

## Содержание

<b>ОСНОВЫ ТЕОРИИ МНОЖЕСТВ.....</b>	<b>4</b>
1) Способы задания множеств.....	4
2) Операции над множествами и их свойства. Диаграммы Венна .....	5
3) Декартово произведение множеств, булеан, мощность множества	6
4) Упорядоченные множества. Проекция множества .....	6
5) Соответствия, основные определения, способы задания .....	7
6) Бинарные отношения и их свойства.....	8
7) Способы задания бинарных отношений.....	9
8) Операции над бинарными отношениями .....	10
<b>ОСНОВЫ АЛГЕБРЫ ЛОГИКИ .....</b>	<b>11</b>
1) Высказывания и операции над ними.....	11
2) Формулы алгебры высказываний и порядок выполнения операций. Таблицы истинности .....	13
3) Равносильности логических формул.....	13
4) Булевы функции и способы их задания.....	14
5) Дизъюнктивные формы представления логических функций. Приведение к ДНФ.....	15
6) Совершенная нормальная дизъюнктивная форма (СНДФ) и её свойства .....	15
7) Конъюнктивные формы представления логических функций. Приведение к КНФ.....	16
8) Совершенная нормальная конъюнктивная форма (СНКФ) и её свойства .....	16
9) Базис представления логических функций(Функционально полная система) .....	17

10) Процедуры приведения ДНФ к КНФ и наоборот.....	17
11) Геометрическое представление логических функций. Контактные схемы .....	18
12) Принцип двойственности в булевой алгебре.....	19
13) Понятие предиката, n-местный предикат, равносильные предикаты.....	19
14) Кванторы, понятие операции навешивания квантора. ....	19
15) Формулы логики предикатов; атомарная, литеральная формулы..	20
16) Основные равносильности, содержащие кванторы. ....	22
17) Предваренная нормальная форма. ....	23
<b>МАТЕМАТИЧЕСКАЯ КОМБИНАТОРИКА.....</b>	<b>23</b>
1) Правила суммы и произведения .....	23
2) Размещения и размещения с повторением .....	25
3) Перестановки и перестановки с повторением .....	25
4) Сочетания и сочетания с повторением.....	26
5) Биномиальные коэффициенты. Основные формулы. Треугольник Паскаля. ....	28
6) Формулы включений и исключений .....	29
<b>ОСНОВЫ ТЕОРИИ ГРАФОВ.....</b>	<b>30</b>
1) Графы, основные понятия и определения .....	30
2) Способы представления графов .....	31
3) Изоморфизм графов .....	32
4) Частичные графы. Подграфы .....	32
5) Маршруты, цепи, циклы .....	32
6) Связность. Цикломатическое число.....	32
7) Плоские и планарные графы. Свойства планарных графов.....	33

8) Операции над графами .....	33
9) Деревья, основные понятия, определения и теоремы.....	35
10) Остовное дерево минимального веса и способы его построения...	36
11) Обходы вершин графа: поиск в ширину и поиск в глубину .....	36
12) Задача о кратчайшем пути в орграфе. Алгоритм Форда .....	37
13) Отношение порядка между вершинами орграфа .....	37
14) Задача о пути максимальной длины в орграфе .....	39
15) Сетевое планирование. Задача о скорейшем пути завершения проекта.....	40
<b>ТЕОРИЯ АВТОМАТОВ .....</b>	<b>41</b>
1. Конечные автоматы, их реализация и применение. ....	41
2. Классификация абстрактных автоматов.....	42
3. Автоматное программирование. Графы переходов. ....	44
4. Теоретико-множественное определение автомата. Инициальные, синхронные и асинхронные автоматы. ....	46
5. Автоматы Мили и Мура.....	47
6. Табличная форма задания автоматов. ....	47
7. Графовая форма задания автоматов.....	49
8. Матричная форма задания автоматов. ....	50
9. Понятие частичного автомата. Реакция автомата. ....	51
10. Переход от автомата Мили к эквивалентному автомату Мура .....	52
11. Переход от автомата Мура к эквивалентному автомату Мили .....	53
12. Минимизация автоматов. ....	54
13. Распознающие автоматы.....	55

## ОСНОВЫ ТЕОРИИ МНОЖЕСТВ

### 1) Способы задания множеств

**Множество** — совокупность различных между собой объектов, объединяемых в целое некоторым общим признаком. Например: множества студентов, книг, законов, чисел и т. п.

**Элементы** — объекты, из которых состоит множество.

**Обозначения:**  $A, B, C, \dots$  — множества,  $a, b, c, \dots$  — элементы (точки) множеств.

Принадлежность:

$a \in A$  —  $a$  принадлежит множеству  $A$ ;

$a \notin A$  —  $a$  не принадлежит множеству  $A$ .

Записью  $a_1, a_2, \dots, a_n \in M$  пользуются в качестве сокращения для записи  $a_1 \in M, a_2 \in M, \dots, a_n \in M$ .

**Задание множеств.** Чтобы задать множество, нужно указать, какие элементы ему принадлежат.

Это можно сделать различными способами:

Перечислением элементов:  $A = \{a_1, a_2, \dots, a_k\}$ ;

Указанием характеристического свойства (хар. предикатом):  $M := \{x \mid P(x)\}$ ;

Порождающей процедурой:  $M := \{x \mid x := f\}$ .

При задании множеств перечислением обозначения элементов обычно заключают в фигурные скобки и разделяют запятыми. Характеристический предикат — это некоторое условие, выраженное в форме логического утверждения или процедуры, возвращающей логическое значение, и позволяющее проверить, принадлежит ли любой данный элемент множеству. Если для данного элемента условие выполнено, то он принадлежит определяемому множеству, в противном случае — не принадлежит. Порождающая процедура — это процедура, которая в процессе работы порождает некоторые объекты, являющиеся элементами определяемого множества.

Пример 1:

$M_9 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ;

$M_9 = \{n \mid n \in \mathbb{N} \ \& \ n < 10\}$ ;

$M_9 = \{n \mid \text{for } i \text{ from } 0 \text{ to } 8 \text{ do } n := i + 1\}$ .

При задании множеств перечислением обозначения элементов иногда снабжают индексами и указывают множество, из которого берутся индексы. В частности, запись  $\{a_i \mid k_i = 1\}$  означает то же, что  $\{a_1, \dots, a_n\}$ , а запись  $M = \{M_a\}_a$  ( $A$  означает, что  $M$  является семейством, элементами которого являются множества  $M_a$ , причём индекс  $a$  «пробегает» множество  $A$ ). Перечислением можно задавать только конечные множества. Бесконечные множества задаются характеристическим предикатом или порождающей процедурой.

Из определения множества следует, что в нем не должно быть неразличимых элементов. Поэтому во множестве не может быть одинаковых элементов. Запись множества  $\{2, 2, 3, 5\}$  следует рассматривать как некорректную и заменить ее на  $\{2, 3, 5\}$ . Так, множество всех простых делителей числа 60 равно  $\{2, 3, 5\}$ . Многие определения теории множеств удобно давать в виде математических выражений, содержащих некоторые логические символы:

$\forall$  — символ, называемый квантором общности и означающий «любой», «каков бы ни был», «для всех»;

$\exists$  — символ, называемый квантором существования и означающий «существует», «найдется хотя бы один»;

$\Rightarrow$  — символ следствия (импликации), означающий «влечет за собой»;

$\Leftrightarrow$  — символ эквивалентности (равносильности), означающий «то же самое, что»;

$\&$  ( $\wedge$ ) — символ конъюнкции, одновременного выполнения условий, означающий «и»;

$\vee$  — символ дизъюнкции, выполнения хотя бы одного из условий, означающий «или»;

$\neg$  ( $\neg$ ) — символ отрицания, невыполнения условия, означающий «не».

## 2) Операции над множествами и их свойства. Диаграммы Венна

Множества  $A$  и  $B$  **равны**,  $A=B$ , тогда и только тогда, когда  $A \subseteq B$  и  $B \subseteq A$ , т. е. состоят из одинаковых элементов, причем порядок следования элементов не имеет значения; в противном случае пишут  $A \neq B$ .

