несущественная) переменная функции $f(x_1, x_2, ..., x_n)$), если для любых значений $\alpha_1, \alpha_2, ..., \alpha_{n-1} \in E_2$ выполняется равенство $f(\alpha_1, \alpha_2, ..., \alpha_{n-1}, 0) = f(\alpha_1, \alpha_2, ..., \alpha_{n-1}, 1)$.*

Переменные функции f, которые не являются фиктивными, называют существенными переменными и говорят, что функция f существенно от них зависит.

Пример 2.1.3 . *Продолжим предыдущий пример. Удалим из таблицы для функции f по одной из каждой пары строк, соответствующих разным значениям переменной y при одинаковых x и z, а также удалим столбец со значениями переменной y. Получим таблицу для новой*

функции $f'(x,z)$*:*

x	z	f'
0	0	1
0	1	0
1	0	0
1	1	1

Функции f *и* f' *по определению различны: у них разная область определения. С другой стороны, если учитывать только существенные переменные, эти две функции полностью совпадают. В дальнейшем будем считать такие функции равными.*

Определение 2.1.2 . *Будем говорить, что две функции* $f(x_1, x_2, ..., x_k)$ *и* $g(x_1, x_2, ..., x_l)$ *равны, если после удаления всех несущественных переменных получаются функции с одинаковыми таблицами. В таком случае будем писать* $f = g$*.*

Замечание 2.1.2 . *При использовании табличного задания функции достаточно указывать тольно набор ее значений, предполагая, что порядок следования наборов аргументов всегда одинаков. Например, функция из примера 2.1.1 может быть определена только записью* $f = (1, 0, 1, 0, 0, 1, 0, 1)$*.*

2.1.2 Формулы

Использование табличного задания функций часто неудобно. Во-первых, число строк в таблице экспоненциально зависит от числа переменных функции. Уже при 10 переменных, число строк таблицы должно быть 1024. Во-вторых, что более важно, при анализе свойств функции нельзя выполнять какие-либо алгебраические преобразования для облегчения этого процесса. В связи с этим будет удобно ввести понятие формулы, как еще одного представления функции.

Выберем некоторую систему функций из P_2: $\mathcal{P} = \{f_1, f_2, ..., f_k\} \subseteq P_2$, $k \geq 1$. Назовем функции из системы $\mathcal{P}$ элементарными функциями. Тогда формула над $\{f_1, f_2, ..., f_k\}$ определяется рекурсивно:

Определение 2.1.3 . *1. Если $f \in \mathcal{P}$ — функция от n аргументов, то $f(x_1, ..., x_n)$ — формула над $\mathcal{P}$.*

2. Если $f \in \mathcal{P}$ — функция от n аргументов и $\mathcal{U}_1$, $\mathcal{U}_2$, ..., $\mathcal{U}_n$ — формулы или логические переменные, то $f(\mathcal{U}_1, ..., \mathcal{U}_n)$ — формула над $\mathcal{P}$.

Замечание 2.1.3 . *Мы определили формулу над $\mathcal{P}$. Формула всегда мыслится в связи с каким-то указанным множеством элементарных функций.*

Каждой формуле можно однозначно сопоставить функцию:
1. Если $\mathcal{U} = f(x_1, ..., x_n) \in \mathcal{P}$, то формуле $\mathcal{U}$ сопоставляется функция $f_{\mathcal{U}} = f(x_1, ..., x_n)$.
2. Пусть $\mathcal{U} = f(\mathcal{U}_1, ..., \mathcal{U}_n)$, где $f(x_1, ..., x_n) \in \mathcal{P}$ и $\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n$ — формулы или логические переменные. Тогда $f_{\mathcal{U}} = f(f_{\mathcal{U}_1}, ..., f_{\mathcal{U}_n})$, где $f_{\mathcal{U}_i}$ — функция, сопоставленная формуле $\mathcal{U}_i$, если $\mathcal{U}_i$ — формула, и $f_{\mathcal{U}_i} = x_i$, если $\mathcal{U}_i = x_i$ — логическая переменная.

Говорят, что формула $\mathcal{U}$ реализует функцию $f_{\mathcal{U}}(x_1, ..., x_n)$.

Пример 2.1.4 . *Пусть дано множество элементарных функций $\mathcal{P} = \{f(x, y), g(x, y), h(x, y)\}$, где функции $f(x, y)$, $g(x, y)$, $h(x, y)$ заданы следующей таблицей:*

x	y	f	g	h
0	0	1	1	0
0	1	0	1	1
1	0	0	0	1
1	1	1	1	1

.

Пусть дана формула $\mathcal{U} = f(g(x, y), h(y, z))$. Найдем функцию $f_{\mathcal{U}}$, реализованную формулой $\mathcal{U}$. Для этого сначала найдем столбцы значений для функций, заданных подформулами $g(x, y)$ и $h(y, z)$, а затем, уже получив нужные значения, подставим их в качестве

аргументов для функции f:

x	y	z	$g(x,y)$	$h(y,z)$	$f(g(x,y),h(y,z))$
0	0	0	1	0	$f(1,0)=0$
0	0	1	1	1	$f(1,1)=1$
0	1	0	1	1	$f(1,1)=1$
0	1	1	1	1	$f(1,1)=1$
1	0	0	0	0	$f(0,0)=1$
1	0	1	0	1	$f(0,1)=0$
1	1	0	1	1	$f(1,1)=1$
1	1	1	1	1	$f(1,1)=1$

.

Пример 2.1.5 . *Для понимания порядка вычислений для получения функции, реализуемой данной формулой, удобно использовать древесную диаграмму.*

Пусть определено множество элементарных функций $\mathcal{P} = \{f(x,y,z), g(x,y), h(x)\}$ *и дана формула* $\mathcal{U} = f(x, g(h(y),0), f(z,x,1))$. *Строение формулы* $\mathcal{U}$ *изображено на рисунке 14. Внутренние узлы представленного дерева помечены символами функций из* $\mathcal{P}$. *концевые узлы (листья) помечены символами переменных или констант. Для вычисления реализуемой функции на произвольном наборе аргументов, необходимо пометить листья соответствующими параметрами и, поднимаясь наверх, последовательно заполнять каждый уровень дерева значениями для промежуточных узлов.*

Определение 2.1.4 . *Будем говорить, что формулы* $\mathcal{U}$ *и* $\mathcal{B}$ *эквивалентны, и писать* $\mathcal{U} = \mathcal{B}$, *если* $f_{\mathcal{U}} = f_{\mathcal{B}}$ *с точностью до несущественных переменных.*

Рассмотрим основные функции, используемые в качестве элементарных функций в алгебре логики.

Всего существует четыре различные функции от одной переменной: тождественный ноль — $f(x) = 0$; тождественная единица — $f(x) = 1$; тождественная функция или тождественный x — $f(x) = x$; отрицание x или "не x" — $f(x) = \neg x$, так же обозначается $\overline{x}$.

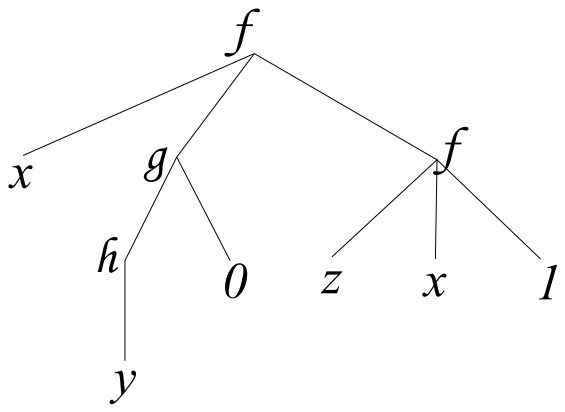

Рисунок 14: Строение формулы $f(x, g(h(y), 0), f(z, x, 1))$.

x	0	1	x	$\neg x$
0	0	1	0	1
1	0	1	1	0

Из них тождественный ноль и тождественная единица не зависят существенно от x. То есть фактически это две функции без аргументов — константы: $f = 0$ и $f = 1$.

Замечание 2.1.4 . *Может показаться, что излишне говорить о* 0, 1 *и* x *здесь, как о функциях. Необходимо понимать, что в различном контексте эти символы могут пониматься различным образом. С одной стороны мы имеем константы* 0 *и* 1 *и некоторую переменную* x*. Но в контексте, предполагающем использование функций, эти символы будут пониматься как отображения, указанные в таблице выше.*

Численное значение этих величин не меняется, но меняется смысл, который мы вкладываем в запись.

Рассмотрим основные булевы функции от двух переменных.

x	y	$x \vee y$	$x \wedge y$	$x \oplus y$	$x \supset y$	$x \equiv y$	$x \mid y$	$x \downarrow y$
0	0	0	0	0	1	1	1	1
0	1	1	0	1	1	0	1	0
1	0	1	0	1	0	0	1	0
1	1	1	1	0	1	1	0	0

$f(x,y) = x \vee y$ — дизъюнкция, логическое "или".

$f(x,y) = x \wedge y$ — конъюнкция, логическое "и", логическое умножение. Также можно использовать обозначения $x\&y$ или xy.

$f(x,y) = x \oplus y$ — сложение по модулю два, логическое исключающее "или". Также можно использовать обозначение $x + y$.

$f(x,y) = x \supset y$ — импликация, "если, то". Также можно использовать обозначение $x \to y$.

$f(x,y) = x \equiv y$ — эквивалентность. Также можно использовать обозначение $x \sim y$.

$f(x,y) = x \mid y$ — штрих Шеффера.

$f(x,y) = x \downarrow y$ — стрелка Пирса.

Всего, как мы помним, существует 16 различных функций от двух переменных. Мы выбрали 7, существенно зависящих от обоих переменных и имеющих наибольшее значение. Добавив к ним функции от одной переменной и константы (функции от 0 переменных) получим систему функций, подмножества которой мы в основном будем испльзовать в качестве множеств элементарных функций для построения формул

$$P = \{0, 1, x, \overline{x}, x \vee y, xy, x \oplus y, x \supset y, x \equiv y, x|y, x \downarrow y\}.$$

Функции $\{\overline{x}, x \vee y, xy, x \oplus y, x \supset y, x \equiv y, x|y, x \downarrow y\}$ будем также называть операциями.

Пример 2.1.6 . *Рассмотрим пример формулы над P:*

$$\mathcal{U} = (((xy) \vee (xz)) \vee (yz)).$$

В этой записи слишком много скобок.

Чтобы облегчить чтение и запись формул можно установить приоритеты выполнения операций. Будем считать, что наивысший приоритет имеет операция отрицания. Из функций от двух переменных наивысший приоритет будет иметь коньюнкция — $\wedge$. Все остальные операции имеют одинаковый приоритет.

Кроме того, можно убедиться, что операции $\wedge$, $\vee$, $\oplus$, $\equiv$ являются ассоциативными. Таким образом, вместо $x \circ (y \circ z)$ или $(x \circ y) \circ z$ можно

писать $x \circ y \circ z$, если $\circ \in \{\wedge, \vee, \oplus, \equiv\}$. Операции $\supset$, $|$, $\downarrow$ не являются ассоциативными.

Пример 2.1.7 . *С учетом указанных договоренностей, формула из примера 2.1.6 примет вид:*

$$\mathcal{U} = xy \vee xz \vee yz.$$

Эта формула задает известную функцию от трех переменных — функцию голосования. Функция голосования носит такое название, поскольку моделирует голосование трех человек: функция принимает значение 1 тогда и только тогда, когда по крайней мере двое проголосовали положительно.

2.1.3 Основные тождества

Как мы помним из определения 2.1.4 формулы называют эквивалентными, когда соответствующие им функции совпадают. Рассмотрим, как можно получить из формулы эквивалентные ей.

Утверждение 2.1.1 (о замене подформул на эквивалентные). *Пусть f — некоторая элементарная функция. Если формулы $\mathcal{U}_1$, $\mathcal{U}_2$, ..., $\mathcal{U}_n$ эквивалентны соответственно формулам $\mathcal{B}_1$, $\mathcal{B}_2$, ..., $\mathcal{B}_n$, то формула $\mathcal{U} = f(\mathcal{U}_1, \mathcal{U}_2, ..., \mathcal{U}_n)$ эквивалентна формуле $\mathcal{B} = f(\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_n)$.*

Доказательство. Действительно, из определения эквивалентности следует, что $f_{\mathcal{U}_i} = f_{\mathcal{B}_i}$, $i = \overline{1, n}$. При этом, формула $\mathcal{U}$ реализует функцию $f_{\mathcal{U}} = f(f_{\mathcal{U}_1}, f_{\mathcal{U}_2}, ..., f_{\mathcal{U}_n})$, а формула $\mathcal{B}$ реализует функцию $f_{\mathcal{B}} = f(f_{\mathcal{B}_1}, f_{\mathcal{B}_2}, ..., f_{\mathcal{B}_n})$. Следовательно $f_{\mathcal{U}} = f_{\mathcal{B}}$.
□

Это позволяет нам, записать ряд тождеств, которые мы сможем использовать для преобразования формул, заменяя подформулы на эквивалентные.

1. Коммутативность: $x \circ y = y \circ x$, если $\circ \in \{\wedge, \vee, \oplus, \equiv, |, \downarrow\}$.

2. Ассоциативность: $x \circ (y \circ z) = (x \circ y) \circ z$, если $\circ \in \{\wedge, \vee, \oplus, \equiv\}$. Мы уже указывали на это свойство раньше.

3. Правила де Моргана: $\overline{x \wedge y} = \overline{x} \vee \overline{y}$, $\overline{x \vee y} = \overline{x} \wedge \overline{y}$.

4. Правила поглощения: $x \vee xy = x, \quad x(x \vee y) = x.$

5. Дистрибутивность:

$x(y \vee z) = xy \vee xz$ — дистрибутивность $\wedge$ относительно $\vee$,

$x \vee yz = (x \vee y)(x \vee z)$ — дистрибутивность $\vee$ относительно $\wedge$,

$x(y \oplus z) = xy \oplus xz$ — дистрибутивность $\wedge$ относительно $\oplus$.

6. Формулы расщепления:

$x = xy \vee x\overline{y}$,

$x = (x \vee y)(x \vee \overline{y})$.

7. $0 = x\overline{x} \ = \ x \wedge 0 \ = \ x \oplus x$,

$1 \ = \ x \vee \overline{x} \ = \ x \vee 1 \ = \ x \equiv x$,

$x \ = \ \neg\neg x \ = \ x \vee x \ = \ xx \ = \ x \wedge 1 \ = \ x \vee 0.$

8. $\overline{x} \ = \ x \oplus 1$,

$x \equiv y \ = \ (x \oplus y) \oplus 1$,

$x \oplus y = (x \vee y)\overline{xy} = x\overline{y} \vee \overline{x}y$,

$x \supset y \ = \ \overline{x} \vee y \ = \ xy \oplus x \oplus 1$,

$x \downarrow y \ = \ \overline{x} \wedge \overline{y} = \overline{x \vee y}$,

$x \mid y \ = \ \overline{x} \vee \overline{y} = \overline{x \wedge y}$.

Используя эти тождества можно выполнять преобразования формул, получая им эквивалентные.

Пример 2.1.8 . *Рассмотрим формулу* $\mathcal{U} = (y \supset x) \vee (x \oplus 1) \vee y$. *Проведем преобразования, используя известные тождества.*

$$(y \supset x) \vee (x \oplus 1) \vee y = (\overline{y} \vee x) \vee (\overline{x}) \vee y = \overline{y} \vee x \vee \overline{x} \vee y = 1 \vee 1 = 1.$$

Таким образом, формула $\mathcal{U}$ *задает тождественно истинную функцию.*

Очевидно, что формула $\mathcal{B} = \overline{\mathcal{U}} = \overline{(y \supset x) \vee (x \oplus 1) \vee y} = \overline{1} = 0$, *то есть* $\mathcal{B}$ *задает тождественно ложную функцию.*

Определение 2.1.5 . *Формула, задающая тождественно истинную функцию, называется тавтологией.*

Определение 2.1.6 . *Формула, задающая тождественно ложную функцию, называется противоречием.*

Определение 2.1.7 . *Формула называется выполнимой, если для нее существует набор аргументов, на котором она принимает значение 1.*

2.1.4 Разложение функции по переменным

Определение 2.1.8 . *Введем следующее обозначение:*

$$x^\sigma = \begin{cases} x, & \sigma = 1, \\ \overline{x}, & \sigma = 0. \end{cases}$$

Также будем говорить x в степени σ, имея в виду запись x^σ определенную выше. Переменную с или без отрицания будем называть литералом.

Замечание 2.1.5 . *Как можно видеть из определения 2.1.8*

$$x^\sigma = 1 \qquad \Longleftrightarrow \qquad x = \sigma.$$

Утверждение 2.1.2 (разложение функции по параметрам). *Пусть $f(x_1, ..., x_n) \in P_2$ и $1 \leq m \leq n$. Тогда*

$$f(x_1, ..., x_m, x_{m+1}, ..., x_n) = \bigvee_{\substack{(\sigma_1,...,\sigma_m): \\ \sigma_i \in E_2,\ i=\overline{1,m}}} x_1^{\sigma_1} x_2^{\sigma_2} \cdots x_m^{\sigma_m} f(\sigma_1, ..., \sigma_m, x_{m+1}, ..., x_n). \quad (27)$$

Доказательство. Рассмотрим произвольный набор аргументов $(\alpha_1, ..., \alpha_n)$: $\alpha_i \in E_2$, $i = \overline{1, n}$. Подставим этот набор в правую часть уравнения (27):

$$\bigvee_{\substack{(\sigma_1,...,\sigma_m): \\ \sigma_i \in E_2,\ i=\overline{1,m}}} \alpha_1^{\sigma_1} \alpha_2^{\sigma_2} \cdots \alpha_m^{\sigma_m} f(\sigma_1, ..., \sigma_m, \alpha_{m+1}, ..., \alpha_n) =$$

Так как $\alpha^\sigma = 1$ тогда и только тогда, когда $\alpha = \sigma$, то из всех членов предыдущего выражения останется только один: когда все σ_i совпадают с α_i.

$$= 1 \cdot 1 \cdots 1 \cdot f(\alpha_1, ..., \alpha_m, \alpha_{m+1}, ..., \alpha_n) = f(\alpha_1, ..., \alpha_m, \alpha_{m+1}, ..., \alpha_n).$$

Таким образом, мы показали, что для произвольного набора значений аргументов функции f левая и правая части формулы (27) совпадают, что и требовалось доказать.
□

Пример 2.1.9 . *Пусть* $f(x,y) = \overline{x}\ |(x \supset \overline{y})$. *Разложим функцию* f *по переменной* x.

$$f(x,y) = \overline{x}\ |(x \supset \overline{y}) = xf(1,y) \vee \overline{x}f(0,y) =$$

Подставим формулу на место функции f.

$$= x(0\ |(1 \supset \overline{y})) \vee \overline{x}(1\ |(0 \supset \overline{y})) = x$$

Последнее равенство следует из того, что $(0 \mid \alpha) = 1$ *для любого* α, *в то время как* $(0 \supset \beta) = 1$ *для любого* β *и, следовательно,* $(1\ |(0 \supset \overline{y})) = (1 \mid 1) = 0$.

Замечание 2.1.6 . *Согласно утверждению 2.1.2, можно строить разложение по любому подмножеству множества переменных функции.*

Пример 2.1.10 . *Пусть* $f(x,y,z) = \overline{(x \supset y) \oplus z}$. *Выполним разложение* f *по переменным* x *и* z.

$$f(x,y,z) = \bigvee_{\substack{(\sigma_1,\sigma_3):\\ \sigma_1 \in E_2,\ \sigma_3 \in E_2}} x^{\sigma_1} z^{\sigma_3} f(\sigma_1, y, \sigma_3) =$$

$$= x^0 z^0 f(0,y,0) \vee x^0 z^1 f(0,y,1) \vee x^1 z^0 f(1,y,0) \vee x^1 z^1 f(1,y,1) =$$

$$= \overline{x}\ \overline{z}\ \overline{(0 \supset y) \oplus 0} \vee \overline{x}\ z\ \overline{(0 \supset y) \oplus 1} \vee x\ \overline{z}\ \overline{(1 \supset y) \oplus 0} \vee x\ z\ \overline{(1 \supset y) \oplus 1} =$$

Таким получилось разложение по переменным. Теперь можно провести преобразования, чтобы упростить результат.

$$= \overline{x}\ \overline{z}\ \overline{1 \oplus 0} \vee \overline{x}\ z\overline{1 \oplus 1} \vee x\ \overline{z}\overline{(1 \supset y) \oplus 0} \vee$$

$$\vee x\ z\overline{(1 \supset y) \oplus 1} = \overline{x}\ z \vee x\ \overline{z}\ \overline{y} \vee x\ z\ y.$$

Утверждение 2.1.3 . *Пусть* $f(x_1, ..., x_n) \in P_2$ *и* $f \neq 0$. *Тогда*

$$f(x_1, ..., x_n) = \bigvee_{\substack{(\sigma_1,...,\sigma_n) \in E_2^n \\ f(\sigma_1,...,\sigma_n)=1}} x_1^{\sigma_1} \cdots x_n^{\sigma_n}.$$

Замечание 2.1.7 . *Здесь и далее запись "$f \neq 0$" понимается в смысле эквивалентности формул. То есть запись "$f \neq 0$" читается "f не является тождественно ложной функцией", а запись "$f \neq 1$" — "f не является тождественно истинной функцией". Аналогично записи "$f = 1$" и "$f = 0$" понимают как "f — тождественно истинна" и "f — тождественно ложна".*

Доказательство. Разложим функцию f по всем ее переменным согласно утверждению 2.1.2.

$$f(x_1, ..., x_n) = \bigvee_{(\sigma_1,...,\sigma_n)\in E_2^n} x_1^{\sigma_1} \cdots x_n^{\sigma_n} \cdot f(\sigma_1, ..., \sigma_n) =$$
$$= \bigvee_{\substack{(\sigma_1,...,\sigma_n)\in E_2^n \\ f(\sigma_1,...,\sigma_n)=1}} x_1^{\sigma_1} \cdots x_n^{\sigma_n}.$$

Поскольку $f \neq 0$, в этом выражении присутствует хотя бы один член. □

2.1.5 Дизъюнктивная и конъюнктивная нормальные формы

Определение 2.1.9 . *Формула вида $x_{i_1}^{\sigma_{i_1}} x_{i_2}^{\sigma_{i_2}} \cdots x_{i_k}^{\sigma_{i_k}}$, где x_{i_j} — логическая переменная, σ_{i_j} — логическая константа, $i_1 < i_2 < ... < i_k$, называется конъюнктом.*

Определение 2.1.10 . *Если $f(x_1, ..., x_n)$ представлена в виде*

$$f(x_1, ..., x_n) = K_1 \vee K_2 \vee ... \vee K_s,$$

где K_1, K_2,..., K_s — различные конъюнкты, то говорят, что f представлена в дизъюнктивной нормальной форме (ДНФ).

Если в каждый K_i входят все переменные $x_1, ..., x_n$, то говорят, что f представлена в совершенной дизъюнктивной нормальной форме (СДНФ).

Так же используют обозначения д.н.ф. и с.д.н.ф.

Утверждение 2.1.4 . *Пусть $f(x_1, ..., x_n) \in P_2$. Если $f \neq 0$, то она представима в виде СДНФ, причем единственным образом (с точностью до перестановки конъюнктов).*

Доказательство. Во-первых, отметим, что разложение функции f по всем переменным, построенное согласно утверждению 2.1.3, будет представлять собой СДНФ. Существование доказано.

Докажем единственность СДНФ. Пусть

$$f(x_1, ..., x_n) = \bigvee_{i=1}^{s} K_i(x_1, ..., x_n) = \bigvee_{j=1}^{s'} K'_j(x_1, ..., x_n),$$

где K_i и K'_j — конъюнкты, $i = \overline{1, s}$, $j' = \overline{1, s'}$. Причем, не умаляя общности, $K_1 \notin \{K'_1, ..., K'_{s'}\}$, поскольку представления в виде СДНФ должны быть различны.

Пусть $K_1(x_1, ..., x_n) = x_1^{\sigma_1} \cdots x_n^{\sigma_n}$.

$$f(\sigma_1, ..., \sigma_n) = \bigvee_{i=1}^{s} K_i(\sigma_1, ..., \sigma_n) = \sigma_1^{\sigma_1} \cdots \sigma_n^{\sigma_n} \vee \bigvee_{i=2}^{s} K_i(\sigma_1, ..., \sigma_n) =$$

$$= 1 \vee \bigvee_{i=2}^{s} K_i(\sigma_1, ..., \sigma_n) = 1,$$

так как $\sigma^\sigma = 1$, для любого $\sigma \in E_2$. С другой стороны,

$$f(\sigma_1, ..., \sigma_n) = \bigvee_{j=1}^{s'} K'_j(\sigma_1, ..., \sigma_n).$$

Нетрудно заметить, что для любого $K'_j = x_1^{\sigma'_1} \cdots x_n^{\sigma'_n} \in \{K'_1, ..., K'_{s'}\}$ выполняется

$$K'_j(\sigma_1, ..., \sigma_n) = \sigma_1^{\sigma'_1} \cdots \sigma_n^{\sigma'_n} = 0,$$

поскольку $\exists m = \overline{1, n}$, $\sigma_m \neq \sigma'_m$ Таким образом $f(\sigma_1, ..., \sigma_n) = \bigvee_{j=1}^{s'} 0 = 0$. Противоречие доказывает, что двух различных представлений функции в СДНФ существовать не может.
□

Определение 2.1.11 . *Формула вида $x_{i_1}^{\sigma_{i_1}} \vee x_{i_2}^{\sigma_{i_2}} \vee \cdots \vee x_{i_k}^{\sigma_{i_k}}$, где x_{i_j} — логическая переменная, σ_{i_j} — логическая константа, $i_1 < i_2 < ... < i_k$, называется дизъюнктом.*

Определение 2.1.12 . *Если $f(x_1, ..., x_n)$ представлена в виде*

$$f(x_1, ..., x_n) = (D_1) \wedge (D_2) \wedge ... \wedge (D_s),$$

где D_1, D_2,..., D_s — различные дизъюнкты, то говорят, что f представлена в конъюнктивной нормальной форме (КНФ).

Если в каждый D_i входят все переменные $x_1, ..., x_n$, то говорят, что f представлена в совершенной конъюнктивной нормальной форме (СКНФ).

Так же используют обозначения к.н.ф. и с.к.н.ф.

Утверждение 2.1.5 . *Пусть $f(x_1, ..., x_n) \in P_2$. Если $f \neq 1$, то она представима в виде СКНФ, причем единственным образом (с точностью до перестановки дизъюнктов).*

Доказательство. Поскольку $f \neq 1$, то $\overline{f} \neq 0$. Тогда по утверждению 2.1.4 у функции $\overline{f}$ существует СДНФ, которая по утверждению 2.1.3 имеет вид:

$$\overline{f}(x_1, ..., x_n) = \bigvee_{i=1}^{s} K_i(x_1, ..., x_n) = \bigvee_{\substack{(\sigma_1,...,\sigma_n) \\ \overline{f}(\sigma_1,...,\sigma_n)=1}} x_1^{\sigma_1} \cdots x_n^{\sigma_n}.$$

Следовательно,

$$f(x_1, ..., x_n) = \neg\left(\bigvee_{\substack{(\sigma_1,...,\sigma_n) \\ f(\sigma_1,...,\sigma_n)=0}} x_1^{\sigma_1} \cdots x_n^{\sigma_n} \right) =$$

Применим правила де Моргана.

$$= \bigwedge_{\substack{(\sigma_1,...,\sigma_n) \\ f(\sigma_1,...,\sigma_n)=0}} \neg\left(x_1^{\sigma_1} \cdots x_n^{\sigma_n}\right) = \bigwedge_{\substack{(\sigma_1,...,\sigma_n) \\ f(\sigma_1,...,\sigma_n)=0}} (\overline{x_1^{\sigma_1}} \vee \cdots \vee \overline{x_n^{\sigma_n}}) =$$

$$= \bigwedge_{\substack{(\sigma_1,...,\sigma_n) \\ f(\sigma_1,...,\sigma_n)=0}} (x_1^{\overline{\sigma_1}} \vee \cdots \vee x_n^{\overline{\sigma_n}}).$$

Теперь покажем единственность СКНФ. Действительно, пусть у некоторой функции $f \neq 1$ существовало две различных СКНФ:

$$f(x_1, ..., x_n) = \bigwedge_{i=1}^{s}(D_i(x_1, ..., x_n)) = \bigwedge_{i=1}^{s'}(D'_i(x_1, ..., x_n)),$$

где D_i и D'_i — дизъюнкты. Возьмем отрицание от функции $f(x_1, ..., x_n)$.

$$\overline{f}(x_1, ..., x_n) = \overline{\bigwedge_{i=1}^{s}(D_i(x_1, ..., x_n))} = \overline{\bigwedge_{i=1}^{s'}(D'_i(x_1, ..., x_n))}.$$

Произведем преобразования по правилам де Моргана и получим

$$\overline{f}(x_1, ..., x_n) = \bigvee_{i=1}^{s} K_i(x_1, ..., x_n) = \bigvee_{i=1}^{s'} K'_i(x_1, ..., x_n),$$

где $K_i = \overline{D_i}$ и $K'_i = \overline{D'_i}$ — конъюнкты, полученные по правилам де Моргана из дизъюнктов СКНФ для функции f. Поскольку наборы дизъюнктов $\{D_1, ..., D_s\}$ и $\{D'_1, ..., D'_{s'}\}$ различны, то и полученные наборы конъюнктов $\{K_1, ..., K_s\}$ и $\{K'_1, ..., K'_{s'}\}$ не совпадают. Таким образом мы получили две различных СДНФ для функции $\overline{f}$, что противоречит утверждению 2.1.4. Противоречие доказывает единственность СКНФ функции.
□

Замечание 2.1.8 . *Из утверждений 2.1.3 и 2.1.5 мы получили формулы, которые удобно использовать для построения СДНФ и СКНФ соответственно.*

$$f(x_1, ..., x_n) = \bigvee_{\substack{(\sigma_1,...,\sigma_n) \\ f(\sigma_1,...,\sigma_n)=1}} x_1^{\sigma_1} \cdots x_n^{\sigma_n}. \tag{28}$$

для функции $f \neq 0$.

$$f(x_1, ..., x_n) = \bigwedge_{\substack{(\sigma_1,...,\sigma_n) \\ f(\sigma_1,...,\sigma_n)=0}} (x_1^{\overline{\sigma_1}} \vee \cdots \vee x_n^{\overline{\sigma_n}}). \tag{29}$$

для функции $f \neq 1$.

Теперь для построения СДНФ согласно формуле (28) необходимо выбрать каждый набор $(\sigma_1, ..., \sigma_n)$, *для которого* $f(\sigma_1, ..., \sigma_n) = 1$, *и сопоставить ему коньюнкт* $x_1^{\sigma_1} \cdots x_n^{\sigma_n}$ *совершенной дизъюнктивной нормальной формы.*

Аналогично строится совершенная конъюнктивная нормальная форма по формуле (29).

Пример 2.1.11 . *Рассмотрим функцию* $f(x, y, z)$, *заданную таблицей:*

x	y	z	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Тогда, согласно (28) СДНФ будет выглядеть следующим образом:

$$f(x, y, z) = x^0 y^0 z^0 \ \vee \ x^0 y^0 z^1 \ \vee \ x^1 y^0 z^1 \ \vee \ x^1 y^1 z^0 =$$

$$= \overline{x}\ \overline{y}\ \overline{z} \ \vee \ \overline{x}\ \overline{y}\ z \ \vee \ x\ \overline{y}\ z \ \vee \ x\ y\ \overline{z}.$$

СКНФ согласно формуле (29) будет иметь вид:

$$f(x, y, z) = (x^{\overline{0}} \vee y^{\overline{1}} \vee z^{\overline{0}}) \wedge (x^{\overline{0}} \vee y^{\overline{1}} \vee z^{\overline{1}}) \wedge (x^{\overline{1}} \vee y^{\overline{0}} \vee z^{\overline{0}}) \wedge (x^{\overline{1}} \vee y^{\overline{1}} \vee z^{\overline{1}}) =$$

$$(x^1 \vee y^0 \vee z^1) \wedge (x^1 \vee y^0 \vee z^0) \wedge (x^0 \vee y^1 \vee z^1) \wedge (x^0 \vee y^0 \vee z^0) =$$

$$= (x \vee \overline{y} \vee z) \wedge (x \vee \overline{y} \vee \overline{z}) \wedge (\overline{x} \vee y \vee z) \wedge (\overline{x} \vee \overline{y} \vee \overline{z}).$$

2.1.6 Полином Жегалкина

Рассмотрим еще одно представление функции в виде формулы заданного вида.

Определение 2.1.13 . *Формула вида*

$$\alpha_0 \oplus \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus ... \oplus \alpha_n x_n \oplus \alpha_{12} x_1 x_2 \oplus ... \oplus \alpha_{12...n} x_1 x_2 ... x_n, \qquad (30)$$

где $x_1, ..., x_n$ *— логические переменные, а* $\alpha_0, \alpha_1, ..., \alpha_{12...n}$ *— логические константы, называется полиномом Жегалкина.*

Замечание 2.1.9 . *Для удобства будем также использовать следующую запись:*

$$\bigoplus_{I \subseteq \{1,...,n\}} \alpha(I) \bigwedge_{i \in I} x_i.$$

Пример 2.1.12 .

$1 \oplus x_1 \oplus x_2 x_3$ *— полином Жегалкина.*

Здесь $n = 3$, $\alpha_0 = \alpha_1 = \alpha_{23} = 1$, *а* $\alpha_2 = \alpha_3 = \alpha_{12} = \alpha_{13} = \alpha_{123} = 0$.

Утверждение 2.1.6 . *Пусть* $f(x_1, ..., x_n) \in P_2$. *Тогда функция* f *может быть представлена полиномом Жегалкина, причем единственным образом.*

Доказательство. Для начала докажем существование такого представления для функции. Прежде всего заметим, что произвольная функция представима в виде формулы $\mathcal{U}_1$ над $\{\neg, \vee, \wedge\}$: если $f \neq 0$, ее можно представить в виде СДНФ, а $f = 0$ представим как $f = x \wedge \overline{x}$. Воспользуемся законом де Моргана $x \vee y = \overline{\overline{x} \wedge \overline{y}}$ и заменим все дизъюнкции в формуле $\mathcal{U}_1$ на конъюнкции и отрицание. Получим представление функции $f(x_1, ..., x_n)$ в виде формулы $\mathcal{U}_2$ над системой элементарных функций $\{\neg, \wedge\}$. Заменим в формуле $\mathcal{U}_2$ все отрицания на сложение по модулю два согласно тождеству $\overline{x} = 1 \oplus x$. Раскроем все скобки, пользуясь дистрибутивностью $\wedge$ относительно $\oplus$, и получим формулу вида (30).

Теперь докажем единственность. Пусть

$$f(x_1, ..., x_n) = \bigoplus_{I \subseteq \{1,...,n\}} \alpha(I) \bigwedge_{i \in I} x_i = \bigoplus_{I \subseteq \{1,...,n\}} \beta(I) \bigwedge_{i \in I} x_i.$$

— два различных полинома Жегалкина для функции f. Полиномы различны, когда отличаются наборы их коэффициентов α и β.

Выберем $I^* \subseteq \{1, ..., n\}$ такой, что $\alpha(I^*) \neq \beta(I^*)$ и $\alpha(I) = \beta(I)$ для любого $I \subset I^*$. Мы можем выбрать такой набор, начав с $I_0 = \{1, ..., n\}$ и последовательно удаляя индексы, если указанное свойство не выполняется. Так, если для набора индексов I_k существует $I \subset I_k$: $\alpha(I) \neq \beta(I)$, то выбираем $I_{k+1} = I$ и так далее, пока не получим I_l с искомыми свойствами.

Пусть

$$\sigma_i = \begin{cases} 1, & i \in I^*, \\ 0, & i \notin I^*, \end{cases} \qquad i = \overline{1, n}.$$

Тогда

$$f(\sigma_1, ..., \sigma_n) = \bigoplus_{I \subseteq \{1,...,n\}} \alpha(I) \bigwedge_{i \in I} \sigma_i = \bigoplus_{I \subseteq I^*} \alpha(I) = \bigoplus_{I \subset I^*} \alpha(I) \oplus \alpha(I^*)$$

и в то же время

$$f(\sigma_1, ..., \sigma_n) = \bigoplus_{I \subseteq \{1,...,n\}} \beta(I) \bigwedge_{i \in I} \sigma_i = \bigoplus_{I \subseteq I^*} \beta(I) = \bigoplus_{I \subset I^*} \beta(I) \oplus \beta(I^*).$$

Таким образом,

$$\bigoplus_{I \subset I^*} \alpha(I) \oplus \alpha(I^*) = \bigoplus_{I \subset I^*} \beta(I) \oplus \beta(I^*) \quad \Longrightarrow \quad \alpha(I^*) = \beta(I^*).$$

Противоречие с нашим исходным предположением о I^* доказывает утверждение.
□

Замечание 2.1.10 . *Полином Жегалкина является формулой над* $\{0, 1, \wedge, \oplus\}$. *0 нам требуется для задания тождественно нулевой функции. Для остальных случаев нам будет достаточно функций* $1, \wedge, \oplus$.

Пример 2.1.13 . *Рассмотрим функцию* f, *заданную формулой* $f(x, y, z) = \overline{xy \supset z} \vee xy$. *Представим ее с помощью формулы над* $\{\neg, \wedge\}$.

$$xy \supset z = \overline{xy} \vee z = \overline{xy \wedge \overline{z}},$$

$$\overline{xy \supset z} \vee xy = \overline{(xy \supset z) \wedge \overline{xy}} = \overline{\overline{xy \wedge \overline{z}} \wedge \overline{xy}}$$

Теперь избавимся от отрицаний и раскроем скобки.

$$\overline{\overline{xy \wedge \overline{z} \wedge \overline{xy}} = 1 \oplus ((1 \oplus (xy \wedge (1 \oplus z))) \wedge (1 \oplus xy)) =$$

$$= 1 \oplus 1 \oplus xy \oplus (xy \wedge (1 \oplus z)) \oplus (xy \wedge (1 \oplus z))xy =$$

$$= xy \oplus (xy \wedge (1 \oplus z)) \oplus (xy \wedge (1 \oplus z)) = xy.$$

Таким образом, единственным ненулевым коэффициентом полинома Жегалкина для функции f оказался α_{12} и сам полином имеет вид $f(x, y, z) = xy$.

Замечание 2.1.11 . *Для построения полинома Жегалкина есть и более удобный способ — метод неопределенных коэффициентов. Предположим, что функция $f(x_1, ..., x_n)$ задана таблицей значений для всех наборов аргументов. Нам известен общий вид полинома Жегалкина для f*

$$\bigoplus_{I \subseteq \{1,...,n\}} \alpha(I) \bigwedge_{i \in I} x_i$$

и требуется только вычислить коэффициенты $\alpha(I)$.

Проходя по всей таблице значений для f будем приравнивать общий вид полинома и известное значение функции на данном наборе, тем самым последовательно вычисляя коэффициенты α.

$f(0, ..., 0) = \bigoplus_{I \subseteq \{1,...,n\}} \alpha(I) \bigwedge_{i \in I} 0 = \alpha(\varnothing)$. Отсюда имеем первый коэффициент:

$$\alpha(\varnothing) = f(0, ..., 0).$$

$f(0, .., 0, 1) = \alpha(\varnothing) \oplus \alpha(\{n\}) \wedge 1$. Таким образом,

$$\alpha(\{n\}) = f(0, ..., 0, 1) \oplus \alpha(\varnothing) = f(0, ..., 0, 1) \oplus f(0, ..., 0, 0).$$

$f(0, ..., 0, 1, 0) = \alpha(\varnothing) \oplus \alpha(\{n-1\}) \wedge 1$ и

$$\alpha(\{n-1\}) = f(0, ..., 0, 1, 0) \oplus \alpha(\varnothing) = f(0, ..., 0, 1, 0) \oplus f(0, ..., 0, 0, 0).$$

$f(0, ..., 0, 1, 1) = \alpha(\varnothing) \oplus \alpha(\{n-1\}) \oplus \alpha(\{n\}) \oplus \alpha(\{n-1, n\})$ и, следовательно,

$$\alpha(\{n-1, n\}) = f(0, ..., 0, 1, 1) \oplus \alpha(\varnothing) \oplus \alpha(\{n-1\}) \oplus \alpha(\{n\}) =$$

$$= f(0,...,0,1,1) \oplus f(0,...,0,0,0) \oplus f(0,...,0,1,0) \oplus$$
$$\oplus f(0,...,0,0,0) \oplus f(0,...,0,1) \oplus f(0,...,0,0) =$$
$$= f(0,...,0,1,1) \oplus f(0,...,0,1,0) \oplus f(0,...,0,0,1) \oplus f(0,...,0,0,0)$$

Продолжая этот процесс мы сможем вычислить все коэффициенты и тем самым получим полином Жегалкина для функции f.