**Объединением** множеств  $A$  и  $B$  (обозначение  $A \cup B$ ) называется множество, состоящее из всех тех и только тех элементов, которые принадлежат хотя бы одному из множеств  $A, B$ :  $A \cup B = \{x \mid x \in A \text{ или } x \in B\}$ . Если система содержит небольшое количество множеств, то их объединение описывается явно:  $A \cup B \cup C \cup D$  и т. д.

Для объединения множеств существуют коммутативный и ассоциативный законы:  $A \cup B = B \cup A$ ,  $(A \cup B) \cup C = A \cup (B \cup C)$ . Рассмотрим  $A \cup \emptyset = A$ . Очевидно, что  $A \cup A = A$  для любого множества  $A$ . Для любых множеств  $A$  и  $B$  верно  $A \subseteq (A \cup B)$ , но неверно  $A \in (A \cup B)$ .

**Пересечением** множеств  $A$  и  $B$  называется множество  $C$ , состоящее из всех элементов, принадлежащих одновременно и  $A$ , и  $B$ :  $C = A \cap B = \{x \mid x \in A \text{ и } x \in B\}$ . Аналогично определяется пересечение (в том числе бесконечной) системы множеств. Для пересечения множеств существуют коммутативный и ассоциативный законы:  $A \cap B = B \cap A$ ,  $(A \cap B) \cap C = A \cap (B \cap C)$ . Рассмотрим  $A \cap \emptyset = \emptyset$ . Очевидно, что  $A \cap A = A$  для любого множества  $A$ . Для любых множеств  $A$  и  $B$  верно  $A \cup (A \cap B) = A$ , но неверно  $A \cap (A \cup B) = A$ .

**Разностью** множеств  $A$  и  $B$  называется множество  $C$ , состоящее из тех элементов множества  $A$ , которые не содержатся в множестве  $B$ :  $C = A \setminus B = \{x \mid x \in A \text{ и } x \notin B\}$ .

В отличие от операций объединения и пересечения данная операция определяется только для двух множеств. Для произвольных множеств  $A$  и  $B$  верны соотношения:

$$A \setminus B = \emptyset \Leftrightarrow A \subseteq B,$$

$$A \setminus \emptyset = A,$$

$$A \setminus B = A \Leftrightarrow A \cap B = \emptyset.$$

**Дополнением** множества  $A$  до универсального множества  $U$  (обозначение  $\overline{A}$ ) называется множество всех элементов  $U$ , не принадлежащих  $A$ :

$$\overline{A} = \{x \mid (x \in U) \& (x \notin A)\},$$

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

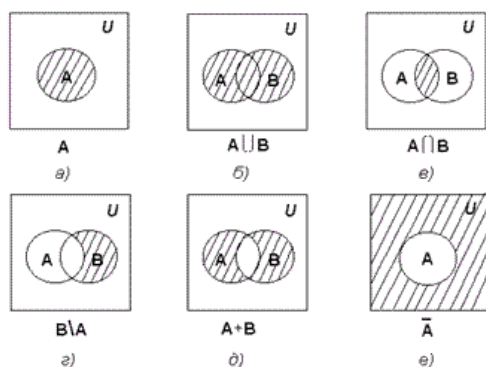
$$A \cap \overline{A} = \emptyset$$

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

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

**Симметрической разностью** множеств  $A$  и  $B$  (обозначение  $A \Delta B$ ) называется множество  $(A \setminus B) \cup (B \setminus A)$ . Оно состоит из всех тех и только тех элементов универсального множества, которые либо принадлежат  $A$  и не принадлежат  $B$ , либо наоборот, принадлежат  $B$ , но не  $A$ .

Названные операции и свойства к ним могут быть проиллюстрированы **диаграммами Венна**



### 3) Декартово произведение множеств, булеан, мощность множества

**Прямое произведение** множеств  $A$  и  $B$  (обозначение  $A \times B$ ) называется множество, состоящее из всех тех и только тех упорядоченных пар, первая компонента которых принадлежит множеству  $A$ , а вторая – множеству  $B$ .

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

Пусть  $A$  и  $B$  – отрезки соответственно длинами  $a$  и  $b$  вещественной оси. Прямое произведение  $A \times B$  представляет собой заштрихованный прямоугольник с длиной  $a$  и шириной  $b$  в декартовой системе координат  $ХОУ$  (рис. 5).

Множество всех подмножеств множества  $A$  называется его **булеаном** (или множеством-степеню) и обозначается через  $P(A)$ . **Пример.** Если  $A = \{a, b, c\}$ , то  $P(A) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a, c\}, \{a, b, c\}\}$ .

Число элементов конечного множества называется его **мощностью**. Если множество  $A$  содержит  $n$  элементов, то будем писать  $|A| = n$ . Если  $A = \emptyset$ , то  $|A| = 0$ .

### 4) Упорядоченные множества. Проекция множества

**Кортеж** – это последовательность элементов, т. е. совокупность элементов, в которой каждый элемент занимает определенное место. Часто кортеж называют вектором, а элементы, образующие кортеж, – его компонентами, или координатами. Компоненты нумеруются слева направо. Число компонент называется длиной, или размерностью, кортежа. Могут быть и бесконечные кортежи. В отличие от элементов множества координаты кортежа могут совпадать. Кортеж будем заключать в круглые скобки. Например,  $a = (a_1, \dots, a_n)$  – кортеж длиной  $n$  с элементами  $a_1, \dots, a_n$ . Иногда скобки и даже запятые опускаются. Два конечных кортежа равны, если имеют одинаковую длину и соответствующие компоненты.

Операция проектирования множества может применяться лишь к таким множествам, элементами которых являются кортежи одинаковой длины. Пусть  $V$  – множество кортежей одинаковой длины. Тогда проекцией множества  $V$  на  $i$ -ю ось называется множество проекций всех векторов из  $V$  на  $i$ -ю ось:  $\text{пр}iV = \{\text{пр}iv \mid v \in V\}$ . Аналогично определяется проекция множества  $V$  на несколько осей:  $\text{пр}i_1, \dots, i_k V = \{\text{пр}i_1, \dots, i_k v \mid v \in V\}$ . **Пример 22.** Пусть  $V := \{(1, 2, 3, 4, 5), (2, 1, 3, 5, 5), (3, 3, 3, 3, 3), (3, 2, 3, 4, 3)\}$ . Тогда  $\text{пр}2V = \{2, 1, 3\}$ ,  $\text{пр}2,4V = \{(2, 4), (1, 5), (3, 3)\}$ .

## 5) Соответствия, основные определения, способы задания

Пусть  $X$  и  $Y$  – два непустых множества. Если определен способ сопоставления элементов  $Y$  элементам  $X$ , то говорят, что между множествами  $X$  и  $Y$  установлено **соответствие**. При этом совершенно необязательно, чтобы в сопоставлении участвовали все элементы множеств  $X$  и  $Y$ . Для того чтобы задать соответствие между множествами  $X$  и  $Y$ , нужно задать множество  $Q \subseteq X \times Y$ , определяющее закон, по которому осуществляется соответствие, т. е. перечисляющий все пары  $(x, y)$ , участвующие в сопоставлении. Таким образом, соответствие  $q$  представляет собой тройку множеств  $q = (X, Y, Q)$ , в которой  $Q \subseteq X \times Y$ .  $X$  называют областью отправления соответствия, вторую компоненту  $Y$  – областью прибытия соответствия, третью компоненту  $Q$  – графиком соответствия. С каждым соответствием неразрывно связаны еще два множества: множество **пр1Q**, называемое областью определения соответствия, которое состоит из всех элементов множества  $X$ , участвующих в сопоставлении, и множество **пр2Q**, называемое областью значений соответствия, которое состоит из всех элементов множества  $Y$ , участвующих в сопоставлении. Если  $(x, y) \in Q$ , то говорят, что элемент  $y$  соответствует элементу  $x$ . Геометрически это удобно изображать стрелкой, направленной от  $x$  к  $y$ . Если  $\text{пр1}Q = X$ , то соответствие называется всюду определенным, или **отображением  $X$  в  $Y$**  (в противном случае соответствие называется частичным). Если  $\text{пр2}Q = Y$ , то соответствие называется **сюръективным** (сюръекцией). Множество всех  $y \in Y$ , соответствующих элементу  $x \in X$ , называется **образом  $x$  в  $Y$**  при соответствии  $q$ . Множество всех  $x \in X$ , которым соответствует элемент  $y \in Y$ , называется **прообразом  $y$  в  $X$**  при соответствии  $q$ . Если  $S \subseteq \text{пр1}Q$ , то образом множества  $S$  называется объединение образов всех элементов  $S$ . Аналогично определяется прообраз множества  $D$  для любого  $D \subseteq \text{пр2}Q$ . Соответствие  $q$  называется **инъективным** (инъекцией), если любые различные  $x_1$  и  $x_2$  из  $\text{пр1}Q$  имеют различные образы и любые различные  $y_1$  и  $y_2$  из  $\text{пр2}Q$  имеют различные прообразы при соответствии  $q$ . Соответствие  $q$  называется **функциональным** (или однозначным), если образом любого элемента  $x \in \text{пр1}Q$  является единственный элемент  $y \in \text{пр2}Q$ . Соответствие  $q$  между множествами  $X$  и  $Y$  называется взаимно однозначным, или **биективным**, биекцией (иногда пишут «1-1-соответствие»), если оно всюду определено, сюръективно и инъективно. Однозначное отображение называется **функцией**. Функция является инъективной, если различным  $x_1$  и  $x_2$  из  $X$  соответствуют различные  $y_1$  и  $y_2$  из  $Y$ , и сюръективной, если она сюръективна как соответствие. Функция называется биективной, если она одновременно инъективна и сюръективна.