Пример 2.1.14 . *Построим полином Жегалкина для функции $f(x,y,z) = \overline{xy \supset z} \vee xy$ из примера 2.1.13 с помощью метода неопределенных коэффициентов. В общем виде полином для функции от трех переменных выглядит следующим образом:*

$$f(x,y,z) = \alpha_0 \oplus \alpha_1 x \oplus \alpha_2 y \oplus \alpha_3 z \oplus \alpha_{1,2} xy \oplus \alpha_{1,3} xz \oplus \alpha_{2,3} yz \oplus \alpha_{1,2,3} xyz.$$

Переберем все возможные наборы аргументов.

$$\begin{array}{ll}
f(0,0,0) = 0 = \alpha_0 & \Rightarrow \ \alpha_0 = 0, \\
f(0,0,1) = 0 = \alpha_0 \oplus \alpha_3 & \Rightarrow \ \alpha_3 = 0, \\
f(0,1,0) = 0 = \alpha_0 \oplus \alpha_2 & \Rightarrow \ \alpha_2 = 0, \\
f(0,1,1) = 0 = \alpha_0 \oplus \alpha_2 \oplus \alpha_3 \oplus \alpha_{2,3} & \Rightarrow \ \alpha_{2,3} = 0, \\
f(1,0,0) = 0 = \alpha_0 \oplus \alpha_1 & \Rightarrow \ \alpha_1 = 0, \\
f(1,0,1) = 0 = \alpha_0 \oplus \alpha_1 \oplus \alpha_3 \oplus \alpha_{1,3} & \Rightarrow \ \alpha_{1,3} = 0, \\
f(1,1,0) = 1 = \alpha_0 \oplus \alpha_1 \oplus \alpha_2 \oplus \alpha_{1,2} & \Rightarrow \ \alpha_{1,2} = 1, \\
f(1,1,1) = 1 = \alpha_0 \oplus \alpha_1 \oplus \alpha_2 \oplus \alpha_3 \oplus \alpha_{1,2} \oplus & \\
\oplus \alpha_{1,3} \oplus \alpha_{2,3} \oplus \alpha_{1,2,3} & \Rightarrow \ \alpha_{1,2,3} = 0.
\end{array}$$

Полином Жегалкина имеет вид $f(x,y,z) = xy$, что совпадает с результатом примера 2.1.13.

Пример 2.1.15 . *Рассмотрим еще один пример использования метода неопределенных коэффициентов. Пусть $f(x,y,z) = (\overline{x} \oplus y) \supset z$. Пусть*

$f(x,y,z) = xy \vee xz \vee yz$.

$$
\begin{array}{ll}
f(0,0,0) = 0 = \alpha_0 & \Rightarrow \ \alpha_0 = 0, \\
f(0,0,1) = 0 = \alpha_0 \oplus \alpha_3 & \Rightarrow \ \alpha_3 = 0, \\
f(0,1,0) = 0 = \alpha_0 \oplus \alpha_2 & \Rightarrow \ \alpha_2 = 0, \\
f(0,1,1) = 1 = \alpha_0 \oplus \alpha_2 \oplus \alpha_3 \oplus \alpha_{2,3} & \Rightarrow \ \alpha_{2,3} = 1, \\
f(1,0,0) = 0 = \alpha_0 \oplus \alpha_1 & \Rightarrow \ \alpha_1 = 0, \\
f(1,0,1) = 1 = \alpha_0 \oplus \alpha_1 \oplus \alpha_3 \oplus \alpha_{1,3} & \Rightarrow \ \alpha_{1,3} = 1, \\
f(1,1,0) = 1 = \alpha_0 \oplus \alpha_1 \oplus \alpha_2 \oplus \alpha_{1,2} & \Rightarrow \ \alpha_{1,2} = 1, \\
f(1,1,1) = 1 = \alpha_0 \oplus \alpha_1 \oplus \alpha_2 \oplus \alpha_3 \oplus \alpha_{1,2} \oplus & \\
\oplus \alpha_{1,3} \oplus \alpha_{2,3} \oplus \alpha_{1,2,3} & \Rightarrow \ \alpha_{1,2,3} = 0.
\end{array}
$$

Следовательно, полином Жегалкина для функции f будет иметь вид $f(x,y,z) = xy \oplus xz \oplus yz$.

2.1.7 Полнота системы функций

Определение 2.1.14 . *Система функций $\mathcal{P} \subseteq P_2$ называется полной, если любую функцию из P_2 можно представить в виде формулы над $\mathcal{P}$.*

Замечание 2.1.12 . *Удобно доказывать полноту системы функций, показывая, что она сводится к уже известной полной системе.*

Лемма 2.1.7 . *Пусть даны две системы функций $\mathcal{P} \subseteq P_2$ и $\mathcal{Q} \subseteq P_2$. Пусть функция $f(x_1, x_2, ..., x_n)$ представима в виде формулы над $\mathcal{P}$, и любая функция из $\mathcal{P}$ представима в виде формулы над $\mathcal{Q}$. Тогда функция f представима в виде формулы над $\mathcal{Q}$.*

Доказательство. Пусть, $\mathcal{U}$ — формула над $\mathcal{P}$, реализующая функцию f. $\mathcal{U}_1$, $\mathcal{U}_2$, ..., $\mathcal{U}_k$ — формулы над $\mathcal{Q}$, реализующие все функции g_1, g_2, ..., g_k множества $\mathcal{P}$. Тогда каждое вхождение функции g_i в формулу $\mathcal{U}$ можно заменить формулой $\mathcal{U}_i$. Проведя такую замену, получим формулу $\mathcal{U}'$ над $\mathcal{Q}$. Очевидно, формула $\mathcal{U}'$ эквивалентна формуле $\mathcal{U}$, поскольку они отличаются заменой подформул на эквивалентные.
□

Теорема 2.1.8 . *Пусть даны две системы функций $\mathcal{P} \subseteq P_2$ и $\mathcal{Q} \subseteq P_2$. Пусть $\mathcal{P}$ — полная система функций, и любая функция из $\mathcal{P}$ представима в виде формулы над $\mathcal{Q}$. Тогда, $\mathcal{Q}$ — полная система функций.*

Доказательство. Непосредственно следует из леммы 2.1.7.
□

Приведем несколько примеров полных систем функций.

Утверждение 2.1.9 . $\{\neg, \vee, \wedge\}$ — *полная система функций.*

Доказательство. Рассмотрим произвольную функцию $f \in P_2$.

Если $f = 0$, то $f = f_{\mathcal{U}}$, где $\mathcal{U} = x \wedge \overline{x}$.

В противном случае $f \neq 0$. Тогда по утверждению 2.1.3 $f = f_{\mathcal{U}}$, где $\mathcal{U} = \bigvee\limits_{\substack{(\sigma_1,\ldots,\sigma_n) \\ f(\sigma_1,\ldots,\sigma_n)=1}} x_1^{\sigma_1} \cdots x_n^{\sigma_n}$.
□

Следствие 2.1.10 . $\{\neg, \wedge\}$ *и* $\{\neg, \vee\}$ — *полные системы функций.*

Доказательство. Докозательство очевидно следует из утверждения 2.1.9 и правил де Моргана, если заметить, что

$$x \vee y = \overline{\overline{x} \wedge \overline{y}}, \quad \text{и} \quad x \wedge y = \overline{\overline{x} \vee \overline{y}}.$$

Таким образом, подставив в любую формулу, выражающую функцию f в виде формулы над $\{\neg, \vee, \wedge\}$, указанные выражения для $\vee$ или $\wedge$, можно получить эквивалентную формулу над $\{\neg, \wedge\}$ или над $\{\neg, \vee\}$.
□

Утверждение 2.1.11 . *1) Системы функций* $\{|\}$ *и* $\{\downarrow\}$ — *полные системы функций.*

2) Других полных систем, состоящих из одной функции от двух переменных нет.

Доказательство. 1) а) $\overline{x} = x \mid x, \quad x \vee y = \overline{x} \mid \overline{y} = (x \mid x) \mid (y \mid y)$.
б) $\overline{x} = x \downarrow x, \quad x \wedge y = \overline{x} \downarrow \overline{y} = (x \downarrow x) \downarrow (y \downarrow y)$.

2) Пусть $h(x, y) \in P_2$ и $\{h\}$ — полная система функций.

Пусть $h(0, 0) = 0$. Тогда, если $f(x)$ задана в виде формулы над $\{h\}$, то $f(0) = 0$. Действительно, функция f имеет вид $f(x) = h(\mathcal{U}_1, \mathcal{U}_2)$, где $\mathcal{U}_i$ — переменная x, или формула того же вида, что и f, $i = 1, 2$. Таким образом, в конце концов оказывается, что $f(0) = h(0, 0) = 0$. Тогда, с помощью только функции $h(x, y)$ не может быть функция отрицания, поскольку $\neg 0 = 1$. Следовательно $h(0, 0) = 1$.

Аналогично можно показать, что $h(1,1) = 0$.

Теперь рассмотрим все функции от двух переменных, удовлетворяющие полученному условию: $h(0,0) = 1$, $h(1,1) = 0$. Всего таких функций четыре (Скажите почему).

x	y	$\downarrow$	$\mid$	$\overline{x}$	$\overline{y}$
0	0	1	1	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	0	0	0	0

Как видно из таблицы, две из этих функций известны нам как стрелка Пирса и штрих Шеффера, а две другие представляют из себя функцию отрицания от аргументов x и y. Поскольку система из одной функции отрицания не является полной системой функций, утверждение доказано.
□

Утверждение 2.1.12 . $\{0, 1, \oplus, \wedge\}$ — *полная система функций.*

Доказательство. Истинность этого утверждения следует из того, что любая функция из P_2 представима в виде полинома Жегалкина (утв. 2.1.6).
□

Следствие 2.1.13 . *Так как* $1 \oplus 1 = 0$, *полной системой функций является и* $\{1, \oplus, \wedge\}$.

Определение 2.1.15 . *Пусть*

$$K_0 = \{f_1(x_1, ..., x_{k_1}), f_2(x_1, ..., x_{k_2}), ..., f_m(x_1, ..., x_{k_m})\}.$$

f — *суперпозиция ранга 1 (элементарная суперпозиция) функций* f_1, ..., f_m, *если* f *получена одним из способов:*

а) переименованием некоторой переменной x_j *функции* f_i, $i \in \{1, ..., m\}$, $j \in \{1, ..., k_i\}$*:*

$$f_i(x_1, ..., x_{j-1}, y, x_{j+1}, ..., x_{k_i}),$$

где y может совпасть с любой переменной;

b) подстановкой некоторой функции f_l вместо переменной x_j функции f_i, $l, i \in \{1, ..., m\}$, $j \in \{1, ..., k_i\}$:

$$f_i(x_1, ..., x_{j-1}, f_l(x_1, ..., x_{k_l}), x_{j+1}, ..., x_{k_i}).$$

Множество суперпозиций ранга 1 функций из K_0 обозначим K_1. Также, множество суперпозиций ранга 1 функций из K_{i-1} обозначим K_i, $i = \overline{1, \infty}$. Функции из множества K_i будем называть суперпозициями ранга i функций из K_0.

Определение 2.1.16 . *Суперпозицией функций из K_0 будем называть суперпозицию любого ранга. Другими словами, f — суперпозиция функций f_1, ..., f_m, если $\exists t \in \mathbb{N}$ такое, что $f \in K_t$.*

Замечание 2.1.13 . *По сути дела утверждение, что f является суперпозицией функций из K_0, эквивалентно утверждению, что f представима в виде формулы над K_0. Таким образом мы можем переформулировать определение полной системы.*

Определение 2.1.17 (2.1.14'). *Система функций $\mathcal{P} \subseteq P_2$ называется полной, если любая функция из P_2 является суперпозицией функций из $\mathcal{P}$.*

Определение 2.1.18 . *Пусть $\mathfrak{M} \subseteq P_2$. Замыканием $\mathfrak{M}$ называется множество*

$$[\mathfrak{M}] = \{f \mid f \text{ — суперпозиция функций из } \mathfrak{M}\}.$$

Определение 2.1.19 . *Пусть $\mathfrak{M} \subseteq P_2$. $\mathfrak{M}$ — замкнутое множество функций, если $\mathfrak{M} = [\mathfrak{M}]$.*

Пример 2.1.16 .

1) $M = \{x, \overline{x}\}$. Тогда $[M] = \{x, \overline{x}\} = M$ и M — замкнуто.

2) $M = \{\overline{x}\}$. Тогда $[M] = \{x, \overline{x}\} \neq M$. Множество M — не замкнуто.

Несложно проверить нижеследующие свойства операции замыкания.

Утверждение 2.1.14 . *Пусть* $M, N \subset P_2$.

1) Замыкание множества содержит само множество:

$$\mathfrak{M} \subseteq [\mathfrak{M}].$$

2) Замыкание произвольного множества замкнуто:

$$[[\mathfrak{M}]] = [\mathfrak{M}].$$

3) Замыкание сохраняет включение множеств:

$$\mathfrak{M} \subseteq \mathfrak{N} \quad \Rightarrow [\mathfrak{M}] \subseteq [\mathfrak{N}].$$

4) Замыкание объединения содержит объединение замыканий:

$$[\mathfrak{M}] \cup [\mathfrak{N}] \subseteq [\mathfrak{M} \cup \mathfrak{N}].$$

Замечание 2.1.14 . *Заметим,что для доказательства замкнутости некоторого класса функций* $\mathfrak{M}$ *достаточно показать, что любая суперпозиция ранга 1 функций из* $\mathfrak{M}$ *лежит в* $\mathfrak{M}$.

Обозначим множество суперпозиций ранга i *функций из* $\mathfrak{M}$ *за* $\mathfrak{M}_i$. *Пусть* $\mathfrak{M}_1 = \mathfrak{M}$. *Тогда* $\mathfrak{M}_2$ *— множество суперпозиций ранга 1 функций из* $\mathfrak{M}_1 = \mathfrak{M}$. *Следовательно,* $\mathfrak{M}_2 = \mathfrak{M}_1 = \mathfrak{M}$. *Аналогично можно убедиться, что* $\mathfrak{M}_i = \mathfrak{M}$, $i = \overline{1, \infty}$.

Далее, когда мы будем доказывать замкнутость классов функций, будем ограничиваться только рассмотрением суперпозиций ранга 1.

С учетом определения замкнутости можно дать еще одно альтернативное определение полноты.

Определение 2.1.20 (2.1.14"). *Система функций* $\mathcal{P} \subseteq P_2$ *— полная, если* $[\mathcal{P}] = P_2$.

Далее мы бы хотели доказать критерий, позволяющий для произвольной системы функций из P_2 определить, является ли она полной. Для этого нам понадобится ввести и изучить свойства нескольких замкнутых классов функций алгебры логики.

2.1.8 Функции, сохраняющие ноль

Определение 2.1.21 . *Пусть* $f(x_1, ..., x_n) \in P_2$. f *называют функцией, сохраняющей ноль, если* $f(0, ..., 0) = 0$.

Множество всех функций сохраняющих 0 обозначим T_0*:*

$$T_0 = \{f(x_1, ..., x_n) \mid f \in P_2,\ f(0, ..., 0) = 0\}.$$

Утверждение 2.1.15 . *Класс функций* T_0 *замкнут.*

Доказательство. Рассмотрим суперпозицию ранга 1 от функций из T_0.
a) Пусть $f(x_1, ..., x_n) \in T_0$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) = f(x_1, ..., x_{j-1}, y, x_{j+1}, ..., x_n).$$

Тогда $g(0, ..., 0) = f(0, ..., 0) = 0$. Следовательно, $g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) \in T_0$.

b) Пусть $f(x_1, ..., x_n) \in T_0$ и $h(y_1, ..., y_m) \in T_0$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) = \\ = f(x_1, ..., x_{j-1}, h(y_1, ..., y_m), x_{j+1}, ..., x_n).$$

Тогда

$$g(0, ..., 0) = f(0, ..., 0, h(0, ..., 0), 0, ..., 0) = f(0, ..., 0, 0, 0, ..., 0) = 0.$$

Получаем, что $g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) \in T_0$.

Таким образом, $[T_0] = T_0$.

□

Замечание 2.1.15 . *Тождественная функция* $f(x) = x$ *лежит в классе* T_0*. Функция отрицания* $f(x) = \overline{x}$ *не лежит в* T_0*. Таким образом,* $T_0 \neq \varnothing$ *и* $T_0 \neq P_2$.

2.1.9 Функции, сохраняющие единицу

Замечание 2.1.16 . *Рассуждения этого пункта повторяют содержание предыдущего с точностью до замены* 0 *на* 1

Определение 2.1.22 . *Пусть* $f(x_1, ..., x_n) \in P_2$. *f называют функцией, сохраняющей единицу, если* $f(1, ..., 1) = 1$.

Множество всех функций сохраняющих 1 обозначим T_1*:*

$$T_1 = \{f(x_1, ..., x_n) \mid f \in P_2,\ f(1, ..., 1) = 1\}.$$

Утверждение 2.1.16 . *Класс функций* T_1 *замкнут.*

Доказательство. Доказательство полностью эквивалентно доказательству утверждения 2.1.15.
□

Замечание 2.1.17 . *Тождественная функция* $f(x) = x$ *лежит в классе* T_1. *Функция отрицания* $f(x) = \overline{x}$ *не лежит в* T_1. *Таким образом,* $T_1 \neq \varnothing$ *и* $T_1 \neq P_2$.

2.1.10 Двойственность

Определение 2.1.23 . *Пусть* $f(x_1, ..., x_n) \in P_2$. *Двойственной функцией к функции* f *называется* $f^*(x_1, ..., x_n) = \overline{f}(\overline{x}_1, ..., \overline{x}_n)$.

Пример 2.1.17 . *Пусть* $f(x, y, z) = \overline{xy \vee z}$. *Тогда* $f^*(x, y, z) = \overline{\overline{x} \cdot \overline{y} \vee \overline{z}} = \overline{x} \cdot \overline{y} \vee \overline{z}$.

Пример 2.1.18 . *Пусть функция* $f(x, y, z)$ *задана таблицей. Тогда, чтобы получить двойственную функцию, нужно перевернуть и инвертировать столбец значений:*

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

.

Определение 2.1.24 . *Функция $f(x_1, ..., x_n) \in P_2$ — самодвойственная, если $f^*(x_1, ..., x_n) = f(x_1, ..., x_n)$.*

Обозначим за S множество всех самодвойственных функций:

$$S = \{f \mid f(x_1, ..., x_n) \in P_2,\ f^*(x_1, ..., x_n) = f(x_1, ..., x_n)\}.$$

Пример 2.1.19 . *Пусть $f = x \vee y$, $f^* = \overline{\overline{x} \vee \overline{y}} = xy$. $f \neq f^*$ и, следовательно, f^* не является самодвойственной.*

Пример 2.1.20 . *Пусть f — функция голосования:*

$$f(x, y, z) = xy \vee xz \vee yz.$$

$$f^*(x, y, z) = \overline{\overline{x} \cdot \overline{y} \vee \overline{x} \cdot \overline{z} \vee \overline{y} \cdot \overline{z}} = \overline{\overline{x} \cdot \overline{y}} \cdot \overline{\overline{x} \cdot \overline{z}} \cdot \overline{\overline{y} \cdot \overline{z}} =$$
$$= (x \vee y)(x \vee z)(y \vee z) = (x \vee xz \vee xy \vee yz)(y \vee z) =$$
$$xy \vee xz \vee xyz \vee xz \vee xy \vee xyz \vee yz \vee yz = xy \vee xz \vee yz \vee xyz =$$
$$= xy \vee xz \vee yz = f(x, y, z).$$

Функция голосования — самодвойственная.

Рассмотрим табличный вид функции голосования:

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

.

Можно видеть, что нижняя половина столбца значений повторяет перевернутую и инвертированную верхнюю. Это верно для любой самодвойственной функции.

Утверждение 2.1.17 (Принцип двойственности).
Пусть формула $\mathcal{U} = f(\mathcal{U}_1, ..., \mathcal{U}_n)$ *реализует функцию* $F(x_1, ..., x_m)$, *где* $\mathcal{U}_i$ *— формулы, реализующие* $f_i(x_{j_1}, ..., x_{j_{k_i}})$, $i = \overline{1,n}$. *Пусть* $\mathcal{U}_i^*$ *— формулы реализующие* f_i^*, $i = \overline{1,n}$. *Тогда формула* $\mathcal{U}^* = f^*(\mathcal{U}_1^*, ..., \mathcal{U}_n^*)$ *реализует функцию* $F^*(x_1, ..., x_m)$.

Доказательство.

$$F(x_1, ..., x_m) = f(f_1(x_{i_1}, ..., x_{i_{k_1}}), ..., f_n(x_{j_1}, ..., x_{j_{k_n}})).$$

Тогда, по определению двойственной функции

$$F^*(x_1, ..., x_m) = \overline{f}(f_1(\overline{x}_{i_1}, ..., \overline{x}_{i_{k_1}}), ..., f_n(\overline{x}_{j_1}, ..., \overline{x}_{j_{k_n}})) =$$

$$= \overline{f}(\overline{f}_1(\overline{x}_{i_1}, ..., \overline{x}_{i_{k_1}}), ..., \overline{f}_n(\overline{x}_{j_1}, ..., \overline{x}_{j_{k_n}})) =$$
$$= \overline{f}(\overline{f^*}_1(x_{i_1}, ..., x_{i_{k_1}}), ..., \overline{f^*}_n(x_{j_1}, ..., x_{j_{k_n}})) =$$
$$= f^*({f^*}_1(x_{i_1}, ..., x_{i_{k_1}}), ..., {f^*}_n(x_{j_1}, ..., x_{j_{k_n}})).$$

С другой стороны, формула $f^*(\mathcal{U}_1^*, ..., \mathcal{U}_n^*)$ реализует функцию

$$f^*(f_{\mathcal{U}_1^*}(x_{i_1}, ..., x_{i_{k_1}}), ..., f_{\mathcal{U}_n^*}(x_{j_1}, ..., x_{j_{k_n}})) =$$
$$= f^*({f^*}_1(x_{i_1}, ..., x_{i_{k_1}}), ..., {f^*}_n(x_{j_1}, ..., x_{j_{k_n}})).$$

Таким образом, один из способов задать функцию $F^*(x_1, ..., x_m)$ формулой имеет вид $\mathcal{U}^* = f^*(\mathcal{U}_1^*, ..., \mathcal{U}_n^*)$.
□

Пример 2.1.21 . *Пусть,*

$$F(x, y, z) = (x \equiv y) \supset (y \mid z) = f(f_1(x, y), f_2(y, z)).$$

$$f^*(x, y) = (x \supset y)^* = \overline{\overline{x} \supset \overline{y}} = \overline{x \vee \overline{y}} = \overline{x}y.$$
$$f_1^*(x, y) = (x \equiv y)^* = \overline{\overline{x} \equiv \overline{y}} = \overline{x \equiv y} = x \oplus y.$$
$$f_2^*(x, y) = (x \mid y)^* = \overline{\overline{x} \mid \overline{y}} = \overline{x \vee y} = \overline{x} \wedge \overline{y} = x \downarrow y.$$

Тогда по утверждению 2.1.17 $f^*(x, y, z)$ *будет иметь вид*

$$F^*(x, y, z) = f^*(f_1^*(x, y), f_2^*(y, z)) = \overline{(x \oplus y)} \wedge (y \downarrow z).$$

Действительно,

$$F^*(x,y,z) = \neg((\overline{x} \equiv \overline{y}) \supset (\overline{y} \mid \overline{z})) = \neg(\overline{(x \equiv y)} \vee (y \vee z)) =$$
$$= \overline{\overline{(x \equiv y)}} \wedge \overline{(y \vee z)} = \overline{(x \oplus y)} \wedge (y \downarrow z).$$

Следствие 2.1.18 . *Пусть $f(x_1, ..., x_n)$ задана формулой $\mathcal{U}$ над множеством функций $\{0, 1, \neg, \vee, \wedge\}$. Тогда $f^*(x_1, ..., x_n)$ задается формулой, полученной из $\mathcal{U}$ заменой: нулей на единицы, единиц на нули, конъюнкций на дизъюнкции, дизъюнкций на конъюнкции.*

Доказательство. Пусть $f = \neg f_1$. Это значит, что $f = f_0(f_1)$, где $f_0(x) = \overline{x}$. Тогда $f_0^* = \overline{\overline{x}} = \overline{x} = f_0$. Следовательно

$$f^* = f_0^*(f_1^*) = f_0(f_1^*) = \neg f_1^*.$$

Пусть $f = f_1 \vee f_2$. Другими словами, $f = f_0(f_1, f_2)$, $f_0(x, y) = x \vee y$. Тогда $f_0^* = \overline{\overline{x} \vee \overline{y}} = x \wedge y$ и

$$f^* = f_0^*(f_1^*, f_2^*) = f_1^* \wedge f_2^*.$$

Пусть $f = f_1 \wedge f_2$. Другими словами, $f = f_0(f_1, f_2)$, $f_0(x, y) = x \wedge y$. Тогда $f_0^* = \overline{\overline{x} \wedge \overline{y}} = x \vee y$ и

$$f^* = f_0^*(f_1^*, f_2^*) = f_1^* \vee f_2^*.$$

Пусть $f = 0$. Тогда $f^* = 1$.
Пусть $f = 1$. Тогда $f^* = 0$.
Пусть $f = x$. Тогда $f^* = \overline{\overline{x}} = x = f$.
□

Пример 2.1.22 . *Пусть, $f(x, y, z) = 0 \wedge x \wedge \overline{y} \vee 1 \wedge y \wedge \overline{z}$. Тогда,*

$$f^*(x,y,z) = \overline{f}(\overline{x}, \overline{y}, \overline{z}) = \neg(0 \wedge \overline{x} \wedge \overline{\overline{y}} \vee 1 \wedge \overline{y} \wedge \overline{\overline{z}}) =$$
$$= \overline{(0 \wedge \overline{x} \wedge y)} \wedge \overline{(1 \wedge \overline{y} \wedge z)} = (1 \vee x \vee \overline{y}) \wedge (0 \vee y \vee \overline{z}).$$

Пример подтверждает утверждение 2.1.18.

Пример 2.1.23 . *Пусть, $f(x, y, z) = (\overline{0 \vee x})(y \vee \overline{x}z)$. Тогда*

$$f^*(x,y,z) = \neg((\overline{0 \vee \overline{x}})(\overline{y} \vee \overline{\overline{x}} \wedge \overline{z})) = \neg(1 \wedge x)(\overline{y} \vee x \wedge \overline{z})) =$$
$$= (\overline{1 \wedge x}) \vee (\overline{\overline{y} \vee x \wedge \overline{z}})) = (\overline{1 \wedge x}) \vee (y \wedge (\overline{x} \vee z))).$$

Пример подтверждает утверждение 2.1.18.

Утверждение 2.1.19 . *Класс функций S замкнут.*

Доказательство. Рассмотрим суперпозицию ранга 1 от функций из S.
a) Пусть $f(x_1, ..., x_n) \in S$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) = f(x_1, ..., x_{j-1}, y, x_{j+1}, ..., x_n).$$

Тогда

$$\overline{g}(\overline{x}_1, ..., \overline{x}_{j-1}, \overline{x}_{j+1}, ..., \overline{x}_n, \overline{y}) = \overline{f}(\overline{x}_1, ..., \overline{x}_{j-1}, \overline{y}, \overline{x}_{j+1}, ..., \overline{x}_n) =$$

$$= f^*(x_1, ..., x_{j-1}, y, x_{j+1}, ..., x_n) = f(x_1, ..., x_{j-1}, y, x_{j+1}, ..., x_n) =$$

$$= g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y).$$

Следоваетльно $g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) \in S$.
b) Пусть $f(x_1, ..., x_n) \in S$ и $h(y_1, ..., y_m) \in S$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) =$$
$$= f(x_1, ..., x_{j-1}, h(y_1, ..., y_m), x_{j+1}, ..., x_n).$$

Тогда

$$\overline{g}(\overline{x}_1, ..., \overline{x}_{j-1}, \overline{x}_{j+1}, ..., x_n, \overline{y}_1, ..., \overline{y}_m) =$$
$$= \overline{f}(\overline{x}_1, ..., \overline{x}_{j-1}, h(\overline{y}_1, ..., \overline{y}_m), \overline{x}_{j+1}, ..., \overline{x}_n) =$$
$$= \overline{f}(\overline{x}_1, ..., \overline{x}_{j-1}, \overline{\overline{h}}(\overline{y}_1, ..., \overline{y}_m), \overline{x}_{j+1}, ..., \overline{x}_n) =$$
$$= \overline{f}(\overline{x}_1, ..., \overline{x}_{j-1}, \overline{h^*}(y_1, ..., y_m), \overline{x}_{j+1}, ..., \overline{x}_n) =$$
$$= f^*(x_1, ..., x_{j-1}, h^*(y_1, ..., y_m), x_{j+1}, ..., x_n) =$$
$$= f(x_1, ..., x_{j-1}, h(y_1, ..., y_m), x_{j+1}, ..., x_n) =$$
$$= g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m)$$

Получаем, что $g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) \in S$.
Таким образом, $[S] = S$.
□

Замечание 2.1.18 . *Тождественная функция $f(x) = x$ лежит в классе S. Конъюнкция $f(x) = x \wedge y$ не лежит в S. Таким образом, $S \neq \varnothing$ и $S \neq P_2$.*

Лемма 2.1.20 (О несамодвойственной функции). *Пусть $f(x_1, ..., x_n) \notin S$. Тогда, подставляя в f вместо аргументов x и $\overline{x}$ можно получить константу.*

Доказательство. Пусть $(\alpha_1, ..., \alpha_n)$, $\alpha_i \in \{0, 1\}$, такой набор, что $f(\alpha_1, ..., \alpha_n) = f(\overline{\alpha}_1, ..., \overline{\alpha}_n)$. Такой набор обязан существовать в силу несамодвойственности функции f.

Рассмотрим функцию $\varphi(x) = f(x^{\alpha_1}, ..., x^{\alpha_n})$. Тогда

$$\varphi(0) = f(0^{\alpha_1}, ..., 0^{\alpha_n}) = f({\alpha_1}^0, ..., {\alpha_n}^0) = f(\overline{\alpha}_1, ..., \overline{\alpha}_n) =$$

$$= f(\alpha_1, ..., \alpha_n) = f(1^{\alpha_1}, ..., 1^{\alpha_n}) = \varphi(1).$$

Следовательно $\varphi(x)$ — константа.
□

Пример 2.1.24 (получение константы из несамодвойственной функции). *Пусть $f(x, y, z) = x \supset (y \oplus z)$. Это несамодвойственная функция, поскольку*

$$f(0, 1, 0) = 0 \supset (1 \oplus 0) = 1 = 1 \supset (0 \oplus 1) = f(1, 0, 1).$$

Тогда $\varphi(x) = f(\overline{x}, x, \overline{x})$ — константа. Действительно,

$$\varphi(x) = \overline{x} \supset (x \oplus \overline{x}) = \overline{x} \supset 1 = 1.$$

2.1.11 Монотонность

Определение 2.1.25 . *Пусть $\widetilde{\alpha}^n = (\alpha_1, ..., \alpha_n)$, $\widetilde{\beta}^n = (\beta_1, ..., \beta_n)$, $\alpha_i, \beta_i \in \{0, 1\}$, $i = \overline{1, n}$.*

Говорят, что набор $\widetilde{\alpha}^n$ предшествует набору $\widetilde{\beta}^n$ (набор $\widetilde{\beta}^n$ следует после набора $\widetilde{\alpha}^n$) и пишут $\widetilde{\alpha}^n \preceq \widetilde{\beta}^n$, если

$$\alpha_1 \leq \beta_1, \quad \alpha_2 \leq \beta_2, \quad ..., \quad \alpha_n \leq \beta_n.$$

Будем говорить, что $\widetilde{\alpha}^n$ строго предшествует $\widetilde{\beta}^n$ и обозначать $\widetilde{\alpha}^n \prec \widetilde{\beta}^n$, если $\widetilde{\alpha}^n \preceq \widetilde{\beta}^n$ и $\widetilde{\alpha}^n \neq \widetilde{\beta}^n$.

Будем говорить, что $\widetilde{\alpha}^n$ непосредственно предшествует $\widetilde{\beta}^n$ и обозначать $\widetilde{\alpha}^n \prec' \widetilde{\beta}^n$, если если $\widetilde{\alpha}^n \prec \widetilde{\beta}^n$ и не существует набора $\widetilde{\gamma}^n$ такого, что $\widetilde{\alpha}^n \prec \widetilde{\gamma}^n \prec \widetilde{\beta}^n$.

Замечание 2.1.19 . *$\preceq$, $\prec$ — отношения частичного порядка.*

Определение 2.1.26 . *Пусть $f(x_1, ..., x_n) \in P_2$. Функция f называется монотонной, если*

$$\widetilde{\alpha}^n \preceq \widetilde{\beta}^n \quad \Longrightarrow \quad f(\widetilde{\alpha}^n) \leq f(\widetilde{\beta}^n).$$

Класс всех монотонных функций будем обозначать M.

Пример 2.1.25 . *Функция x является монотонной. Функция $\overline{x}$ — не монотонна.*

Утверждение 2.1.21 . *Класс функций M замкнут.*

Доказательство. Рассмотрим суперпозицию ранга 1 от функций из M.
a) Пусть $f(x_1, ..., x_n) \in M$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) = f(x_1, ..., x_{j-1}, y, x_{j+1}, ..., x_n).$$

Тогда монотонность g непосредственно следует из монотонности функции f.

b) Пусть $f(x_1, ..., x_n) \in M$ и $h(y_1, ..., y_m) \in M$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) = \\ = f(x_1, ..., x_{j-1}, h(y_1, ..., y_m), x_{j+1}, ..., x_n).$$

Пусть $\widetilde{\alpha}^{n+m-1} \preceq \widetilde{\beta}^{n+m-1}$. Тогда, в частности,

$$(\alpha_n, ..., \alpha_{n+m-1}) \preceq (\beta_n, ..., \beta_{n+m-1}).$$

Поскольку h монотонна, то

$$h(\alpha_n, ..., \alpha_{n+m-1}) \leq h(\beta_n, ..., \beta_{n+m-1})$$

и, следовательно,

$$(\alpha_1, ..., \alpha_{j-1}, h(\alpha_n, ..., \alpha_{n+m-1}), \alpha_j, ..., \alpha_{n-1}) \preceq$$

$$\preceq (\beta_1, ..., \beta_{j-1}, h(\beta_n, ..., \beta_{n+m-1}), \beta_j, ..., \beta_{n-1}).$$

Таким образом, из монотонности f следует, что

$$g(\widetilde{\alpha}^{n+m-1}) = f(\alpha_1, ..., \alpha_{j-1}, h(\alpha_n, ..., \alpha_{n+m-1}), \alpha_j, ..., \alpha_{n-1}) \leq$$

$$\leq f(\beta_1, ..., \beta_{j-1}, h(\beta_n, ..., \beta_{n+m-1}), \beta_j, ..., \beta_{n-1}) = g(\widetilde{\beta}^{n+m-1}).$$

Получаем, что $g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) \in M$.

Таким образом, $[M] = M$.

□

Замечание 2.1.20 . *Тождественная функция $f(x) = x$ лежит в классе M. Функция отрицания $f(x) = \overline{x}$ не лежит в M. Таким образом, $M \neq \varnothing$ и $M \neq P_2$.*

Лемма 2.1.22 (О немонотонной функции). *Пусть $f(x_1, ..., x_n) \notin M$. Тогда, подставляя в f вместо аргументов 0, 1, x можно получить $\overline{x}$.*

Доказательство. Так как $f \notin M$, существуют два набора $\widetilde{\alpha}^n$ и $\widetilde{\beta}^n$, такие что $\widetilde{\alpha}^n \preceq \widetilde{\beta}^n$ и $f(\widetilde{\alpha}^n) > f(\widetilde{\beta}^n)$. Очевидно, что $f(\widetilde{\alpha}^n) = 1$ и $f(\widetilde{\beta}^n) = 0$.

Пусть $\widetilde{\alpha}^n$ отличен от $\widetilde{\beta}^n$ в t позициях.

a) Пусть $t = 1$. Тогда $\widetilde{\alpha}^n \prec' \widetilde{\beta}^n$ для некоторого i верно, что

$$\widetilde{\alpha}^n = (\alpha_1, ..., \alpha_{i-1}, 0, \alpha_{i+1}, ..., \alpha_n)$$

и

$$\widetilde{\beta}^n = (\alpha_1, ..., \alpha_{i-1}, 1, \alpha_{i+1}, ..., \alpha_n).$$

Определим функцию $\varphi(x)$ следующим образом:

$$\varphi(x) = f(\alpha_1, ..., \alpha_{i-1}, x, \alpha_{i+1}, ..., \alpha_n).$$

Тогда:

$$\varphi(0) = f(\alpha_1, ..., \alpha_{i-1}, 0, \alpha_{i+1}, ..., \alpha_n) = 1$$

$$\varphi(1) = f(\alpha_1, ..., \alpha_{i-1}, 1, \alpha_{i+1}, ..., \alpha_n) = 0.$$

То есть $\varphi(x) = \overline{x}$, что и требовалось доказать.

b) Пусть теперь $t > 1$. В этом случае построим последовательность наборов

$$\widetilde{\alpha}^n = \widetilde{\gamma}^n(0) \prec \widetilde{\gamma}^n(1) \prec ... \prec \widetilde{\gamma}^n(t-1) \prec \widetilde{\gamma}^n(t) = \widetilde{\beta}^n,$$

где каждая пара наборов $\widetilde{\gamma}^n(i-1)$ и $\widetilde{\gamma}^n(i)$ отличаются только в одной позиции, $i = \overline{1,t}$. Это не трудно сделать, последовательно заменяя каждую позицию, в которой наборы $\widetilde{\alpha}^n$ и $\widetilde{\beta}^n$ различаются с нуля на единицу. Поскольку $f(\widetilde{\alpha}^n) = 1$ и $f(\widetilde{\beta}^n) = 0$, найдется такое k, что $\widetilde{\gamma}^n(k-1) = 1$ и $\widetilde{\gamma}^n(i) = 0$. Такая ситуация возвращает нас к пункту a) доказательства.
□

Пример 2.1.26 (Построение $\overline{x}$ с помощью немонотонной функции). *Пусть $f(x,y,z) = x \supset (y \oplus z)$. Эта функция немонотонна, так как*

$$f(0,0,0) = 0 \supset (0 \oplus 0) = 1 > 0 = 1 \supset (1 \oplus 1) = f(1,1,1).$$

Рассмотрим последовательность наборов

$$(0,0,0) \prec (0,0,1) \prec (0,1,1) \prec (1,1,1)$$

и значения функции f на этих наборах

$$1 = f(0,0,0) = f(0,0,1) = f(0,1,1) > f(1,1,1) = 0.$$

Тогда $\varphi(x) = f(x,1,1) = \overline{x}$. Действительно,

$$\varphi(x) = x \supset (1 \oplus 1) = x \supset 0 = \overline{x}.$$

2.1.12 Линейность

Определение 2.1.27 . *Пусть $f(x_1,...,x_n) \in P_2$. Функция f — линейная, если ее полином Жегалкина имеет вид*

$$f(x_1,...,x_n) = \alpha_0 \oplus \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus ... \oplus \alpha_n x_n.$$

Обозначим L — класс всех линейных функций.