**Способы задания:** табличный, графический, матрицей, аналитический (т.е. формулой), способ перечисления пар .

## 6) Бинарные отношения и их свойства

Бинарное отношение - это отношение между двумя объектами. Бинарное отношение можно определить как совокупность упорядоченных пар, указывающих объекты, находящиеся в данном отношении. В общем случае, если два элемента  $a, b$  находятся в данном отношении  $R$ , то этот факт записывают  $(a, b) \in R$  или  $aRb$ . Если эти элементы не находятся в отношении  $R$ , то это записывают так:  $(a, b) \notin R$ , или  $a \bar{R} b$ . Некоторым из наиболее известных отношений присваивают специальные названия и обозначения. Примеры: эквивалентность ( $\equiv$ ), отношение порядка ( $>$ ) или ( $<$ ), равенство ( $=$ ), параллельность ( $\parallel$ ), перпендикулярность ( $\perp$ ) и т. д. Очевидно, что всякое бинарное отношение  $R$  можно рассматривать как подмножество прямого произведения некоторых множеств  $A$  и  $B$ :  $R \subseteq A \times B$ . Левой областью бинарного отношения называют множество всех первых компонент упорядоченных пар, составляющих данное отношение, то есть  $R_- = \{a \mid (a, b) \in R\}$ . Правой областью бинарного отношения  $R$  называют множество всех вторых компонент упорядоченных пар, составляющих данное отношение, то есть  $R_+ = \{b \mid (a, b) \in R\}$ .

Поле бинарного отношения  $R$  называют объединение его левой и правой областей:  $F(R) = R_- \cup R_+$ . Бинарное отношение  $R^{-1}$  называют обратным к отношению  $R$ , если  $(a, b) \in R^{-1}$  тогда и только тогда, когда  $(b, a) \in R$  то есть  $R^{-1} = \{(a, b) \mid (b, a) \in R\}$ .

Пересечением бинарного отношения  $R$  по элементу  $a \in F(R)$  называют совокупность всех вторых (различных) компонентов упорядоченных пар, составляющих данное отношение, и таких, у которых первой компонентой есть элемент  $a$ . Обозначение:  $R_a$ .

Композицией бинарных отношений  $R$  и  $S$  называют бинарное отношение  $T$ , состоящее из всех упорядоченных пар  $(a, b)$ , для каждой из которых существует элемент  $\tilde{c} \in \tilde{E}_+ \cap S$  такой, что  $(a, \tilde{c}) \in R$ ,  $(\tilde{c}, b) \in S$  (то есть  $aR\tilde{c}$ ,  $\tilde{c}Sb$ ). Операцию композиции записывают так:  $T = R \circ S$ . Например, пусть  $R = \{(1, 1), (1, 2), (2, 3), (3, 3)\}$ ,  $S = \{(2, 4), (2, 5), (3, 2), (5, 5)\}$ . Тогда  $R \circ S = \{(1, 4), (1, 5), (2, 2), (3, 2)\}$ ,  $S \circ R = \{(3, 3)\}$ .

**Свойства бинарных отношений.** Бинарное отношение  $R$  называют **рефлексивным**, если для любого элемента  $a \in F(R)$  имеет место  $aRa$  (находится в отношении  $R$  с самим собой). Бинарное отношение  $R$  называют **антирефлексивным**, если для любого элемента поля  $a \in F(R)$  имеет место  $a \bar{R} a$ . Бинарное отношение  $R$  **симметричное**, если из  $aRb$  следует  $bRa$ . Бинарное отношение  $R$  **асимметрично**, если из  $aRb$  следует  $b \bar{R} a$ . Бинарное отношение  $R$  называют **антисимметричным**, если из  $aRb$  и  $bRa$  следует, что  $a = b$ . Бинарное отношение  $R$  называют **транзитивным**, если из  $aRb$  и  $bRc$  следует  $aRc$ . В противном случае отношение  $R$  называют нетранзитивным. Бинарное отношение называют отношением **эквивалентности**, если оно рефлексивно, симметрично и транзитивно.

Классом эквивалентности  $R$   $a$  называют множество всех вторых компонент упорядоченных пар отношения эквивалентности  $R$ , у которых первой компонентой является элемент  $a$ :  $R_a = \{b \mid (a, b) \in R\}$ .



## 7) Способы задания бинарных отношений

1. Бинарное отношение  $R$  можно задать перечислением всех упорядоченных пар, находящихся в отношении  $R$ . Очевидно, что такой способ задания отношений приемлем для относительно небольшого числа упорядоченных пар. Например:  $R_1 = \{(1, 1), (2, 2), (3, 3), (4, 4)\}$ .
2. Задать формулой.
3. Графическое задание бинарного отношения предполагает графическое представление элементов левой и правой областей отношения в виде точек в этих областях, соединенных дугами (направленными отрезками). Каждая дуга представляет некоторую упорядоченную пару, находящуюся в данном отношении. Дуга начинается в точке, соответствующей первой компоненте упорядоченной пары, и заканчивается в точке, соответствующей второй компоненте упорядоченной пары.
4. Бинарное отношение можно задать в табличной форме. В этой таблице указываются все элементы поля отношения и соответствующие им пересечения данного отношения по выбранному элементу.
5. Бинарное отношение можно задать матрицей  $\|a_{i,j}\|$ , в которой строки и столбцы соответствуют полю отношения. В этой матрице  $i$ -я строка соотносится с некоторым элементом левой области отношения, а  $j$ -й столбец – с некоторым элементом правой области отношения. Тогда  $a_{i,j} = 1$ , если соответствующие элементы находятся в данном отношении, и  $a_{i,j} = 0$  в противном случае.

## 8) Операции над бинарными отношениями

Если  $R$  – бинарное отношение, то в качестве универсального множества в этом случае рассматривают множество  $U = F(R) \times F(R)$ , где  $F(R)$  – поле отношения  $R$ . Если совместно рассматривается несколько бинарных отношений, то в качестве универсального множества рассматривают множество  $U = A \times A$ , где  $A$  есть объединение полей каждого из рассматриваемых отношений. Так как всякое бинарное отношение – это множество упорядоченных пар, то над бинарными отношениями можно выполнять все теоретико-множественные операции: объединение, пересечение, разность, дополнение.

Композицией бинарных отношений  $R$  и  $S$  называют бинарное отношение  $T$ , состоящее из всех упорядоченных пар  $(a, b)$ , для каждой из которых существует элемент  $c \in R_+ \cap S$  такой, что  $aRc, cSb$ . Операцию композиции записывают так:  $T = R \circ S$ . Например, пусть  $R = \{(1, 1), (1, 2), (2, 3), (3, 3)\}$ ,  $S = \{(2, 4), (2, 5), (3, 2), (5, 5)\}$ . Тогда  $R \circ S = \{(1, 4), (1, 5), (2, 2), (3, 2)\}$ ,  $S \circ R = \{(3, 3)\}$ .