Пример 2.1.27 . *x — линейная функция. Функция $x \vee y$ не является линейной.*

Утверждение 2.1.23 . *Класс функций L замкнут.*

Доказательство. Рассмотрим суперпозицию ранга 1 от функций из L.
а) Пусть $f(x_1, ..., x_n) \in L$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) = f(x_1, ..., x_{j-1}, y, x_{j+1}, ..., x_n).$$

Если

$$f(x_1, ..., x_n) = \alpha_0 \oplus \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus ... \oplus \alpha_n x_n,$$

то

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) = \\ = \alpha_0 \oplus \alpha_1 x_1 \oplus ... \oplus \alpha_{j-1} x_{j-1} \oplus \alpha_j y \oplus \alpha_{j+1} x_{j+1} \oplus ... \oplus \alpha_n x_n.$$

Следовательно $g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y) \in L$.
b) Пусть $f(x_1, ..., x_n) \in L$ и $h(y_1, ..., y_m) \in L$ и

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) = \\ = f(x_1, ..., x_{j-1}, h(y_1, ..., y_m), x_{j+1}, ..., x_n).$$

Пусть

$$f(x_1, ..., x_n) = \alpha_0 \oplus \alpha_1 x_1 \oplus \alpha_2 x_2 \oplus ... \oplus \alpha_n x_n$$

и

$$h(y_1, ..., y_m) = \beta_0 \oplus \beta_1 y_1 \oplus \beta_2 y_2 \oplus ... \oplus \beta_m y_m.$$

Тогда

$$g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) = \alpha_0 \oplus \alpha_1 x_1 \oplus ... \oplus \alpha_{j-1} x_{j-1} \oplus \\ \oplus \alpha_j(\beta_0 \oplus \beta_1 y_1 \oplus ... \oplus \beta_m y_m) \oplus \alpha_{j+1} x_{j+1} \oplus ... \oplus \alpha_n x_n = \\ = (\alpha_0 \oplus \alpha_j \beta_0) \oplus \alpha_1 x_1 \oplus ... \oplus \alpha_{j-1} x_{j-1} \oplus \alpha_{j+1} x_{j+1} \oplus ... \oplus \alpha_n x_n \oplus \\ \oplus \alpha_j \beta_1 y_1 \oplus ... \oplus \alpha_j \beta_m y_m.$$

Если некоторая переменная x_i совпадает с переменной y_j, сложим коэффициенты по модулю 2. Полученный полином Жегалкина линеен. Следовательно, $g(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n, y_1, ..., y_m) \in L$.

Таким образом, $[L] = L$.

□

Замечание 2.1.21 . *Тождественная функция $f(x) = x$ лежит в классе L. Дизъюнкция $f(x) = x \vee y$ не лежит в L. Таким образом, $L \neq \varnothing$ и $L \neq P_2$.*

Лемма 2.1.24 (О нелинейной функции). *Пусть $f(x_1, ..., x_n) \notin L$. Тогда, подставляя в f вместо аргументов константы, x, y, $\overline{x}$, $\overline{y}$ и, возможно, навешивая отрицание над f можно получить $x \wedge y$.*

Доказательство. Пусть $f(x_1, ..., x_n) = \bigoplus_{I \subseteq \{1,...,n\}} \alpha(I) \bigwedge_{i \in I} x_i$ и $f \notin L$. Тогда существует $I^* \subseteq \{1, ..., n\}$: $|I^*| \geq 2$, $\alpha(I^*) \neq 0$. Не умаляя общности, положим $\{1, 2\} \subseteq I^*$.

$$f(x_1, ..., x_n) = \\ = x_1 x_2 f_{1,2}(x_3, ..., x_n) \oplus x_1 f_1(x_3, ..., x_n) \oplus x_2 f_2(x_3, ..., x_n) \oplus f_0(x_3, ..., x_n),$$

причем $\exists \alpha_3, ..., \alpha_n$: $f_{1,2}(\alpha_3, ..., \alpha_n) = 1$. Действительно, такие $\alpha_3, ..., \alpha_n$ существуют, поскольку, если бы $f_{1,2}(\sigma_3, ..., \sigma_n) = 0$, $\forall \sigma_3, ..., \sigma_n \in \{0, 1\}$, то функция f приняла бы вид

$$f(x_1, ..., x_n) = x_1 f_1(x_3, ..., x_n) \oplus x_2 f_2(x_3, ..., x_n) \oplus f_0(x_3, ..., x_n),$$

что противоречит нашему предположению, что $\{1, 2\} \subseteq I^*$ и $\alpha(I^*) = 1$.

Рассмотрим

$$\psi(x, y) = f(x, y, \alpha_3, ..., \alpha_n) = xy \oplus x f_1(\alpha_3, ..., \alpha_n) \oplus \\ \oplus y f_2(\alpha_3, ..., \alpha_n) \oplus f_0(\alpha_3, ..., \alpha_n) = xy \oplus x\beta \oplus y\gamma \oplus \delta.$$

Теперь определим $\varphi(x, y)$, как $\varphi(x, y) = \psi(x \oplus \gamma, y \oplus \beta) \oplus \gamma\beta \oplus \delta$. Тогда

$$\varphi(x, y) = ((x \oplus \gamma)(y \oplus \beta) \oplus (x \oplus \gamma)\beta \oplus (y \oplus \beta)\gamma \oplus \delta) \oplus \gamma\beta \oplus \delta = \\ = xy \oplus x\beta \oplus y\gamma \oplus \gamma\beta \oplus x\beta \oplus \gamma\beta \oplus y\gamma \oplus \gamma\beta \oplus \delta \oplus \gamma\beta \oplus \delta = xy.$$

Таким образом, мы получили функцию $\varphi(x, y) = x \wedge y$, причем

$$\varphi(x, y) = f(x \oplus f_2(\alpha_3, ..., \alpha_n), y \oplus f_1(\alpha_3, ..., \alpha_n), \alpha_3, ..., \alpha_n) \oplus \\ \oplus f_1(\alpha_3, ..., \alpha_n) f_2(\alpha_3, ..., \alpha_n) \oplus f_0(\alpha_3, ..., \alpha_n),$$

где добавление к x и y констант $f_2(\alpha_3, ..., \alpha_n)$ и $f_1(\alpha_3, ..., \alpha_n)$ равносильно навешиванию отрицания над переменной, если соответствующая

константа равна 1, а добавление константы $f_1(\alpha_3, ..., \alpha_n)f_2(\alpha_3, ..., \alpha_n) \oplus f_0(\alpha_3, ..., \alpha_n)$ к f означает возможное навешивание отрицания над этой функцией.
□

Пример 2.1.28 (Построение $x \wedge y$ с помощью нелинейной функции). *Пусть $f(x, y, z) = x \supset (y \oplus z)$. Построим полином Жегалкина этой функции методом неопределенных коэффициентов.*

$$f(x, y, z) = \alpha_0 \oplus \alpha_1 x \oplus \alpha_2 y \oplus \alpha_3 z \oplus \alpha_{1,2} xy \oplus \alpha_{1,3} xz \oplus \alpha_{2,3} yz \oplus \alpha_{1,2,3} xyz.$$

x	y	z	$f(x,y,z)$	
0	0	0	1	$\alpha_0 = 1$
0	0	1	1	$\alpha_3 = 0$
0	1	0	1	$\alpha_2 = 0$
0	1	1	1	$\alpha_{2,3} = 0$
1	0	0	0	$\alpha_1 = 1$
1	0	1	1	$\alpha_{1,3} = 1$
1	1	0	1	$\alpha_{1,2} = 1$
1	1	1	0	$\alpha_{1,2,3} = 0$

Таким образом $f(x, y, z) = 1 \oplus x \oplus xy \oplus xz$ и функция f нелинейна.

Построим функцию $\varphi(x, y) = x \wedge y$. В нашем случае коэффициент при xy не равен нулю; выделим в полиноме Жегалкина переменные x и y.

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

Тогда можно положить $z = 0$ и в терминах леммы 2.1.24 получим, что $\beta = 1$, $\gamma = 0$, $\delta = 1$ и

$$\varphi(x, y) = f(x \oplus \gamma, y \oplus \beta, 0) \oplus \gamma\beta \oplus \delta = \neg f(x, \overline{y}, 0)$$

Проверим, что φ соответствует конъюнкции:

$$\varphi(x, y) = \overline{(x \supset (\overline{y} \oplus 0))} = \overline{x \supset \overline{y}} = \overline{\overline{x} \vee \overline{y}} = x \wedge y.$$

2.1.13 Критерий полноты системы функций

Итак, мы рассмотрели пять классов функций T_0, T_1, S, M, L.

	T_0	T_1	S	M	L
$\neg x$	$-$	$-$	$+$	$-$	$+$
0	$+$	$-$	$-$	$+$	$+$
1	$-$	$+$	$-$	$+$	$+$
xy	$+$	$+$	$-$	$+$	$-$

Каждый из этих классов функций замкнут и, как можно видеть из таблицы, ни один не совпадает с P_2.

Теорема 2.1.25 (Теорема Поста). *Для полноты системы функций $\mathcal{P} \subseteq P_2$ необходимо и достаточно, чтобы $\mathcal{P}$ не лежал полностью ни в одном из классов T_0, T_1, S, M, L:*

$$\mathcal{P} \not\subseteq T_0, \quad \mathcal{P} \not\subseteq T_1, \quad \mathcal{P} \not\subseteq S, \quad \mathcal{P} \not\subseteq M, \quad \mathcal{P} \not\subseteq L.$$

Доказательство. Необходимость. Если система функций $\mathcal{P}$ лежит полностью в одном из классов $R \in \{T_0, T_1, S, M, L\}$, то, поскольку все эти классы замкнуты и не совпадают с P_2, $[\mathcal{P}] \subseteq [R] \neq P_2$. Тогда система $\mathcal{P}$ — не является полной.

Докажем достаточность. Пусть $f_0, f_1, f_S, f_M, f_L \in \mathcal{P}$ такие функции, что $f_0 \notin T_0$, $f_1 \notin T_1$, $f_S \notin S$, $f_M \notin M$, $f_L \notin L$ (некоторые из функций могут совпадать). Проведем доказательство в несколько этапов.

1) Покажем, что с помощью f_0, f_1, f_S можно получить 0 и 1.

a) Пусть $f_0(1, ..., 1) = 1$. Пусть $\varphi(x) = f_0(x, ..., x)$. Тогда $\varphi(0) = \varphi(1) = 1$. Значит $\varphi(x) = 1$ и, имея единицу, можно получить вторую константу $0 = f_1(1, ..., 1)$.

b) Пусть теперь $f_0(1, ..., 1) = 0$. Тогда $\varphi(x) = f_0(x, ..., x) = \overline{x}$. Подставляя в f_S x и $\overline{x}$ по лемме о несамодвойственной функции получаем константу 0 или 1 и с помощью $\overline{x}$ получаем вторую константу.

2) По лемме о немонотонной функции, подставляя константы в f_M можно получить $\neg x$.

3) Используя f_L, константы и $\neg x$, по лемме о нелинейной функции можно получить $x \wedge y$.

Так как $\{\neg, \wedge\}$ — полная системя функций, то и система $\mathcal{P}$ — полная. $\square$

Пример 2.1.29 . *Требуется проверить на полноту систему функций* $\mathcal{P} = \{0,\ 1,\ xy,\ x \oplus y \oplus z\}$. *Рассмотрим принадлежность функций* $\mathcal{P}$ *классам* T_0, T_1, S, M, L *и заполним таблицу.*

	T_0	T_1	S	M	L
0	+	−	−	+	+
1	−	+	−	+	+
xy	+	+	−	+	−
$x \oplus y \oplus z$	+	+	+	−	+

Рассмотрим, например, проверку функции $x \oplus y \oplus z$*:*

a) $0 \oplus 0 \oplus 0 = 0 \quad \Rightarrow \quad x \oplus y \oplus z \in T_0$*;*

b) $1 \oplus 1 \oplus 1 = 1 \quad \Rightarrow \quad x \oplus y \oplus z \in T_1$*;*

c) $\overline{\overline{x} \oplus \overline{y} \oplus \overline{z}} = 1 \oplus (1 \oplus x) \oplus (1 \oplus y) \oplus (1 \oplus z) = x \oplus y \oplus z \quad \Rightarrow \quad x \oplus y \oplus z \in S$*;*

d) $(1,0,0) \prec (1,1,0)$*, но* $1 = 1 \oplus 0 \oplus 0 > 1 \oplus 1 \oplus 0 = 0 \quad \Rightarrow \quad x \oplus y \oplus z \notin M$*;*

e) Очевидно, функция является линейной: $x \oplus y \oplus z \in L$.

Теперь, заполнив и проанализировав таблицу, можно убедиться, что система функций $\mathcal{P}$ *является полной, так как в каждом столбце, соответствующем одному из классов присутствует хотя бы один минус. В то же время ни одно подмножество* $\mathcal{P}$ *полной системой не является, поскольку, если вычеркнуть в таблице хотя бы одну строку, появится столбец не имеющий минуса.*

Определение 2.1.28 . *Пусть* $\mathfrak{M}$ *— замкнутый класс функций. Пусть* $\mathcal{B} \subseteq \mathfrak{M}$*.* $\mathcal{B}$ *называется базисом класса* $\mathfrak{M}$*, если*

1) $[\mathcal{B}] = \mathfrak{M}$*;*

2) $\forall f \in \mathcal{B} \quad \Rightarrow \quad [\mathcal{B} \setminus \{f\}] \neq \mathfrak{M}$.

Пример 2.1.30 . *1) Система из примера 2.1.29 является базисом* P_2.

2) Система $\{0, 1, xy, x \oplus y\}$ *полная, но базисом* P_2 *не является.*

Базисом P_2 будет ее подсистема $\{1, xy, x \oplus y\}$.

	T_0	T_1	S	M	L
0	$+$	$-$	$-$	$+$	$+$
1	$-$	$+$	$-$	$+$	$+$
xy	$+$	$+$	$-$	$+$	$-$
$x \oplus y$	$+$	$-$	$-$	$-$	$+$

3 Теория алгоритмов

3.1 Машины Тьюринга

3.1.1 Понятие алгоритма

Интуитивно алгоритм можно рассматривать как последовательность действий, которая обладает рядом свойств.
1) Входные данные. Поскольку алгоритм решает некоторую общую задачу, он обычно получает некоторые инициирующие данные, в зависимости от которых выполняется.
2) Выходные данные. Любой алгоритм должен давать какой-то результат от своего выполнения.
3) Определенность. В каждый момент времени мы должны точно знать, какое действие выполнить следующим.
4) Элементарность. Это свойство значит, что каждый шаг алгоритма должен быть достататочно прост, чтобы его выполнение не требовало разбиения на меньшие шаги.
5) Конечность. Чтобы выдать результат алгоритм обычно должен завершить свою работу.

Для того, чтобы дать строго формальное определение алгоритма, введем понятие машины Тьюринга.

3.1.2 Машина Тьюринга

Определение 3.1.1 . *Машиной Тьюринга называется совокупность пяти объектов* $< A, \square, Q, q_1, I >$*:*

1. Алфавит $A = \{a_1, ..., a_n\}$ *— множество символов данной машины Тьюринга;* $|A| \geq 1$*. Иногда алфавит называют множеством внешних состояний машины Тьюринга.*

2. Пустой символ $\square \notin A$*.*

3. Множество состояний $Q = \{q_1, ..., q_k\}$*,* $|Q| \geq 1$*. В некоторых случаях это множество также называют множеством внутренних состояний машины Тьюринга.*

4. Начальное состояние $q_1 \in Q$*.*

5. Программа I *— множество команд вида* $\mathbf{q_i a} \rightarrow \mathbf{q_j b}$*, где* $q_i, q_j \in Q$*,*

$a \in \{\square\} \cup A$, $b \in \{\square, L, R\} \cup A$. *Здесь* L *и* R *выделенные символы, такие что* $L, R \notin \{\square\} \cup A$. *Кроме того, для любых* $q_i \in Q$ *и* $a \in \{\square\} \cup A$ *в программе* I *может быть не больше одной команды* $q_i a \to q_j b$ *(иными словами, в* I *нет двух команд с одинаковой левой частью).*

Процесс функционирования машины Тьюринга. Машина Тьюринга (рис. 15) состоит из бесконечной в обе стороны ленты, разбитой на одинаковые ячейки, а также читающе-пишущей головки. В ячейках ленты могут быть записаны символы алфавита или пустой символ.

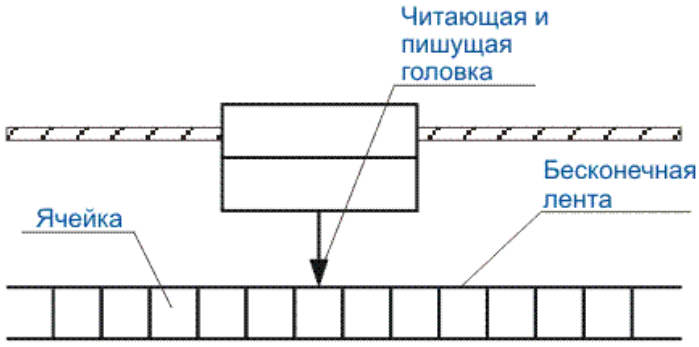

Рисунок 15: Машина Тьюринга

Работа машины Тьюринга понимается следующим образом (рис. 16). В начальный момент времени на ленте записано конечное слово α в

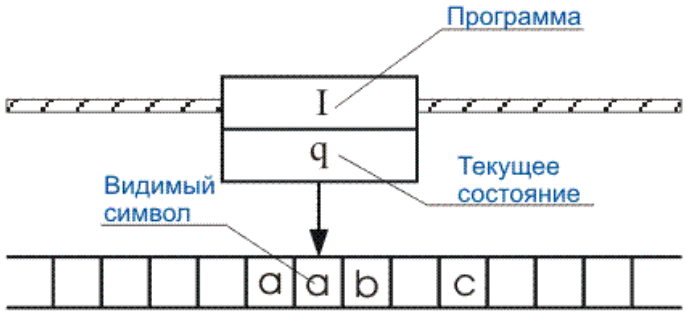

Рисунок 16: Функционирование машины Тьюринга.

алфавите A (или несколько слов разделенных пустым символом $\square$) окруженное слева и справа бесконечным количеством пустых символов $\square$; машина Тьюринга находится в состоянии q_1 и ее головка обозревает самую левую ячейку слова α. Пусть там записан символ a_{i_1}. В I ищется команда с $q_1 a_{i_1}$ в левой части. Если такая команда не найдется, машина останавливается. Если команда найдется и это $q_1 a_{i_1} \to q' a_{i'}$, машина переходит в состояние q' и выполняется одно из действий в зависимости от значения $a_{i'}$: если $a_{i'} \in A \cup \{\square\}$, в обозреваемую машиной ячейку записывается символ $a_{i'}$; если $a_{i'} \in \{L, R\}$, машина передвигается на одну ячейку влево, при $a_{i'} = L$, или на одну ячейку вправо, при $a_{i'} = R$.

В каждый следующий момент времени, если машина Тьюринга находится в состоянии q_i и обозревает некоторую ячейку, в которой записан символ $a \in \{\square\} \cup A$, в программе ищется команда $q_i a \to q_j b$. Если такая команда не будет найдена, машина останавливается. Если команда найдется, выполняются соответствующие действия, как для первого шага работы.

Если машина Тьюринга останавливается, то говорят, что эта машина *принимает* слово α, а оставшееся на ленте слово β (или несколько слов, разделенных пустым символом) считается результатом ее работы.

Замечание 3.1.1 . *Можно заметить, что машина Тьюринга удовлетворяет всем пунктам интуитивного определения алгоритма:*
1) Входные данные. Входными данными машины Тьюринга является информация на ленте перед началом работы.
2) Выходные данные. Результатом работы машины Тьюринга является содержание ленты после ее остановки.
3) Определенность. Условие на программу, не допускающее двух команд с одинаковой левой частью, гарантирует, что в любой момент времени у нас будет не более одной подходящей команды для следующего шага.
4) Элементарность. Все возможные действия машины Тьюринга сводятся к сдвигам головки на одну позицию, а также чтению/записи символа из/в текущую ячейку.
5) Конечность. По своему определению машина Тьюринга может не остановиться. Отвественность за конечность ее работы лежит на программе.

Пример 3.1.1 . *Машина* M_3. $A = \{1\}, \quad Q = \{q_1, q_2, q_3\}$.

$$
\begin{aligned}
I: \quad & q_1\square \to q_1 1 \\
& q_1 1 \to q_2 L \\
& q_2\square \to q_2 1 \\
& q_2 1 \to q_3 L \\
& q_3\square \to q_3 1
\end{aligned}
$$

Пусть, в начале работы на ленте записано слово Λ *— пустое слово (все ячейки на ленте содержат пустой символ). Тогда в по окончании работы на ленте будет слово* 111.

Выпишем конфигурации, в которых будет находится машина M_3 *в процессе своей работы (рис. 17).*

			q_1		
$\square$	$\square$	$\square$	$\square$	$\square$	— начальная конфигурация
			q_1		
$\square$	$\square$	$\square$	1	$\square$	
		q_2			
$\square$	$\square$	$\square$	1	$\square$	
		q_2			
$\square$	$\square$	1	1	$\square$	
	q_3				
$\square$	$\square$	1	1	$\square$	
	q_3				
$\square$	1	1	1	$\square$	— конечная конфигурация

Рисунок 17: Конфигурации в процессе рабоы машины M_3

Очевидно, что аналогично можно построить машину M_n *с* n *состояниями, которая будет строить на пустой ленте слово из* n *единиц.*

3.1.3 Способы записи машины Тьюринга

Для однозначного определения машины Тьюринга достаточно задать ее программу. Действительно, если мы сравним две машины с одинаковой

программой, то окажется, что они могут отличаться только элементами из множеств A и Q какой-то из машин, которые не используются ни в одной из команд. В таком случае, это символы алфавита, на которые данная машина никак (за исключением остановки) не отреагирует, или состояния, в которые она никогда не перейдет. С точки зрения обработки входных данных, между такими машинами не будет никакой разницы.

Прямой записью машины Тьюринга будет простое перечисление команд программы через запятую или в столбец, как мы это сделали в примере 3.1.1. Этот способ не очень удобен для понимания структуры программы и поиска нужной команды.

Более удобным представлением может оказаться таблица $(n+1) \times k$, в которой по вертикали перечисляются символы из $A \cup \{\square\}$, а по горизонтали состояния машины. В ячейке в строке, соответствующей символу a, и столбце, соответствующем состоянию q_i записывается $q_j b$, если в программе данной машины есть команда $q_i a \to q_j b$.

Например, таблица для программы из примера 3.1.1 будет выглядеть следующим образом:

	q_1	q_2	q_3
$\square$	$q_1 1$	$q_2 1$	$q_3 1$
1	$q_2 L$	$q_3 L$	

При этом представлении гораздо проще быстро определить, какая команда должна выполняться в следующий момент. Кроме того, из таблицы очевидно, машина может остановиться в том и только в том случае, если она в какой-то момент окажется в состоянии q_3 и ее головка будет обозревать символ 1.

Наиболее точно представляет структуру работы машины Тьюринга *граф переходов*. Граф переходов для машины из примера 3.1.1 представлен на рисунке 18. Каждая вершина такого графа соответствует одному из состояний машины Тьюринга, а каждая стрелка — команде программы этой машины.

Например, команде $q_1\square \to q_1 1$ соответствует стрелка от вершины q_1 к ней же самой, а над стрелкой указана замена символа $\square$ на символ 1: "$\square \ : \ 1$"; команде $q_1 1 \to q_2 L$ соответствует стрелка от вершины q_1 к вершине q_2 с пометкой "$1 \ : \ L$".

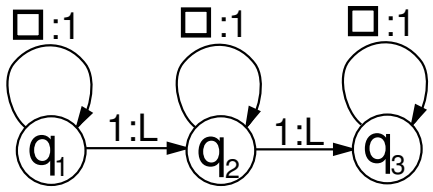

Рисунок 18: Граф переходов машины M_3

3.1.4 Стандартные конфигурации

Пусть M — машина Тьюринга с алфавитом A и множеством состояний Q.

Определение 3.1.2 . *Пусть* $..., \gamma_{-3}, \gamma_{-2}, \gamma_{-1}, \gamma_0, \gamma_1, \gamma_2, \gamma_3, ...$ *— символы на ленте* M.

Пусть существуют $i, j: \quad i \leq j, \ \gamma_i, \gamma_j \in A$, *и* $\gamma_k = \square$, *при* $k < i$ *или* $k > j$. *Тогда* $\gamma = \gamma_i, \gamma_{i+1}, ..., \gamma_j$ *— слово на ленте машины* M.

Если для любых i *имеем* $\gamma_i = \square$, *то говорят, что на ленте записано пустое слово:* $\gamma = \Lambda$.

Определение 3.1.3 . *Расширением слова на ленте называется запись* $\delta\gamma\sigma$, *где* γ *— слово на ленте,* δ *и* σ *слова конечной или нулевой длины из символа* $\square$.

Определение 3.1.4 . *Конфигурация, в которой находится машина Тьюринга в определенный момент времени, может быть представлена записью вида*

$$\tau_k, \tau_{k+1}, ..., \tau_{l-1}, q, \tau_l, ..., \tau_m,$$

где $\tau_k, \tau_{k+1}, ..., \tau_m$ *— расширение слова на ленте,* q *— состояние, в котором находится машина* M, *а* l *— номер ячейки, которую обозревает головка машины.*

Определение 3.1.5 . *Стандартная начальная конфигурация имеет вид* $\square q_1 \alpha \square$, *где* α *— слово на ленте.*

Определение 3.1.6 . *Стандартная конечная конфигурация имеет вид* $\square q\beta\square$, *где* $q \in Q$, β — *слово на ленте;* $\beta = \Lambda$ *или* $\beta = \beta_i, ..., \beta_j$, $\beta_s \neq \square$, *при* $i \leq s \leq j$.

Определение 3.1.7 . *Длина слова на ленте* γ —

$$L(\gamma) = \begin{cases} j - i + 1, & \gamma = \gamma_i, ..., \gamma_j, \\ 0, & \gamma = \Lambda. \end{cases}$$

Определение 3.1.8 . *Пусть,* M — *машина Тьюринга.* $M(\alpha)$ — *конфигурация, в которой останавливается машина Тьюринга* M, *запущенная из стандартной начальной конфигурации* $\square q_1\alpha\square$.

$M(\alpha)$ *неопределена, если* M *не остановится.*

Определение 3.1.9 . *Говорят, что машина Тьюринга* M *принимает слово* α, *если* $M(\alpha)$ — *стандартная конечная конфигурация.*

3.1.5 Вычислимые функции

Далее в нашем курсе будем рассматривать только машины Тьюринга с алфавитом $A = \{1\}$. Каждой такой машине M соответствует частичная функция $f_M : \mathbb{N}_0 \to \mathbb{N}_0$.

Определение 3.1.10 . *Пусть,* M — *машина Тьюринга. Тогда, для любого* $i \in \mathbb{N}_0$,

$$f_M(i) = \begin{cases} L(\beta), & \text{если } M(\underbrace{11...1}_{i}) = \square q\beta\square \text{ — стандартная} \\ & \text{конечная конфигурация;} \\ \text{неопред.}, & \text{иначе.} \end{cases}$$

Пример 3.1.2 . *Машина* $M_{\times 2}$. $A = \{1\}, \quad Q = \{q_1, q_2, ..., q_9\}$.

$\alpha = 111...1$ — *слово из* n *единиц.* $\beta = 111...1$ — *слово из* $2n$ *единиц.*

Программа машины $M_{\times 2}$ *представлена графом переходов на рисунке 19.*

Выпишем конфигурации, в которых будет находиться машина $M_{\times 2}$ *в процессе своей работы, если* $\alpha = 1$ *(рисунок 20). Результатом работы оказывается слово* 11, *что соответствует удвоенному входному слову.*

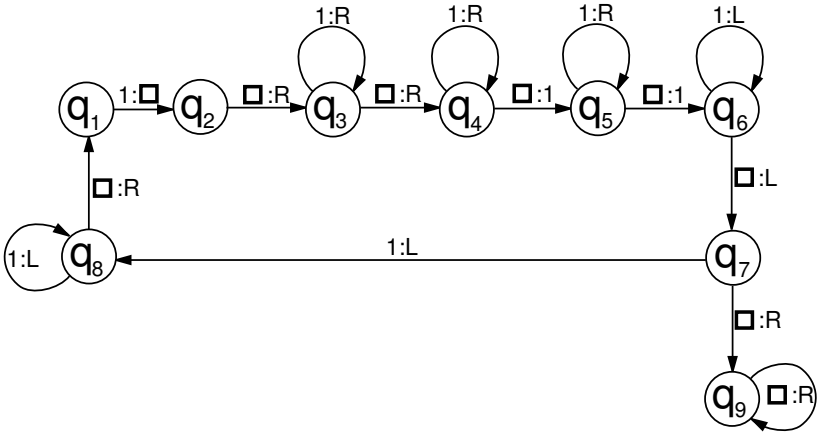

Рисунок 19: Граф переходов машины удвоения

□	$q_1 1$	□	□	□	□
□	$q_2 □$	□	□	□	□
□	□	$q_3 □$	□	□	□
□	□	□	$q_4 □$	□	□
□	□	□	$q_5 1$	□	□
□	□	□	1	$q_5 □$	□
□	□	□	1	$q_6 1$	□
□	□	□	$q_6 1$	1	□
□	□	$q_6 □$	1	1	□
□	$q_7 □$	□	1	1	□
□	□	$q_9 □$	1	1	□
□	□	□	$q_9 1$	1	□

Рисунок 20: Конфигурации в процессе работы машины $M_{\times 2}$

Работа машины $M_{\times 2}$ на входном слове из двух и более единиц рассматривается аналогично.

Очевидно, $f_{M_{\times 2}}(i) = 2i$ для любых $i \in \mathbb{N}_0$.

Определение 3.1.11 . *Функция $f : \mathbb{N}_0 \to \mathbb{N}_0$ называется вычислимой, если существует такая машина Тьюринга M, что $f = f_M$. В противном случае, f — невычислимая функция.*

Утверждение 3.1.1 (О существовании невычислимых функций). *Существует невычислимая функция $f : \mathbb{N}_0 \to \mathbb{N}_0$.*

Доказательство. 1) Для начала покажем, что множество машин Тьюринга счетно.

Во-первых, множество машин Тьюринга не может быть конечным. Мы ранее упомянали, что для любого $n \in \mathbb{N}$ можно построить машину M_n, с n состояниями, которая рисует n единиц, начиная работу с пустой ленты. Если мы построим последовательность машин Тьюринга M_1, M_2, ..., M_n, ..., которая будет иметь счетную длину. Значит, множество машин Тьюринга не менее, чем счетно.

Как было замечено ранее в параграфе 3.1.3, машина Тьюринга практически полностью определяется своей программой. Значит, чтобы пересчитать различные машины Тьюринга, достаточно пересчитать различные программы.

Сопоставим любому множеству команд I, отвечающих определению программы машины Тьюринга число как описано ниже.

Рассмотрим алфавит $X = \{\square, 1, *, L, R\}$. Пусть $\square < 1 < * < L < R$. Тогда на множестве слов алфавита X можно определить строгий линейный (например, лексикографический) порядок. Пронумеруем все слова алфавита X следующим образом: сначала по порядку идут все слова алфавита X длины 1, следующие номера получают в том же порядке слова длины 2, затем — длины 3 и так далее. Так мы пронумеруем натуральными числами все слова алфавита X. Каждой команде программы $q_i a \to q_j b$ сопоставим слово

$$\underbrace{*\ldots*}_{i} a \underbrace{*\ldots*}_{j} b$$

в алфавите X. Такое слово однозначно описывает команду. Действительно, для определения состояний нам достаточно знать их номера (никакая другая информация об элементах множества Q машиной Тьюринга не используется), а для любых $a \in \{\square\} \cup A = \{\square, 1\}$, $b \in \{\square, L, R\} \cup A = \{\square, L, R, 1\}$ у нас имеются подходящие символы в алфавите X.

Программе I можно сопоставить конкатенацию таких слов; очевидно, что разным программам не могут соответствовать совпадающие конкатенации и по такой конкатенации можно однозначно восстановить исходную программу. Но, поскольку команды в программе не упорядочены, одной программе могут соответствовать записи, полученные одна из другой изменением порядка команд. Из всех записей, сопоставленных одной и той же программе будем выбирать ту, номер которой, в выбранной ранее нумерации всех слов алфавита X, будет наименьшим; этот номер сопоставим программе I.

Таким образом, все различные программы машин Тьюринга пронумерованы числами из некоторого подмножества множества натуральных чисел и, следовательно, число машин Тьюринга не более чем счетно.

С другой стороны, множество машин Тьюринга не может быть конечным. Чтобы показать это, достаточно вспомнить про множество машин Тьюринга M_n, $i = \overline{1, \infty}$, из примера 3.1.1, которые рисуют n единичек, начиная работу с пустой ленты. Все эти машины различны и их счетное число. Следовательно число машин Тьюринга счетно.

2) Теперь покажем, что число частичных функций $f : \mathbb{N}_0 \to \mathbb{N}_0$ более чем счетно.

Рассмотрим множество всех частичных функций $\mathbb{N}_0 \to \mathbb{N}_0$:

$$A = \{f \mid f : \mathbb{N}_0 \to \mathbb{N}_0\}$$

и рассмотрим подмножество этого множества

$$B = \{f \mid f : \mathbb{N}_0 \to \{0, 1\},\ f \text{ — всюду определена на } \mathbb{N}_0\}.$$

Если мы докажем, что множество B более чем счетно, то, очевидно, это будет верно и для множества A.

Предположим, что множество B счетно. Тогда все функции из B можно пронумеровать натуральными числами. Пусть тогда, $B = \{f_1, f_2, f_3, ...\}$.

Рассмотрим функцию f, определенную следующим выражением:

$$f(i) = 1 - f_i(i), \quad i = \overline{1, \infty}. \tag{31}$$

По определению это функция $f : \mathbb{N}_0 \to \{0, 1\}$ и всюду определена на $\mathbb{N}_0$. Тогда должен существовать индекс k, которым функция f пронумерована в множестве B, но из формулы 31 следует, что $f(k) = 1 - f_k(k) \neq f_k$?!

Это противоречие доказывает, что как бы мы не нумеровали элементы B, всегда найдется ненумерованная функция. Следовательно, множество B имеет несчетную мощность, а, значит, и множество A также более чем счетно.

3) Поскольку число машин Тьюринга счетно, а число частичных функций $f : \mathbb{N}_0 \to \mathbb{N}_0$ более чем счетно, мы не можем сопоставить каждой функции свою машину Тьюринга. Таким образом, должны существовать функции, которые не будут являться вычислимыми, что и требовалось доказать.

□

3.1.6 Алгоритмически неразрешимые задачи

В силу интуитивности понятия алгоритма, вопрос о существовании алгоритмического решения некоторого класса задач оставался без ответа. Избавиться от неопределенности позволил тезис Тьюринга-Черча, который устанавливает соответствие между интуитивным понятием алгоритмически разрешимой задачи и понятием машины Тьюринга, которое возможно изучать с формально-математической точки зрения.

Тезис Тьюринга-Чёрча: *Любой алгоритм в интуитивном смысле этого слова может быть представлен эквивалентной машиной Тьюринга.*

Физический тезис Тьюринга-Чёрча: *Любая функция, которая может быть вычислена физическим устройством, может быть вычислена машиной Тьюринга.*

Этот тезис позволяет нам четко определить алгоритмически неразрешимую проблему, как проблему, для которой не существует решающей ее машины Тьюринга. Из теоремы 3.1.1 следует, что существуют функции, которые нельзя вычислить с помощью машин Тьюринга, а значит алгоритмически неразрешимые задачи существуют. Далее в этом параграфе приведем пример такой функции.

Определим функцию "Продуктивность" следующим образом:

$$\text{Продуктивность}(M) = \begin{cases} f_M(0), & f_M(0) \text{ — определена,} \\ 0, & \text{иначе .} \end{cases}$$

Пример 3.1.3 . *Машина Тьюринга на рисунке 21 имеет продуктивность 2, поскольку запущенная на пустой ленте она рисует две единички и останавливается. Машина Тьюринга на рисунке*

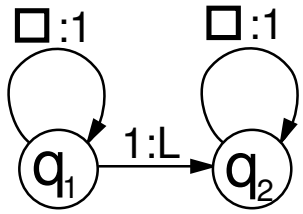

Рисунок 21: Машина с двумя состояниями и продуктивностью 2

22, запущенная на пустой ленте, не останавливается, а рисует бесконечную последовательность единиц. Ее продуктивность — 0.

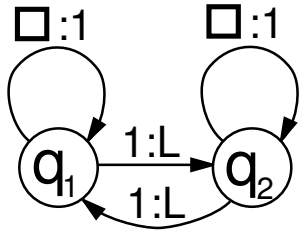

Рисунок 22: Машина с двумя состояниями и продуктивностью 0

Определение 3.1.12 . *Введем функцию максимальной продуктивности машины Тьюринга с n состояниями.*

$$p(n) = \max_{M-\text{ м.Т. с n состояниями}} \textit{Продуктивность}(M), \quad n \in \mathbb{N}.$$

Чтобы получить функцию $\mathbb{N}_0 \to \mathbb{N}_0$, *доопределим функцию* p *в нуле:* $p(0) = 0$.

Лемма 3.1.2 (о свойствах фукнции $p(n)$). *Пусть* $n \in \mathbb{N}_0$. *Тогда*
1) $p(n+1) > p(n)$ *(монотонность),*
2) $p(n+9) \geq 2n$.

Доказательство. 1) Пусть M — такая машина Тьюринга с n состояниями, что Продуктивность$(M) = p(n)$. Построим машину M' на основе машины M согласно рисунку 23, добавляя к машине одно состояние и увеличивая продуктивность на единицу. $M(\Lambda) = \square q_n \underbrace{1\ldots1}_{p(n)} \square$,

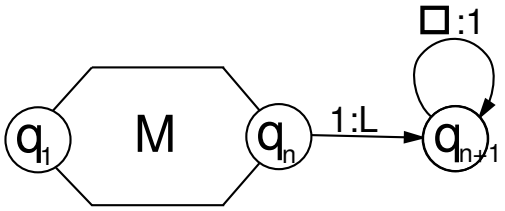

Рисунок 23: Машина M'

а машина M' содержит дополнительное состояние и две новые команды $q_n 1 \to q_{n+1} L$ и $q_{n+1}\square \to q_{n+1}1$ которые сдвигают головку на одну ячейку влево и дорисовывают дополнительную единицу, так что $M'(\Lambda) = \square q_{n+1} \underbrace{1\ldots1}_{p(n)+1} \square$. Мы получили машину Тьюринга M' с $n+1$ состоянием и продуктивностью $p(n)+1$. Следовательно

$$p(n+1) \geq \text{Продуктивность}(M') = p(n) + 1 > p(n).$$

2) Построим машину M'' на основе машин M_n из примера 3.1.1 и $M_{\times 2}$ из примера 3.1.2 согласно рисунку 24: машина M_n рисует n единиц а

машина $M_{\times 2}$ удваетвает это число. Поскольку у машины M_n n состояний, а у $M_{\times 2}$ — 9 состояний, то у машины M'' $n+9$ состояний. Таким образом, $M''(\Lambda) = \square q_{n+9} \underbrace{1...1}_{2n} \square$. Тогда

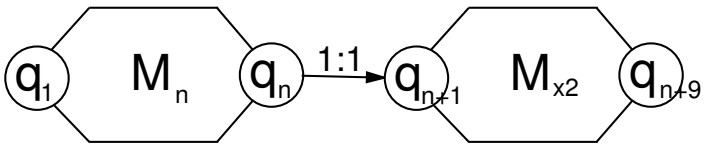

Рисунок 24: Машина M''

$$p(n+9) \geq \text{Продуктивность}(M'') = 2n.$$

□

Теорема 3.1.3 (об алгоритмической неразрешимости задачи вычисления функции $p(n)$). *Не существует машины Тьюринга T, для которой $f_T(n) = p(n)$ для всех $n \in \mathbb{N}_0$.*

Доказательство. Пусть такая машина T существует и пусть у нее k состояний. Построим машину M с $n + 2k$ состояниями таким образом, чтобы ее продуктивность была равна $p(p(n))$:

$$M(\Lambda) = \square q_{n+2k} \underbrace{1...1}_{p(p(n))} \square.$$

Машина M получается путем соединения машины M_n с n состояниями из примера 3.1.1 с двумя копиями машины T (рис. 25); всего у машины M $n + 2k$ состояний. Проверим, что продуктивность такой машины $p(p(n))$. Сначала на пустом слове работает машина M_n и рисует на ленте n единичек:

$$M_n(\Lambda) = \square q_n \underbrace{1...1}_{n} \square.$$

Затем работает копия машины T с состояниями $q_{n+1}, ..., q_{n+k}$. Получая в качестве аргумента n, машина вычисляет функцию $p(n)$:

$$T(\square \underbrace{1...1}_{n} \square) = \square q_{n+k} \underbrace{1...1}_{p(n)} \square.$$

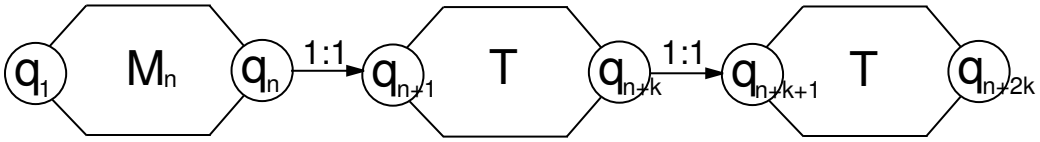

Рисунок 25: Машина с продуктивностью $p(p(n))$

Последней работает копия машины T с состояниями $q_{n+k+1}, ..., q_{n+2k}$, вычисляя максимальную продуктивность от своего аргумента:

$$T(\square q_{n+k} \underbrace{1...1}_{p(n)} \square) = \square q_{n+2k} \underbrace{1...1}_{p(p(n))} \square.$$

Следовательно, искомая машина M построена.