## ОСНОВЫ АЛГЕБРЫ ЛОГИКИ

### 1) Высказывания и операции над ними

Высказывание – это повествовательное предложение, о котором можно сказать истинно оно или ложно. Каждое из высказываний принято обозначать латинской буквой. Если высказывание представляет собой одно утверждение - его называют элементарным или простым. Используя такие логические операции, как не, или, и, можно построить новые, так называемые составные высказывания комбинируя более простые.

Логической операцией наз-ся построение нового высказывания из исходных высказываний

**Отрицанием** произвольного высказывания  $x$  называется высказывание вида  $(\bar{x})$ , чье истинностное значение строго противоположно значению  $X$ . Определяющая таблица истинности отрицания высказывания приведена в табл.

$x$	$\bar{x}$
1	0
0	1

**Конъюнкцией** или логическим умножением двух высказываний  $X$  и  $Y$  называют составное высказывание вида  $z = x \wedge y (x * y)$ . Оно принимает истинное значение только в том случае, когда истинны обе его составные части.

$x$	$y$	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

**Дизъюнкцией** или логическим сложением двух высказываний  $X$  и  $Y$  называется составное высказывание  $z = x \vee y (x + y)$ . Оно истинно, если хотя бы одна из ее составных частей имеет истинное значение.

$x$	$y$	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

**Импликацией** двух высказываний  $X$  и  $Y$  называется составное высказывание  $z = x \rightarrow y$ . Оно будет ложью когда  $x$  – истинно, а  $y$  – ложно, в остальных случаях будет истиной.

$x$	$y$	$x \rightarrow y$
0	0	0
0	1	0
1	0	1
1	1	0

**Эквиваленцией** двух высказываний  $X$  и  $Y$  называется составное высказывание  $z = x \leftrightarrow y$ . Оно истинно, только когда оба высказывания либо истины, либо ложны.

$x$	$y$	$x \leftrightarrow y$
0	0	1
0	1	0
1	0	0
1	1	1

**Исключающее** двух высказываний  $X$  и  $Y$  называется составное высказывание  $z = x(+ )y$ . Оно истинно, только когда оба высказывания не совпадают.

$x$	$y$	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

## 2) Формулы алгебры высказываний и порядок выполнения операций. Таблицы истинности

С помощью логических операций над высказываниями можно строить различные формулы. Порядок операций:  $\neg, \wedge, \vee, (+), \rightarrow, \leftrightarrow$ . Всякое высказывание, кот. м.б. получено из элементарных высказываний путём применения конечного числа логических операций, наз. **формулой алгебры логики**. Логическое значение формулы определяется заданными логическими значениями входящих в неё элементарных высказываний. Если ставится задача определить все возможные формулы в зависимости от значения входящих в неё – решается через построение **таблицы истинности**.

## 3) Равносильности логических формул

Две формулы алгебры логики A и B называются равносильными, если они принимают одинаковые логические значения при любом наборе значений входящих в формулы элементарных высказываний (переменных).

Обозначение.  $A \equiv B$ . Формула A называется **тождественно истинной** (тавтологией), если она принимает значение 1 при всех значениях входящих в её переменных. Формула A называется **тождественно ложной**, если она принимает значение 0 при всех значениях входящих в её переменных.

Свойства: 1)  $A=A$  –рефлексивность, 2) Если  $A=B$ , то  $B=A$  – симметричность, 3)  $A=B$ ,  $B=C$ , то  $A=C$  – транзитивность 4)  $x*/+y=y*/+x$  – коммутативность, 5)  $(x*/+y)*/+z=(x*/+z)*/+y$  – ассоциативность, 6)  $(x*/+y)+/*z=(x+/*z)*/+ (y+/*z)$  – дистрибутивность.

### 1. Основные равносильности.

- $x \wedge x \equiv x, x \vee x \equiv x$  – законы идемпотентности;
- $x \wedge 1 \equiv x$ ;
- $x \vee 1 \equiv 1$ ;
- $x \wedge 0 \equiv 0$ ;
- $x \vee 0 \equiv x$ ;
- $x \wedge \bar{x} \equiv 0$  – закон противоречия;
- $x \vee \bar{x} \equiv 1$  – закон исключенного третьего;
- $\bar{\bar{x}} \equiv x$  – закон снятия двойного отрицания;
- $x \vee (y \wedge x) \equiv x, x \wedge (y \vee x) \equiv x$  – законы поглощения.

### 1. Равносильности, выражающие одни логические опера

- $x \leftrightarrow y \equiv (x \rightarrow y) \wedge (y \rightarrow x)$ ;
- $x \rightarrow y \equiv \bar{x} \vee y$ ;
- $\overline{x \wedge y} \equiv \bar{x} \vee \bar{y}$ ;
- $\overline{x \vee y} \equiv \bar{x} \wedge \bar{y}$ ;
- $\overline{\overline{x \wedge y}} \equiv x \wedge y$ ;
- $\overline{\overline{x \vee y}} \equiv x \vee y$ ;
- $\overline{\overline{x \wedge y}} \equiv x \wedge y$ .

Формулы расщепления :  $(x*y)+(x*\bar{y})=x$  и  $(x+y)*(x+\bar{y})=x$

#### 4) Булевы функции и способы их задания

Булевой функцией (БФ) или функцией алгебры логики (ФАЛ), от  $n$  переменных называется функция  $f(x_1, x_2, x_3, \dots, x_n)$ , которая на любом наборе своих аргументов может принимать одно из двух значений: 0 или 1. Под набором аргументов понимается совокупность значений переменных, каждая из которых может быть равна 0 или 1. Для функции от  $n$  аргументов количество возможных наборов равно  $2^n$ .

##### 1) Задание булевой функции таблицей истинности

	$x_1$	$x_2$	...	$x_n$	$f(x_1, x_2, \dots, x_n)$
0	0	0	...	0	$f(0, 0, \dots, 0)$
1	0	0	...	1	$f(0, 0, \dots, 1)$
...	...	...	...	...	...
$a$	$a_1$	$a_2$	...	$a_n$	$f(a_1, a_2, \dots, a_n)$
...	...	...	...	...	...
$2^n - 1$	1	1	...	1	$f(1, 1, \dots, 1)$

2) Задание булевой функции характеристическими множествами. Так называются два множества:

$M^1_f$ , состоящее из всех наборов, на которых функция принимает значение 1, то есть  $M^1_f = \{\alpha \in B^n : f(\alpha) = 1\}$ ;

$M^0_f$ , состоящее из всех наборов, на которых функция принимает значение 0, то есть  $M^0_f = \{\alpha \in B^n : f(\alpha) = 0\}$ .

3) Задание булевой функции вектором ее значений.

$$\varphi = f(0, 0, \dots, 0) f(0, 0, \dots, 1) \dots f(1, 1, \dots, 1).$$

4) Задание булевой функции матрицей Грея. Булево пространство задается матрицей Грея, и наборы (клетки матрицы), на которых булева функция  $f(x_1, \dots, x_n)$  принимает значение 1, отмечаются и называются *точками*.

5) Интервальный способ задания булевой функции. Булеву функцию  $f(x_1, \dots, x_n)$  можно задать множеством интервалов  $I_f = \{I_1, I_2, \dots, I_k\}$ , объединение которых образует характеристическое множество  $M^1_f$ , то есть  $I_1 \cup I_2 \cup \dots \cup I_k = M^1_f$ . Множество интервалов  $I_f$  называется *достаточным* для функции  $f(x_1, \dots, x_n)$ .

6) Задание булевой функции формулами.

## 5) Дизъюнктивные формы представления логических функций. Приведение к ДНФ

ДНФ логической функции называется дизъюнкция любого конечного множества попарно различных элементарных конъюнкций.  $f(x_1, x_2, x_3) = x_1 \bar{x}_2 \vee x_2 \bar{x}_3 \vee \bar{x}_1 x_3$ .