По построению, у машины M $n + 2k$ состояний. Следовательно,

$$p(p(n)) = \text{Продуктивность}(M) \leq p(n + 2k).$$

Тогда, $p(n) \leq n + 2k$ по первому пункту леммы 3.1.2. Так как последнее неравенство верно для любого натурального n, то оно верно и для $n + 9$: $p(n + 9) \leq n + 2k + 9$. Из второго пункта леммы 3.1.2 следует, что

$$2n \leq p(n + 9) \leq n + 2k + 9.$$

Следовательно, $n \leq 2k + 9$, что противоречит произвольности выбора номера n, так как $2k + 9$ заведомо известная константа. Противоречие доказывает невозможность существования машины Тьюринга T, вычисляющей функцию максимальной продуктивности.
□

Задача 3.1.1 (проблема остановки машины Тьюринга). *Пусть $N(M)$ — номер машины Тьюринга M согласно нумерации, использованной в доказательстве утверждения 3.1.1. Существует ли такая машина Тьюринга M_s, что для любой машины Тьюринга M*

$$M_s(\underbrace{1...1}_{N(M)}) = \begin{cases} \square q1\square, & M(\Lambda) \text{ — } \textit{определено}, \\ \square q\square, & M(\Lambda) \text{ — } \textit{неопределено}. \end{cases}$$

Следствие 3.1.4 . *Проблема остановки машины Тьюринга алгоритмически неразрешима.*

Доказательство этого факта основывается на теореме 3.1.3. Его идея сводится к тому, что если бы существовала машина, способная определить остановится ли произвольная машина Тьюринга, нам удалось бы построить и машину Тьюринга, вычисляющую максимальную продуктивность машины Тьюринга с n состояниями.

Смысл алгоритмической неразрешимости проблемы остановки машины Тьюринга состоит в том, что не может существовать алгоритма, который, получая на вход описание произвольного алгоритма, мог бы определить, завершит ли последний работу за конечное время, или его выполнение будет зациклино.

Можно привести другие примеры. Например, алгоритмически неразрешимой является задача о выводимости формулы в исчислении предикатов.

Задача 3.1.2 . *Пусть $\mathcal{A}$ — формула исчисления предикатов. Выводима ли $\mathcal{A}$ в исчислении предикатов?*

3.2 Теория NP-полных задач

3.2.1 Сложность алгоритма

Когда идет речь о сложности алгоритма, обычно имеется в виду скорость выполнения алгоритма (временная сложность) или требуемая для его работы память (емкостная сложность). При оценке времени выполнеия, память чаще всего считается неограниченной и не оценивается. Дальше будем говорить в основном о скорости выполнения, но практически все наши определения и рассуждения будут аналогичны и для оценки емкостной сложности алгоритмов.

Время выполнения или требуемая память для одного и того же алгоритма будет разниться в зависимости от конкретных входных данных. Оценить сложность в этом смысле безотносительно параметров невозможно. С другой стороны оценка сложности для каждого набора входных параметров отдельно будет слишком трудоемкой. По-этому, необходимо разделить задачу на группы частных случаев, чтобы оценить сложность для каждой такой группы.

Определение 3.2.1 . *Массовая задача — задача с параметрами.*

Определение 3.2.2 . *Индивидуальная задача — это массовая задача, для всех параметров которой заданы конкретные значения.*

Пример 3.2.1 . *Например, задача сортировки массива длины n, где не указаны n и значения элементов массива является массовой.*

Задача отсортировать массив $\{8, 10, 4, 5, 1, 3, 7\}$ *является индивидуальной задачей к массовой задаче сортировки.*

Для оценки скорости выполнения имеет значение, на каком именно вычислительном устройстве работает алгоритм. В теории сложности алгоритмов различают машины Тьюринга с одной лентой, с k лентами и с произвольным доступом к памяти. Они имеют разные возможности и, следовательно, алгоритмы и время их работы на этих машинах будет различной.

Кроме того, для выбранной машины, реальная длительность выполения данного алгоритма может зависить от быстродействия

машины (длительности выполнения доступных ей операций), что затрудняет использование полученной оценки. По этой причине оценивают не время по абсолютной величине, а число элементарных операций, которые необходимо выполнить при работе данного алгоритма на машине выбранного типа.

Замечание 3.2.1 . *На практике не так часто исследуют алгоритм в применении к машинам Тьюринга. Чаще рассматривают возможности некоего обобщенного устройства, понимающего язык высокого уровня. При этом важно помнить, что одна команда высокого уровня может требовать большого количества элементарных операций вычислительной системы.*

Также обычно не считают все элементарные операции алгоритма. Обычно выбирается множество ключевых операций, которые оказывают наибольшее влияние на оцениваемую величину. Например, сравнение чисел, обращения к медленной памяти и т.п.

Определение 3.2.3 . *Пусть заданы следующие объекты:*
1) массовая задача;
2) класс индивидуальных задач данной массовой задачи;
3) алгоритм, решающий эту задачу;
4) тип машины, на которой будет работать этот алгоритм;
5) ключевые операции, играющие в алгоритме основную роль.

Сложность алгоритма на этом классе индивидуальных задач — максимальное число выделенных ключевых операций, которые выполняет данный алгоритм при решении индивидуальной задачи из выделенного класса; то есть число ключевых операций, которые необходимо выполнить в худшем случае.

Существуют различные подходы к тому, как выбрать класс индивидуальных задач. Первый способ состоит в том, чтобы зафиксировать один (или несколько) из параметров массовой задачи и разбить ее на такие классы, что в каждый класс входят все задачи с заданным значением этого (этих) параметра. Остальные параметры меняются произвольно в допустимых для них областях. Тогда, сложность алгоритма — функция $T(n)$.

Пример 3.2.2 . *Например, массовую задачу сортировки, обычно, разбивают на классы индивидуальных задач по параметру n — длине сортируемого массива. Конкретным классом индивидуальных задач в этом случае могли бы быть все задачи сортировки с массивами длины $n = 7$.*

Второй подход состоит в том, чтобы закодировать условия каждой индивидуальной задачи некоторым "разумным" способом и выбрать в качестве критерия разбиения на классы индивидуальных задач длину N слов, кодирующих задачи, — то есть включать в один класс все индивидуальные задачи, длины кодов которых не превосходят N. Величины сложности на таких классах будут описывать функцию сложности $T(N)$.

Разумность кодирования здесь подразумевает выбор способа исходя из здравого смысла. Например, не стоит кодировать числа в единичной системе исчисления.

В общем случае и величины фиксированных параметров и длина кода описания задачи характеризуют ее размер и сложность задается функцией от размера.

Часто сложность алгоритма не имеет точного представления формулой над хорошо известными функциями. В этом случае для предъявления функции сложности пришлось бы предъявлять бесконечную таблицу значений. Тогда имеет смысл оценить сложность более простой функцией, например, полиномом некоторой степени.

Введем следующие обозначения (O-символика):
Будем писать $f(n) = O(g(n))$, если существуют $n_0 \in \mathbb{N}$ и $c \in \mathbb{R}_+$:

$$f(n) \leq c \cdot g(n), \quad \forall n \geq n_0.$$

Будем писать $f(n) = \Omega(g(n))$, если существуют $n_0 \in \mathbb{N}$ и $c \in \mathbb{R}_+$:

$$f(n) \geq c \cdot g(n), \quad \forall n \geq n_0.$$

Будем писать $f(n) = \Theta(g(n))$, если существуют $n_0 \in \mathbb{N}$ и $c_1, c_2 \in \mathbb{R}_+$:

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n), \quad \forall n \geq n_0.$$

Известны следующие свойства: Пусть f и g — функции от натурального аргумента. Тогда

1) $f(n) = O(f(n))$;
2) $c \cdot O(f(n)) = O(f(n))$, где c - константа;
3) $O(f(n)) + O(f(n)) = O(f(n))$;
4) $O(O(f(n))) = O(f(n))$;
5) $O(f(n)) \cdot O(g(n)) = O(f(n) \cdot g(n))$;
6) $O(f(n) \cdot g(n)) = f(n) \cdot O(g(n))$.

Если $f(n) = O(g(n))$ будем говорить, что функция $f(n)$ имеет порядок $O(g(n))$.

В реальных ситуациях обычно достаточно выяснить вид функции, наиболее точно ограничивающей сложность алгоритма решения некоторой массовой задачи: полином, логарифм, экспонента и т.п. Если сложность имеет порядок полинома, чаще всего достаточно указать только старшую его степень. Если сложность оценивается экспонентой a^n, важно основание степени — a.

Пример 3.2.3 . *Известны алгоритмы решения задачи сортировки с оценкой сложности* $O(n \cdot \log n)$.

Какую сложность можно считать практически приемлимой. Рассмотрим следующую таблицу (рисунок 26), на которой представлено сравнение длительности работы алгоритмов разной сложности на некоторой гипотетической машине. По горизонтали отмерены размеры входных параметров, по вертикали — $T(n)$, сложность алгоритмов.

$T(n)$	10	20	30	40	50	60
n	10^{-5} с	$2 \cdot 10^{-5}$ с	$3 \cdot 10^{-5}$ с	$4 \cdot 10^{-5}$ с	$5 \cdot 10^{-5}$ с	$6 \cdot 10^{-5}$ с
n^2	10^{-4} с	0.0004 с	0.0009 с	0.0016 с	0.0025 с	0.0036 с
n^3	10^{-3} с	0.008 с	0.027 с	0.064 с	0.125 с	0.216 с
n^5	0.1 с	3.2 с	24.3 с	1.7 мин	5.2 мин	13.0 мин
2^n	0.001 с	1.0 с	17.9 мин	12.7 дней	35.7 лет	366 стол
3^n	0.059 с	58 мин	6.5 лет	3855 стол	$2 \cdot 10^8$ стол	$1.3 \cdot 10^{13}$ стол

Рисунок 26: Сравнение времени выполнения алгоритмов различной сложности в зависимости от размера параметра.

Как можно видеть, удовлетворительными можно считать только алгоритмы, сложность которых не превосходит некоторого полинома фиксированной (не зависящей от размеров входных параметров задачи) степени. Такие алгоритмы называют полиномиальными. Алгоритмы

экспоненциальной сложности могут быть выполнены за приемлемое время только при небольших размерах входных данных.

Быть может развитие технологий и ускорение вычислительных устройств позволит нам в скором времени выполнять и алгоритмы экспоненциальной сложности. Рассмотрим таблицу на рисунке 27. Там представлены максимальные размеры входных данных гипотетических задач различной сложности, выполняемых за час на одной и той же машине, и на сколько эти размеры могут вырости при ускорении работы компьютеров в 100 и в 1000 раз.

T(n)	Современный компьютер	в 100 раз быстрее	в 1000 раз быстрее
n	N_1	$100 \cdot N_1$	$1000 \cdot N_1$
n^2	N_2	$10 \cdot N_2$	$31.6 \cdot N_2$
n^3	N_3	$4.63 \cdot N_3$	$10 \cdot N_3$
n^5	N_4	$2.5 \cdot N_4$	$3.98 \cdot N_4$
2^n	N_5	$N_5 + 6.64$	$N_5 + 9.97$
3^n	N_6	$N_6 + 4.19$	$N_6 + 6.29$

Рисунок 27: Увеличение размера задачи, выполняемой за фиксированное время при увеличении мощности компьютера.

Видно, что в разы увеличивается только размер входных параметров для полиномиальных алгоритмов. Чтобы они могли быть выполненными в то же время, параметры экспоненциальных алгоритмов можно увеличить лишь на единицы.

3.2.2 Полиномиальная сводимость

Можно столкнуться с разными случаями сравнения алгоритмов по сложности. Если один из алгоритмов имеет полиномиальную сложность, а другой требует, например, экспоненциального количества шагов (памяти), как показано выше, использование экспоненциальных алгоритмов может иметь смысл только для небольших входных параметров задачи. На больших данных любой полиномиальный алгоритм будет лучше любого экспоненциального.

В случаях, когда для двух алгоритмов, решающих одну задачу, известна полиномиальная оценка сложности, имеет смысл находить

наиболее точно прибиженное к величине сложности значение полиномов и сравнивать их сначала по максимальной степени, а если она окажется одинаковой, по коэффициенту перед максимальной степенью.

Принципиально иная ситуация имеет место в том случае, если вопрос о существовании полиномиального алгоритма решения некоторой задачи остается открытым. Тогда в первую очередь нужно выяснить ответ на этот вопрос и потом предпринимать действия по дальнейшей оптимизации. Если ответ положительный, можно пытаться искать наилучший из полиномиальных алгоритмов решения. Если полиномиального алгоритма решения задачи не существует, имеет смысл рассмотреть возможность нахождения "быстрого" алгоритма приближенного решения задачи.

При выяснении существования для задачи полиномиального алгоритма, все оценки можно делать с точностью до полиномиальных преобразований. Например, перед решением задачу можно преобразовать в другую форму, если затраты на этот процесс не превысят полиномиальное время, поскольку это не скажется на выводе о существовании полиномиального алгоритма для исходной задачи.

Также при такой постановке вопроса не важен и тип машины, на которой будет выполняться алгоритм. Таблица на рисунке 28 показывает, какова сложность моделирования машины Тьюринга одного типа с помощью машины другого типа. Рассматриваются машины Тьюринга с одной леной, с k лентами и машины с произвольным доступом к памяти.

Моделируемая МТ A	Моделирующая МТ B		
	МТ 1	МТ k	МТ с п. д. к п.
МТ с 1 лентой	—	$O(T(n))$	$O(T(n)\log T(n))$
МТ с k лентами	$O(T^2(n))$	—	$O(T(n)\log T(n))$
МТ с п. д. к памяти	$O(T^3(n))$	$O(T^3(n))$	—

Рисунок 28: Время моделирования машин Тьюринга.

В любом случае понядок сложности моделирования не больше куба от сложности алгоритма исходной машины. Из этого следует, что, если существует полиномиальный алгоритм решения задачи на машине

Тьюринга одного типа, то полиномиальные алгоритмы для этой задачи существуют для любого типа машин Тьюринга.

Помочь доказать существование или отсутствие полиномиального алгоритма решения некоторой задачи может понятие полиномиальной сводимости.

Определение 3.2.4 . *Пусть имеются две массовые задачи S_1 и S_2. Пусть A_1 — произвольный алгоритм решения задачи S_1. Пусть существуют два алгоритма полиномиальной сложности P_{21} и P_{12}: P_{21} получает на входе описание индивидуальной задачи типа S_2 и преобразует его в описание некоторой индивидуальной задачи типа S_1; P_{12} получает на вход решение задачи типа S_1 и преобразует его в решение задачи типа S_2.*

Если алгоритмы P_{21} и P_{12} таковы, что после преобразования описания индивидуальной задачи S_2 алгоритмом P_{21} в описания индивидуальной задачи S_1, решения полученной задачи S_1 и преобразования полученного решения с помощью P_{12}, мы получим решение исходной индивидуальной задачи S_2, говорят, что задача S_2 полиномиально сводится к задаче S_1, и пишут $S_2 \propto S_1$ (рис. 29).

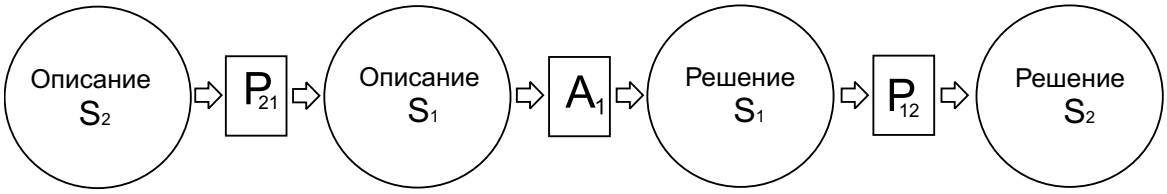

Рисунок 29: Полиномиальная сводимость.

Иными словами, $S_2 \propto S_1$, если связка алгоритмов $A_2 = P_{12}A_1P_{21}$ решает индивидуальную задачу s типа S_2:

$$P_{12}(A_1(P_{21}(\text{условия } s))) = \text{ответ на } s.$$

Пусть $S_2 \propto S_1$. Тогда можно сказать, что задача S_1 в некотором смысле не легче, чем задача S_2. Действительно, если извстно, что у задачи S_1 существует решающий ее полиномиальный алгоритм A_p, то, очевидно, алгоритм $P_{12}A_pP_{21}$, решающий задачу S_2, тоже будет

полиномиальным. С другой стороны, если известно, что для задачи S_2 не существует полиномиального решения, то такого решения не может иметь и задача S_1.

Если задача S_1 не может быть решена за полиномиалное время, это не позволяет сделать выводов о сложности задачи S_2. Также, если за полиномиальное время можно решить задачу S_2, это не позволяет сказать, на сколько эффективно может быть решена задача S_1.

Определение 3.2.5 . *Если $S_2 \propto S_1$ и $S_1 \propto S_2$, говорят, что задачи S_1 и S_2 полиномиально эквивалентны.*

Рассмотрим пример полиномиальной сводимости.

Задача 3.2.1 . *Задача о выполнимости: Дано логическое выражение в конъюнктивной нормальной форме. Является ли функция, реализуемая этой формулой, выполнимой?*

Другими словами, дана булева функция $f(x_1, x_2, ..., x_n)$, определенная своей КНФ: $(D_1) \wedge (D_2) \wedge \cdots \wedge (D_k)$. Требуется ответить на вопрос, существует ли такой набор значений логических переменных a_1, a_2, ..., a_n, что $f(a_1, a_2, ..., a_n) = 1$.

Замечание 3.2.2 . *Подробнее о конъюнктивных нормальных формах булевых функций написано в параграфе 2.1.5.*

Пример 3.2.4 . *Дана конъюнктивная нормальная форма*

$$f(x_1, x_2, x_3, x_4, x_5) = (x_1 \vee \overline{x}_2) \wedge (x_3 \vee \overline{x}_5 \vee \overline{x}_2) \wedge x_4.$$

Можно убедиться, что $f(0, 0, 0, 1, 1) = 1$. Значит ответ на задачу — "Да".

Замечание 3.2.3 . *Очевидно, существует всего 2^n различных наборов аргументов для $f(x_1, x_2, ..., x_n)$. Значит для решения задачи о выполнимости достаточно просто перебрать все варианты. То есть, вопрос о разрешимости тут не стоит, но остается вопрос о возможности эффективного решения задачи.*

Задача 3.2.2 . *Задача о k-выполнимости — это задача о выполнимости, в условиях которой в каждом дизъюнкте не более k литералов.*

Теорема 3.2.1 . *Задача о выполнимости полиномиально сводится к задаче о 3-выполнимости: ВЫП $\propto$ 3-ВЫП.*

Доказательство. Пусть $E(x_1, x_2, ..., x_n) = (D_1) \wedge (D_2) \wedge ... \wedge (D_m)$ — логическое выражение в конъюнктивной нормальной форме. Пусть $D_i = \alpha_1 \vee \alpha_2 \vee ... \vee \alpha_k$ — один из дизъюнктов в E, состоящий не менее, чем из 4 литералов. Заменим в E дизъюнкт D_i на выражение

$$D_i' = (\alpha_1 \vee \alpha_2 \vee z) \wedge (\alpha_3 \vee .... \vee \alpha_k \vee \overline{z}),$$

где z — новая переменная. Обозначим результат $\widehat{E}$.

Покажем, что $E(x_1, x_2, ..., x_n)$ выполнима тогда и только тогда, когда выполнима $\widehat{E}(x_1, x_2, ..., x_n, z)$. Пусть $a_1, a_2, ..., a_n$ — набор аргументов, на котором E принимает значение 1. Тогда все дизъюнкты формы E принимают на этом наборе аргументов значение 1. В частности $D_i(a_1, a_2, ..., a_n) = 1$. Тогда существует $i \in \{1, ..., k\}$: $\alpha_i \Big|_{a_1, a_2, ..., a_n} = 1$. Выберем значение для z следующим образом

$$b = \begin{cases} 0, & i = 1 \text{ или } i = 2; \\ 1, & i \in \{3, ..., k\}. \end{cases}$$

Тогда $\widehat{E}(a_1, a_2, ..., a_n, b) = 1$, то есть $\widehat{E}$ — выполнима.

Пусть теперь E невыполнима: при любых занчениях x_1, ..., x_n, $E(x_1, x_2, ..., x_n) = 0$. Рассмотрим некоторый произвольный набор значений a_1, ..., a_n. Поскольку $E(a_1, a_2, ..., a_n) = 0$, то один из дизъюнктов формы равен нулю. Если $D_j(a_1, a_2, ..., a_n) = 0$ и $j \neq i$, то и $\widehat{E}(a_1, a_2, ..., a_n, z) = 0$ для любого z.

Пусть нулю равен именно i-тый дизъюнкт. Следовательно равны нулю все литералы α_1, α_2, ..., α_k. Если выбрать значение $z = 1$, то $(\alpha_3 \vee .... \vee \alpha_k \vee \overline{1}) = 0$. Если же выбрать значение $z = 0$, то $(\alpha_1 \vee \alpha_2 \vee 0) = 0$. В любом случае $D_i' = 0$. Следовательно $\widehat{E}$ также невыполнима.

Итак, мы заменили исходную конъюнктивную нормальную форму E на эквивалентную ей в плане выполнимости конъюнктивную нормальную

форму $\widehat{E}$, причем один дизъюнкт D_i с $k > 3$ литералами был заменен на два новых дизъюнкта, один из которых содержит три литерала, а другой $k-1$ литерал. Повторяя описанный процесс нужное число раз можно прийти к конъюнктивной нормальной форме, в которой не останется дизъюнктов с более чем тремя литералами.

Сложность: Всего нам придется провести не более $n-3$ циклов уменьшения длины для каждого дизъюнкта. То есть сложность алгоритма имеет порядок $n \cdot m$ — полиномиальна относительно размера входных данных.
□

Следствие 3.2.2 . *Задача о выполнимости полиномиально эквивалентна задаче о 3-выполнимости.*

3.2.3 Классы задач в форме распознавания свойств

Определение 3.2.6 . *Будем говорить, что задача сформулирована в форме распознавания свойств, если ответ на задачу "Да" или "Нет".*

Другими словами, нам дан предикат и набор его параметров; необходимо ответить на вопрос: принимает ли предикат на этих параметрах значение 1 или 0?

Замечание 3.2.4 . *Если нам встретилась задача не в форме распознавания, ее можно переформулировать таким образом, чтобы она попала в наш класс задач.*

Пример 3.2.5 . *Например, рассмотрим следующую задачу:*
Дана контурная карта с обозначенными на ней n странами и требуется раскрасить ее в разные цвета таким образом, чтобы любые два государства, соприкасающиеся участком границы, имели различный цвет.

Задача в оптимизационной форме: Дана контурная карта. Требуется раскрасить ее в минимально возможное число цветов.

Задача переформулированная в форме распознавания свойств: Дана контурная карта и число k. Существует ли допустимая раскраска этой карты в k цветов.

Очевидно, если решена задача в оптимизационной форме, то ответ на переформулированную задачу в форме распознавания будет получен простым сравнением полученного минимального числа цветов со значением k. Следовательно, в форме распознавания задача не становится сложнее исходной задачи.

С другой стороны, если решена задача в форме распознавания, может быть решена задача о нахождении числа цветов в минимальной раскраске. Для ее решения достаточно перебрать все k от 1 до n. То есть, сложность задачи нахождения минимального числа цветов в допустимой раскраске не выше $n \cdot T(n)$, где $T(n)$ — оценка сложности алгоритма решения задачи в форме распознавания. Эта задача полиномиально эквивалентна задаче в форме распознавания.

Рассмотрим два класса задач в форме распознавания свойств, — P и NP, — изучение которых имеет большое значение для теории алгоритмов.

Определение 3.2.7 . *Класс P определяется, как класс всех задач в форме распознавания, для которых существует полиномиальный алгоритм решения.*

Пример 3.2.6 . *К классу P, например относятся переформулированная в форме распознавания свойств задача нахождения минимального элемента массива:*
Дан массив чисел длины n и номер ячейки k. Правда ли, что в k-той ячейке находится минимальный элемент массива. Сложность решения этой задачи $O(n)$

Определение 3.2.8 . *Класс NP (от non-determinstic polynomial) определяется, как класс всех задач в форме распознавания, для которых существует недетерминированная машина Тьюринга, вычисляющая ответ на эту задачу за полиномиальное время.*

Недетерминированная машина Тьюринга отличается от машин Тьюринга, с которыми мы сталкивались ранее (детерминированных), тем, что в ее программе допускается наличие нескольких команд с одинаковой левой частью. Другими словами, в некоторый момент времени недетерминированная машина Тьюринга может оказаться в

конфигурации, из которой у нее определено более, чем один вариант дальнейших действий. Таким образом условие определенности алгоритма здесь не выполняется. В общем случае у недетерминированного алгоритма есть целое дерево различных вариантов хода выполнения.

Что делать, если мы столкнулись с такой развилкой? В таком случае недетерминированная машина может копировать себя необходимое число раз и все копии начинают выполнять каждая свою возможную последовательность действий. Копии машины работают независимо и одновременно (рисунок 30). Как только хотя бы одна копия машины заканчивает вычисления с ответом "Да", все копии завершают свою работу. В случае отрицательного ответа, машина может не остановиться.

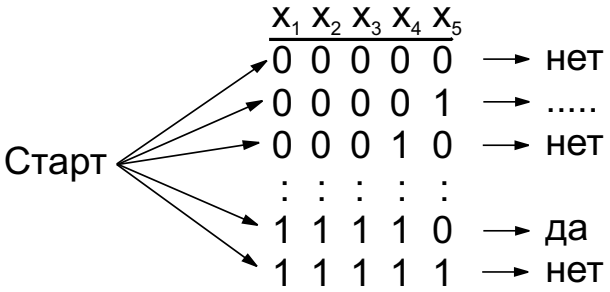

Рисунок 30: Работа недетерминированного алгоритма

Можно также говорить об угадывающем устройстве. Тогда считаем, что подойдя к развилке машина безошибочно угадывает, какой именно переход нужно выбрать, чтобы получить ответ "Да", если это возможно.

Определение 3.2.9 . *Говорят, что недетерминированная машина Тьюринга принимает слово α, если существует хотя бы одна ветвь дерева вычислений, где машина останавливается и дает ответ "Да".*

Можно привести альтернативное определение класса NP

Определение 3.2.10 . *Задача в форме распознавания принадлежит классу NP, если, когда на задачу дан ответ "Да" и приведено доказательство решения — некоторые дополнительные сведения, — можно за полиномиальное время проверить, что это правда.*

Пример 3.2.7 . *Задача о выполнимости лежит в классе NP. Действительно, если для некоторой КНФ дан ответ, что она выполнима, и в подтверждение приведен набор значений аргументов $x_1, x_2, ..., x_n$, то проверить, что значение формулы действительно 1, можно за полиномиальное время, просто вычислив заданную формой функцию.*

Утверждение 3.2.3 . *Определения 3.2.8 и 3.2.10 эквивалентны.*

Доказательство. Если задача удовлетворяет определению 3.2.8, то в качестве доказательства ответа "Да" можно привести номера веток дерева вариантов выполнения, которые нужно выбирать недетерминированной машине, чтобы получить ответ за полиномиальное время.

Пусть задача удовлетворяет определению 3.2.10. Тогда построим недетерминированную машину, которая в начальный момент времени создает необходимое число копий и начнет рассмотрение всех возможных вариантов доказательства правильности ответа. Каждый отдельный процесс будет проводить проверку доказательства за полиномиальное время.

□

Ответы "Да" и "Нет" в случае недетерминированной машины Тьюринга не симметричны.

Пример 3.2.8 . *Рассмотрим задачу о невыполнимости:*

Задача 3.2.3 . *Дано логическое выражение в конъюнктивной нормальной форме. Является ли функция, реализуемая этой формулой, невыполнимой.*

Другими словами, дана функция $f(x_1, x_2, ..., x_n)$, определенная своей КНФ: $(D_1) \wedge (D_2) \wedge \cdots \wedge (D_k)$. Требуется ответить на вопрос, правда ли, что для любых наборов значений логических переменных a_1, a_2, ..., a_n, выполняется равенство $f(a_1, a_2, ..., a_n) = 0$.

Очевидно, что эта задача обратная к задаче о выполнимости, но она не лежит в классе NP, поскольку для нее не может существовать недетерминированного полиномиального алгоритма

решения: в любом случае, чтобы получить положительный ответ на данную задачу, мы должны проверить все возможные наборы логических переменных, а сложность этого процесса имеет порядок $O(2^n)$. *Недетерминированность в данном случае не может улучшить ситуацию.*

Действительно, если даже машина копирует себя 2^n *раз и каждая копия проверяет свой набор аргументов функции, ни одна копия не сможет узнать, какой ответ получили оставные, и значит не сможет дать положительный ответ на задачу.*

Если же нам дан ответ "Да", то мы не сможем предоставить доказательство этого ответа, которое можно было бы проверить за полиномиальное время. В любом случае нам придется проверить все 2^n *наборов аргументов, чтобы убедиться в правильности ответа.*

3.2.4 Отношение между классами P и NP

Ясно, что недетерминированная машина Тьюринга является частным случаем детерминированной, так что $P \subseteq NP$. Более полную систему отношений между классами задач в форме распознавания можно увидеть на рисунке 31.

Кажется, что $NP \neq P$, поскольку недетерминированная машина может выполнять несколько процессов одновременно, при условии, что хотя бы один из них остановится и выдаст положительный результат.

С другой стороны, с помощью детерминированной машины Тьюринга можно смоделировать работу недетерминированной машины, используя дополнительные состояния, когда встречается неоднозначность, подобно тому, как это делают псевдомногозадачные операционные системы. Работа такой моделирующей машины требует значительно большего времени, чем работа оригинальной недетерминированной машины Тьюринга, но не известно на сколько больше — сохранит ли полученный алгоритм свойство полиномиальности.

Вопрос о том, найдется ли для любой задачи из класса NP эффективный алгоритм решения является классическим вопросом о равенстве классов P и NP, который уже много лет остается одной из центральных открытых научных проблем. Вопрос равенства классов

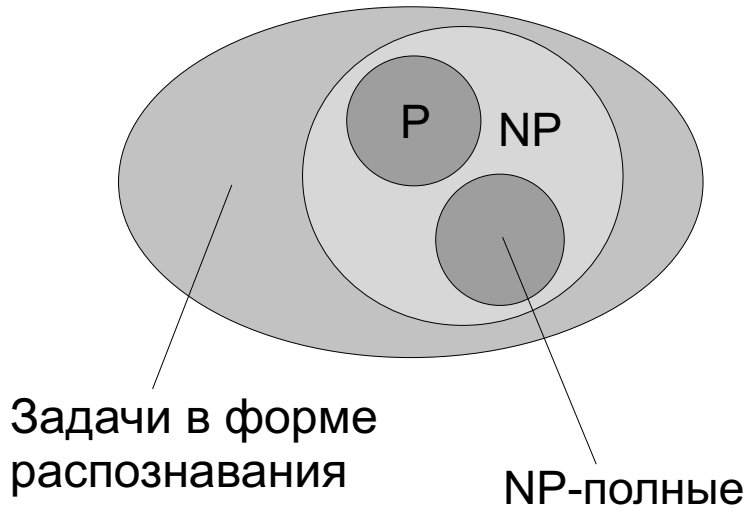

Рисунок 31: Отношение между классами задач в форме распознавания.

NP и P по сути состоит в следующем: если положительный ответ на задачу можно проверить за полиномиальное вермя, то правда ли, что этот ответ можно также за полиномиальное время и найти. Действительно ли задачи легче проверить, чем решить?

Если бы было доказано, что $P = NP$, это позволило бы решить целый ряд задач, для которых до сих пор не известно полиномиальных алгоритмов решения. Большое количество классических задач лежит в классе NP. В то же время, это могло бы повлечь и проблемы, связанные, например, с ненадежностью (в этом случае) некоторых алгоритмов шифрования, основанных на предполагаемой невозможности быстро проводить определенные вычисления.

В настоящее время большинство математиков считает, что эти классы не равны, но доказательство остается ненайденым.

3.2.5 NP-полные задачи

Следующая теорема была доказана Куком в 1971 году.

Теорема 3.2.4 . *Любая задача из класса NP полиномиально сводится к задаче о выполнимости.*

Таким образом, задача о выполнимости в некотором смысле не проще всех остальных задач из класса NP. Все подобные ей задачи названы NP-полными.

Определение 3.2.11 . *Задача называется NP-полной, если она принадлежит классу NP и к ней полиномиально сводятся все задачи из класса NP.*

Замечание 3.2.5 . *Все NP-полные задачи попарно полиномиально эквивалентны.*

Определение 3.2.12 . *Задача называется NP-трудной, если к ней полиномиально сводятся все задачи из класса NP.*

Пример 3.2.9 . *Можно доказать, что задача о раскраске контурной карты из примера 3.2.5 в оптимизационной форме является NP-трудной, а ее переформулировка в форме распознавания свойств является NP-полной задачей.*

Теперь для доказательства NP-полноты задачи из класса NP достаточно показать, что к ней полиномиально сводится некоторая известная NP-полная задача.

Пример 3.2.10 . *Из теоремы 3.2.1 следует, что и задача о 3-выполнимости также является NP-полной.*

Можно заметить, что задача о 2-выполнимости лежит в классе P.

Понятие NP-полной задачи может помочь прояснить ситуацию с отношением между классами P и NP. Если будет найден полиномиальный алгоритм решения одной из NP-полных задач, то любая задача из класса NP сможет быть решена за полиномиальное время и, следовательно, $P = NP$.

Если для произвольной задачи из класса NP будет доказано, что она за полиномиаьное время не решается, то окажется верно, что $P \neq NP$ и любая NP-полная задача не сможет лежать в P.

На практике полезно уметь распознавать NP-полные задачи. Если будет доказано, что рассматриваемая вами задача NP-полна, то нахождение полиномиального алгоритма решения этой задачи будет равносильно решению вопроса о равенстве или неравенстве классов P и NP, который остается без ответа уже более тридцати лет. С практической точки зрения такие задачи можно считать неразрешимыми эффективно и искать приближенные алгоритмы их решения.

4 Теория графов

Теория графов — один из основных разделов дискретной математики Основным объектом изучения является граф. Граф исходно графический объект, рисунок (рис. 32). Это способ изобразить некоторую структуру с помощью простой схемы. У графа есть некоторое количество узлов

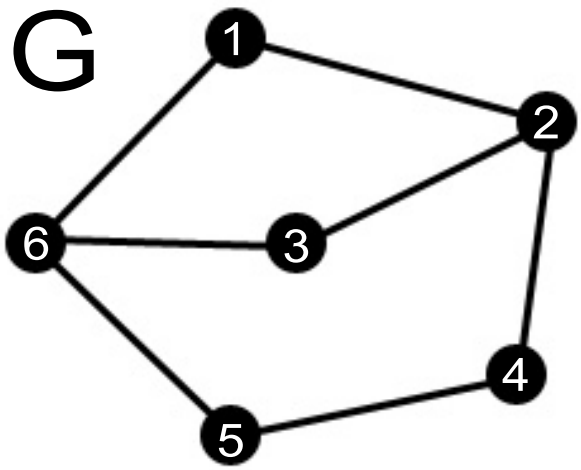

Рисунок 32: Пример графа

(точек, вершин), которые соеденины между собой линиями (ребрами).

Примером графа служит схема метрополитена. Станции — вершины графа; переезды между станциями — ребра графа.

Другим примером может служить система родственных связей в каком-нибудь городе. Вершинами такого графа служили бы все жители города. Люди, находящиеся в непосредственном родстве (братья, сестры, родитель и ребенок) были бы соеденены ребрами, а между дальними родственниками можно было бы найти цепочку из ребер близкого родства. Между людьми не находящимися в родстве в этом графе не нашлось бы никакой связи.

Теорию графов часто не интересует конкретное изображение графа и имена вершин. Основное внимание уделяется структуре: сколько вершин

имеет граф, какие из них соеденены ребрами, а для *ориентированных* графов еще от к какой вершины к какой направлены ребра.

4.1 Определения графа

Существует несколько способов сказать, что такое граф, а под словом граф в разных случаях понимают структуры, отличающиеся по своим свойствам. В каждом конкретном случае, когда вы имеете дело с графами следует понимать, что имеется в виду под этим словом, и использовать те свойства, которые характерны для данного класса графов.

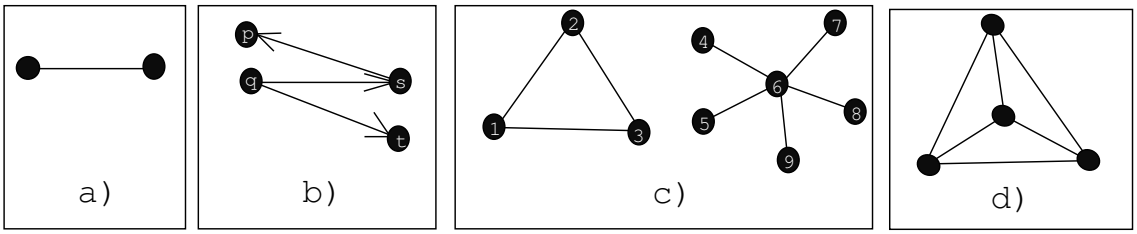

Рисунок 33: Примеры графов

Самым наглядным способом задания графа является его графическое изображение. Само название графа появилось благодаря этому представлению. На рисунке 33 инзображены примеры графов. Графы a), b) и d) — связные; графы b) и c) — помеченные; графы a) и d) — полные, граф b) — ориентированный [1].

Далее мы попробуем прояснить, что же такое граф и в чем особенности разных способов его определения.

4.1.1 Общее определение

Рассмотрим наиболее общее определение графа. Определения большинства понятий подразумевающихся под термином "граф" являются некоторыми сужениями этого определения.

Определение 4.1.1 . *Пусть заданны конечные множества произвольной природы* V *и* E, $V \neq \varnothing$, $V \cap E = \varnothing$, *и трехместный*

[1]Перечисленные свойства графов будут определены в дальнейшем.