Алгоритм приведения формулы к ДНФ:

1. Выражают все логические операции в формуле через дизъюнкцию, конъюнкцию и отрицание.
2. Используя законы Де-Моргана переносят все отрицания к переменным и сокращают двойные отрицания.
3. Используют закон дистрибутивности конъюнкции относительно дизъюнкции, преобразуют формулу так, чтобы все конъюнкции встречались раньше дизъюнкции.

## 6) Совершенная нормальная дизъюнктивная форма (СНДФ) и её свойства

Совершенная дизъюнктивная нормальная форма формулы (СДНФ) - это дизъюнкция конъюнкций, обладающая свойствами:

1. Каждое логическое слагаемое формулы содержит все переменные, входящие в функцию  $F(x_1, x_2, \dots, x_n)$
2. Все логические слагаемые формулы различны
3. Ни одно логическое слагаемое не содержит переменную и её отрицание
4. Ни одно логическое слагаемое формулы не содержит одну и ту же переменную дважды. СДНФ можно получить или с помощью таблиц истинности или с помощью равносильных преобразований.

## 7) Конъюнктивные формы представления логических функций. Приведение к КНФ

**Что значит нормальна форма:** Нормальная форма логической формулы не содержит знаков импликации, эквиваленции и отрицания неэлементарных формул.

**Конъюнктивная нормальная форма**, т. е. конъюнкция нескольких дизъюнкций

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

Для приведение к КНФ:

1. Привести функцию к виду, используя базовые операции – конъюнкцию, дизъюнкцию и инверсию;
2. преобразовать инверсию с помощью законов де Моргана до отдельных букв;
3. уничтожить все суммы произведений, используя законы поглощения, с помощью второго закона дистрибутивности.

## 8) Совершенная нормальная конъюнктивная форма (СНКФ) и её свойства

**Совершенно конъюнктивная НФ** - конъюнкция дизъюнкций, причём в каждой дизъюнкции (в каждой скобке) присутствуют все переменные, входящие в формулу, либо их отрицание, нет одинаковых дизъюнкций, в каждой дизъюнкции нет одинаковых слагаемых.

**Свойства: 1.** в ней нет одинаковых элементарных дизъюнкций

**2.** в каждой дизъюнкции нет одинаковых пропозициональных переменных

**3.** каждая элементарная дизъюнкция содержит каждую пропозициональную букву из входящих в данную кнф пропозициональных букв.



## 9) Базис представления логических функций (Функционально полная система)

*Функционально полная система логических элементов* - это такой набор элементов, используя который можно реализовать любую сколь угодно сложную логическую функцию. Поскольку любая логическая функция представляет собой комбинацию простейших функций - дизъюнкции, конъюнкции и инверсии, то набор из элементов трех типов, реализующих соответственно функции И, ИЛИ и НЕ, естественно, является функционально полным.

Система логических функций  $\Sigma = \{f_1, \dots, f_m\}$  называется *Функционально полной системой*, если любая логическая функция может быть выражена через функции  $f_1, \dots, f_m$  с помощью их суперпозиции

## 10) Процедуры приведения ДНФ к КНФ и наоборот

Алгоритм этого перехода следующий: ставим над ДНФ два отрицания и с помощью правил де Моргана (не трогая верхнее отрицание) приводим отрицание ДНФ снова к ДНФ. При этом приходится раскрывать скобки с использованием правила поглощения (или правила Блейка). Отрицание (верхнее) полученной ДНФ (снова по правилу де Моргана) сразу дает нам КНФ:

$$xy \vee y \bar{z} = \overline{\overline{xy \vee y \bar{z}}} = \overline{\overline{xy} \cdot \overline{y \bar{z}}} = \overline{(\bar{x} \vee \bar{y})(\bar{y} \vee z)}$$

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

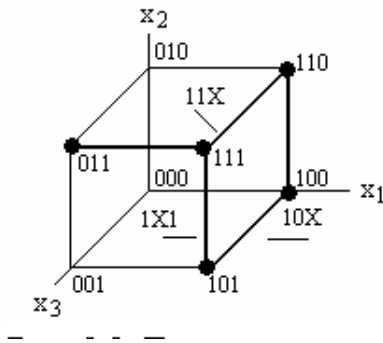
Заметим, что КНФ можно получить и из первоначального выражения, если вынести  $y$  за скобки;

б) переход от КНФ к ДНФ. Этот переход осуществляется простым раскрытием скобок (при этом опять-таки используется правило поглощения)

$$(x \vee \bar{y} \vee \bar{z})(\bar{x} \vee \bar{y})(\bar{y} \vee \bar{z}) = (\bar{y} \vee \bar{x}\bar{z})(x \vee \bar{y} \vee \bar{z}) = \bar{y} \vee \bar{x}\bar{z}. \text{ Таким образом, получили ДНФ.}$$

### 11) Геометрическое представление логических функций. Контактные схемы

В геометрическом смысле каждый набор значений переменных  $(x_1, x_2, \dots, x_n)$  можно рассматривать как  $n$ -мерный вектор, определяющий точку в  $n$ -мерном пространстве. Все множество двоичных наборов значений аргументов образует геометрическое множество вершин  $n$ -мерного единичного куба. Выделяя вершины, на которых значение функции равно 1, можно получить геометрический образ истинностной функции.



**Контактная схема** (англ. *contact circuit*) представляет собой ориентированный ациклический граф, на каждом ребре которого написана переменная или ее отрицание. При операции конъюнкции (умножение) идёт последовательное соединение контактов, при операции дизъюнкции – параллельное. При отрицании – ключ замкнут.

## 12) Принцип двойственности в булевой алгебре

Булева функция  $f^*(x_1, \dots, x_n)$  называется двойственной булевой функции  $f(x_1, \dots, x_n)$ , если она получена из  $f(x_1, \dots, x_n)$  инверсией всех аргументов и самой функции.

Пример:  $(x \vee y)^* = (\overline{x \vee y}) = \overline{x} \overline{y} = x y$ . • Функция  $f$  называется самодвойственной если  $f^*=f$ .

### ТЕОРЕМА (Закон двойственности)

Если формула  $f_1$  равносильна формуле  $f_2$ , то формула  $f_1^*$  равносильна формуле  $f_2^*$ .

### ТЕОРЕМА (Принцип двойственности)

Двойственная к булевой формуле может быть получена заменой констант 0 на 1, 1 на 0,  $\wedge$  на  $\vee$ ,  $\vee$  на  $\wedge$  и сохранением структуры формулы.

## 13) Понятие предиката, n-местный предикат, равносильные предикаты.

Предикат - это выражение, которое зависит от одного или нескольких объектов и выражает свойство или отношение между ними. В логике и математике предикаты часто используются для формулирования утверждений или условий, которые могут быть истинными или ложными в зависимости от значений, которые принимают их аргументы.

1. n-местный предикат: Это предикат, который зависит от n объектов или аргументов. Например, предикат "больше", который принимает два числа в качестве аргументов, является двухместным предикатом.

2. Равносильные предикаты: Два предиката называются равносильными, если их истинностные значения совпадают для всех возможных значений их аргументов. Другими словами, если для любого значения аргументов оба предиката либо оба истинны, либо оба ложны. Например, предикаты " $x > y$ " и " $y < x$ " равносильны, так как они оба истинны или ложны для любых значений  $x$  и  $y$ .

## 14) Кванторы, понятие операции навешивания квантора.

Кванторы - это логические операторы, которые используются для выражения утверждений о множествах объектов. В общем виде, кванторы могут быть универсальными и существенными.

1. Универсальный квантор ( $\forall$ ): Обозначается как "для всех" или "для любого". Утверждение с универсальным квантором верно, если оно

истинно для всех элементов множества. Например, выражение " $\forall x P(x)$ " означает, что предикат  $P(x)$  истинен для каждого  $x$  из домена.

2. Существенный квантор ( $\exists$ ): Обозначается как "существует" или "найдется". Утверждение с существенным квантором верно, если оно истинно хотя бы для одного элемента множества. Например, выражение " $\exists x P(x)$ " означает, что существует хотя бы один элемент  $x$  из домена, для которого предикат  $P(x)$  истинен.