предикат P

$$P(x,u,y) : V \times E \times V \mapsto \{True, False\}.$$

Будем говорить, что задан граф $G = (V, E, P)$ *с множеством вершин* V*, множеством ребер* E *и инцидентором* P*, если*

1. P *определен для всех упорядоченных троек* (v, e, w)*, где* $v, w \in V$ *и* $e \in E$*;*

2. $(\forall e \in E)(\exists v, w \in V)\{P(v,e,w)\&(\forall v', w' \in V)[P(v',e,w') \Rightarrow (v = v' \ \& \ w = w') \vee (v = w' \ \& \ w = v')]\}$*. То есть для любого ребра всегда существует две вершины (не обязательно различные), которые оно соединяет, и никакие другие две вершины не могут быть соеденины тем же ребром.*

Кроме того для каждого ребра $e \in E$ *должно выполняться одно и только одно из соотношений:*

A. $(\exists v, w \in V)[v \neq w \ \& \ P(v,e,w) \ \& \ P(w,e,v)]$

B. $(\exists v, w \in V)[v \neq w \ \& \ P(v,e,w) \ \& \ \overline{P}(w,e,v)]$

C. $(\exists v \in V)P(v,e,v)$

Ребро удовлетворяющее соотношению A называют *неориентированным*, удовлетворяющее соотношению B — *ориентированным* ребром или *дугой*; ребра удовлетворяющие соотношению C называют *петлями*. Ребро $e \in E$ назовем *кратным*, если

$$(\exists v, w \in V)(\exists e_1 \in E)[P(v,e,w)\&P(v,e_1,w)].$$

Определение 4.1.2 . *Граф будем называть ориентированным или орграфом, если в нем отсутствуют ребра типа A (неориентированные ребра).*

Граф будем называть неориентированным, если в нем отсутствуют ребра типа B (ориентированные ребра).

Если граф содержит и ориентированные и неориентированные ребра, его называют смешанным.

Неориентированные ребра рисуют линиями, соединяющими точки, соответствующие вершинам. Ориентированные ребра обозначают стрелками. При этом для ребра, для которого выполняется $P(v, e, w)$, стрелка направлена от вершины v и к вершине w. Ориентированные ребра также называют дугами. Для петель ориентация не имеет смысла, так что они могут изображаться и со стрелкой и без, когда это удобно.

Определение 4.1.3 . *Порядком графа $G = (V, E, P)$ называют число $n = |V|$.*

Определение 4.1.4 . *Две вершины $v, w \in V$ называются смежными, если $(\exists e \in E)[P(v, e, w) \vee P(w, e, v)]$*

Определение 4.1.5 . *Вершина $v \in V$ и ребро $e \in E$ инцидентны, если $(\exists w \in V)[P(v, e, w) \vee P(w, e, v)]$*

Определение 4.1.6 . *Два ребра $e, f \in E$ смежны, если существует вершина $v \in V$ которой они оба инцидентны.*

Замечание 4.1.1 . *До сих пор мы говорили о конечных графах. Существуте также понятие бесконечного графа, для которого множества V и E могут быть счетными. Мы такие графы рассматривать не будем.*

4.1.2 Виды графов

Определение 4.1.7 . *Псевдографом будем называть общий случай графа описанного в определении 4.1.1.*

Определение 4.1.8 . *Мультиграфом назовем граф, удовлетворяющий определению 4.1.1, который, дополнительно, не может содержать ребрер типа C (петель).*

Псевдограф может содержать кратные ребра и петли. Мультиграф не может содержать петель, хотя может содержать любое количество кратных ребер. Пример псевдографа приведен на рисунке 34 a), пример мультиграфа — на рисунке 34 b).

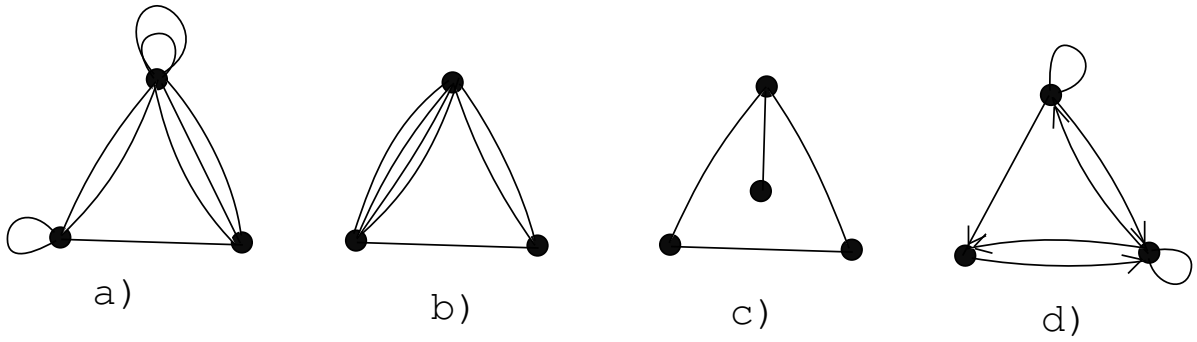

Рисунок 34: Псевдограф, мультиграф, обыкновенный граф и граф Бержа.

При необходимости можно говорить об ориентированных и неориентированных псевдографах и мультиграфах.

Приведем еще одно определение графа, которое, впрочем не является частным случаем определения 4.1.1.

Определение 4.1.9 . *Гиперграфом назовем пару $G = (V, E)$, где V — конечное множество элементов произвольной природы, а $E \subset 2^V$ — произвольное множество подмножеств множества V.*

Это самый широкий класс, включающий в себя все описанные выше классы неориентированных графов. Больше мы с ним не встретимся.

4.1.3 Обыкновенный граф

В самом простом случае, мы имеем дело с так называемыми *обыкновенными* графами (рис. 32). Такие графы не ориентированы (ребра не имеют направления, концевые вершины ребер равнозначны) и между любыми двумя вершинами может быть не более одного ребра; петель нет.

Таким образом в обыкновенный граф входят только ребра типа A (неориентированные ребра), причем не более одного ребра между любыми двумя вершинами.

Примеры обыкновенных гарфов приведены на рисунках 33 a), c), d) и рисунке 34 c). В большинстве случаев мы будем иметь дело именно с такими графами.

Обыкновенные графы удобно представлять в виде пары множеств (V, E), где V — произвольное непустое множество (множество вершин графа), а $E \subseteq V^{(2)} = \{\{u, v\} \mid u, v \in V;\ u \neq v\}$ — множество двухэлементных подмножеств множества V (множество ребер графа). Каждая пара $\{u, v\} \in E$ соответствует ребру соединяющему вершины u и v. Когда известно, что речь идет об обыкновенных графах, фигурные скобки часто заменяют на круглые для простоты записи и восприятия. Подразумвается, что пары тем не менее неупорядоченные ($(u, v) = (v, u)$).

Например, граф G на рисунке 33 является обыкновенным. Его множество вершин $V = \{1, 2, 3, 4, 5, 6\}$ и множество ребер $E = \{(1, 2), (1, 6), (2, 3), (2, 4), (3, 6), (4, 5), (5, 6)\}$.

Будем предполагать, что имеем дело с конечными графами: $|V| < \infty$. Будем писать $G = (V, E)$ для обозначения, что граф G имеет множество вершин V и множество ребер E. Если нам понадобится указать, что некоторое множество является множеством вершин или ребер именно графа G, будем писать $V(G)$ или $E(G)$ соответственно.

Определение 4.1.10 . *Порядком графа G называется число его вершин — $|V(G)|$.*

Определение 4.1.11 . *Будем говорить, что вершины u и v смежны в графе G, если они соеденины ребром: $(u, v) \in E(G)$. В этом случае u и v называются концевыми вершинами ребра $e = (u, v)$.*

Определение 4.1.12 . *Вершина v и ребро e называются инцидентными, если v является одной из концевых вершин e: $e = (u, v)$.*

Определение 4.1.13 . *Два ребра e_1 и e_2 называются смежными, если у них есть общая концевая вершина: $e_1 = (u, v)$, $e_2 = (v, w)$.*

Для удобства будем полагать, что $G = (V, E)$, $|V| = n$, $|E| = m$. Будем называть такой граф, имеющий n вершин и m ребер (n, m)-графом.

Определение 4.1.14 . *Степенью вершины $v \in V$ в графе G назовем число $\deg_G v$ инцидентных ей ребер.*

Вершина со степенью 0 называется изолированной. Вершина со степенью 1 называется висячей.

Определение 4.1.15 . *Степенью графа* $\Delta(G)$ *называют максимальную из степеней вершин графа. Минимальную из степеней графа обозначают* $\delta(G)$.

Определение 4.1.16 . *Регулярным графом называется граф, у которого все вершины имеют одинаковую степень.*

Теорема 4.1.1 . *В обыкновенном графе число вершин нечетной степени четно.*

Доказательство. Для графа с n вершинами и m ребрами имеет место равенство $\sum_{i=1}^{n} \deg_G v_i = 2m$, где $\deg_G v_i$ — степень i-той вершины.
□

4.1.4 Примеры графов

Рассмотрим несколько классов обыкновенных графов.

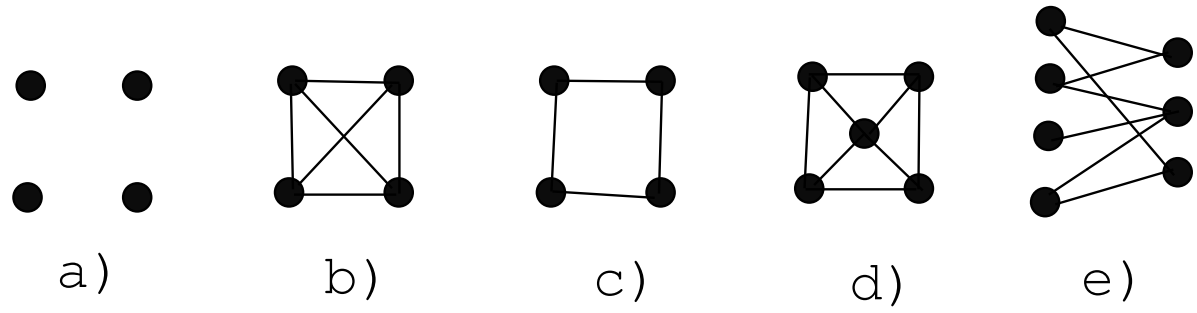

Рисунок 35: O_4, F_4, C_4, W_5 и двудольный граф.

1) Полный граф (35 b)). Граф называется полным, если $E = V^{(2)}$. Другими словами, каждая вершина графа смежна с каждой другой. Такие графы обычно обозначают K_n или F_n. На рисунке 36 пункты a) и b) изображены полные графы K_2 и K_5 соответственно.

2) Пустой граф (35 a)). Граф называется пустым, если у него нет ни одного ребра: $E = \varnothing$. Пустые графы обозначают O_n. Пример пустого графа O_3 изображен на рисутке 36 c).

3) Двудольный граф (35 e)). Граф G называется двудольным, если его множество вершин можно разбить на два подмножества, таким

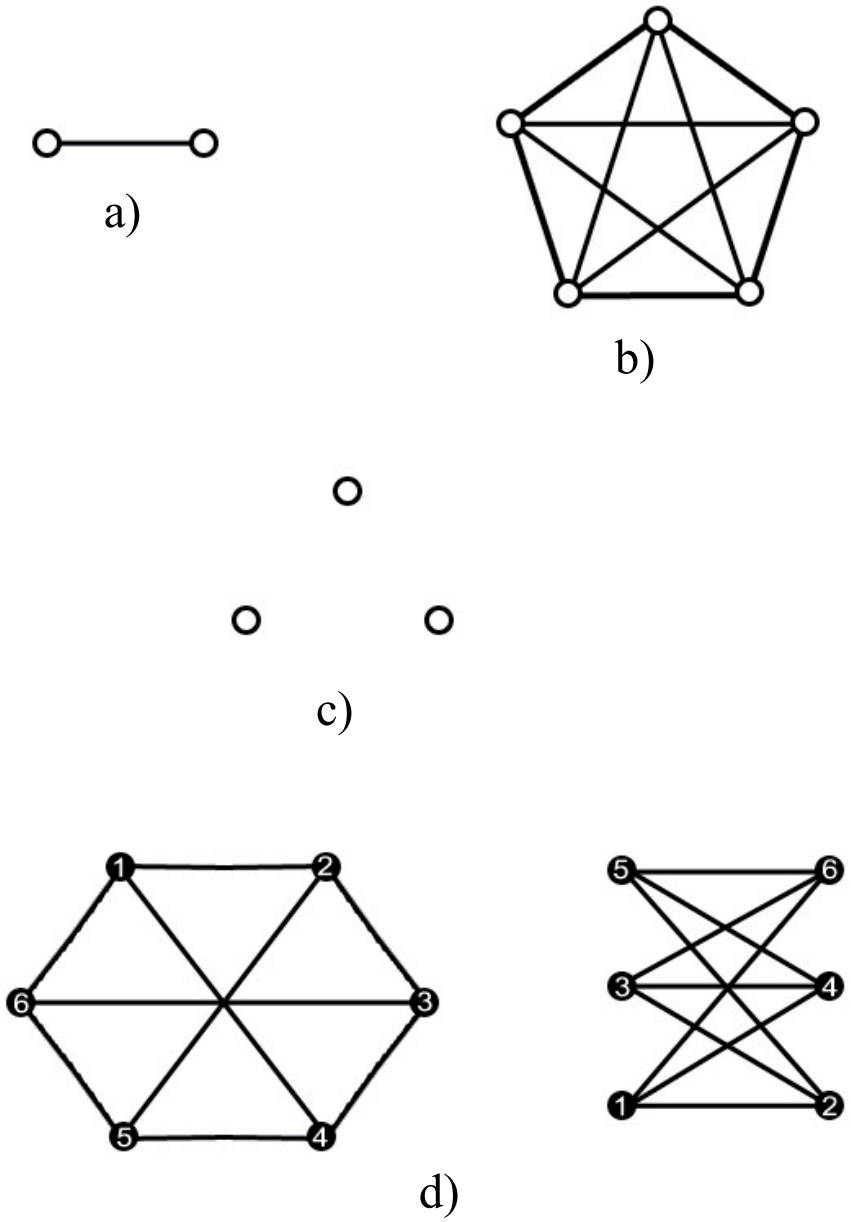

Рисунок 36: Примеры полного, пустого, двудольного графов

образом, чтобы все ребра графа начинались в одном из подмножеств и заканчиваются в другом. То есть существуют такие множества V_1 и V_2, что $V(G) = V_1 \cup V_2$, $V_1 \cap V_2 = \varnothing$ и

$$\forall (u, v) \in E(G) \quad \Rightarrow \quad v \in V_1 \text{ и } u \in V_2, \text{ или } u \in V_1 \text{ и } v \in V_2.$$

Граф называется полным двудольным графом, если дополнительно $\forall u \in V_1$, $\forall v \in V_2 \Rightarrow (u, v) \in E(G)$. Полный двудольный грав обозначают K_{n_1,n_2}.

На рисунке 36 d) слева приведено изображение полного двудольного графа $K_{3,3}$. На рисунке 36 d) справа этот же граф изображен так, чтобы было лучше видно множества V_1 и V_2.

4) Циклы (35 c)). Циклом называется граф, состоящий из n вершин последовательно соединены ребрами в кольцо[2].

Цикл из n вершин обозначают C_n. Примеры циклов изображены на рисунке 37: цикл C_6 слева и C_5 справа.

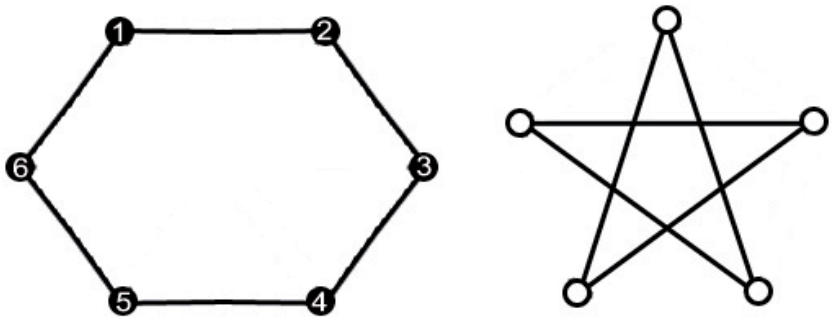

Рисунок 37: Примеры циклов

5) Колесо (35 d)) W_n — цикл C_{n-1}, к которому добавлена еще одна, смежная со всеми остальными, вершина.

6) Звезда $K_{1,n-1}$ (на рис. 38 изображена звезда $K_{1,5}$).

[2]Подробнее о циклах смотри тему про маршруты.

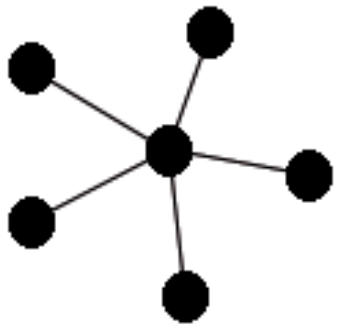

Рисунок 38: Граф "Звезда"

4.1.5 Графы Бержа

Самым распространенным видом ориентированных графов являются графы Бержа.

Определение 4.1.17 . *Граф $G = (V, E, P)$ называется графом Бержа, если он удовлетворяет условию*

$$(\forall e, f \in E)(\forall v, w \in V)[P(v, e, w) \& P(v, f, w) \Rightarrow (e = f)].$$

Таким образом граф Бержа — это ориентированный граф, который имеет не более одной дуги между любыми двумя вершинами в одном направлении и не более одной петли при каждой вершине.

Примеры графов Бержа приведены на рисунках 33 b) и рисунке 34 d).

Не трудно видеть, что для задания графа Бержа достаточно указать отображение $F : V \to 2^V$, ставящее в соответствие каждой v вершине графа некоторое подмножество множества его вершин — вершины, в которые из v идут дуги. То есть, определение может выглядеть так:

Определение 4.1.18 . *Графом Бержа будем называть пару (V, F), где V — множество элементов произвольной природы, называемых вершинами, и $F : V \to 2^V$.*

Определение 4.1.19 . *По другому граф Бержа может быть определен, как пара $G = (V, E)$, где V — конечное множество элементов произвольной природы, а $E \subset V \times V$ — множество упорядоченных пар элементов из V.*

Определение 4.1.20 . *Полустепенью исхода вершины $v \in V$ ориентированного графа G называют число* $\operatorname{odeg} v$ *выходящих из v ребер, то есть*

$$\operatorname{odeg} v = |\{e \in E \mid (\exists w \in V) P(v, e, w)\}|$$

Полустепенью захода вершины $v \in V$ ориентированного графа G называют число $\operatorname{ideg} v$ *входящих в v ребер, то есть*

$$\operatorname{ideg} v = |\{e \in E \mid (\exists w \in V) P(w, e, v)\}|$$

Свойства смежности и инцидентности для ориентированных графов определены точно так же, как для неориентированных.

4.2 Изоморфизм графов

Очень важным для теории графов является понятие изоморфизма.

В обыкновенном графе с n вершинами может быть не более $\binom{n}{2}$ ребер. Следовательно, если рассматривать все возможные способы расстановки ребер, оказывается, что существует всего $2^{\binom{n}{2}}$ помеченных графов. При этом, далеко не все графы различаются структурно.

На рисунке 39 изображены два графа. Очевидно, что их структура

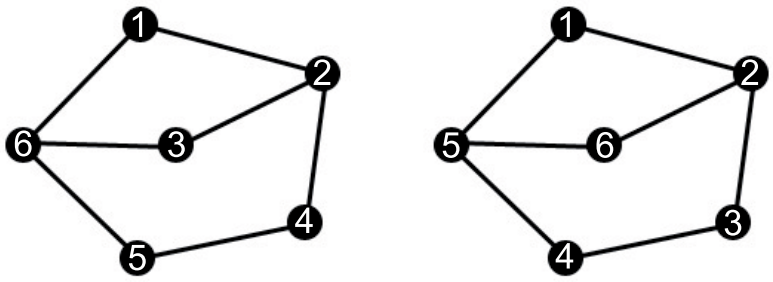

Рисунок 39: Пример изоморфизма

ничем не различается, но строго по определению они не равны, поскольку их множества ребер не совпадают: ребро $(1, 6)$ есть в левом гафе, но его нет в правом.

Определение 4.2.1 . *Два графа* $G_1 = (V_1, E_1, P_1)$ *и* $G_2 = (V_2, E_2, P_2)$ *изоморфны, если существуют биекции* $\varphi : V_1 \mapsto V_2$ *и* $\psi : E_1 \mapsto E_2$, *такие что*

$$(\forall v, w \in V_1)(\forall e \in E_1)[P_1(v, e, w) \Leftrightarrow P_2(\varphi(v), \psi, (e), \varphi(w))].$$

Несложно показать, что изоморфизм на множестве графов является отношением эквивалентности. Действительно, чтобы доказать рефлексивность, достаточно взять в качестве φ и ψ тождественные отображения $V_1 \mapsto V_1$ и $E_1 \mapsto E_1$. Симметричность доказывается, использованием в качестве искомых отображений φ^{-1} и ψ^{-1}. Транзитивность будет следовать из биективности композиции биективных отображений.

Таким образом по свойству отношений эквивалентности все множество графов делится на классы, любые два графа в которых — изоморфны. В дальнейшем, когда нам не будут принципиальны особенности конкретных графов, мы будем обращаться со всем классом изоморфных графов, как с одним и тем же графом.

Запись $G_1 \cong G_2$ в дальнейшем будет значить, что графы G_1 и G_2 изоморфны.

Для обыкновенных графов и графов Бержа определение изоморфизма можно можно переформулировать проще.

Определение 4.2.2 . *Графы* $G_1 = (V_1, E_1)$ *и* $G_2 = (V_2, E_2)$ *называют изоморфными, если существует такая биекция* $\varphi : V_1 \to V_2$, *что*

$$\forall u, v \in V_1, (u, v) \in E_1 \quad \Leftrightarrow \quad (\varphi(u), \varphi(v)) \in E_2. \tag{32}$$

Пример 4.2.1 . *Рассмотрим графы на рисунке 40. Выберем отображение* φ *следующим образом*

v	$\varphi(v)$
1	1
2	2
3	4
4	3
5	6
6	5

.

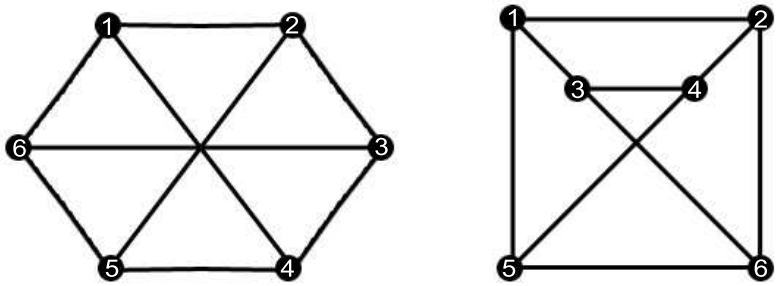

Рисунок 40: Изоморфные графы

Несложно убедиться, что условие (32) *для графов на рисунке 40 выполняется.*

Далее, можем говорить о помеченных и непомеченных графах. Будем говорить, что граф помеченный, если каждая вершина графа имеет индивидуальное имя. С помеченными графами имеют дело, когда необходимо хранить какую-то индивидуальную информацию о каждой вершине. Если же нам интересна только структура графа, а конкретные названия вершин не имеют значения, можно говорить, что граф непомеченный и рассматривать его с точностью до изоморфизма.

4.2.1 Инварианты графа

Определение 4.2.3 . *Функции, принимающие одинаковые значения для изоморфных графов называются инвариантами.*

Определение 4.2.4 . *Систему инвариантов назовем полной системой инвариантов, если, из того что для некоторых двух графов все эти инварианты принимают одинаковые значения, следует, что данные графы изоморфны.*

Приведем примеры инвариантов графа.

Пример 4.2.2 . *1) Простейшими примерами инвариантов графа G являются $n = |V(G)|$ и $m = |E(G)|$ — количества вершин и ребер графа. Очевидно, что у всех изоморфных графов эти значения совпадают.*

2) Другим инвариантом является вектор степеней вершин, упорядоченных по возрастанию: $(d_1, d_2, \ldots, d_n)$.
3) Хроматическое число.

Определение 4.2.5 . *Хроматическим числом графа* G *назовем наименьшее целое число* $\chi(G)$*, такое что существует разбиение множества вершин графа* $V(G) = V_1 \cup V_2 \cup \cdots \cup V_{\chi(G)}$*, удовлетворяющее следующему условию*

$$(\forall x, y \in V_i)(\forall e \in E(G))\overline{P}(x, u, y), \quad i = \overline{1, \chi(G)}.$$

Другими словами ни в одном подмножестве V_i *нет смежных вершин.*

Заметим, что графы с $\chi(G) = 2$ *и* $\chi(G) = 1$ *будут двудольными.*
4) Размер максимальной клики в графе (плотность). Кликой будем называть полный подграф графа G*. Количество вершин максимальной клики в графе* G *обозначается* $\omega(G)$*; иногда это значение называют плотностью графа и обозначают* $\varphi(G)$.

Очевидно, что $\omega(G) \leq \chi(G)$*, поскольку в каждое множество* V_i *из определения хроматического числа может не может входить болше одной вершины из максиальной клики. Графы, для всех подграфов которых выполняется равенство* $\omega(G) = \chi(G)$*, называются совершенными.*
5) Число вершинной независимости (неплотность). Независимым множеством вершин графа G *назовем множество вершин какого-нибудь пустого подграфа* G*, то есть множество попарно несмежных вершин. Числом вершинной независимости* $\alpha(G)$ *(неплотностью) называется число вершин в максимальном независимом множестве графа* G*. Очевидно* $\alpha(G) = \omega(\overline{G})$.
6) Размер минимального кликового покрытия. Размером минимального кликового покрытия графа G *назовем наименьшее число* $\overline{\chi}(G)$ *клик* $G_1, G_2, \ldots, G_{\overline{\chi}(G)}$ *графа* G*, удовлетворяющих условию*

$$(\forall v \in V(G))(\exists G' \in \{G_1, G_2, \ldots, G_{\overline{\chi}(G)}\})[v \in V(G')]$$

Нетрудно видеть, что $\overline{\chi}(G) = \chi(\overline{G})$*. Также легко доказывается неравенство* $\alpha(G) \leq \overline{\chi}(G)$.

4.3 Операции

4.3.1 Основные операции над графами

Существует множество операций над графами и разные авторы определяют операции, которые им нужны. Рассмотрим несколько основных операций для примера.

1) Добавление вершины: $G_1 = G + v$.
$V(G_1) = V(G) \cup \{v\}, \quad E(G_1) = E(G)$.

2) Добавление ребра: $G_1 = G + e$.
$V(G_1) = V(G), \quad E(G_1) = E(G) \cup \{e\}$. Операция применима, если $e = (v, u)$ и $u, v \in V(G)$.

3) Удаление ребра: $G_1 = G - e$.
$V(G_1) = V(G), \quad E(G_1) = E(G) \setminus \{e\}$. Операция применима, если $e \in E(G)$.

4) Удаление вершины: $G_1 = G - v$.
$V(G_1) = V(G) \setminus \{v\}, \quad E(G_1) = \{e \mid e \in E(G), \ \nexists u \in V(G) : \ e = (v, u)\}$.

5) Объединение графов: $G_3 = G_1 \cup G_2$.
$V(G_3) = V(G_1) \cup V(G_2), \quad E(G_3) = E(G_1) \cup E(G_2)$.

Как видно из определения, если $V(G_1) \cap V(G_2) = \varnothing$ и $E(G_1) \cap E(G_2) = \varnothing$, изображение графа G_3 можно получить просто нарисовав рядом графы G_1 и G_2.

6) Произведение графов: $G_3 = G_1 \times G_2$.
$V(G_3) = V(G_1) \times V(G_2)$, $E(G_3) = \{((u_1, u_2), (v_1, v_2)) \mid u_1, v_1 \in V(G_1), \ u_2, v_2 \in V(G_2), \ (u_1 = v_1$ и $(u_2, v_2) \in E(G_2))$ или $(u_2 = v_2$ и $(u_1, v_1) \in E(G_1))\}$.

Пример произведения графов можно увидеть на рисунке 41.

4.3.2 Подграфы

Определение 4.3.1 . *Рассмотрим граф $G = (V, E, P)$. Пусть $V' \subseteq V$, $E' \subseteq E$, а P' — предикат, индуцированный (полученный сужением) инцидентором P на подмножествах V' и E'. Если V' и E' выбраны так, что $G' = (V', E', P')$ удовлетворяет определению графа, то G' называется частью графа G.*

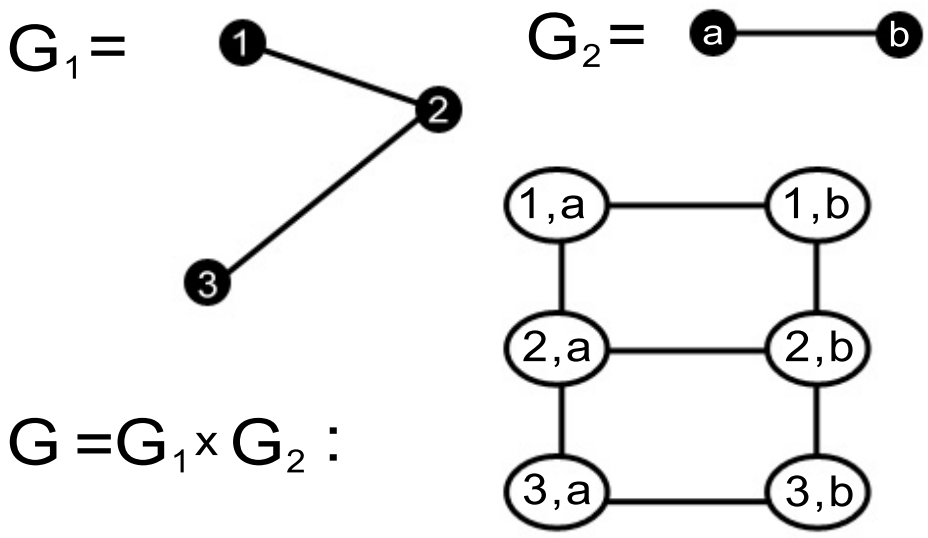

Рисунок 41: Произведение графов

Определение 4.3.2 . *Часть* $G' = (V, E', P')$ *графа* $G = (V, E, P)$ *называется суграфом.*

Определение 4.3.3 . *Часть* $G' = (V', E', P')$ *графа* $G = (V, E, P)$ *называется подграфом, если* $(\forall v, w \in V')(\forall e \in E)[P(v, e, w) \Rightarrow e \in E']$

Приведем более удобные определения для случая обыкновенного графа $G = (V, E)$.

Определение 4.3.4 . *Граф* $G_1 = (V_1, E_1)$ *называется подграфом графа* $G = (V, E)$, *если* $V_1 \subseteq V$ *и* $E_1 = \{e \mid e = (u, v) \in E, \ \ u, v \in V_1\}$.

Такой подграф иногда называют подграфом порожденным множеством вершин V_1, *поскольку он включает все ребра графа* G, *которые соединяют вершины из* V_1.

Определение 4.3.5 . *Граф* $G_1 = (V_1, E_1)$ *называется частью графа* $G = (V, E)$, *если* $V_1 \subseteq V$ *и* $E_1 \subseteq \{e \mid e = (u, v) \in E, \ \ u, v \in V_1\}$.

Определение 4.3.6 . *Граф* $G_1 = (V, E_1)$ *называется суграфом графа* $G = (V, E)$, *если* $E_1 \subseteq E$.

Можно сказать, что чтобы получить суграф, нужно удалить из графа часть ребер. Чтобы получить порожденный подграф, нужно удалить из графа часть вершин и каждое ребро, хотябы один конец которого был удален. Чтобы получить часть графа, нужно взять суграф от порожденного подграфа графа.

Примером задачи, связанной с подграфами, является поиск наибольшей клики графа (рис. 42).

Определение 4.3.7 . *Кликой графа называется максимальный по включению вершин полный подграф графа.*

Таким образом, поиск наибольшей клики графа то же, что и поиск наибольшего полного подграфа графа.

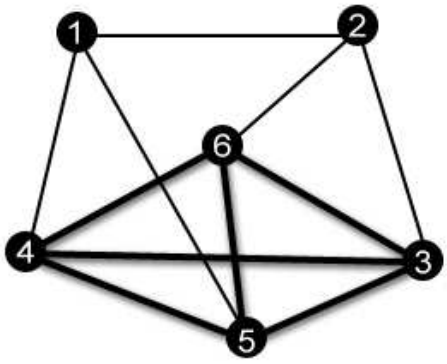

Рисунок 42: Наибольшей клика графа

Неизвестно эффективного алгоритма, решающего эту задачу. Переформулируем эту задачу переформулировать в форме распознавания: Дан граф G и $b \in \mathbb{R}_+$. Ответить на вопрос существует ли в этом графе полный подграф размера b? Эта задача является NP-полной задачей. Доказательство NP-полноты задачи о клике будет рассмотрено позже, в параграфе 4.9.

4.3.3 Дополнение графа

Определение 4.3.8 . *Дополнением графа $G = (V, E)$ называется граф*

$\overline{G}$, *такой, что* $V(\overline{G}) = V(G)$ *и* $(\{u, v\} \in E(\overline{G})) \Leftrightarrow (\{u, v\} \notin E(G))$, *то есть множество ребер* $E(\overline{G})$ *— дополнение* $E(G)$ *до множества ребер полного графа с данным числом вершин.*

Определение 4.3.9 . *Граф изоморфный своему дополнению называется самодополнительным.*

Несложно показать и следующие свойства дополнения графа

1) Двойное дополнение равно исходному графу: $\overline{\overline{G}} = G$,

2) Если графы изоморфны, то изоморфны и их дополнения:
$G_1 \cong G_2 \quad \Leftrightarrow \quad \overline{G}_1 \cong \overline{G}_2$.

4.4 Маршруты и связность

Определение 4.4.1 . *Маршрутом в графе* G *называется последовательность вершин и ребер* $v_0, e_1, v_1, e_2, v_2, ..., e_l, v_l$, *где* $v_0, v_1, ..., v_l \in V(G)$, $e_1, ..., e_l \in E(G)$ *и* e_i *инцидентна* v_{i-1} *и* v_i, $i = \overline{1, l}$.

Указанный маршрут называется маршрутом из вершины v_0 *в вершину* v_l. *Длиной маршрута называется число* l.

Проще говоря, маршрут представляет собой последовательность вершин и переходов между ними по ребрам. Определение выше избыточно, так как если указана первая вершина маршрута, остальные вершины однозначно определяются ребрами по которым проходит маршрут. Если граф обыкновенный или граф Бержа, для полного задания маршрута достаточно указать только последовательность проходимых вершин.

Определение 4.4.2 . *Маршрут называется цепью, если ребра в нем не повторяются.*

Определение 4.4.3 . *Цепь называется простой цепью, если вершины в ней не повторяются.*

Определение 4.4.4 . *Замкнутым называется такой маршрут, где* $v_0 = v_l$.

Определение 4.4.5 . *Циклом называется замкнутая цепь.*

Определение 4.4.6 . *Прстым циклом называется замкнутая простая цепь. В этом случае, запрет на повторение вершин не распространяется на вершины v_0 и v_l.*

Определение 4.4.7 . *Ориентированным маршрутом в ориентированном графе G называется последовательность вершин и дуг $v_0, e_1, v_1, e_2, v_2, ..., e_l, v_l$, где $v_0, v_1, ..., v_l \in V(G)$, $e_1, ..., e_l \in E(G)$ и $e_i = (v_{i-1}, v_i)$ — дуга из v_{i-1} в v_i, $i = \overline{1, l}$.*

Определение 4.4.8 . *Ориентированный маршрут называется путем, если он не проходит по одной дуге дважды.*

Определение 4.4.9 . *Путь называется простым путем, если вершины в нем не повторяются.*

Определение 4.4.10 . *Замкнутый путь называется контуром.*

Определение 4.4.11 . *Замкнутый простой путь называется простым контуром.*

Определение 4.4.12 . *Граф G называется связным, если между любыми двумя вершинами этого графа существует маршрут.*

Определение 4.4.13 . *Компонентой связности графа называется максимальный по включению вершин связный подграф графа.*

Любой граф можно считать объединением его компонент связности.

Теорема 4.4.1 . *Для любого графа G верно, что хотя бы один из графов G и $\overline{G}$ — связный граф.*

Доказательство. Если граф G связен, теорему можно считать доказанной.

Пусть, граф G несвязен. Тогда существуют больше одной компоненты связности: $V(G) = V_1 \cup V_2$, $V_1 \neq \varnothing$, $V_2 \neq \varnothing$, $\forall u \in V_1$, $\forall v \in V_2$, $(u, v) \notin E(G)$.

Докажем, что граф $\overline{G}$ связен. Для этого рассмотрим две любые вершины w_1 и w_2 графа и докажем, что между ними есть маршрут.

Рассмотрим два варианта. 1) Пусть $(w_1, w_2) \in E(G)$. Тогда вершины w_1 и w_2 одновременно лежат или в множестве V_1 или в множестве V_2. Пусть, не умаляя общности, $w_1, w_2 \in V_1$.

Рассмотрим произвольную вершину $u \in V_2$. Тогда $(u, w_1) \notin E(G)$ и $(u, w_2) \notin E(G)$. Следовательно $(u, w_1) \in E(\overline{G})$ и $(u, w_2) \in E(\overline{G})$. То есть между вершинами w_1 и w_2 нашелся маршрут длины 2: w_1, u, w_2.

2) Пусть $(w_1, w_2) \notin E(G)$. Тогда по определению дополнения $(w_1, w_2) \in E(\overline{G})$. То есть между вершинами w_1 и w_2 нашелся маршрут длины 1.

Между двумя прозвольными вершинами графа $\overline{G}$ существует соединяющий их маршрут, что и требовалось доказать.
□

4.5 Деревья

Определение 4.5.1 . *Деревом называется связный граф без циклов.*

Граф без циклов называется лесом. Каждая компонента связности леса является деревом.

Теорема 4.5.1 . *Пусть $G = (V, E)$, $|V| = n$, $|E| = m$. Тогда следующие утверждения эквивалентны:*
1) G — связный граф без циклов (дерево),
2) G — граф без циклов и $m = n - 1$,
3) G — связный граф и $m = n - 1$.

Доказательство. 1)⇒ 2) Необходимо доказать, что $m = n-1$. Докажем по индукции. База индукции очевидна: Дерево с 1 вершиной не имеет ребер и соотношение выполняется.

Пусть соотношение выполняется для любого дерева не более чем с $n-1$ вершиной. Рассмотрим дерево с n вершинами, $n > 1$.

Выберем произвольное ребро $e = (u, v) \in E(G)$. Рассмотрим граф $G - e$. Этот граф не может быть связным. Действительно, если бы граф $G-e$ был связным, в нем был бы маршрут между вершинами u и v. Этот маршрут вместе с ребром e образовывал бы цикл в графе G.

Значит в графе $G - e$ две компоненты связности G_1 и G_2. Обе они являются связными графами без циклов и для них выполняется

индукционное предположение. Значит $m_1 = n_1 - 1$, $m_2 = n_2 - 1$ и для исходного графа G: $m = m_1 + m_2 + 1 = n_1 - 1 + n_2 - 1 + 1 = n - 1$.

2)$\Rightarrow$ 3) Пусть граф G не имеет циклов и число ребер на единицу меньше числа вершин. Докажем, что граф G связный.

Пусть граф G не связен. Пусть G_1, G_2, ..., G_k все компоненты связности G. Тогда каждый граф G_i связный граф без циклов — дерево.

Мы уже доказали утверждение 1)$\Rightarrow$ 2), так что можем этим пользоваться. Следовательно для каждого из графов G_i верно $E(G_i) = V(G_i) - 1$. Таким образом

$$n = \sum_{i=1}^{k} V(G_i),$$

$$m = \sum_{i=1}^{k} E(G_i) = \sum_{i=1}^{k} (V(G_i) - 1) = n - k.$$

Но нам известно, что число ребер в графе на единицу меньше числа вершин, то есть $k = 1$. Следовательно в графе G может быть только одна компонента связности.

3)$\Rightarrow$ 1) Пусть граф G связный и число ребер на единицу меньше числа вершин. Докажем, что граф G не имеет циклов.