Операция навешивания квантора заключается в применении квантора к предикату или формуле. Например, предикат " $P(x)$ " может быть преобразован в " $\forall x P(x)$ " (для всех  $x$   $P(x)$ ), что означает, что предикат  $P(x)$  истинен для всех элементов множества. Аналогично, " $P(x)$ " может быть преобразован в " $\exists x P(x)$ " (существует  $x$   $P(x)$ ), что означает, что существует хотя бы один элемент множества, для которого предикат  $P(x)$  истинен. Эта операция позволяет указать область применения утверждения и определить его истинность или ложность в контексте данной области.

### 15) Формулы логики предикатов; атомарная, литеральная формулы.

В логике предикатов формулы строятся на основе предикатов, переменных и кванторов. Давайте определим базовые термины:

1. Атомарная формула: Это выражение, которое состоит из предиката и его аргументов (переменных или констант). Например, " $P(x)$ ", " $Q(y)$ ", " $R(z)$ " - это атомарные формулы, где " $P$ ", " $Q$ " и " $R$ " - это предикаты, а " $x$ ", " $y$ ", " $z$ " - переменные.

2. Литерал: Литерал - это атомарная формула или отрицание атомарной формулы. То есть, литерал может быть предикатом или его отрицанием. Например, если " $P(x)$ " - атомарная формула, то и " $P(x)$ " и " $\neg P(x)$ " (отрицание " $P(x)$ ") являются литералами.

Теперь давайте рассмотрим формулы, которые строятся из атомарных формул, литералов и других логических операций:

1. Конъюнкция: Если  $A$  и  $B$  - это формулы, то их конъюнкция  $(A \wedge B)$  также является формулой. Пример: " $P(x) \wedge Q(y)$ ".

2. Дизъюнкция: Если  $A$  и  $B$  - это формулы, то их дизъюнкция  $(A \vee B)$  также является формулой. Пример: " $P(x) \vee Q(y)$ ".

3. Импликация: Если  $A$  и  $B$  - это формулы, то их импликация  $(A \rightarrow B)$  также является формулой. Пример: " $P(x) \rightarrow Q(y)$ ".

4. Эквиваленция: Если  $A$  и  $B$  - это формулы, то их эквиваленция  $(A \leftrightarrow B)$  также является формулой. Пример: " $P(x) \leftrightarrow Q(y)$ ".

5. Кванторы: Если  $\Phi$  - это формула, а  $x$  - переменная, то универсальная квантификация  $(\forall x \Phi)$  и существенная квантификация  $(\exists x \Phi)$  также являются формулами.

Это основные составляющие формул логики предикатов, которые могут быть атомарными, литералами или составными выражениями, построенными с использованием различных логических операций.

## 16) Основные равносильности, содержащие кванторы.



### ChatGPT

В логике предикатов существуют некоторые основные равносильности, которые могут быть полезны при доказательствах и преобразованиях формул. Вот несколько таких равносильностей:

1. **Де Моргановы законы для кванторов:**
  - $\neg(\forall x P(x)) \equiv \exists x \neg P(x)$
  - $\neg(\exists x P(x)) \equiv \forall x \neg P(x)$
2. **Дистрибутивность кванторов:**
  - $\forall x (P(x) \wedge Q(x)) \equiv (\forall x P(x)) \wedge (\forall x Q(x))$
  - $\exists x (P(x) \vee Q(x)) \equiv (\exists x P(x)) \vee (\exists x Q(x))$
3. **Перестановка кванторов:**
  - $\forall x \exists y P(x, y) \equiv \exists y \forall x P(x, y)$
4. **Связь между кванторами и логическими операциями:**
  - $\forall x P(x) \wedge Q \equiv \forall x (P(x) \wedge Q)$
  - $\exists x P(x) \wedge Q \equiv \exists x (P(x) \wedge Q)$
  - $\forall x P(x) \vee Q \equiv \forall x (P(x) \vee Q)$
  - $\exists x P(x) \vee Q \equiv \exists x (P(x) \vee Q)$

Эти равносильности могут быть использованы для упрощения или преобразования формул с кванторами, облегчая анализ и выводы в логике предикатов.

## 17) Предваренная нормальная форма.



**ChatGPT**

Предваренная нормальная форма (Prenex Normal Form, PNF) - это форма логической формулы, в которой все кванторы вынесены в начало выражения. В PNF кванторы обычно идут подряд, а затем следуют чистые предикаты без кванторов.

Преобразование формулы в предваренную нормальную форму состоит из двух основных шагов:

1. **Вынесение кванторов:** Кванторы выносятся в начало выражения, при этом сохраняется порядок кванторов и предикатов.
2. **Сокращение кванторов:** Кванторы, примененные к одной и той же переменной в последовательности, могут быть объединены или сокращены, если это возможно.

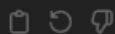
Пример преобразования формулы в PNF:

Исходная формула:  $\forall x (P(x) \wedge \exists y Q(x, y))$

Шаг 1 (вынесение кванторов):  $\forall x \exists y (P(x) \wedge Q(x, y))$

Теперь формула находится в предваренной нормальной форме. В этом примере оба квантора вынесены в начало выражения, и чистый предикат  $P(x) \wedge Q(x, y)$  идет после них.

Предваренная нормальная форма удобна для анализа и преобразований формул в логике предикатов. Она упрощает выражение кванторов и делает структуру формулы более ясной.



## МАТЕМАТИЧЕСКАЯ КОМБИНАТОРИКА

### 1) Правила суммы и произведения

Пусть  $X$  – конечное множество такое, что  $|X| = n$ . Тогда говорят, что объект  $x$  из  $X$  может быть выбран  $n$  способами. Пусть  $X_1, \dots, X_n$  – попарно непересекающиеся множества, то есть  $X_i \cap X_j = \emptyset$  при любых  $i \neq j$ . Тогда, очевидно, выполняется равенство

$$\left| \bigcup_{i=1}^n X_i \right| = \sum_{i=1}^n |X_i|.$$

В комбинаторике этот факт называется **правилом суммы**. Для  $n = 2$  оно формулируется следующим образом: «Если объект  $x$  может быть выбран  $m$  способами, а объект  $y$  – другими  $n$  способами, то выбор “либо  $x$ , либо  $y$ ” может быть осуществлен  $m + n$  способами».

**Правило произведения:** Если объект  $x_1$  может быть выбран  $n_1$  способами, после чего объект  $x_2$  может быть выбран  $n_2$  способами и для любого  $i$ , где  $2 \leq i \leq m - 1$ , после выбора объектов  $x_1, \dots, x_i$  объект  $x_{i+1}$  может быть выбран  $n_{i+1}$  способами, то выбор кортежа  $(x_1, x_2, \dots, x_m)$  длины  $m$  может быть осуществлен  $n_1 \cdot n_2 \cdot \dots \cdot n_m$  способами. Сформулируем частный случай этого правила для кортежа длины 2: «Если объект  $x$  может быть выбран  $m$  способами и после каждого из таких выборов объект  $y$  в свою очередь может быть выбран  $n$  способами, то выбор упорядоченной пары  $(x, y)$  может быть осуществлен  $m \cdot n$  способами».



## 2) Размещения и размещения с повторением

Размещениями называют комбинации, составленные из  $n$  различных элементов по  $m$  элементов, которые отличаются либо составом элементов, либо их порядком. Число всех возможных размещений вычисляется по формуле:

$$A_n^m = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n-m+1) = \frac{n!}{(n-m)!}$$

( $n, k$ )-размещением с повторениями называется упорядоченная ( $n, A$ ;-выборка, элементы в которой могут повторяться;

( $n, k$ )-размещением без повторений называется упорядоченная ( $n, /c$ )-выборка, элементам в которой повторяться запрещено;

### Размещения с повторением

Предположим, что мы берем элементы  $x_1, x_2, \dots, x_n$  из множества  $X$  мощности  $k$ . Каждый такой набор принято называть выборкой объема  $k$  из  $n$  элементов или, иначе, ( $n, k$ )-выборкой. Выборка называется упорядоченной, если порядок следования элементов в ней задан. При этом две упорядоченные выборки, различающиеся лишь порядком следования элементов, считаются разными. Если же порядок следования элементов в выборке не имеет значения, то выборка называется неупорядоченной.

( $n, k$ )-размещением с повторениями называется упорядоченная ( $n, k$ )-выборка, элементы в которой могут повторяться;

**Пример.** Всевозможные размещения с повторениями из трех элементов  $a, b, c$  по 2:

$aa, ab, ac, ba, bb, bc, ca, cb, cc$ .

**Теорема.** Число всевозможных размещений с повторениями из  $n$  элементов по  $k$  равно

$$n^k.$$

## 3) Перестановки и перестановки с повторением

Перестановками называют комбинации, состоящие из одних тех же  $n$  различных элементов и отличающиеся только порядком их расположения.

Число всех возможных перестановок находится по формуле:

$$P_n = 1 \cdot 2 \cdot \dots \cdot n = n!,$$

### Перестановка с повторением

Если в основном множестве  $k$  элементов  $a_1, a_2, \dots, a_k$  и выборка  $n$  элементов составляется так:

элемент  $a_1$  повторяется  $n_1$  раз,

элемент  $a_2$  повторяется  $n_2$  раз,

...

элемент  $a_k$  повторяется  $n_k$  раз,

такие выборки называются перестановками с повторениям.

Их возможное количество вычисляется по формуле:

$$\overline{P}_n = P_{n_1, n_2, \dots, n_k} = \frac{n!}{n_1! \cdot n_2! \cdot \dots \cdot n_k!}$$

Пример:

Сколько различных пятибуквенных слов можно составить из букв слова «манна»?

Решение:

В слове буквы а и н повторяются 2 раза, а буква м один раз.

$$\overline{P}_5 = P_{2,2,1} = \frac{5!}{2! \cdot 2! \cdot 1!} = \frac{2 \cdot 3 \cdot 4 \cdot 5}{2 \cdot 2} = 30$$

#### 4) Сочетания и сочетания с повторением

Сочетаниями называют комбинации, составленные из  $n$  различных элементов по  $m$  элементов, которые отличаются хотя бы одним элементом.

Число всех возможных сочетаний находится по формуле:

$$C_n^m = \frac{n!}{(n-m)! \cdot m!}.$$

- неупорядоченная  $(n,k)$ -выборка с повторяющимися элементами называется  $(n,k)$ -сочетанием с повторениями
- неупорядоченная  $(n,k)$ -выборка без повторяющихся элементов называется  $(n,k)$ -сочетанием без повторений.

#### Сочетания с повторением

Предположим, что мы берем элементы  $x_1, x_2, \dots, x_n$  из множества  $X$  мощности  $k$ . Каждый такой набор принято называть выборкой объема  $k$  из  $n$  элементов или, иначе,  $(n, k)$ -выборкой. Выборка называется упорядоченной, если порядок следования элементов в ней задан. При этом две упорядоченные выборки, различающиеся лишь порядком следования

элементов, считаются разными. Если же порядок следования элементов в выборке не имеет значения, то выборка называется неупорядоченной.

Неупорядоченная  $(n, k)$ -выборка с повторяющимися элементами называется  $(n, k)$ -**сочетанием с повторениями**.

Теорема. Число сочетаний с повторениями из  $n$  элементов по  $k$  выражается формулой

$$\tilde{C}_n^k = \frac{(n+k-1)!}{k!(n-1)!} = C_{n+k-1}^k.$$

Пусть имеется три элемента ( $n = 3$ ):  $a$ ,  $b$  и  $c$ . Тогда из этих трёх элементов можно составить шесть сочетаний с повторениями по два элемента ( $k = 2$ ):  $ab$ ,  $ac$ ,  $bc$ ,  $aa$ ,  $bb$ ,  $cc$  (порядок неважен!)

5) Биномиальные коэффициенты. Основные формулы. Треугольник Паскаля.

Числа  $C(n, k)$  возникают как коэффициенты при раскрытии скобок в бинOME  $(a + b)^n$ .

В общем случае, раскрывая скобки в бинOME  $(a + b)^n$ , мы будем получать члены вида

$a^{n-k} b^k$  (где  $k$  принимает каждое из значений от 0 до  $n$ ) при перемножении символов  $b$ , взятых из  $k$  скобок, и  $a$ , взятых из оставшихся  $(n - k)$  скобок. Так как есть ровно  $C(n, k)$  способов выбора  $k$  скобок из  $n$ , то у нас будет в точности  $C(n, k)$  членов вида  $a^{n-k} b^k$  при  $A$ :

$= 0, 1, \dots, n$ . Следовательно  $(a+b)^n = C(n, 0)a^n + C(n, 1)a^{n-1}b + C(n, 2)a^{n-2}b^2 + \dots + C(n, n)b^n$ .

Эта формула называется биномом Ньютона. Ровно поэтому коэффициенты  $C(n,k)$  часто называют биномиальными коэффициентами.

Биномиальные коэффициенты полезно выстроить в так называемый треугольник Паскаля

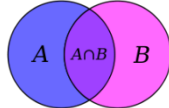


## 6) Формулы включений и исключений

**Формула включений-исключений** — комбинаторная формула, позволяющая определить мощность объединения конечного числа множеств, которые в общем случае могут пересекаться друг с другом. Например, в случае двух множеств  $A$ ,  $B$  формула включений-исключений имеет вид:

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

Действительно, сумме  $|A| + |B|$  элементы пересечения  $A \cap B$  учтены дважды, и чтобы компенсировать это мы вычитаем  $|A \cap B|$  из правой части формулы



Таким же образом и в случае  $n > 2$  множеств процесс нахождения количества элементов объединения  $A_1 \cup A_2 \cup \dots \cup A_n$  состоит во включении всего, затем исключении лишнего, затем включении ошибочно исключенного и т. д. Отсюда и происходит название формулы. Формула включений-исключений утверждает:

$$\left| \bigcup_{i=1}^n A_i \right| = \sum_i |A_i| - \sum_{i < j} |A_i \cap A_j| + \sum_{i < j < k} |A_i \cap A_j \cap A_k| - \dots$$

$$+ (-1)^{n-1} |A_1 \cap A_2 \cap \dots \cap A_n|.$$

## ОСНОВЫ ТЕОРИИ ГРАФОВ

### 1) Графы, основные понятия и определения

Пусть  $V$  – непустое множество и  $E$  – набор пар элементов множества  $V$ , причем в парах могут быть одинаковые элементы и допускается повторение пар. Тогда совокупность  $(V, E)$  называется графом  $G$ . Будем обозначать иногда этот граф как  $G(V, E)$ . Элементы множества  $V$  называются вершинами графа, а элементы множества  $E$  – ребрами. Ребра графа могут представляться как неупорядоченными парами  $\{v_i, v_j\}$ , так и упорядоченными  $(v_i, v_j)$ . В последнем случае ребро называется ориентированным, или дугой,  $v_i$  – начальной вершиной (началом),  $v_j$  – конечной вершиной (концом) данной дуги. Ребро  $\{v_i, v_i\}$  или дуга  $(v_i, v_i)$  называется петлей. Граф, состоящий из вершин и соединяющих их ребер, называется неориентированным, а граф, состоящий из вершин и соединяющих их дуг, – ориентированным (орграфом). Графы, содержащие как ребра, так и дуги, именуются смешанными. Вершины, соединенные между собой хотя бы одним ребром или дугой, называются смежными. Аналогично, два ребра, имеющие хотя бы одну общую вершину, называются смежными. 4 и е5. Если ребро  $e_k$  соединяет две вершины, т. е.  $e_k = \{v_i, v_j\}$ ,  $e_k = (v_i, v_j)$  или  $e_k = (v_j, v_i)$ , то ребро  $e_k$  называется инцидентным вершинам  $v_i$  и  $v_j$  или вершины  $v_i$  и  $v_j$  называются инцидентными ребру  $e_k$ . Граф называется конечным, если множества его вершин и ребер конечны (пустое множество тоже рассматривается как конечное). Назовем граф обыкновенным или простым, если в нем отсутствуют петли и кратные ребра (дуги). Граф, имеющий кратные ребра (дуги), называется мультиграфом, а граф, в котором есть хотя бы одна петля, называется псевдографом. Два графа  $G$  и  $H$  изоморфны (записывается  $G \cong H$ ), если между их множествами вершин существует взаимно однозначное соответствие, сохраняющее смежность. Если ребра ориентированы, то их направления также должны соответствовать друг другу. Обыкновенный неориентированный граф называется полным, если любые две различные его вершины смежны. Обыкновенный орграф называется полным, если в нем любые две различные вершины соединены парой антипараллельных дуг. Число ребер неориентированного графа, инцидентных вершине  $v$ , называется степенью, или порядком, этой вершины. При подсчете числа ребер, инцидентных вершине  $v$ , некоторую неопределенность вносит петля, так как ее можно считать и как единственное, и как двойное ребро. Будем обозначать степень вершины  $v$  через  $d(v)$  (или  $\deg(v)$ ). Изолированными называются вершины, которые не являются концами ребер и не связаны ни между собой, ни с другими вершинами. Изолированность вершины  $v$  в неориентированном графе эквивалентна условию  $d(v) = 0$ . Вершина степени 1 (единица) называется концевой, или висячей, вершиной, если петля считается двойной. Маршрутом в графе называется чередующаяся последовательность вершин и ребер. Если  $v_q \rightarrow v_k$ , то маршрут замкнут, иначе – открыт. Если все ребра различны, то маршрут называется цепью. Замкнутая цепь называется циклом, замкнутая простая цепь называется простым циклом. Число циклов в графе  $G$  обозначается  $z(G)$ . Граф без циклов называется ациклическим. Для орграфов цепь называется путем, а цикл – контуром. Цепь(цикл) называется гаметоновым, если он проходит через все вершины графа по одному разу.