Пусть в графе G есть циклы. Рассмотрим произвольное ребро e_1, которое принадлежит какому-то циклу графа G. Граф $G - e_1$ связен, поскольку между любыми двумя вершинами в цикле существовало как минимум две цепи.

Можем повторять процесс удаления ребер до тех пор, пока в графе $G - e_1 - e_2 - ... - e_i$ остаются циклы.

Рассмотрим граф $F = G - e_1 - e_2 - ... - e_s$, полученный после размыкания всех циклов графа G. Граф F связный и в нем нет циклов. Еще раз воспользуемся доказанным переходом 1)$\Rightarrow$ 2):

$$E(F) = V(F) - 1 = V(G) - 1 = E(G),$$

то есть после удаления s ребер число ребер в графе не изменилось. Противоречие вызвано предположением, что в графе G есть циклы. Следовательно, циклов нет, что и требовалось доказать.

□

4.6 Матрицы, связанные с графом

Одним из удобных математических представлений графа является задание связанных с ним матриц. Две самых распространенных из них — это матрица смежности и матрица инцидентности, которые хранят информацию соответственно об отношении смежности между вершинами и отношении инцидентности между вершинами и ребрами.

4.6.1 Матрица смежности

Пусть $G = (V, E)$ — неориентированный граф, $V = \{1, 2, ..., n\}$. Матрица смежности $A(G)$ для графа G представляет собой квадратную (0,1)-матрицу $n \times n$. Элементы $a_{i,j}(G)$ матрицы $A(G)$ заданы следующим соотношением:

$$a_{i,j}(G) = \begin{cases} 1, & (i,j) \in E(G) \\ 0, & (i,j) \notin E(G) \end{cases}. \tag{33}$$

Матрица смежности обыкновенного графа симметрична, поскольку $(i,j) \in V(G) \Leftrightarrow (j,i) \in V(G)$.

На диагонали стоят нулевые элементы, поскольку не допускаются петли.

Сумма элементов по i-той строке или по i-тому столбцу равна степени вершины i.

Сумма всех элементов матрицы смежности равна удвоенному числу ребер графа:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j}(G) = 2|E(G)|.$$

Пример 4.6.1 . *Рассмотрим граф на рисунке 43. Матрица смежности этого графа имеет следующий вид.*

$$A(G) = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

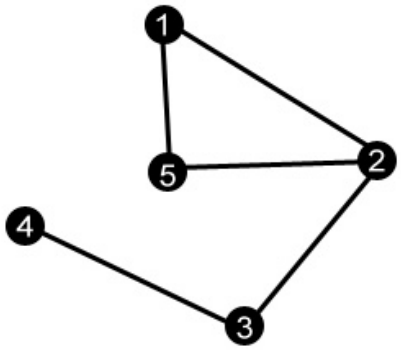

Рисунок 43: Неориентированный граф

Для графа Бержа определение матрицы смежности выглядит так же, как описано в (33), но ребра уже будут считаться ориентированными, так что и симметричности матрицы может не быть. Так же диагональ матрицы смежности для графа Бержа может быть не нулевой, так как он допускает петли.

Сумма элементов по i-той строке равна полустепени исхода вершины i. Сумма элементов по j-тому столбцу равна полустепени захода вершины j.

Сумма всех элементов матрицы смежности ориентированного графа равна числу дуг графа:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}(G) = |E(G)|.$$

Пример 4.6.2 . *Рассмотрим граф на рисунке 44. Матрица смежности этого графа имеет следующий вид.*

$$A(G) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

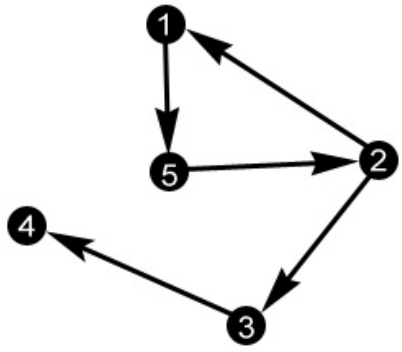

Рисунок 44: Ориентированный граф

Замечание 4.6.1 . *Если мы захотим определить матрицу смежности для мультиграфа или псевдографа, придется выйти за рамки значений 0 и 1. Например, можно сказать, что элемент $a_{i,j}(G)$ равен числу ребер (дуг) соединяющих вершины i и j (идущих из вершины i в вершину j).*

Теорема 4.6.1 . *Пусть $G = (V, E)$ — ориентированный или неориентированный граф и $A(G)$ — его матрица смежности.*

Тогда элемент $a_{i,j}^{(k)}$, стоящий на пересечении i-той строки и j-того столбца матрицы $A^k(G) = (A(G))^k$, равен числу маршрутов из вершины i в вершину j длины k, $i = \overline{1,n}$, $j = \overline{1,n}$. В случае ориентированного графа имеется в виду ориентированный маршрут.

Доказательство. Докажем по индукции. Для степени 1 утверждение верно: элемент матрицы смежности равен числу ребер из i в j.

Предположим, что формула верна для степени $k-1$, где $1 < k$.

Рассмотрим i, j-тый элемент $A^k(G) = A^{k-1}(G) \cdot A(G)$:

$$a_{i,j}^{(k)} = \sum_{t=1}^{n} a_{i,t}^{(k-1)}(G) \cdot a_{t,j}(G). \qquad (*)$$

Здесь $a_{i,t}^{(k-1)}(G)$ — число маршрутов из i в t длины $k-1$, а $a_{t,j}(G)$ указывает наличие или отсутствие ребра из t в j. Таким образом

произведение $a_{i,t}^{(k-1)}(G) \cdot a_{t,j}(G)$ равно числу маршрутов из i в j длины k, где t является предпоследней вершиной. Просуммировав в формуле (*) это произведение по всем возможным промежуточным вершинам t, мы получим число всех маршрутов из i в j длины k, что и требовалось доказать.
□

Пример 4.6.3 . *Продолжим пример 4.6.1. Рассмотрим квадрат матрицы смежности графа с рисунка 43.*

$$A^2(G) = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}^2 = \begin{pmatrix} 2 & 1 & 1 & 0 & 1 \\ 1 & 3 & 0 & 1 & 1 \\ 1 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 2 \end{pmatrix}.$$

Глядя на рисунок можно убедиться, что что на диагонали стоят степени соответствующих вершин. Это логично, так как маршрут из вершины i в нее же длины 2 является проходом по по каждому ребру графа смежному с i туда и обратно. Остальные элементы матрицы равны 1 в тех случаях, если между соответствующими вершинами есть цепь длины 2.

Замечание 4.6.2 . *Можно заметить, что изоморфным графам могут соответствовать разные матрицы смежности. Но это различие не произвольно. Изоморфные графы по сути отличаются порядком нумерации вершин. Этому порядку соответствует порядок нумерации строк и столбцов матрицы смежности. Таким образом матрицы смежности для изоморфных графов отличаются друг от друга одинаковой перестановкой строк и столбцов.*

4.6.2 Матрица инцидентности

Второй распространенной матрицей ассоциированной с графом является матрица инцидентности. Она определяется как прямоугольная матрица $B(G)$ размерности $|V(G)| \times |E(G)|$, где строки соответствуют вершинам, а

столбци — ребрам графа. Пусть $G = (V, E)$ — неориентированный граф, $V = \{1, 2, ..., n\}$ $E = \{e_1, e_2, ..., e_m\}$.

$$b_{i,j}(G) = \begin{cases} 1, & \exists l : e_j = (i, l) \in E(G) \\ 0, & \text{иначе} \end{cases}, \quad i = \overline{1, n}, j = \overline{1, m}. \tag{34}$$

В каждом столбце матрицы инцидентности ровно 2 единицы. Сумма по i-той строке матрицы инциденции равна степени вершины i.

Сумма всех элементов матрицы инцидентности графа равна удвоенному числу его ребер:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} b_{i,j}(G) = 2|E(G)|.$$

Пример 4.6.4 .

Для обыкновенного графа на рисунке 43 матрица инцидентности может выглядеть следующим образом

$$B(G) = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Для определения матрицы инцидентности графа Бержа потребуется использовать дополнительное значение -1 для указания, куда заходит соответствующая дуга.

$$b_{i,j}(G) = \begin{cases} 1, & \exists l : e_j = (i, l) \in E(G) \\ -1, & \exists l : e_j = (l, i) \in E(G) \\ 0, & \text{иначе} \end{cases}, \quad i = \overline{1, n}, j = \overline{1, m}. \tag{35}$$

Пример 4.6.5 . *Для графа Бержа на рисунке 44 матрица инцидентности будет выглядеть следующим образом*

$$B(G) = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Матрицы инцидентности изоморфных графов отличаются друг от друга перестановкой столбцов.

4.6.3 Список ребер

Список ребер — это еще один способ представления графа. В этом случае, каждое ребро задается двумя числами — номерами вершин, которые соединяет это ребро. Можно сказать, что это вариант матрицы инцидентности, где номера строк с единицами заданы в явном виде.

4.7 Обходы графов

4.7.1 Эйлеров цикл

Определение 4.7.1 . *Эйлеровым циклом в графе G называется цикл, который проходит через все ребра графа ровно по одному разу.*

Определение 4.7.2 . *Граф G называется эйлеровым графом, если в нем есть эйлеров цикл.*

Для проверки графа на эйлеровость есть удобный критерий.

Лемма 4.7.1 . *Пусть граф G является эйлеровым графом. Тогда степени всех вершин графа G четны.*

Доказательство. Пусть в графе G существует эйлеров цикл $C = (v_{i_1}, (v_{i_1}, v_{i_2}), v_{i_2}, (v_{i_2}, v_{i_3}), ..., v_{i_l}, (v_{i_l}, v_{i_1}), v_{i_1})$. В этом цикле присутствуют все ребра графа G причем каждое ровно один раз.

В цикле C проходя некоторую вершину v_{i_j} мы входим и выходим из нее по разным ребрам, что добавляет двойку к степени этой вершины. Значит степень вершины v графа G равна удвоенному числу проходов цикла C через вершину v. Следовательно степень v четна.
□

Теорема 4.7.2 . *Пусть G — связный граф. Тогда в графе G существует эйлеров цикл тогда и только тогда, когда степени всех вершин графа G четны.*

Доказательство. Необходимость следует из леммы 4.7.1.

Достаточность. Пусть $\forall v \in V(G)$: $\deg_G v$ — четна. С помощью двухшагового процесса разобьем все ребра графа на циклы.

1) Начнем с произвольной вершины $u \in V(G)$ строить цепь переходя по любым ребрам от вершины к вершине. Поскольку степени вершин четны, войдя в некоторую вершину по одному ребру мы обязательно можем из нее выйти по второму. Таким образом мы точно можем обходить вершины, пока не вернемся в исходную вершину u. Так мы построили некоторый цикл $P_1 = P_1(u)$. По лемме 4.7.1, степени всех вершин в $P_1(u)$ четны.

2) Удалим из G все ребра, входящие в цикл $P_1(u)$. Поскольку степени всех вершин в G и $P_1(u)$ четны, то и в графе $G - E(P_1(u))$ степень каждой вершины четна.

Дальше мы можем перейти к пункту 1) и искать цикл P_2 в графе $G - E(P_1(u))$. Процесс можно повторять до тех пор, пока в графе $G - \sum P_i$ остаются вершины с ненулевой степенью.

Наконец мы получим набор циклов P_1, P_2, ..., P_t:

$$E(G) = E(P_1) \cup E(P_2) \cup ... \cup E(P_t), \qquad E(P_i) \cap E(P_j) = \varnothing,\ i \neq j.$$

Поскольку граф G связен, эти циклы пересекаются по некоторым вершинам. Рассмотрим какие-нибудь два из этих циклов P_i и P_j, у которых есть общая вершина v. Тогда

$$P_i = (v, (v, v_{s_2}), v_{s_2}, (v_{s_2}, v_{s_3}), ..., v_{s_{l_i}}, (v_{s_{l_i}}, v), v),$$

$$P_j = (v, (v, v_{r_2}), v_{r_2}, (v_{r_2}, v_{r_3}), ..., v_{r_{l_j}}, (v_{r_{l_j}}, v), v).$$

Объединим эти циклы в точке v и получим новый цикл

$$P_{i,j} = (v, (v, v_{s_2}), v_{s_2}, (v_{s_2}, v_{s_3}), ..., v_{s_{l_i}}, (v_{s_{l_i}}, v), v, \\ (v, v_{r_2}), v_{r_2}, (v_{r_2}, v_{r_3}), ..., v_{r_{l_j}}, (v_{r_{l_j}}, v), v).$$

Продолжим аналогично объединять циклы, пока не получим один общий цикл, включающий все ребра графа, причем каждое ровно один раз.

□

4.7.2 Гамильтонов цикл

Определение 4.7.3 . *Гамильтоновым циклом в графе G называется цикл, который проходит через все вершины графа ровно по одному разу.*

Определение 4.7.4 . *Граф G называется гамильтоновым графом, если в нем есть гамильтонов цикл.*

Определение гамильтонова графа похоже на определение эйлерова графа, но здесь нет такого удобного критерия для определения, является ли граф гамильтоновым. Как мы покажем позже, проверка графов на гамильтоновость является NP-полной задачей.

4.8 Задачи и алгоритмы

4.8.1 Остов минимального веса

Задача 4.8.1 . *Пусть имеется n населенных пунктов и между некоторыми из них проложены не асфальтированные дороги известной длины. Необходимо заасфальтировать часть дорог таким образом, чтобы была возможность проехать по новой дороге из любого пункта в любой другой, и при этом, чтобы суммарная длина заасфальтированных дорог была минимальной.*

Сформулируем задачу на языке графов.

Задача 4.8.2 . *Дан граф $G = (V, E)$ и весовая функция W, сопоставляющая каждму ребру положительное значение:*

$$W : E \to \mathbb{R}^+.$$

Будем считать весом любого подграфа H графа G суммарный вес его ребер:

$$W(H) = \sum_{e \in E(H)} W(e).$$

Требуется найти такой связный остовной подграф $G_1 = (V, E_1)$, чтобы его вес $W(G_1)$ был минимальным из весов всех остовных подграфов.

Очевидно, искомый подграф — дерево. С одной стороны он должен быть связен. С другой, если бы в нем имелся хотябы один цикл, мы смогли бы удалить любое ребро этого цикла и получить подграф, удовлетворяющий нашим требованиям, но меньшего веса.

Пример минимального остовного дерева для взвешенного графа приведен на рисунке 45.

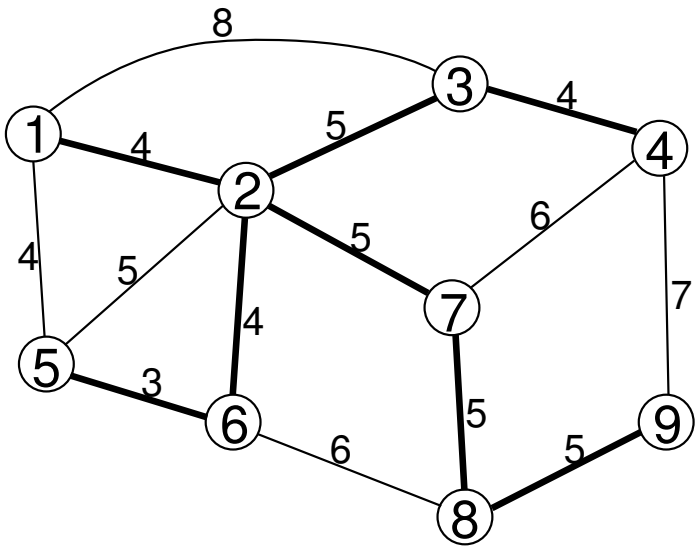

Рисунок 45: Остов миниального веса

Приведем алгоритм решения этой задачи.

Алгоритм 4.8.1 (Алгоритм Краскала).
Пусть $G = (V, E),\ |V| = n,\ W : E \to \mathbb{R}_+$.
1) $e_1 = \arg\min_{e \in E} W(e)$.
2) $i = 1; \quad G_i = (V, E_i) = O_n + e_1$.
3) $while(i < n - 1)\{$

$$e_{i+1} = \arg \min_{\substack{e \in E, \\ \text{в } G_i + e \text{ нет циклов}}} W(e);$$

$$G_{i+1} = G_i + e_{i+1};$$

$$i = i + 1;$$

$\}$

На каждом шаге этого алгоритма мы выбираем самое "легкое" ребро графа, при добавлении которого у нас не образуется циклов вместе с уже выбранными ранее ребрами. Процесс завершается, когда мы выберем $n-1$ ребро. Чтобы убедиться, что этот алгоритм решает задачу об остове минимального веса, нужно показать, что 1) мы действительно можем довести процесс до конца (всегда найдется такое $n-1$ ребро); 2) построенный граф будет деревом; 3) это дерево будет минимальным по весу из всех остовных деревьев G.

Теорема 4.8.1 . *Пусть граф G — связный граф. Тогда алгоритм Краскала строит минимальное остовное дерево.*

Доказательство. 1) Во-первых, покажем, что агоритм всегда позволяет построить граф G_{n-1}.

Пусть на некотором шаге i ($i < n-1$) мы не смогли выбрать следующее ребро e_{i+1}. Значит не существует ни одного такого ребра $e \in E$, что в $G_i + e$ нет циклов.

Граф G_i не имеет циклов по построению. Если бы граф G_i был связен, то по теореме 4.5.1 в нем было бы $n-1$ ребро. Так как $i < n-1$, граф G_i не является связным.

Пусть H произвольная компонента связности графа G. Поскольку граф G связен, в нем существует ребро $e = (u, v)$, соединяющее некоторую вершину u компоненты H с вершиной v, которая не принадлежит H.

По нашему предположению, $G_i + e$ будет иметь цикл. Этот цикл должен проходить по ребру e (иначе цикл был бы и в графе G_i). Но тогда этот цикл состоит из маршрута между вершинами u и v и ребра (u, v). То есть в графе G_i должен быть маршрут между вершинами u и v и они обе принадлежать одной компоненте связности H ?!

2) Покажем, что построенный в алгоритме граф G_{n-1} есть дерево.

Действительно, выбирая каждое ребро мы проверяем, что у нас не должно появляться циклов. Когда мы выберем $n-1$ ребро, по теореме 4.5.1 они образуют связный граф без циклов — остовное дерево графа G.

3) Наконец докажем, что построено дерево минимального веса.

Пусть $W_{\min}(G)$ — вес минимального остовного дерева графа G. Пусть

$$S = \{T \mid T = (V, E(T)) \text{ — остовное дерево грава } G, W(T) = W_{\min}(G)\}.$$

Пусть T^* остовное дерево менимального веса, которое имеет наибольшее число общих ребер с графом G_{n-1}:

$$T^* = \arg\max_{T \in S} |E(G_{n-1}) \cap E(T)|.$$

Будем доказывать от противного. Пусть G_{n-1} не является минимальным остовным деревом. Тогда T^* не совпадает с G_{n-1}.

Пусть e_i — первое ребро в G_{n-1}, которого нет в T^*:

$$e_i = (u, v) \in E(G_{n-1}) : e_i \notin E(T^*), \text{ и } \forall j < i \;\Rightarrow\; e_j \in E(T^*).$$

Рассмотрим граф $T^* + e_i$. В нем обязательно есть цикл C, так как T^* уже был связным графом — имел маршрут между вершинами u и v. Так как в графе G_{n-1} нет циклов, в цикле C есть хотябы одно ребро e', которого нет в G_{n-1}.

Рассмотрим граф $T^* + e_i - e'$. В этом графе нет циклов и он связен. Значит $T^* + e_i - e'$ остовное дерево. Вес $T^* + e_i - e'$ не может быть меньше веса минимального остовного дерева.

$$W(T^*) + W(e_i) - W(e') = W(T^* + e_i - e') \geq W(T^*). \qquad (\square)$$

Следовательно

$$W(e_i) \geq W(e'). \qquad (*)$$

Посмотрим с другой стороны. На i-том шаге алгоритма мы выбирали ребро e_i как ребро минимального веса из всех невыбранных ребер, не образующих циклов с ребрами e_j, $j < i$. Все ребра e_j, $j < i$ входят и в дерево T^*, а значит и ребро e' не образует с ними цикла. Таким образом, на i-том шаге алгоритма ребро e' было кандидатом на выбор и было бы выбрано, если бы $W(e') < W(e_i)$. Следовательно

$$W(e_i) \leq W(e'). \qquad (**)$$

Из неравенств $(*)$ и $(**)$ следует, что $W(e_i) = W(e')$ и по формуле $(\square)$

$$W(T^* + e_i - e') = W(T^*) + W(e_i) - W(e') = W(T^*) = W_{\min}(G).$$

То есть $T^* + e_i - e'$ является остовным деревом минимального веса. При этом у графа $T^* + e_i - e'$ на одно общее ребро с графом G_{n-1} больше, чем у T^*, что противоречит выбору T^*.

Противоречие вызвано предположением, что G_{n-1} не является деревом минимального веса. Следовательно G_{n-1}, полученный с помощью алгоритма Краскала, есть минимальное остовное дерево, что и требовалось доказать.
□

Замечание 4.8.1 . *Заметим, что алгоритм Краскала является алгоритмом полиномиальной сложности. Не будем доказывать это строго. Для этого потребовалось бы определять, какие операции мы считаем элементарными, описывать алгоритм в терминах этих операций и подсчитывать их число в худшем случае в зависимости от размеров задачи. Рассмотрим лишь возможность выполнения за полиномиальное время основных действий алгоритма.*

В основном цикле алгоритма ровно $n-1$ шаг. Самым трудоемким пунктом в этом цикле является выбор ребра, который можно разделить на два основных действия:

1) Выбор ребра e минимального веса.

2) Проверка, появится ли цикл в графе после добавления ребра e.

1) Можно однократно стоставить список ребер, упорядоченных по весу (алгоритм сортировки имеет сложность порядка $n \cdot \log n$). Тогда будем каждый раз брать самое легкое ребро и удалять его из списка. Всего за все итерации цикла мы в худшем случае переберем все ребра.

2) Пусть $e = (u, v)$ — ребро под вопросом. Необходимо определить, появится ли в графе $G_i + e$ цикл. Достаточно проверить, был ли в графе G_i маршрут между вершинами u и v.

Будем приписывать метки вершинам начиная с вершины u и далее все смежные с уже помеченными вершинами. Остановимся, если смежных с помеченными непомеченных вершин больше нет (рис. 46). Максимум нам удастся пометить $i+1$ вершину, так как больше соединено ребрами быть не может. По окончании процесса нам останется только проверить, есть ли метка у вершины v. Если есть, в графе $G_i + e$ присутствовал бы цикл. Если нет, ребро e может быть выбрано в качестве e_{i+1}.

Итого, нам потребуется порядка $n \cdot \log n$ действий, чтобы отсортировать ребра, не более m действий выбора легких ребер, для

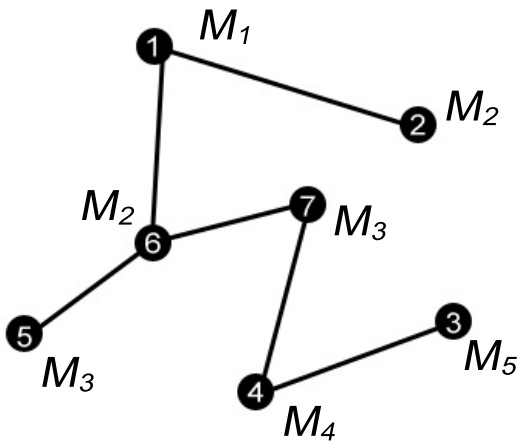

Рисунок 46: Проверка наличия маршрута между вершинами

каждого из которых нужно проверить наличие цикла, и порядка $\text{const} \cdot (n-1)$ *других действий*

Замечание 4.8.2 . *Этот алгоритм относится к так называемым жадным алгоритмам. Суть жадных алгоритмов состоит в том, чтобы на каждом шаге выбирать элемент с минимальным (или максимальным) значением некоторого параметра в итоге получая оптимальное решение.*

Пример 4.8.1 . *Легко убедиться, что жадный алгоритм применим не всегда. Например, рассмотрим задачу нахождения минимума суммы двух целых чисел с условием на значение их произведения.*

$$x + y \to \min,$$

$$x \cdot y = 30, \qquad x, y \in \{2, 3, 5, 6, 10, 15\}.$$

Очевидно, что решением этой задачи являются числа 5 *и* 6*. В то же время следуя жадной стратегии мы должны были бы сначала выбрать самое маленькое значение* $x = 2$*, а затем подобрать наименьшее значение для* y*, для которого выполняется условие* $x \cdot y = 30 - y = 15$*. Решение* $(2, 15)$ *очевидно не является минимумом.*

Жадный алгоритм здесь не дал правильного ответа.

4.9 NP-полные задачи теории графов

4.9.1 Задача коммивояжера

Рассмотрим задачу, которая по формулировке напоминает задачу об остове минимального веса, но по сложности принципиально отличается от нее.

Задача 4.9.1 . *Пусть имеется n городов, соединенные дорогами известной длины. Коммивояжер, живущий в одном из городов, должен объехать их все и вернуться домой таким образом, чтобы проделанный им путь был минимальным. Но есть еще одно ограничение. Поскольку продукция у коммивояжера не очень надежная, коммивояжер не может позволить себе вернуться в один из пройденных городов.*

Сформулируем задачу в терминах графов.

Задача 4.9.2 . *Дан граф* $G = (V, E)$ *и весовая функция* W*:*

$$W : E \to \mathbb{R}^+.$$

Требуется найти замкнутый маршрут минимального веса, который проходил бы по всем вершинам графа ровно по одному разу.

Задача разпадается на два вопроса: Во-первых, стоит вопрос, существует ли вообще замкнутый маршрут, который проходил бы по всем вершинам графа ровно по одному разу. Во-вторых, если такие маршруты существуют, требуется найти тот из них, который имеет минимальный вес.

Чтобы избежать такой неоднозначности вопроса, можно добавить к G ребра бесконечно большого веса между всеми несмежными вершинами. То есть мы договориваемся, что данный в задаче граф полный, а весовая функция может быть бесконечной для некоторых ребер. Тогда ответ на вопрос о существовании искомого маршрута всегда положителен и можно сосредоточиться на поиске минимума.

Покажем, что эта задача, если ее сформулировать в форме распознавания, будет NP-полной задачей. Для доказательства этого факта построим цепочку NP-полных задач, к каждой из которых полиномиально сводятся предыдущие.

Можно переформулировать задачу коммивояжера.

Задача 4.9.3 . *Дан полный граф $G = K_n$ и весовая функция W:*

$$W : E(K_n) \to \mathbb{R}^+.$$

Требуется найти гамильтонов цикл минимального веса.

Докажем NP-полноту задачи коммивояжера, построив последовательность полиномиальных сводимостей задач теории графов.

4.9.2 Задача о клике

Напомним, что клика — это максимальный по включению вершин полный подграф графа.

В оптимизационной форме задача о клике выглядит следующим образом:

Задача 4.9.4 . *Дан граф $G = (V, E)$. Необходимо найти наибольшую клику графа (клику графа с наибольшим для этого графа числом вершин).*

Используем обозначение $\varphi(G)$ — размер наибольшей клики в гарфе G. Тогда задачу о клики можно переформулировать в форме распознавания следующим образом.

Задача 4.9.5 (КЛИКА). *Дан граф $G = (V, E)$ и целое число $b > 0$. Правда ли, что $\varphi(G) \geq b$.*

То есть необходимо ответить на вопрос, существует ли в графе в графе G полный подграф с b вершинами.

Докажем NP-полноту задачи о клике, сведя к ней известную нам NP-полную задачу о выполнимости.

Задача 4.9.6 (ВЫПОЛНИМОСТЬ). *Дана конъюнктивная нормальная форма $A = D_1 \wedge D_2 \wedge ... \wedge D_k$. Требуется ответить на вопрос, является ли эта формула выполнимой (то есть, найдется ли набор значений переменных, чтобы формула A принимала значение 1.*

Теорема 4.9.1 . *ВЫПОЛНИМОСТЬ $\propto$ КЛИКА.*

Доказательство. Покажем, что задача о выполнимости сводится за полиномиальное время к задаче о клике.

Пусть нам поставлена следующая задача о выполнимости: дана конъюнктивная нормальная форма $A = D_1 \wedge D_2 \wedge ... \wedge D_k$, где D_i — дизъюнкт, $i = \overline{1,k}$. Построим граф $G = (V, E)$ следующим образом:

$$V(G) = \{(\alpha, i) \mid \alpha \text{ — литерал в } D_i\},$$
$$E(G) = \{((\alpha, i), (\beta, j)) \mid i \neq j, \alpha \neq \overline{\beta}\}.$$

1) Пусть A выполнима. Значит на определенном наборе значений переменных $A = 1$. Тогда в каждом дизъюнкте D_i найдется хотя бы один литерал $\alpha_i = 1$.

Рассмотрим вершины (α_i, i) и (α_j, j) при $i \neq j$. Поскольку на выбранном наборе значений переменных $\alpha_i = 1$ и $\alpha_j = 1$, то $\alpha_i \neq \overline{\alpha}_j$. Следовательно вершины (α_i, i) и (α_j, j) соеденины ребром в графе G. Таким образом множество вершин $\{(\alpha_i, i) \mid i = \overline{1,k}\}$ порождает полный подграф в графе G.

2) Пусть теперь в графе G есть клика размера k с вершинами $(\alpha_1, 1)$, $(\alpha_2, 2)$, $...(\alpha_k, k)$. Тогда $\alpha_i \neq \overline{\alpha}_j$, $\forall i, j = \overline{1,k}$. Слендовательно можно таким образом подобрать значения переменных, чтобы все α_i принимали значение 1 одновременно. Следовательно A — выполнима.

3) Итак, мы показали, что формула A является выполнимой тогда и только тогда, когда в графе G имеется клика размера k.

Осталось заметить, что построение графа G можно выполнить за полиномиальное время от размера СКНФ A.

□

Переформулированная в форме распознавания задача о клике принадлежит классу NP, так как, если ответ на задачу "да" и нам дано множество вершин, образующих полный подграф в графе G, мы можем за полиномиальное время проверить правда ли каждая из этих вершин смежна с каждой.

Следствие 4.9.2 . *Задача о клике является NP-полной задачей.*

4.9.3 Задача о вершинном покрытии

Пусть дан граф $G = (V, E)$.

Определение 4.9.1 . *Множество вершин $R \subseteq V$ называется вершинным покрытием графа G, если*

$$\forall e = (u, v) \in E : \quad \{u, v\} \cap R \neq \varnothing.$$

Теорема 4.9.3 . *Пусть $G = (V, E)$ — граф. Тогда $G_1 = (V_1, E_1)$ — полный подграф графа G тогда и только тогда, когда $V \setminus V_1$ — вершинное покрытие в графе $\overline{G}$.*

Доказательство. $\Rightarrow$) Пусть $G_1 = (V_1, E_1)$ — полный подграф графа G. В граф $\overline{G}$ входят те и только те ребра, которых нет в графе G. Следовательно подграф графа $\overline{G}$, порожденный множеством вершин V_1 не имеет ребер. Другими словами, если в графе $\overline{G}$ и найдется ребро, то хотябы один его конец будет принадлежать множеству $V \setminus V_1$, что и требовалось доказать.

$\Leftarrow$) Пусть $V \setminus V_1$ — вершинное покрытие в графе $\overline{G}$.

Для любых вершин $u, v \in V_1$ выполняется неравенство $u, v \cap (V \setminus V_1) = 0$. Следовательно, по определению вершинного покрытия между вершинами u и v нет ребра в графе $\overline{G}$. А значит ребро (u, v) присутствует в графе G.

Поскольку это выполняется для произвольных вершин из V_1, множество вершин V_1 порождает полный подграф в графе G.

□

Задача о вершинном покрытии в оптимизационной форме формулируется следующим образом.

Задача 4.9.7 . *Дан граф $G = (V, E)$. Найти вершинное покрытие G наименьшей мощности.*

Переформулируем задачу в форме распознавания

Задача 4.9.8 (ВЕРШИННОЕ ПОКРЫТИЕ). *Дан граф $G = (V, E)$ и число $b > 0$. Существует ли вершинное покрытие G мощности b.*

Следствие 4.9.4 . *КЛИКА $\propto$ ВЕРШИННОЕ ПОКРЫТИЕ.*

Доказательство. Для любого графа G мы можем построить дополнение и для него решить задачу о вершинном покрытии.

□

Задача о вершинном покрытии принадлежит классу NP, так как, если ответ на задачу "да" и нам дано множество из b вершин, мы можем перебрав все ребра убедиться, что хотябы один конец каждого из них лежит в данном множестве.

Следствие 4.9.5 . *Задача о вершинном покрытии является NP-полной задачей.*

4.9.4 Задача о гамильтоновом цикле

Как мы помним, гамильтонов цикл — это цикл проходящий через каждую вершину графа ровно один раз.

Задача 4.9.9 (ГАМИЛЬТОНОВ ЦИКЛ). *Дан граф $G = (V, E)$. Существует ли в графе G гамильтонов цикл?*

Теорема 4.9.6 . *ВЕРШИННОЕ ПОКРЫТИЕ $\propto$ ГАМИЛЬТОНОВ ЦИКЛ.*

Доказательство этой теоремы приводить не будем из-за его излишней громоздкости.

Задача о гамильтоновом цикле лежит в классе NP, поскольку, если ответ на задачу "да" и нам дана последовательность прохождения ребер гарфа G, мы можем за полиномиальное время убедиться, что вершины могут быть пройдены в этой последовательности по ребрам G.

Следствие 4.9.7 . *Задача о гамильтонове цикле является NP-полной задачей.*

4.9.5 Снова задача коммивояжера

Итак, задачу коммивояжера мы сформулировали следующим образом (Задача 4.9.3):

Дан полный граф $G = K_n$ и весовая функция W:

$$W : E(K_n) \to \mathbb{R}^+.$$

Требуется найти гамильтонов цикл минимального веса.

Существует также общий (несимметричный) случай задачи.

Задача 4.9.10 . *Пусть $G = (V, A)$ — полный ориентированный граф (то есть, ориентированный граф в которым любые две вершины связаны двумя противоположнонаправленными дугами). Весовая функция сопоставляет каждой дуге положительное число. Противоположные ребра могут иметь различный вес.*

Требуется найти гамильтонов контур минимального веса.

Переформулируем симметричную задачу в форме распознавания:

Задача 4.9.11 (ЗАДАЧА КОММИВОЯЖЕРА). *Дан полный граф $G = K_n$, весовая функция W и целое число $b > 0$. Существует ли в G гамильтонов цикл веса не привосходящего b.*

Теорема 4.9.8 . *ГАМИЛЬТОНОВ ЦИКЛ $\propto$ ЗАДАЧА КОММИВОЯЖЕРА.*

Доказательство. Пусть $G_1 = (V_1, E_1)$ граф, в котором необходимо проверить наличие гамильтонова цикла. Построим граф $G_2 = (V_2, E_2)$ и весовую функцию W следующим образом:

$$(u, v) \in E_1 \quad \Rightarrow \quad (u, v) \in E_2,\ W((u, v)) = 1;$$

$$(u, v) \notin E_1 \quad \Rightarrow \quad (u, v) \in E_2,\ W((u, v)) = 2.$$

Нетрудно видеть, что в G_1 есть гамильтонов цикл $\Leftrightarrow$ задача коммивояжера с графом G_2 и границей $b = |V|$ имеет ответ "да".
□

Следствие 4.9.9 . *Задача коммивояжера является NP-полной задачей.*

Следствие 4.9.10 . *Несимметричная задача коммивояжера является NP-полной задачей, так как симметричная задача коммивояжера является ее частным случаем.*

4.9.6 Алгоритм дерева

Как мы помним, для NP-полных задач не известно полиномиальных алгоритмов решения и поиск их представляется крайне сложной задачей.

Алгоритм дерева — это приближенный алгоритм решения, который применим к особому случаю симметричной задачи коммивояжера с неравенством треугольника:

Пусть $G = (V, E)$ — полный неориентированный граф с n вершинами, $W: E \to \mathbb{R}_+$ — функция весов ребер:

$$W((u, w)) \leq W((v, u)) + W((u, w)), \text{ для всех } u, v, w \in V. \qquad (36)$$

При изложении алгоритма дерева нам понадобится два вспомогательных алгоритмов: построение минимального остовного дерева (алгоритм 4.8.1) и алгоритм построения эйлерова цикла.

Алгоритм построения эйлерова цикла напоминает структуру доказательства теоремы 4.7.2 (рис. 47).

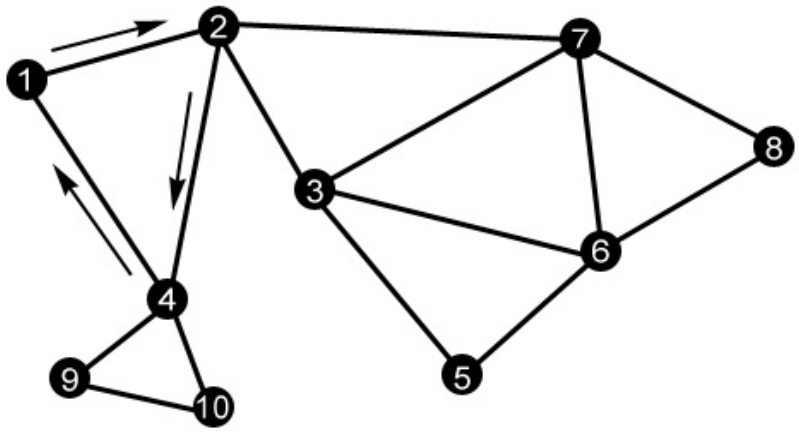

Рисунок 47: Поиск эйлерова цикла

Пусть дан эйлеров граф $G = (V, E)$ — связный и все вершины имеют четные степени.

Алгоритм 4.9.1 (Эйлеров цикл).
function ЭЙЛЕР (H, v_1) *{*
if ($\deg_H(v_1) = 0$*) {*
ЭЙЛЕР=(v_1*)*
} else {
1. Построить любой цикл $(v_1, v_2, ..., v_k, v_1)$*, который ни по одному ребру не проходит дважды:* $(v_i, v_{i+1}) \neq (v_j, v_{j+1})$, $1 \leq i, j \leq k$
2. $H = H - (v_1, v_2) - (v_2, v_3) - \cdots - (v_k, v_1)$
3. ЭЙЛЕР = (ЭЙЛЕР(H, v_1*), ЭЙЛЕР(*H, v_2*), ..., ЭЙЛЕР(*H, v_k*),* v_1*)*
}
}
main() {
v *= любая вершина из* V
ω *= ЭЙЛЕР(G,* v*)*
}

Замечание 4.9.1 . *Заметим, что параметр H в функции ЭЙЛЕР передается по ссылке. То есть все преобразования, которые производятся с графом в функции, (удаление ребер) оказывают воздействие на исходный объект G. Таким образом, если мы удалили некоторые ребра в одной из ветвей рекурсивных вызовов функции, то ребра будут отсутствовать для всех последующих вызовов, в том числе в других ветвях рекурсии.*

На вход алгоритма дерева подается граф $G = (V, E)$ и функция весов $W : E \to \mathbb{R}_+$, удовлетворяющая неравенству треугольника.

Алгоритм 4.9.2 (Алгоритм дерева).
1) $T = (V, E_1)$ *= Минимальное Остовное Дерево (G)*
2) H *= мультиграф, полученный удвоением ребер из дерева* T
3) ω *= ЭЙЛЕР(*H, v*), где* v *— любая из вершин* V
4) τ *= (Удалить повторение вершин (*ω*),* v*)*

Пример 4.9.1 . *Предположим, что мы уже нашли минимальное остовное дерево и удвоили его ребра. Результат изображен на рисунке 48. В ходе выполнения алгоритма поиска эйлерова цикла мы могли*

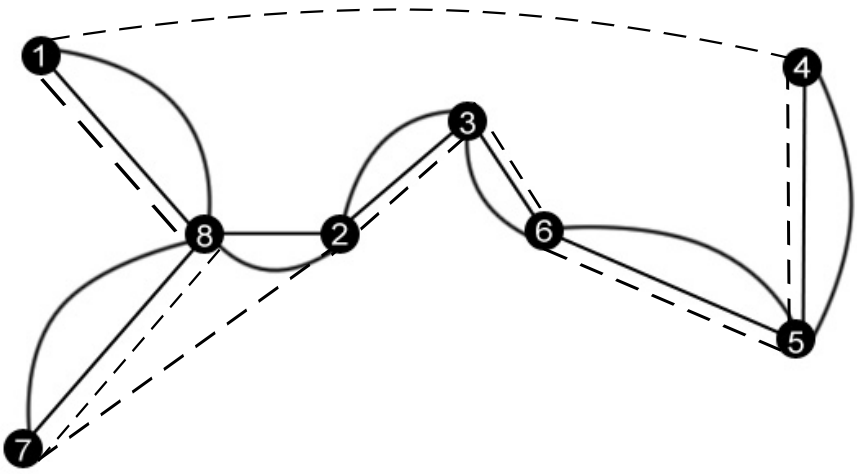

Рисунок 48: Построение гамильтонова цикла с помощью алгоритма дерева

обойти сначала, скажем, цикл (1,8,1). Тогда искомый цикл должен получиться в результате вызова функций в выражении
ω = *(ЭЙЛЕР(H,1), ЭЙЛЕР(H,8), 1).*

Вершина 1 к этому моменту уже имеет степень 0, так что ЭЙЛЕР(H,1) = (1).

ЭЙЛЕР(H,8) может найти, например, цикл (8, 2, 3, 2, 8). В этом случае мы выполним вызов (ЭЙЛЕР(H,8),ЭЙЛЕР(H,2),ЭЙЛЕР(H,3),ЭЙЛЕР(H,2),8). Здесь ЭЙЛЕР(H,2) = (2) в обоих случаях. ЭЙЛЕР(H,8) мог бы оказаться (8,7,8), а ЭЙЛЕР(H,3) — (3,6,5,4,5,6,3).

Подставим эти циклы в
(ЭЙЛЕР(H,8),ЭЙЛЕР(H,2),ЭЙЛЕР(H,3),ЭЙЛЕР(H,2),8)
и получим (8,7,8,2,3,6,5,4,5,6,3,2,8). Теперь подставим это вместо ЭЙЛЕР(H,8) в выражение для ω *и получим эйлеров цикл*

$$\omega = (1, 8, 7, 8, 2, 3, 6, 5, 4, 5, 6, 3, 2, 8, 1).$$

После удаления повторений в этой последовательности получим цикл $\tau = (1, 8, 7, 2, 3, 6, 5, 4, 1)$ *(на рисунке 48 изображен пунктирной линией). Это и есть результат работы алгоритма — гамильтонов*

цикл.

Степень близости веса полученного в алгоритме дерева гамильтонова цикла к оптимальному описывает следующая теорема.

Определение 4.9.2 . *Пусть $\tau_{opt}(G, W)$ — оптимальный цикл в задаче коммивояжера для графа G и функции весов W, а $\tau_A(G, W)$ — маршрут полученный с помощью некоторого алгоритма A.*

Тогда алгоритм A называют ε-приближенным, если

$$\frac{W(\tau_A(G,W)) - W(\tau_{opt}(G,W))}{W(\tau_{opt}(G,W))} \leq \varepsilon.$$

Утверждение 4.9.11 . *Алгоритм дерева — 1-приближенный алгоритм.*

Доказательство. Пусть $\tau_{opt}(G, W)$ — оптимальный цикл в задаче коммивояжера для графа G и функции весов W, а $\tau_T(G, W)$ — маршрут полученный с помощью алгоритма дерева. Так же предполагаем, что функция W удовлетворяет неравенству треугльника (36), поскольку в прочих случаях алгоритм дерева мы не рассматриваем.

При удалении любого ребра e из $\tau_{opt}(G, W)$ мы получим некоторое остовное дерево графа G. Вес этого дерева не может быть меньше веса минимального остовного дерева T графа G. Следовательно,

$$W(\tau_{opt}(G,W)) > W(\tau_{opt}(G,W) - e) \geq W(T).$$

С другой стороны $\tau_T(G, W)$ построен из T с удвоенными ребрами (и весом $2W(T)$) заменой последовательностей ребер $(v_{i_1}, v_{i_2}), (v_{i_2}, v_{i_3}), ..., (v_{i_{j-1}}, v_{i_j})$ на одиночные ребра (v_{i_1}, v_{i_j}). Из неравенства треугольника следует, что

$$W((v_{i_1}, v_{i_2})) + W((v_{i_2}, v_{i_3})) + \cdots + W((v_{i_{j-1}}, v_{i_j})) \geq W((v_{i_1}, v_{i_j})).$$

Таким образом, $W(\tau_T(G, W)) \leq 2 \cdot W(T)$ и следовательно $W(\tau_T(G, W)) \leq 2 \cdot W(\tau_{opt}(G, W))$.

Осталось записать формулу для ϵ-приближенности.

$$\frac{W(\tau_T(G,W)) - W(\tau_{opt}(G,W))}{W(\tau_{opt}(G,W))} \leq \frac{2W(\tau_{opt}(G,W)) - W(\tau_{opt}(G,W))}{W(\tau_{opt}(G,W))} = 1.$$

□

Хотя мы и не можем пока ответить на вопрос о существовании полиномиального алгоритма для NP-трудной задачи коммивояжера, в некоторых случаях (при выполнении дополнительных условий) мы можем находить приближенное решение этой задачи.

5 Математическая логика: Исчисления высказываний и предикатов

5.1 Исчисление высказываний

5.1.1 Пример задачи логики высказываний

Рассмотрим пример. Пусть мы хотим выяснить, является ли определенная последовательность рассуждений логически правильной.

Простые высказывания:
A = капиталовложения останутся постоянными;
B = возрастут правительственные расходы;
C = возрастет безработица;
D = налоги будут снижены.

На основе простых высказываний построим сложные высказывания:
$\mathcal{A} = A \supset (B \vee C)$,
$\mathcal{B} = \neg B \supset D$,
$\mathcal{C} = (D \wedge A) \supset \neg C$.

Здесь высказывание $\mathcal{A}$ можно прочитать как "если капиталовложения останутся постоянными, то возрастут правительственные расходы, или возрастет безработица". Аналогично можно читать и другие сложные высказывания.

Замечание 5.1.1 . *Для определенности здесь и далее простые высказывания будем обозначать большими латинскими буквами A, B, C,..., а сложные высказывания округлыми большими латинскими буквами $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$,...*

Определение 5.1.1 . *Здесь A, B, C — пропозициональные буквы, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ — пропозициональные формы, построенные из пропозициональных букв с помощью связок $\{\neg, \vee, \wedge, \supset, \equiv\}$ по правилам построения формул алгебры логики.*

Рассмотрим следующее рассуждение:

$$\frac{\mathcal{A}, \mathcal{B}, \mathcal{C}, A}{B} \quad \begin{matrix} \text{— посылки} \\ \text{— заключение} \end{matrix}$$

Определение 5.1.2 . *Высказывание $\mathcal{A}$ логически влечет высказывание $\mathcal{B}$ ($\mathcal{B}$ — следствие $\mathcal{A}$), если пропозициональная форма $\mathcal{A} \supset \mathcal{B}$ — тавтология.*

Определение 5.1.3 . *$\mathcal{A}$ логически эквивалентна $\mathcal{B}$, если $\mathcal{A} \equiv \mathcal{B}$ — тавтология.*

Итак, нам необходимо проверить, будет ли тавтологией следующая пропозициональная форма:

$$\underbrace{\mathcal{A} \wedge \mathcal{B} \wedge \mathcal{C} \wedge A}_{\text{конъюнкция посылок}} \supset \underbrace{B}_{\text{заключение}} = 1 \ ?$$

Прежде чем перейти непосредственно к вычислению этой формы, необходимо выполнить предварительную проверку непротиворечивости посылок. Если окажется, что посылки в нашем рассуждении не могут быть истинными в одно и то же время, то, исходя из предположения, что все они выполняются, мы не сможем получить достоверных результатов.

Кроме того, по определению импликации, если $\mathcal{A} = 0$, то $\mathcal{A} \supset \mathcal{B} = 1$ для любых значений $\mathcal{B}$, что совсем не будет свидетельствовать о правильном рассуждении. Это согласуется с одним из интуитивных логических законов: исходя из ложных посылок можно вывести как истинное, так и ложное заключение.

Проверим, выполнима ли формула $\mathcal{A} \wedge \mathcal{B} \wedge \mathcal{C} \wedge A$. Такую проверку для посылок нужно выполнять каждый раз.

$$\begin{aligned}
&(A \supset (B \vee C))(\overline{B} \supset D)(DA \supset \overline{C})A = \\
&= (\overline{A} \vee B \vee C)(B \vee D)(\overline{D} \vee \overline{A} \vee \overline{C})A = \\
&= (\overline{A}B \vee \overline{A}D \vee B \vee BD \vee CB \vee CD)(\overline{D}A \vee \overline{C}A) = \\
&= \overline{A}B\overline{D}A \vee \overline{A}B\overline{C}A \vee \overline{A}D\overline{D}A \vee \overline{A}D\overline{C}A \vee B\overline{D}A \vee B\overline{C}A \vee \\
&\vee BD\overline{D}A \vee BD\overline{C}A \vee CB\overline{D}A \vee CB\overline{C}A \vee CD\overline{D}A \vee CD\overline{C}A = \\
&= B\overline{D}A \vee B\overline{C}A \vee BD\overline{C}A \vee CB\overline{D}A = AB\overline{D} \vee AB\overline{C}
\end{aligned}$$

Легко убедиться, что, например, на наборе значений $A = 1,\ B = 1,\ C = 1,\ D = 0$ полученное выражение обращается в единицу. Таким

образом, $\mathcal{A} \wedge \mathcal{B} \wedge \mathcal{C} \wedge A$ — выполнимая функция, т.е. посылки непротиворечивы.

Вычислим теперь значение выражения $\mathcal{A} \wedge \mathcal{B} \wedge \mathcal{C} \wedge A \supset B$, воспользовавшись результатами предыдущих выкладок:

$$
\begin{aligned}
&(AB\overline{D} \vee AB\overline{C}) \supset B = \overline{AB\overline{D} \vee AB\overline{C}} \vee B = \\
&= \overline{AB\overline{D}} \cdot \overline{AB\overline{C}} \vee B = (\overline{A} \vee \overline{B} \vee D)(\overline{A} \vee \overline{B} \vee C) \vee B = \\
&= \overline{A} \vee \overline{A}\,\overline{B} \vee \overline{A}C \vee \overline{B} \vee \overline{B}C \vee D\overline{A} \vee D\overline{B} \vee DC \vee B = \\
&= (\overline{A} \vee \overline{B} \vee DC) \vee B = 1
\end{aligned}
$$

Исследуемая формула оказалась тавтологией. Значит, высказывание B действительно оказалось логическим следствием посылок $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ и A.

При этом мы нигде не обсуждаем истинность самих высказываний $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, A; возможно они неверны. Тем не менее, если они истинны, то верным окажется и заключение.

Заметим, что мы рассматривали истинность только логики рассуждений независимо от смысла, который мы приписываем элементарным высказываниям A, B, C, D. Если мы опеределим отличные значения для элементарных высказываний, истинность или ложность логики рассуждений останется неизменной. Если при новых определениях будут верны посылки, верным будет и заключение.

5.1.2 Формальные теории

Определение 5.1.4 . *Формальная теория — это совокупность четырех объектов:*

1) Алфавит $\mathfrak{A}$ — произвольное множество элементов, которые, в этом случае, называют символами.

2) Множество формул $\mathfrak{F}$ — некоторое множество слов в алфавите $\mathfrak{A}$:

$$\mathfrak{F} \subseteq \mathfrak{A}^* = \bigcup_{i=1}^{\infty} \mathfrak{A}^i.$$

3) Множество аксиом $\mathfrak{B}$ — некоторое множество формул: $\mathfrak{B} \subseteq \mathfrak{F}$.

4) Множество правил вывода $\mathfrak{R}$ — множество отношений R на множестве формул:

$$R \subseteq \underbrace{\mathfrak{F} \times \mathfrak{F} \times \cdots \times \mathfrak{F}}_{k+1}.$$

Определение 5.1.5 . *Пусть $R \in \mathfrak{R}$ — некоторое $k+1$-арное правило вывода. Если $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_k$, $\mathcal{A}_{k+1}$ — формулы из $\mathfrak{F}$ и $(\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_k, \mathcal{A}_{k+1}) \in R$, то говорят, что $\mathcal{A}_{k+1}$ непосредственно выводима из $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_k$ (или $\mathcal{A}_{k+1}$ непосредственное следствие $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_k$) по правилу вывода R.*

Определение 5.1.6 . *Вывод — это последовательность формул $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_n$, такая что для каждого $i \in \{1, ..., n\}$ верно одно из двух:*
a) $\mathcal{A}_i \in \mathfrak{B}$ — аксиома, или
b) $\mathcal{A}_i$ непосредственно выводима из формул $\mathcal{A}_{j_1}$, $\mathcal{A}_{j_2}$, ..., $\mathcal{A}_{j_k}$, $1 \leq j_s < i$, по некоторому правилу вывода данной формальной теории.

Определение 5.1.7 . *Говорят, что формула $\mathcal{A}$ формальной теории T является теоремой (выводима в формальной теории T), если для нее существует вывод $\mathcal{A}_1$, $\mathcal{A}_2$, ...,$\mathcal{A}_n = \mathcal{A}$.*

Пишут $\vdash_T \mathcal{A}$.

Определение 5.1.8 . *Аксиоматическая теория — формальная теория, в которой существует алгоритм, позволяющий для любой формулы $\mathcal{A}$ определить, является ли $\mathcal{A}$ аксиомой.*

Определение 5.1.9 . *Разрешимая теория — формальная теория, в которой существует алгоритм, позволяющий для любой формулы $\mathcal{A}$ определить, является ли $\mathcal{A}$ теоремой.*

Неразрешимая теория — теория не являющаяся разрешимой.

Определение 5.1.10 . *Пусть $\Gamma \subseteq \mathfrak{F}$ — некоторое множество формул теории T. Формула $\mathcal{A}$ выводима в формальной теории T из множества посылок Γ ($\mathcal{A}$ — следствие формул множества Γ), если для нее существует последовательность формул $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_n = \mathcal{A}$, такая что для каждого $i \in \{1, ..., n\}$ верно одно из трех:*
a) $\mathcal{A}_i \in \mathfrak{B}$ — аксиома,

b) $\mathcal{A}_i \in \Gamma$ *— посылка, или*
c) $\mathcal{A}_i$ *непосредственно выводима из формул* $\mathcal{A}_{j_1}$, $\mathcal{A}_{j_2}$, ..., $\mathcal{A}_{j_k}$, $1 \le j_s < i$, *по некоторому правилу вывода теории* T.

Такую последовательность формул будем называть выводом из посылок Γ. *Пишут* $\Gamma \vdash_T \mathcal{A}$.

Пример 5.1.1 . *Формальная теория* M:
1) Алфавит: $\mathfrak{A} = \{|\}$.
2) Формулы — последовательности из символа $|$ *произвольной длины:* $\mathfrak{F} = \mathfrak{A}^*$
3) Аксиома: $\mathfrak{B} = \{||\}$.
4) Правило вывода: $R = \{(\alpha, \beta) \mid \alpha$ *— формула,* $\beta = \alpha\alpha\}$.

Вывод:
1. $||$ *— аксиома.*
2. $||||$ *— из строки 1 по правилу вывода* R.
3. $||||||||$ *— из строки 2 по правилу вывода* R.

Таким образом, теоремами этой формальной теории являются последовательности из символа $|$ *длины* 2^t *для любого* $t \in \mathbb{N}$.

Формальная теория M *является аксиоматической и разрешимой.*

5.1.3 Формальная теория исчисление высказываний

Определим аксиоматическую теорию L — исчисление высказываний.

1. Алфавит: $\{A_1, A_2, ..., A_n, ..., \neg, \supset, (,)\}$.
2. Формулы: 1) A_i — формула, для любого $i \in \mathbb{N}$.
   2) Если $\mathcal{A}$ и $\mathcal{B}$ — формулы, то $\neg\mathcal{A}$ и $(\mathcal{A} \supset \mathcal{B})$ — формулы.
   3) Других формул нет.
3. Аксиомы: Пусть $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ — произвольные формулы. Тогда следующие формулы являются аксиомами:

   1^a $(\mathcal{A} \supset (\mathcal{B} \supset \mathcal{A}))$,
   2^a $((\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})) \supset ((\mathcal{A} \supset \mathcal{B}) \supset (\mathcal{A} \supset \mathcal{C})))$,
   3^a $((\neg\mathcal{B} \supset \neg\mathcal{A}) \supset ((\neg\mathcal{B} \supset \mathcal{A}) \supset \mathcal{B}))$.

Замечание 5.1.2 . *Три указанные формулы являются схемами аксиом: при подстановке в одну из схем произвольных формул теории* L *мы получим аксиому. Таким образом существует счетное число аксиом*

теории L. В дальнейшем мы будем говорить об аксиоме 1, 2 или 3, имея в виду одну из аксиом по схеме 1, 2 или 3 соответственно.

4. Правило вывода modus ponens (MP):

$$\mathrm{MP} = \{(\mathcal{A}, (\mathcal{A} \supset \mathcal{B}), \mathcal{B}) \mid \mathcal{A}, \mathcal{B} \in \mathfrak{F}\}.$$

То есть, если $\mathcal{A}$ и $\mathcal{B}$ произвольные формулы теории L, то $\mathcal{B}$ непосредственно выводима из формул $\mathcal{A}$ и $(\mathcal{A} \supset \mathcal{B})$ по правилу вывода modus ponens.

Замечание 5.1.3 . *В алфавите формальной теории L нет символов $\vee$, $\wedge$, $\equiv$. Тем не менее, мы можем использовать эти символы, как краткую форму записи в некоторых сложных выражениях:*

$$(\mathcal{A} \vee \mathcal{B}) = (\neg\mathcal{A} \supset \mathcal{B}),$$
$$(\mathcal{A} \wedge \mathcal{B}) = (\neg(\mathcal{A} \supset \neg\mathcal{B})),$$
$$(\mathcal{A} \equiv \mathcal{B}) = ((\mathcal{A} \supset \mathcal{B}) \wedge (\mathcal{B} \supset \mathcal{A})) = (\neg((\mathcal{A} \supset \mathcal{B}) \supset \neg(\mathcal{B} \supset \mathcal{A}))).$$

Для простоты иногда будем опускать внешние скобки в записи формул: вместо $(\mathcal{A} \supset \mathcal{B})$ — писать $\mathcal{A} \supset \mathcal{B}$.

Далее в этой главе мы будем говорить только о формальной теории исчисление высказываний; для простоты, вместо $\vdash_L \mathcal{A}$ будем писать $\vdash \mathcal{A}$.

Лемма 5.1.1 . *Пусть, $\mathcal{A}$ — формула. Тогда $\vdash (\mathcal{A} \supset \mathcal{A})$.*

Доказательство. Построим вывод для формулы $(\mathcal{A} \supset \mathcal{A})$:

1. $((\mathcal{A} \supset ((\mathcal{A} \supset \mathcal{A}) \supset \mathcal{A})) \supset ((\mathcal{A} \supset (\mathcal{A} \supset \mathcal{A})) \supset (\mathcal{A} \supset \mathcal{A})))$ — аксиома 2 после подстановки вместо $\mathcal{A}$ формулы $\mathcal{A}$, вместо $\mathcal{B}$ — $(\mathcal{A} \supset \mathcal{A})$ и вместо $\mathcal{C}$ — $\mathcal{A}$.

2. $(\mathcal{A} \supset ((\mathcal{A} \supset \mathcal{A}) \supset \mathcal{A}))$ — аксиома 1 после подстановки $\mathcal{A} \leftarrow \mathcal{A}$, $\mathcal{B} \leftarrow (\mathcal{A} \supset \mathcal{A})$.

3. $((\mathcal{A} \supset (\mathcal{A} \supset \mathcal{A})) \supset (\mathcal{A} \supset \mathcal{A}))$ — применение правила вывода MP к строкам 2 и 1.

4. $(\mathcal{A} \supset (\mathcal{A} \supset \mathcal{A}))$ — аксиома 1 после подстановки $\mathcal{A} \leftarrow \mathcal{A}$, $\mathcal{B} \leftarrow \mathcal{A}$.

5. $(\mathcal{A} \supset \mathcal{A})$ — MP к строкам 3 и 4.

□

Замечание 5.1.4 . *Формулы, для которых мы доказали, что они являются теоремами данной формальной теории, в дальнейшем можно использовать в выводах наравне с аксиомами. Действительно, раз для такой формулы существует вывод, мы всегда можем привести его как составную часть других выводов, но не будем этого делать для краткости.*

Пример 5.1.2 . *Рассмотрим формализацию следующего рассуждения. "Если Вася баскетболист,то Вася высокий. Вася баскетболист. Следовательно Вася высокий."*

Обозначим через A высказывание "Вася баскетболист" и через B — "Вася высокий". Тогда наше рассуждение будет имеь вид:

$$\frac{A \supset B,\ A}{B}.$$

Чтобы убедиться, что оно верно, проверим, является ли формула $(A \supset B)A \supset B$ тавтологией.

$$(A \supset B)A \supset B = \overline{(\overline{A} \vee B)A} \vee B = \overline{\overline{A} \vee B} \vee \overline{A} \vee B = A\overline{B} \vee \overline{A} \vee B =$$
$$= (A \vee \overline{A})(\overline{B} \vee \overline{A}) \vee B = \overline{B} \vee \overline{A} \vee B = 1.$$

Теперь рассмотрим это рассуждение с точки зрения теории L.

1) A — посылка.

2) $A \supset B$ — посылка.

3) B — MP к 1 и 2.

Следовательно, $A, A \supset B \vdash B$.

Согласно определению 5.1.2, если формула $\mathcal{B}$ получина по правилу вывода MP из предыдущих, формула $\mathcal{B}$ является их логическим следствием.

5.1.4 Теоремы исчисления высказываний

Теорема 5.1.2 (Теорема о дедукции). *Пусть $\mathcal{A}$ и $\mathcal{B}$ — формулы, Γ — некоторое множество формул теории L. Тогда из $\Gamma, \mathcal{A} \vdash \mathcal{B}$ следует, что $\Gamma \vdash (\mathcal{A} \supset \mathcal{B})$.*

Доказательство. Пусть $\mathcal{B}_1$, $\mathcal{B}_2$, ..., $\mathcal{B}_n = \mathcal{B}$ — вывод $\mathcal{B}$ из множества посылок $\Gamma \cup \{\mathcal{A}\}$. Докажем по индукции, что $\Gamma \vdash \mathcal{A} \supset \mathcal{B}_i$, $i = 1, 2, ..., n$.

1) Пусть $i = 1$. $\mathcal{B}_1$ — аксиома, или $\mathcal{B}_1 \in \Gamma$, или $\mathcal{B}_1 = \mathcal{A}$.

a) Пусть $\mathcal{B}_1$ — аксиома.

1. $\mathcal{B}_1$ — аксиома.
2. $\mathcal{B}_1 \supset (\mathcal{A} \supset \mathcal{B}_1)$ — аксиома 1.
3. $\mathcal{A} \supset \mathcal{B}_1$ — MP из 1 и 2.

Следовательно $\vdash \mathcal{A} \supset \mathcal{B}_1 \Rightarrow \Gamma \vdash \mathcal{A} \supset \mathcal{B}_1$.

b) Пусть $\mathcal{B}_1 \in \Gamma$.

1. $\mathcal{B}_1$ — посылка.
2. $\mathcal{B}_1 \supset (\mathcal{A} \supset \mathcal{B}_1)$ — аксиома 1.
3. $\mathcal{A} \supset \mathcal{B}_1$ — MP из 1 и 2.

Следовательно $\mathcal{B}_1 \vdash \mathcal{A} \supset \mathcal{B}_1 \Rightarrow \Gamma \vdash \mathcal{A} \supset \mathcal{B}_1$.

c) Пусть $\mathcal{B}_1 = \mathcal{A}$. По лемме 5.1.1 $\vdash \mathcal{A} \supset \mathcal{A} = \mathcal{A} \supset \mathcal{B}_1 \Rightarrow \Gamma \vdash \mathcal{A} \supset \mathcal{B}_1$.

2) Пусть для всех $l < k$ верно, что $\Gamma \vdash \mathcal{A} \supset \mathcal{B}_l$. Докажем, что $\Gamma \vdash \mathcal{A} \supset \mathcal{B}_k$. $\mathcal{B}_k$ — аксиома, или $\mathcal{B}_k \in \Gamma$, или $\mathcal{B}_k = \mathcal{A}$, или $\mathcal{B}_k$ получена по правилу вывода modus ponens. Первые три случая доказываются аналогично соответствующим пунктам базы индукции.

d) $\mathcal{B}_k$ — результат применения MP к формулам $\mathcal{B}_i$ и $\mathcal{B}_j$, $i, j < k$. Тогда, $\Gamma \vdash \mathcal{A} \supset \mathcal{B}_i$, $\Gamma \vdash \mathcal{A} \supset \mathcal{B}_j$ и $\mathcal{B}_j = \mathcal{B}_i \supset \mathcal{B}_k$. Построим вывод:

1. $(\mathcal{A} \supset (\mathcal{B}_i \supset \mathcal{B}_k)) \supset ((\mathcal{A} \supset \mathcal{B}_i) \supset (\mathcal{A} \supset \mathcal{B}_k))$ — аксиома 2 ($\mathcal{A} \leftarrow \mathcal{A}$, $\mathcal{B} \leftarrow \mathcal{B}_i$, $\mathcal{C} \leftarrow \mathcal{B}_k$).
2. $(\mathcal{A} \supset (\mathcal{B}_i \supset \mathcal{B}_k))$ — посылка.
3. $((\mathcal{A} \supset \mathcal{B}_i) \supset (\mathcal{A} \supset \mathcal{B}_k))$ — MP из 2 и 1.
4. $(\mathcal{A} \supset \mathcal{B}_i)$ — посылка.
5. $(\mathcal{A} \supset \mathcal{B}_k)$ — MP из 4 и 3.

Таким образом, $(\mathcal{A} \supset (\mathcal{B}_i \supset \mathcal{B}_k)), (\mathcal{A} \supset \mathcal{B}_i) \vdash (\mathcal{A} \supset \mathcal{B}_k)$. Следовательно, $\Gamma \vdash (\mathcal{A} \supset \mathcal{B}_k)$. Следовательно, $\Gamma \vdash (\mathcal{A} \supset \mathcal{B})$.

□

Пример 5.1.3 . *С помощью теоремы о дедукции мы могли бы гораздо проще доказать лемму 5.1.1:*

1) $\mathcal{A}$ — посылка.

2) $\mathcal{A} \supset \mathcal{A}$ — по теореме о дедукции, поскольку $\mathcal{A} \vdash \mathcal{A}$.

Следствие 5.1.3 . *Пусть, $\mathcal{A}$, $\mathcal{B}$ и $\mathcal{C}$ — некоторые формулы теории L. Тогда $\mathcal{A} \supset \mathcal{B},\ \mathcal{B} \supset \mathcal{C} \ \vdash\ \mathcal{A} \supset \mathcal{C}$.*

Доказательство.

1) $\mathcal{A}$ — посылка.
2) $\mathcal{A} \supset \mathcal{B}$ — посылка.
3) $\mathcal{B}$ — MP из 1 и 2.
4) $\mathcal{B} \supset \mathcal{C}$ — посылка.
5) $\mathcal{C}$ — MP из 3 и 4.

Таким образом, $\mathcal{A}$, $\mathcal{A} \supset \mathcal{B}$, $\mathcal{B} \supset \mathcal{C} \vdash \mathcal{C}$. Следовательно, по теореме о дедукции $\mathcal{A} \supset \mathcal{B}$, $\mathcal{B} \supset \mathcal{C} \vdash \mathcal{A} \supset \mathcal{C}$.
□

Следствие 5.1.4 . *Пусть, $\mathcal{A}$, $\mathcal{B}$ и $\mathcal{C}$ — некоторые формулы теории L. Тогда $\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C}),\ \mathcal{B} \ \vdash\ \mathcal{A} \supset \mathcal{C}$.*

Доказательство.

1) $\mathcal{A}$ — посылка.
2) $\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})$ — посылка.
3) $\mathcal{B} \supset \mathcal{C}$ — MP из 1 и 2.
4) $\mathcal{B}$ — посылка.
5) $\mathcal{C}$ — MP к 4 и 3.

$\mathcal{A}$, $\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})$, $\mathcal{B} \vdash \mathcal{C}$. Следовательно, по теореме о дедукции $\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})$, $\mathcal{B} \vdash \mathcal{A} \supset \mathcal{C}$.
□

Лемма 5.1.5 (Список теорем). *Пусть $\mathcal{A}$ и $\mathcal{B}$ — произвольные формулы теории L. Тогда следующие формулы являются теоремами L:*

1. $\neg\neg\mathcal{B} \supset \mathcal{B}$ — закон отрицания отрицания.
2. $\mathcal{B} \supset \neg\neg B$ — закон отрицания отрицания.
3. $\neg\mathcal{A} \supset (\mathcal{A} \supset \mathcal{B})$ — из ложных посылок можно вывести что угодно.
4. $(\neg\mathcal{B} \supset \neg\mathcal{A}) \supset (\mathcal{A} \supset \mathcal{B})$ — доказательство от противного.
5. $(\mathcal{A} \supset \mathcal{B}) \supset (\neg\mathcal{B} \supset \neg\mathcal{A})$ — доказательство от противного.
6. $\mathcal{A} \supset (\neg\mathcal{B} \supset \neg(\mathcal{A} \supset \mathcal{B}))$ — из истинных посылок нельзя вывести ложное заключение.

7. $(\mathcal{A} \supset \mathcal{B}) \supset ((\neg\mathcal{A} \supset \mathcal{B}) \supset \mathcal{B})$ *— если* $\mathcal{B}$ *выводима и из* $\mathcal{A}$ *и из его отрицания, то* $\mathcal{B}$ *истинна независимо от посылок.*

Доказательство. Докажем, например, для формул 1 и 3.

1.

1) $(\neg\mathcal{B} \supset \neg\neg\mathcal{B}) \supset ((\neg\mathcal{B} \supset \neg\mathcal{B}) \supset \mathcal{B})$ — аксиома 3.
2) $\neg\neg\mathcal{B}$ — посылка.
3) $\neg\neg\mathcal{B} \supset (\neg\mathcal{B} \supset \neg\neg\mathcal{B})$ — аксиома 1.
4) $\neg\mathcal{B} \supset \neg\neg\mathcal{B}$ — MP из 2 и 3.
5) $(\neg\mathcal{B} \supset \neg\mathcal{B}) \supset \mathcal{B}$ — MP из 4 и 1.
6) $\neg\mathcal{B} \supset \neg\mathcal{B}$ — лемма 5.1.1.
7) $\mathcal{B}$ — MP из 6 и 5.

Таким образом $\neg\neg\mathcal{B} \vdash \mathcal{B}$ и по теореме о дедукции $\vdash \neg\neg\mathcal{B} \supset \mathcal{B}$.

3.

1) $\mathcal{A} \supset (\neg\mathcal{B} \supset \mathcal{A})$ — аксиома 1.
2) $\neg\mathcal{A} \supset (\neg\mathcal{B} \supset \neg\mathcal{A})$ — аксиома 1.
3) $\mathcal{A}$ — посылка.
4) $\neg\mathcal{B} \supset \mathcal{A}$ — MP из 3 и 1.
5) $\neg\mathcal{A}$ — посылка.
6) $\neg\mathcal{B} \supset \neg\mathcal{A}$ — MP из 5 и 2.
7) $(\neg\mathcal{B} \supset \neg\mathcal{A}) \supset ((\neg\mathcal{B} \supset \mathcal{A}) \supset \mathcal{B})$ — аксиома 3.
8) $((\neg\mathcal{B} \supset \mathcal{A}) \supset \mathcal{B})$ — MP из 6 и 7.
9) $\mathcal{B}$ — MP из 4 и 8.

Таким образом $\mathcal{A}$, $\neg\mathcal{A} \vdash \mathcal{B}$. Тогда по теореме о дедукции, $\neg\mathcal{A} \vdash \mathcal{A} \supset \mathcal{B}$ и $\vdash \neg\mathcal{A} \supset (\mathcal{A} \supset \mathcal{B})$.

Остальные пункты утверждения предлагаются читателю для самостоятельного доказательства.

□

5.1.5 Теорема о полноте исчисления высказываний

Лемма 5.1.6 . *Аксиомы теории* L *являются тавтологиями.*

Доказательство. 1)

$$(\mathcal{A} \supset (\mathcal{B} \supset \mathcal{A})) = \overline{\mathcal{A}} \vee \overline{\mathcal{B}} \vee \mathcal{A} = 1.$$

2)

$$((\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})) \supset ((\mathcal{A} \supset \mathcal{B}) \supset (\mathcal{A} \supset \mathcal{C}))) =$$
$$= \overline{\overline{\mathcal{A}} \vee \overline{\mathcal{B}} \vee \mathcal{C}} \vee \overline{\overline{\mathcal{A}} \vee B} \vee \overline{\mathcal{A}} \vee \mathcal{C} = \mathcal{A}\mathcal{B}\overline{\mathcal{C}} \vee \mathcal{A}\overline{\mathcal{B}} \vee \overline{\mathcal{A}} \vee \mathcal{C} =$$
$$= (\mathcal{A}\mathcal{B} \vee \mathcal{C})(\overline{\mathcal{C}} \vee \mathcal{C}) \vee (\mathcal{A} \vee \overline{\mathcal{A}})(\overline{\mathcal{B}} \vee \overline{\mathcal{A}}) = \mathcal{A}\mathcal{B} \vee \mathcal{C} \vee \overline{\mathcal{B}} \vee \overline{\mathcal{A}} =$$
$$= (\mathcal{A} \vee \overline{\mathcal{A}})(\mathcal{B} \vee \overline{\mathcal{A}}) \vee \mathcal{C} \vee \overline{\mathcal{B}} = \mathcal{B} \vee \overline{\mathcal{A}} \vee \mathcal{C} \vee \overline{\mathcal{B}} = 1$$

3)

$$((\neg\mathcal{B} \supset \neg\mathcal{A}) \supset ((\neg\mathcal{B} \supset \mathcal{A}) \supset \mathcal{B}) = \overline{\mathcal{B} \vee \overline{\mathcal{A}}} \vee \overline{\mathcal{B} \vee \mathcal{A}} \vee \mathcal{B} =$$
$$= \overline{\mathcal{B}}\mathcal{A} \vee \overline{\mathcal{B}}\,\overline{\mathcal{A}} \vee B = \overline{\mathcal{B}}(\mathcal{A} \vee \overline{\mathcal{A}}) \vee B = \overline{\mathcal{B}} \vee \mathcal{B} = 1.$$

□

Лемма 5.1.7 . *Любая теорема теории L является тавтологией.*

Доказательство. Пусть для формулы $\mathcal{A}$ теории L существует вывод $\mathcal{A}_1$, $\mathcal{A}_2$, ..., $\mathcal{A}_n = \mathcal{A}$. Докажем по индукции, что $\mathcal{A}_i$ — тавтология, $i = \overline{1,n}$. По определению вывода, $\mathcal{A}_1$ является аксиомой и по лемме 5.1.6 она является тавтологией.

Пусть формулы $\mathcal{A}_i$ — тавтология, $\forall i = \overline{1,k}$. Рассмотрим $\mathcal{A}_{k+1}$: она является аксиомой, или получена по правилу вывода modus ponens из предыдущих формул. Если $\mathcal{A}_{k+1}$ — аксиома, то она является тавтологией по лемме 5.1.6. Пусть $\mathcal{A}_{k+1}$ получена по правилу modus ponens из формул $\mathcal{A}_i$ и $\mathcal{A}_j = \mathcal{A}_i \supset \mathcal{A}_{k+1}$, $i, j \leq k$. По индукционному предположению, $\mathcal{A}_i$ и $\mathcal{A}_j$ — тавтологии. Пусть, на некотором наборе значений пропозициональных букв формула $\mathcal{A}_{k+1}$ принимает значение 0. Тогда на этом наборе

$$\mathcal{A}_j = \mathcal{A}_i \supset \mathcal{A}_{k+1} = 1 \supset 0 = 0. \;\; ?!$$

Противоречие. Следовательно $\mathcal{A}_{k+1}$ — тавтология. Следовательно $\mathcal{A}$ — тавтология.

□

Пусть $\mathcal{A}$ — формула, в которую входят пропозициональные буквы B_1, B_2, ..., B_k. Когда нам будет необходимо подчеркнуть, от каких значений

рассматривается $\mathcal{A}$, будем писать $\mathcal{A}(\alpha_1, \alpha_2, ..., \alpha_k)$, имея в виду, что вместо пропозициональной буквы B_i подставляется значение α_i, $i = \overline{1,k}$.

Как и раньше, будем использовать знак степени для обозначения наличия или отсутствия отрицания:

$$\mathcal{B}^\sigma = \begin{cases} \mathcal{B}, & \sigma = 1, \\ \neg\mathcal{B}, & \sigma = 0. \end{cases}$$

Лемма 5.1.8 . *Пусть B_1, B_2, ..., B_k — все пропозициональные буквы, которые входят в формулу $\mathcal{A}$, и пусть σ_1, σ_2, ..., σ_k — логические константы. Тогда*

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_k^{\sigma_k} \vdash \mathcal{A}^{\mathcal{A}(\sigma_1,\sigma_2,...,\sigma_k)}.$$

Доказательство. Докажем по индукции по числу j логических связок в формуле $\mathcal{A}$.

1) Пусть $j = 0$. Тогда $\mathcal{A} = B_1$, $\mathcal{A}(\sigma_1) = \sigma_1$, $\mathcal{A}^{\mathcal{A}(\sigma_1)} = B_1^{\sigma_1}$.

$$B^{\sigma_1} \vdash B^{\sigma_1} = \mathcal{A}^{\mathcal{A}(\sigma_1)}.$$

2) Пусть утверждение верно для любого $j < n$. Докажем для случая n логических связок в $\mathcal{A}$. Формула $\mathcal{A}$, в зависимости от своей последней логической связки, может быть одного из двух видов: $\mathcal{A} = \neg\mathcal{A}_1$ или $\mathcal{A} = (\mathcal{A}_1 \supset \mathcal{A}_2)$.

а) Пусть $\mathcal{A} = \neg\mathcal{A}_1$. Рассмотрим два варианта

Пусть $\mathcal{A}_1(\sigma_1, ..., \sigma_k) = 0$. По индукционному предположению $B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_k^{\sigma_k} \vdash \neg\mathcal{A}_1$. $\neg\mathcal{A}_1 = \mathcal{A} = \mathcal{A}^{\neg\mathcal{A}_1(\sigma_1,...,\sigma_k)} = \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)}$. Следовательно,

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_k^{\sigma_k} \vdash \mathcal{A}^{\mathcal{A}(\sigma_1,\sigma_2,...,\sigma_k)}.$$

Пусть $\mathcal{A}_1(\sigma_1, ..., \sigma_k) = 1$.

$$\mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} = \mathcal{A}^{\neg\mathcal{A}_1(\sigma_1,...,\sigma_k)} = \neg\mathcal{A} = \neg\neg\mathcal{A}_1.$$

По индукционному предположению $B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_k^{\sigma_k} \vdash \mathcal{A}_1$. Построим вывод:

1. $\mathcal{A}_1 \supset \neg\neg\mathcal{A}_1$ — пункт 2 из леммы 5.1.5.

2. $\mathcal{A}_1$ — посылка.
3. $\neg\neg\mathcal{A}_1$ — MP из 2 и 1.

Таким образом, $\mathcal{A}_1 \vdash \neg\neg\mathcal{A}$ и

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_k^{\sigma_k} \ \vdash \ \neg\neg\mathcal{A}_1 = \mathcal{A}^{\mathcal{A}(\sigma_1,\sigma_2,...,\sigma_k)}.$$

b) Пусть $\mathcal{A} = (\mathcal{A}_1 \supset \mathcal{A}_2)$. Рассмотрим несколько вариантов.

Пусть $\mathcal{A}_1(\sigma_1, ..., \sigma_k) = 0$, $\mathcal{A}_2(\sigma_1, ..., \sigma_k) = 0$. Тогда

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \neg\mathcal{A}_1, \qquad B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \neg\mathcal{A}_2.$$

$$\mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} = \mathcal{A}^{0\supset 0} = \mathcal{A} = (\mathcal{A}_1 \supset \mathcal{A}_2).$$

Рассмотрим вывод:

1. $\neg\mathcal{A}_1$ — посылка.
2. $\neg\mathcal{A}_1 \supset (\mathcal{A}_1 \supset \mathcal{A}_2)$ — пункт 3 из леммы 5.1.5.
3. $(\mathcal{A}_1 \supset \mathcal{A}_2)$ — MP из 1 и 2.

Таким образом, $\neg\mathcal{A}_1 \vdash (\mathcal{A}_1 \supset \mathcal{A}_2) = \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} \Rightarrow$

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)}.$$

Пусть $\mathcal{A}_1(\sigma_1, ..., \sigma_k) = 0$, $\mathcal{A}_2(\sigma_1, ..., \sigma_k) = 1$. Тогда

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \neg\mathcal{A}_1, \qquad B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}_2.$$

$$\mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} = \mathcal{A}^{0\supset 1} = \mathcal{A} = (\mathcal{A}_1 \supset \mathcal{A}_2).$$

Рассмотрим вывод:

1. $\neg\mathcal{A}_1$ — посылка.
2. $\neg\mathcal{A}_1 \supset (\mathcal{A}_1 \supset \mathcal{A}_2)$ — пункт 3 из леммы 5.1.5.
3. $(\mathcal{A}_1 \supset \mathcal{A}_2)$ — MP из 1 и 2.

Таким образом, $\neg\mathcal{A}_1 \vdash (\mathcal{A}_1 \supset \mathcal{A}_2) = \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} \Rightarrow$

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)}.$$

Пусть $\mathcal{A}_1(\sigma_1, ..., \sigma_k) = 1$, $\mathcal{A}_2(\sigma_1, ..., \sigma_k) = 0$. Тогда

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}_1, \qquad B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \neg\mathcal{A}_2.$$

$$\mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} = \mathcal{A}^{1\supset 0} = \mathcal{A}^0 = \neg(\mathcal{A}_1 \supset \mathcal{A}_2).$$

Рассмотрим вывод:

1. $\mathcal{A}_1$ — посылка.
2. $\neg\mathcal{A}_2$ — посылка.
3. $\mathcal{A}_1 \supset (\neg\mathcal{A}_2 \supset \neg(\mathcal{A}_1 \supset \mathcal{A}_2))$ — пункт 6 из леммы 5.1.5.
4. $\neg\mathcal{A}_2 \supset \neg(\mathcal{A}_1 \supset \mathcal{A}_2)$ — MP из 1 и 3.
5. $\neg(\mathcal{A}_1 \supset \mathcal{A}_2)$ — MP из 2 и 4.

Таким образом, $\mathcal{A}_1, \neg\mathcal{A}_2 \vdash \neg(\mathcal{A}_1 \supset \mathcal{A}_2) = \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} \Rightarrow$

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)}.$$

Пусть $\mathcal{A}_1(\sigma_1, ..., \sigma_k) = 1$, $\mathcal{A}_2(\sigma_1, ..., \sigma_k) = 1$. Тогда

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}_1, \qquad B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}_2.$$

$$\mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} = \mathcal{A}^{1\supset 1} = \mathcal{A} = (\mathcal{A}_1 \supset \mathcal{A}_2).$$

Рассмотрим вывод:

1. $\mathcal{A}_2$ — посылка.
2. $\mathcal{A}_2 \supset (\mathcal{A}_1 \supset \mathcal{A}_2)$ — аксиома 1.
3. $(\mathcal{A}_1 \supset \mathcal{A}_2)$ — MP из 1 и 2.

Таким образом, $\mathcal{A}_2 \vdash (\mathcal{A}_1 \supset \mathcal{A}_2) = \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)} \Rightarrow$

$$B_1^{\sigma_1}, ..., B_k^{\sigma_k} \ \vdash \ \mathcal{A}^{\mathcal{A}(\sigma_1,...,\sigma_k)}.$$

Утверждение доказано.

□

Пример 5.1.4 . *Пусть,* $\mathcal{A} = B_1 \supset B_2$. *Если* $\sigma_1 = 1$, $\sigma_2 = 0$, *то* $\mathcal{A}(\sigma_1, \sigma_2) = 0$. *Следовательно, по лемме 5.1.8,*

$$B_1, \neg B_2 \ \vdash \ \neg(B_1 \supset B_2).$$

Покажем, что это действительно так. Вывод:

1) $B_1 \supset (\neg B_2 \supset \neg(B_1 \supset B_2))$ — *лемма 5.1.5 пункт 6.*

2) B_1 — *посылка.*

3) $\neg B_2$ — *посылка.*

4) $(\neg B_2 \supset \neg(B_1 \supset B_2))$ — *MP из 2 и 1.*

5) $\neg(B_1 \supset B_2)$ — *MP из 3 и 4.*

Что и требовалось доказать.

Теорема 5.1.9 (о полноте исчисления высказываний). *Формула $\mathcal{A}$ формальной теории L есть тавтология тогда и только тогда, когда $\mathcal{A}$ — теорема теории L.*

Доказательство. Достаточность следует из леммы 5.1.7.

Покажем необходимость. Пусть $\mathcal{A}$ — тавтология. Пусть B_1, B_2, ..., B_k — все пропозициональные буквы, которые входят в формулу $\mathcal{A}$, и пусть σ_1, σ_2, ..., σ_k — логические константы. По лемме 5.1.8

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_{k-1}^{\sigma_{k-1}}, B_k^{\sigma_k} \vdash \mathcal{A}^{\mathcal{A}(\sigma_1,\sigma_2,...,\sigma_k)}.$$

Поскольку $\mathcal{A}$ — тавтология, $\mathcal{A}^{\mathcal{A}(\sigma_1,\sigma_2,...,\sigma_k)} = \mathcal{A}$ для любых наборов σ_i. Тогда,

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_{k-1}^{\sigma_{k-1}}, B_k \vdash \mathcal{A},$$
$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_{k-1}^{\sigma_{k-1}}, \neg B_k \vdash \mathcal{A}.$$

По теореме о дедукции,

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_{k-1}^{\sigma_{k-1}} \vdash B_k \supset \mathcal{A},$$

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_{k-1}^{\sigma_{k-1}} \vdash \neg B_k \supset \mathcal{A}.$$

Построим вывод:

1) $B_k \supset \mathcal{A}$ — посылка.
2) $\neg B_k \supset \mathcal{A}$ — посылка.
3) $(B_k \supset \mathcal{A}) \supset ((\neg B_k \supset \mathcal{A}) \supset \mathcal{A})$ — теорема 7 из леммы 5.1.5.
4) $((\neg B_k \supset \mathcal{A}) \supset \mathcal{A})$ — MP из 1 и 3.
5) $\mathcal{A}$ — MP из 2 и 4.

Следовательно,

$$B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_{k-1}^{\sigma_{k-1}} \vdash \mathcal{A}.$$

Таким образом, мы избавились от B_k в списке посылок для вывода формулы $\mathcal{A}$. Повторяя процесс, мы можем показать, что $B_1^{\sigma_1}, B_2^{\sigma_2}, ..., B_{k-2}^{\sigma_{k-2}} \vdash \mathcal{A}$ и так далее, пока не получим, что $\vdash \mathcal{A}$.
□

Замечание 5.1.5 . *Формальная теория L является разрешимой, поскольку для любой формулы мы можем проверить, является ли она тавтологией и, следовательно, теоремой L.*

Определение 5.1.11 . *Формальная теория T с операцией $\neg$ непротиворечива, если в ней не существует такой формулы $\mathcal{A}$, что $\vdash_T \mathcal{A}$ и $\vdash_T \neg\mathcal{A}$.*

Следствие 5.1.10 . *Формальная теория L непротиворечива.*

Доказательство. Если $\vdash \mathcal{A}$, то $\mathcal{A}$ — тавтология. Тогда $\neg\mathcal{A}$ не является тавтологией и по теореме 5.1.9: $\nvdash \neg\mathcal{A}$.
□

5.1.6 Независимость аксиом исчисления высказываний

Определение 5.1.12 . *Пусть $\mathfrak{B}$ — множество аксиом формальной теории T. Тогда $X \subseteq \mathfrak{B}$ — независимое множество аксиом, если существует такая формула $\mathcal{A} \in X$, что $\mathcal{A}$ не может быть выведена в T без использования аксиом из $\mathfrak{B}$:* $\mathfrak{B} \setminus X \nvdash \mathcal{A}$.

Теорема 5.1.11 (о независимости схем аксиом теории L). *Каждая из схем аксиом 1,2,3 теории L независима.*

Замечание 5.1.6 . *Имеется в виду, что множество всех аксиом по схеме i ($i = 1$, 2, или 3) независимо.*

Доказательство. 1) Докажем независимость схемы 1:

$$(\mathcal{A} \supset (\mathcal{B} \supset \mathcal{A})).$$

Поскольку вывод формул в формальной теории L происходит в соответствии с правилами вывода и независим от наших знаний от приписанной операциям логике, мы можем сопоставить символам $\supset$ и

$\neg$ любые значения. Определим таблицы следующим образом:

A	$\neg A$
0	1
1	1
2	0

A	B	$A \supset B$
0	0	0
0	1	2
0	2	2
1	0	2
1	1	2
1	2	0
2	0	0
2	1	0
2	2	0

Можно показать непосредственной проверкой, что, при таком задании операций, значения всех аксиом по схемам 2 и 3 всегда будет 0.

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$((\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})) \supset ((\mathcal{A} \supset \mathcal{B}) \supset (\mathcal{A} \supset \mathcal{C})))$
0	0	0	$(0 \supset 0) \supset (0 \supset 0) = 0 \supset 0 = 0$
0	0	1	$(0 \supset (0 \supset 1)) \supset ((0 \supset 0) \supset (0 \supset 1)) =$ $= (0 \supset 2) \supset (0 \supset 2) = 2 \supset 2 = 0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
2	2	2	$(2 \supset (2 \supset 2)) \supset ((2 \supset 2) \supset (2 \supset 2)) =$ $= (2 \supset 0) \supset (0 \supset 0) = 0 \supset 0 = 0$

$\mathcal{A}$	$\mathcal{B}$	$\mathcal{C}$	$((\neg\mathcal{B} \supset \neg\mathcal{A}) \supset ((\neg\mathcal{B} \supset \mathcal{A}) \supset \mathcal{B})$
0	0	0	$(1 \supset 1) \supset ((1 \supset 0) \supset 0) = 2 \supset (2 \supset 0) = 2 \supset 0 = 0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
2	2	2	$(0 \supset 0) \supset ((0 \supset 2) \supset 2) = 0 \supset (2 \supset 2) = 0 \supset 0 = 0$

Рассмотрим, как работает правило вывода modus ponens: если $\mathcal{A} = 0$ и $\mathcal{A} \supset \mathcal{B} = 0$, то согласно нашему описанию операции $\supset$ будет верно и $\mathcal{B} = 0$. При этом аксиома 1 может принимать ненулевые значения:

$$(A \supset (B \supset A))\Big|_{A=1,B=2} = 1 \supset (2 \supset 1) = 1 \supset 0 = 2.$$

Следовательно, пользуясь правилом вывода modus ponens из аксиом 2 и 3 нельзя получить любую аксиому схемы 1.

2) Доказательство независимости схемы аксиом 2 аналогично доказательству предыдущего пункта. Определим таблицы для операций $\neg$ и $\supset$ следующим образом:

A	$\neg A$
0	1
1	0
2	1

A	B	$A \supset B$
0	0	0
0	1	2
0	2	1
1	0	0
1	1	2
1	2	0
2	0	0
2	1	0
2	2	0

Можно показать непосредственной проверкой, что значения аксиом 1 и 3, при таком определении, всегда 0, а также, что modus ponens сохраняет 0. В то же время

$$\begin{aligned}&((\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})) \supset ((\mathcal{A} \supset \mathcal{B}) \supset (\mathcal{A} \supset \mathcal{C})))\big|_{A=B=0,C=1} = \\ &= ((0 \supset (0 \supset 1)) \supset ((0 \supset 0) \supset (0 \supset 1))) = ((0 \supset 2) \supset (0 \supset 2)) = \\ &= 1 \supset 1 = 2.\end{aligned}$$

Следовательно, пользуясь правилом вывода modus ponens из аксиом 1 и 3 нельзя получить любую аксиому схемы 2.

3) Докажем независимость схемы 3:

$$((\neg\mathcal{B} \supset \neg\mathcal{A}) \supset ((\neg\mathcal{B} \supset \mathcal{A}) \supset \mathcal{B})).$$

Рассмотрим оператор h, сопоставляющий произвольной формуле $\mathcal{A}$ формулу $h(\mathcal{A})$, полученную удалением всех символов $\neg$. Тогда, для любой аксиомы $\mathcal{A}$ по схеме 1 или 2, $h(\mathcal{A})$ также будет аксиомой по схеме, соответственно, 1 или 2. Следовательно, по лемме 5.1.7, для любой аксиомы $\mathcal{A}$ по схеме 1 или 2, $h(\mathcal{A})$ — тавтология.

Пусть $\mathcal{A}$ и $\mathcal{A} \supset \mathcal{B}$ таковы, что $h(\mathcal{A})$ и $h(\mathcal{A} \supset \mathcal{B})$ — тавтологии. Тогда $h(\mathcal{A}) \supset h(\mathcal{B}) = h(\mathcal{A} \supset \mathcal{B})$ — тавтология и $h(\mathcal{B})$, по свойствам операции

импликации, тоже тавтология. Следовательно, h от любой формулы, выводимой из аксиом 1 и 2, — тавтология.

В то же время,

$$h(((\neg B \supset \neg A) \supset ((\neg B \supset A) \supset B)))\Big|_{A=B=0} =$$

$$= ((B \supset A) \supset ((B \supset A) \supset B))\Big|_{A=B=0} =$$
$$= ((0 \supset 0) \supset ((0 \supset 0) \supset 0) = 1 \supset (1 \supset 0) = 1 \supset 0 = 0.$$

То есть оператор h от аксиомы по схеме 3 не обязательно тавтология. Следовательно, пользуясь правилом вывода modus ponens из аксиом 1 и 2 нельзя получить любую аксиому схемы 3.
□

5.2 Исчисление предикатов

5.2.1 Пример задачи логики предикатов

Рассмотрим пример.

Пусть есть следующее рассуждение. Посылки:
(1) Всякий, кто находится в здравом уме, может понимать математику.
(2) Ни один из сыновей Гегеля не может понимать математику.
(3) Сумасшедшие не допускаются к голосованию.

Заключение:
(4) Никто из сыновей Гегеля не допускается к голосованию.

Формализуем рассуждение.
$A(x)$ — x находится в здравом уме. Здесь x предметная переменная, A — предикат;
$B(x)$ — x может понимать математику;
$C(x, y)$ — x есть сын y;
$D(x)$ — x допускается к голосованию.

С помощью этих предикатов можно переписать посылки и заключение:
(1) $\mathcal{A} = \forall x(A(x) \supset B(x))$;
(2) $\mathcal{B} = \forall x(C(x, \text{Гегель}) \supset \neg B(x))$;
(3) $\mathcal{C} = \forall x(\neg A(x) \supset \neg D(x))$;
(4) $\mathcal{D} = \forall x(C(x, \text{Гегель}) \supset \neg D(x))$.
Здесь $\forall x$ — квантор всеобщности, а "Гегель" — предметная константа.

Как и раньше, будем говорить, что рассуждение верно, если коньюнкция посылок импликация заключение истинна. В нашем примере

$$\underbrace{\mathcal{A} \wedge \mathcal{B} \wedge \mathcal{C}}_{\text{конъюнкция посылок}} \quad \supset \quad \underbrace{\mathcal{D}.}_{\text{заключение}}$$

Можно отметить следующие отличия от исчисления высказываний:

1) Вместо пропозициональных букв, принимающих всего два значения, используются предикаты, зависящие от переменных, которые принимают значения из некоторого множества объектов.

Пример 5.2.1 . *Если раньше мы могли использовать элементарное утверждение $A =$ "Вася — баскетболист", которое могло быть только истинно или ложно, то теперь мы бы написали $A(x)$ — "x —*

баскетболист" имея в виду, что $A(x)$ принимает значение 1 для всех x из заданного множества, для которых верно свойство "быть баскетболистом".

2) Появились кванторы $\forall x$ и $\exists x$ для записи выражений "для любых" и "существует такое" соответственно.

3) В исчислении высказываний формула была логическим законом, если принимала значение "истина" на любом наборе логических констант, подставляемых вместо пропозициональных букв. Теперь мы используем предикаты с переменными из некоторого множества, так что проверить, что формула истинна на всех наборах не так просто. Кроме того, одной и той же формуле можно приписать различные области определения и по-разному определять истинное или ложное значение входящих в нее предикатов на данных аргументах. Здесь нам будет необходимо новое понятие — интерпретация.

Опишем интерпретацию для предикатов из нашего примера. Пусть M — произвольное множество; область интерпретации. A_M, B_M, D_M — его произвольные подмножества. C_M — произвольное подмножество $M \times M$. $m \in M$ — произвольный элемент.

Будем полагать, что в формуле

$$(\forall x(A(x) \supset B(x))) \wedge (\forall x(C(x, \text{Гегель}) \supset \neg B(x)))\wedge \\ \wedge (\forall x(\neg A(x) \supset \neg D(x))) \supset (\forall x(C(x, \text{Гегель}) \supset \neg D(x))) \quad (37)$$

предикатам A, B, C и D сопоставлены соответственно множества A_M, B_M, C_M и D_M, "Гегелю" сопоставлен элемент m, а x принимает произвольные значения из M.

Тогда $A(x) = 1$ тогда и только тогда, когда $x \in A_M$. Аналогично для $B(x)$ и $D(x)$. $C(x, \text{Гегель}) = 1$ тогда и только тогда, когда $(x, m) \in C_M$.
$\forall x(A(x) \supset B(x)) = 1$, если $A_M \subseteq B_M$.
$\forall x A(x) = 1$, если $A_M = M$.

Будем говорить, что M вместе с сопоставленными предикатам A, B, C, D подмножествами и соответствующим "Гегелю" элементом m есть интерпретация нашей формулы. Если задана интерпретация, то формула в этой интерпретации получает значение "истина" или "ложь".

В общем случае безотносительно интерпретации говорить значении формулы исчисления предикатов бессмысленно.

Пример 5.2.2 . *Рассмотрим следующую интерпретацию формулы* (37)*: Область интерпретации* $M = \mathbb{N}$.

$$A(x) = \begin{cases} 1, & x \vdots 4, \\ 0, & \textit{иначе.} \end{cases}$$

$$B(x) = \begin{cases} 1, & x \vdots 2, \\ 0, & \textit{иначе.} \end{cases}$$

$$C(x,y) = \begin{cases} 1, & x \textit{ и } y \textit{ — взаимнопросты}, \\ 0, & \textit{иначе.} \end{cases}$$

$$D(x) = \begin{cases} 1, & x \vdots 100, \\ 0, & \textit{иначе.} \end{cases}$$

$$\textit{Гегель} = 10.$$

Легко убедиться, что в данной интерпретации и посылки и заключение истинны. Например, поскольку в данной интерпретации A_M *— множество всех чисел кратных четырем, а* B_M *— множество всех чисел кратных двум, то* $A_M \subset B_M$ *и* $\forall x(A(x) \supset B(x)) = 1$.

Следовательно $\mathcal{A} \wedge \mathcal{B} \wedge \mathcal{C} \supset \mathcal{D} = 1$ *и наше рассуждение является верным в данной интерпретации.*

Замечание 5.2.1 . *Заметим, что наше исходное словесное рассуждение на счет сыновей Гегеля не является интерпретацией, поскольку не определена область интерпретации* M.

Определение 5.2.1 . *Будем говорить, что формула тождественно истинна (логически общезначима), если ее значение "истина" в любой интерпретации*

В общем случае определение, является ли формула логически общезначимой, — алгоритмически неразрешимая задача. Тем не менее, в некоторых частных случаях можно доказать, что формула является тождественно истинной. Покажем это для формулы (37).

Пусть существует интерпретация, в которой наша формула принимает значение "ложь":

$$(\forall x(A(x) \supset B(x))) \wedge (\forall x(C(x, \text{Гегель}) \supset \neg B(x))) \wedge$$
$$\wedge (\forall x(\neg A(x) \supset \neg D(x))) \supset (\forall x(C(x, \text{Гегель}) \supset \neg D(x))) = 0.$$

Тогда, в этой интерпретации все посылки нашего рассуждения истинны, а заключение ложно. Таким образом существует x_o:

$$\begin{aligned} C(x_o, \text{Гегель}) \supset \neg D(x_o) = 0 \quad &\Rightarrow \quad C(x_o, \text{Гегель}) = 1, \quad D(x_o) = 1. \\ \neg A(x_o) \supset \neg D(x_o) = 1 \quad &\Rightarrow \quad A(x_o) = 1. \\ A(x_o) \supset B(x_o) = 1 \quad &\Rightarrow \quad B(x_o) = 1. \\ C(x_o, \text{Гегель}) \supset \neg B(x_o) = 1 \quad &\Rightarrow \quad B(x_o) = 0. \end{aligned}$$

Противоречие доказывает, что такого x_o не существует, что и требовалось доказать.

5.2.2 Формальная теория исчисление предикатов

Определим формальную теорию исчисление предикатов.

1. Алфавит.

1) Предметные переменные: x_1, x_2, ..., x_n, ...

2) Предметные константы: a_1, a_2, ..., a_n, ...

3) Функциональные символы: f_1^1, f_2^1, ..., f_1^2, f_2^2, ... f_1^n, f_2^n, ...

4) Предикатные символы: A_1^1, A_2^1, ..., A_1^2, A_2^2, ... A_1^n, A_2^n, ...

5) Квантор всеобщности: $\forall x$, где x — предметная переменная.

Символы логических операций: $\supset$, $\neg$.

Скобки и запятая:), (, ,.

2. Формулы.

Формула определяется с использованием понятия "терм".

Определение 5.2.2 .

1) x_i и a_i — термы.

2) Если t_1, t_2, ..., t_n — термы и f_i^n — функциональный символ, то $f_i^n(t_1, t_2, ..., t_n)$ — терм.

3) Других термов нет.

Определение 5.2.3 .

1) Пусть t_1, t_2, ..., t_n — термы, A_i^n — предикатный символ. Тогда $A_i^n(t_1, t_2, ..., t_n)$ — формула.

2) Если $\mathcal{A}$ и $\mathcal{B}$ — формулы и x — предметная переменная, то $\neg\mathcal{A}$, $\forall x\mathcal{A}$ и $(\mathcal{A} \supset \mathcal{B})$ — формулы.

3) Других формул нет.

Определение 5.2.4 . *Областью действия квантора $\forall x$ в формуле $\forall x\mathcal{A}$ называют подформулу $\mathcal{A}$.*

Переменная x_i называется связанной, если она входит в квантор $\forall x_i$, или в область его действия. Иначе переменная x_i называется свободной.

Пример 5.2.3 .

$$\neg A_1^2(x_1, a_1) \wedge \forall x_2(\exists x_3 A_1^2(x_1, f_1^2(x_2, x_3)) \supset (A_1^2(x_2, x_1) \vee A_1^2(x_2, a_1))).$$

В этой формуле переменные x_1 свободные, а переменные x_2 и x_3 — связанные.

Определение 5.2.5 . *Если в формуле нет свободных переменных, она называется замкнутой.*

Пример 5.2.4 . $(\forall x_1(A_1^1(x_1) \supset A_2^1(x_1)) \wedge A_1^1(a_1)) \supset A_2^1(a_1)$ *— замкнутая формула.*

В дальнейшем будем писать $\mathcal{A}(x_i)$, подразумевая, что x_i — свободная переменная формулы $\mathcal{A}$.

Определение 5.2.6 . *Будем говорить, что терм t — свободный для переменной x_i в формуле $\mathcal{A}(x_i)$, если в t нет такой переменной x_j, что переменная x_i в $\mathcal{A}(x_i)$ попадает в область действия квантора $\forall x_j$.*

Пример 5.2.5 . *Пусть $\mathcal{A}(x_1) = \forall x_2 A_1^2(x_1, x_2)$. Тогда терм $t' = x_3$ — свободный для x_1 в $\mathcal{A}(x_1)$, терм $t'' = f_1^2(x_2, x_3)$ — не является свободным для x_1 в $\mathcal{A}(x_1)$.*

Будем обозначать $\mathcal{A}(t)$ формулу, полученную из $\mathcal{A}(x_i)$ подстановкой терма t вместо всех вхождений x_i. При этом иногда специально оговаривается, что t — свободный для переменной x_i в формуле $\mathcal{A}(x_i)$.

3. Аксиомы.

Как и в исчислении предикатов, введем счетное множество аксиом через схемы, подставляя в которые любые формулы получим аксиому формальной теории. Пусть $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ — произвольные формулы. Тогда следующие формулы являются аксиомами:

1^a $(\mathcal{A} \supset (\mathcal{B} \supset \mathcal{A}))$.

2^a $((\mathcal{A} \supset (\mathcal{B} \supset \mathcal{C})) \supset ((\mathcal{A} \supset \mathcal{B}) \supset (\mathcal{A} \supset \mathcal{C})))$.

3^a $((\neg\mathcal{B} \supset \neg\mathcal{A}) \supset ((\neg\mathcal{B} \supset \mathcal{A}) \supset \mathcal{B}))$.

4^a $(\forall x_i \mathcal{A}(x_i) \supset \mathcal{A}(t))$, где t — терм, свободный для x_i в формуле $\mathcal{A}$. Если в $\mathcal{A}$ нет свободной переменной x_i, аксиома принимает вид $\forall x_i \mathcal{A} \supset \mathcal{A}$.

5^a $(\forall x_i(\mathcal{A} \supset \mathcal{B}) \supset (\mathcal{A} \supset \forall x_i \mathcal{B}))$, где в $\mathcal{A}$ нет свободной переменной x_i.

Если больше аксиом нет, получаем исчисление предикатов первого порядка. Если, кроме указанных, добавлены еще аксиомы, — формальную теорию первого порядка.

4. Правила вывода.

1) Правило вывода modus ponens (MP): если $\mathcal{A}$ и $\mathcal{B}$ произвольные формулы, то $\mathcal{B}$ непосредственно выводима из формул $\mathcal{A}$ и $\mathcal{A} \supset \mathcal{B}$ по правилу вывода modus ponens.

2) Правило обобщения (Gen): если $\mathcal{A}$ — произвольная формула, то из $\mathcal{A}$ по правилу обобщения непосредственно выводима формула $\forall x_i \mathcal{A}$.

$$\text{Gen} = \{(\mathcal{A}, \forall x_i \mathcal{A}) \mid \mathcal{A} \text{ — формула}\}.$$

Замечание 5.2.2 . *В алфавит не входят символы* $\vee$, $\wedge$, $\equiv$, $\exists x$. *Тем не менее, мы можем использовать эти символы, как краткую форму записи в некоторых сложных выражениях:*

$$\mathcal{A} \vee \mathcal{B} = (\neg\mathcal{A} \supset \mathcal{B}),$$
$$\mathcal{A} \wedge \mathcal{B} = (\neg(\mathcal{A} \supset \neg\mathcal{B})),$$
$$\mathcal{A} \equiv \mathcal{B} = ((\mathcal{A} \supset \mathcal{B}) \wedge (\mathcal{B} \supset \mathcal{A})) = (\neg((\mathcal{A} \supset \mathcal{B}) \supset \neg(\mathcal{B} \supset \mathcal{A}))),$$
$$\exists x_i \mathcal{A} = \neg\forall x_i \neg\mathcal{A}.$$

5.2.3 Интерпретация

Понятие интерпретации позволяет поставить в соответствие формуле некоторый смысл. Для этого каждому символу алфавита, который

используется в формуле, интерпретация сопоставляет некоторый объект, как описано ниже.

1) Для предметных переменных в интерпретации задается произвольное множество M — область интерпретации. Каждая предметная переменная может принимать произвольное значение из M.

2) В интерпретации каждой предметной константе сопоставляется единственный элемент из M.

3) Каждому функциональному символу f_i^n в интерпретации сопоставляется функция

$$f_i^n \ : \ \underbrace{M \times M \times \cdots \times M}_{n} \ \to \ M.$$

Здесь n — число аргументов функции, а i — номер функции с данным числом аргументов.

Функциональные символы в частности и термы в общем позволяют косвенно ссылаться на элемент в M.

4) Каждому предикатному символу A_i^n в интерпретации сопоставляется функция

$$A_i^n \ : \ \underbrace{M \times M \times \cdots \times M}_{n} \ \to \ \{0,1\}.$$

Другими словами, каждому предикатному символу A_i^n сопоставляется n-арное отношение на множестве M.

Предикатный символ представляет элементарное высказывание, утверждающее, что некоторые n объектов находятся в некотором отношении.

Пример 5.2.6 . *Запишем уравнение* $y = x^2 + 2x + 1$ *на языке предикатов. Область интерпретации* $M = \mathbb{R}$. *В уравнениисть одно бинарное отношение "=" и действия умножения и сложения над элементами из области интерпретации.*

Пусть $A_1^2(x_1, x_2) = 1 \quad \Leftrightarrow \quad x_1 = x_2$. *Пусть* $f_1^2(x_1, x_2) = x_1 + x_2$ *и* $f_2^2(x_1, x_2) = x_1 \cdot x_2$. *Пусть* $c_1 = 1$.

Тогда $\mathcal{A}(y, x) = A_1^2(y, f_1^2(f_2^2(x, x), f_1^2(f_2^2(f_1^2(c_1, c_1), x), c_1))) = 1$ *тогда и только тогда, когда* $y = x^2 + 2x + 1$.

Можно было бы упростить запись использовав другую интерпретацию. Например, введя $f_1^1(x) = x^2 + 2x + 1$, *нашу формулу можно будет записать как* $\mathcal{A}(y, x) = A_1^2(y, f_1^1(x))$.

5) Рассмотрим, как интерпретируются формулы, построенные из символов описанных выше с использованием логических операций и квантора всеобщности. Будем обозначать $\mathcal{A}_I$ формулу, полученную из $\mathcal{A}$ заменой символов исчисления предикатов на соответствующие отношения, операции и константы интерпретации I.

Пусть $\mathcal{A}(x_{i_1}, x_{i_2}, ..., x_{i_k})$ — формула, $x_{i_1}, x_{i_2}, ..., x_{i_k}$ — все ее свободные переменные. I — интерпретация формулы $\mathcal{A}$ с областью интерпретации M.

Для каждого предикатного символа мы знаем, какая функция ему сопоставляется интерпретацией. Пусть формуле $\mathcal{A}$ в интерпретации I соответствует функция

$$\mathcal{A}_I \ : \ \underbrace{M \times M \times \cdots \times M}_{k} \ \to \ \{0, 1\}.$$

Тогда будем полагать, что

$$(\neg\mathcal{A}(x_{i_1}, x_{i_2}, ..., x_{i_k}))_I = 1 \quad \Longleftrightarrow \quad \mathcal{A}_I(x_{i_1}, x_{i_2}, ..., x_{i_k}) = 0,$$

$$(\forall x_{i_t}\mathcal{A}(x_{i_1}, ..., x_{i_{t-1}}, x_{i_t}, x_{i_{t+1}}, ..., x_{i_k}))_I = 1 \quad \Longleftrightarrow$$
$$\Longleftrightarrow \mathcal{A}_I(x_{i_1}, ..., x_{i_{t-1}}, x, x_{i_{t+1}}, ..., x_{i_k}) = 1, \forall x \in M,$$

если квантор берется по свободной переменной формулы $\mathcal{A}$, и

$$(\forall x_s\mathcal{A}(x_{i_1}, x_{i_2}, ..., x_{i_k}))_I = \mathcal{A}_I(x_{i_1}, x_{i_2}, ..., x_{i_k}),$$

если у $\mathcal{A}$ нет свободной переменной x_s.

Пусть $\mathcal{A}(x_{i_1}, x_{i_2}, ..., x_{i_k})$ и $\mathcal{B}(x_{j_1}, x_{j_2}, ..., x_{j_l})$ — формулы, которым в интерпретации I сопоставлены соответственно функции $\mathcal{A}_I$ и $\mathcal{B}_I$. Тогда

$$(\mathcal{A}(x_{i_1}, ..., x_{i_k}) \supset \mathcal{B}(x_{j_1}, ..., x_{j_l}))_I = 1 \quad \Longleftrightarrow$$
$$\Longleftrightarrow \mathcal{A}_I(x_{i_1}, ..., x_{i_k}) = 1 \text{ влечет } \mathcal{B}_I(x_{j_1}, ..., x_{j_l}) = 1.$$

Если задана интерпретация, замкнутая формула принимает фиксированное значение — "истина" или "ложь".

Пример 5.2.7 . *Рассмотрим интерпретацию для формулы*

$$\mathcal{A}(x_1) = \neg A_1^2(x_1, a_1) \wedge \forall x_2(\exists x_3 A_1^2(x_1, f_1^2(x_2, x_3)) \supset \\ \supset (A_1^2(x_2, x_1) \vee A_1^2(x_2, a_1))).$$

Пусть область интерпретации $M = \mathbb{N}$, A_1^2 *— отношение равенства,* f_1^2 *— операция умножения,* $a_1 = 1$. *Тогда формулу можно переписать следующим образом:*

$$\mathcal{A}_I(x_1) = (x_1 \neq 1) \wedge \forall x_2(\exists x_3(x_1 = x_2 \cdot x_3) \supset ((x_1 = x_2) \vee (x_2 = 1))).$$

Таким образом, в данной интерпретации A_I *— функция равная 1 тогда и только тогда, когда ее единственный аргумент* x_1 *— простое число.*

Пример 5.2.8 . *Пусть задан алфавит:*
1) Предметные переменные: x_1, x_2, ...
2) Предикатные символы A_1^3, A_2^3.
3) $\supset$, $\neg$, $\forall x_i$, (,), ,.

Рассмотрим интерпретацию I: $M = 2^D = \{X \mid X \subseteq D\}$, *где* D *— некоторое множество.*

$$A_1^3(x_1, x_2, x_3) = 1 \quad \Leftrightarrow \quad x_3 = x_1 \cup x_2.$$

$$A_2^3(x_1, x_2, x_3) = 1 \quad \Leftrightarrow \quad x_3 = x_1 \cap x_2.$$

Задача: простроить формулы для предикатов

$$\mathcal{A}_\varnothing(x_1) = 1 \quad \Leftrightarrow \quad x_1 = \varnothing.$$
$$\mathcal{A}_=(x_1, x_2) = 1 \quad \Leftrightarrow \quad x_1 = x_2.$$
$$\mathcal{A}_\subseteq(x_1, x_2) = 1 \quad \Leftrightarrow \quad x_1 \subseteq x_2.$$
$$\mathcal{A}_1(x_1) = 1 \quad \Leftrightarrow \quad |x_1| = 1.$$

Для одного логического высказывания может существовать много формул описывающих его на формальном языке. Можно предложить, например, такое решение поставленной задачи:

$$\mathcal{A}_\varnothing(x_1) = \forall x_2 A_1^3(x_1, x_2, x_2).$$
$$\mathcal{A}_=(x_1, x_2) = A_1^3(x_1, x_1, x_2).$$

$$\mathcal{A}_{\subseteq}(x_1, x_2) = A_2^3(x_1, x_2, x_1).$$
$$\mathcal{A}_1(x_1) = \neg\mathcal{A}_{\varnothing}(x_1) \wedge \forall x_2(\mathcal{A}_{\subseteq}(x_1, x_2) \vee$$
$$\vee \exists x_3(A_2^3(x_1, x_2, x_3) \wedge \mathcal{A}_{\varnothing}(x_3))).$$

Определение 5.2.7 . *Будем говорить, что формула $\mathcal{A}$ выполнима в I, если $\exists d_1 \in M$, $\exists d_2 \in M$, ..., $\exists d_k \in M$: $\mathcal{A}_I(d_1, d_2, ..., d_k) = 1$.*

Определение 5.2.8 . *Будем говорить, что формула $\mathcal{A}$ истинна в I, если $\forall d_1 \in M$, $\forall d_2 \in M$, ..., $\forall d_k \in M$: $\mathcal{A}_I(d_1, d_2, ..., d_k) = 1$.*

Пример 5.2.9 . *Рассмотрим интерпретацию для формулы $\mathcal{A} = (\forall x_1(A_1^1(x_1) \supset A_2^1(x_1)) \wedge A_1^1(a_1)) \supset A_2^1(a_1)$. Пусть M — множество всех когда-либо живших на земле; $A_1^1(x) = 1$ тогда и только тогда, когда x — человек; $A_2^1(x) = 1$ тогда и только тогда, когда x — смертен; $a_1 =$ Сократ.*

Тогда формула $\mathcal{A}$ может читаться следующим образом: "Любой человек смертен и Сократ — человек. Следовательно, Сократ — смертен."

Формула замкнута и в интерпретации принимает фиксированное значение 1. Таким образом, формула $\mathcal{A}$ — истинна в данной интерпретации.

Определение 5.2.9 . *Будем говорить, что формула $\mathcal{A}$ ложна в I, если $\forall d_1 \in M$, $\forall d_2 \in M$, ..., $\forall d_k \in M$: $\mathcal{A}_I(d_1, d_2, ..., d_k) = 0$.*

Определение 5.2.10 . *Будем говорить, что I — модель для множества формул Γ, если любая формула $\mathcal{A} \in \Gamma$ будет истинна в I.*

Определение 5.2.11 . *Будем говорить, что формула $\mathcal{A}$ выполнима, если существует интерпретация I, такая что $\mathcal{A}$ выполнима в I.*

Определение 5.2.12 . *$\mathcal{A}$ — логически общезначима, если $\mathcal{A}$ истинна в любой интерпретации.*

Определение 5.2.13 . *Будем говорить, что формула $\mathcal{A}$ противоречие, если $\neg\mathcal{A}$ — логически общезначима.*

Определение 5.2.14 . *Будем говорить, что $\mathcal{A}$ логически влечет $\mathcal{B}$ ($\mathcal{B}$ логическое следствие $\mathcal{A}$), если $(\mathcal{A} \supset \mathcal{B})$ — логически общезначима.*

Замечание 5.2.3 . *Раньше мы уже приводили определение 5.2.12 под номером 5.2.1. Здесь повторяем его, пользуясь строго определенными понятиями.*

Логически общезначимые формулы — аналог тавтологий в исчислении высказываний в том смысле, что они представляют собой логические законы математической логики, любая интерпретация которых будет давать нам логически истинное высказывание. В отличии от исчисления высказываний здесь нет алгоритма, позволяющего для произвольной формулы определить, является ли она логически общезначимой, хотя можно построить формальную теорию, в которой будут выводиться только логически общезначемые формулы.

Теорема 5.2.1 . *Пусть $\mathcal{A}$ — формула исчисления предикатов. Тогда, $\mathcal{A}$ — логически общезначима тогда и только тогда, когда $\vdash_{ИП} \mathcal{A}$.*

Замечание 5.2.4 . *Поскольку не существует алгоритма, позволяющего определить, является ли заданная формула логически общезначимой, формальная теория исчисление предикатов не является разрешимой.*

Литература

[1] *Нефедов В.Н., Осипова В.А.* Курс дискретной математики. М.: Изд-во МАИ, 1992.

[2] *Грэхэм Р., Кнут Д., Паташник О.* Конкретная математика. М.: Мир, 1998.

[3] *Романовский И.В.* Дискретный анализ. СПб.: Невский диалект, 2000.

[4] *Яблонский С.В.* Введение в дискретную математику. М.: Наука, 1979.

[5] Гаврилов Г.П., Сапоженко А.А. Сборник задач по дискретной математике. 1977. 367 с.

[6] Гаврилов Г.П., Сапоженко А.А. Задачи и упражнения по дискретной математике. — М.: Физматлит, 2004.

[7] Лавров И.А., Максимова Л.Л. Задачи по теории множеств, математической логике и теории алгоритмов. — М.: Физматлит, 1995.

[8] *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.:Мир, 1982.