## 2) Способы представления графов

**Матрицей смежности вершин неориентированного графа**  $G$  называется квадратная матрица  $A(G) = (a_{ij})$  порядка  $p$  ( $p$  – количество вершин графа), элементы  $a_{ij}$  которой равны числу ребер, соединяющих вершины  $v_i$  и  $v_j$  (при этом петля может означать одно или два ребра по договоренности).

**Матрицей смежности вершин ориентированного графа**  $G$  называется квадратная матрица  $A(G) = (a_{ij})$  порядка  $p$  ( $p$  – количество вершин графа), элементы  $a_{ij}$  которой равны числу дуг, исходящих из вершины  $v_i$  и заходящих в вершину  $v_j$ .

**Матрицей инцидентности неориентированного графа**  $G$  называется матрица  $B(G) = (b_{ij})$  размерности  $p \times q$  ( $p$  и  $q$  – количество вершин и ребер графа соответственно), элементы которой определяются следующим образом: 1, если вершина инцидентна ребру ; 0, если вершина не инцидентна ребру. **Матрицей инцидентности орграфа**  $G$  с вершинами  $p$  и ребрами  $q$  называется матрица  $B(G) = (b_{ij})$  размерности  $p \times q$ , элементы которой определяются следующим образом: 1, если вершина начало дуги ; -1, если вершина конец дуги ; 1, если начало; 0, если вершина не инцидентна дуге.

**Список смежности.** Представляет собой структуру данных, которая для каждой вершины графа хранит список смежных с ней вершин. Список представляет собой массив указателей,  $i$ -ый элемент которого содержит указатель на список вершин, смежных с  $i$ -ой вершиной.

**Список инцидентности.** Эта структура содержит для каждой вершины  $v \in V$  список таких вершин  $u \in V$ , что существует дуга  $(v, u)$  в орграфе или ребро  $(v, u)$  в неориентированном графе. Каждый элемент такого списка содержит два поля: информацию о вершине и указатель на следующий элемент в списке.

### 3) Изоморфизм графов

Два графа  $G$  и  $H$  изоморфны (записывается  $G \cong H$ ), если между их множествами вершин существует взаимно однозначное соответствие, сохраняющее смежность. Если ребра ориентированы, то их направления также должны соответствовать друг другу. Очевидно, что отношение изоморфизма графов является эквивалентностью, т. е. оно рефлексивно, симметрично и транзитивно. Следовательно, множество всех графов разбивается на классы так, что графы из одного класса попарно изоморфны, а графы из разных классов не изоморфны. Изоморфные графы естественно отождествлять, т. е. считать совпадающими (их можно изображать одним рисунком). На практике доказывается с помощью перестановки строк и столбцов матрицы смежности.

### 4) Частичные графы. Подграфы

Граф  $H(X)$  называется **частичным** для графа  $G(X)$ , если все ребра  $H(X)$  являются ребрами  $G(X)$  и множество вершин графа  $H(X)$  совпадает с множеством вершин графа  $G(X)$ . Частичный граф содержит **часть ребер** (дуг). Подграф содержит **часть вершин** вместе с ребрами, соединяющими эти вершины. Граф называется связным, если любые две его вершины можно соединить цепью. Ребро графа называется перешейком, если его удаление приводит к тому, что граф становится несвязным. Граф из одних перешейков называется деревом.

### 5) Маршруты, цепи, циклы

Маршрутом в графе называется чередующаяся последовательность вершин и рёбер. Если  $vq \dots vk$ , то маршрут замкнут, иначе — открыт. Если все рёбра различны, то маршрут называется цепью. Замкнутая цепь называется циклом, замкнутая простая цепь называется простым циклом. Число циклов в графе  $G$  обозначается  $z(G)$ . Граф без циклов называется ациклическим. Для орграфов цепь называется путем, а цикл — контуром. Цепь(цикл) называется гамильтоновым, если он проходит через все вершины графа по одному разу.

### 6) Связность. Цикломатическое число

**1.** Неориентированный граф считается связным, если из любой вершины есть путь в любую другую вершину (путь может состоять из любого количества рёбер) - Чтобы граф с  $n$  вершинами был связным, он должен иметь не менее  $(n-1)$  рёбер. - Если граф имеет не менее  $(n^2 - 3n + 4)/2$  рёбер, то он гарантированно связный. - Если граф связный, у него обязательно есть вершины степени не менее 2, то есть вершины, каждая из которых имеет не менее двух смежных вершин. - Если граф связный и без циклов (то есть это дерево), то удаление любого ребра приведёт к потере связности. **2.**

Величина  $k = N - n + 1$  называется **цикломатическим числом** связного графа.  $N$  — ребра,  $n$  — вершины. В формуле  $q=1$  — для связного графа,  $q$  — количество компонентов связности.



## 7) Плоские и планарные графы. Свойства планарных графов

**Плоский граф** – это граф, который нарисован на плоскости так, что никакие два его ребра не пересекаются. **Планарный граф** – это граф, изоморфный плоскому графу. Для связного плоского графа справедливо следующее соотношение между количеством вершин  $V(G)$ , рёбер  $E(G)$  и граней, формула Эйлера:

$$F(G) - |V(G)| - |E(G)| + |F(G)| = 2$$

Плоский граф – это граф уложенный на плоскость, а планарный граф – это граф который можно уложить на плоскость.

## 8) Операции над графами

- 1) **Дополнением графа**  $G_1(V_1, E_1)$  (обозначение —  $\overline{G_1}(V_1, E_1)$ ) называется граф  $G_2(V_2, E_2)$ , где  $V_2 = V_1$  &  $E_2 = \overline{E_1} = \{e \in V_1 \times V_1 \mid e \notin E_1\} = V \times V \setminus E_1$ .
- 2) **Объединением** (дизъюнктивным) графов  $G_1(V_1, E_1)$  и  $G_2(V_2, E_2)$  (обозначение —  $G_1(V_1, E_1) \cup G_2(V_2, E_2)$ , при условии  $V_1 \cap V_2 = \emptyset$ ) называется граф  $G(V, E)$ , где  $V = V_1 \cup V_2$  &  $E = E_1 \cup E_2$ .
- 3) **Соединением графов**  $G_1(V_1, E_1)$  и  $G_2(V_2, E_2)$  (обозначение —  $G_1(V_1, E_1) + G_2(V_2, E_2)$ , при условии  $V_1 \cap V_2 = \emptyset$ ) называется граф  $G(V, E)$ , где  $V = V_1 \cup V_2$  &  $E = E_1 \cup E_2 \cup \{e = (v_1, v_2) \mid v_1 \in V_1 \text{ & } v_2 \in V_2\}$ .
- 4) **Удаление вершины**  $v$  из графа  $G_1(V_1, E_1)$  (обозначение —  $G_1(V_1, E_1) - v$  при условии  $v \in V_1$ ) даёт граф  $G_2(V_2, E_2)$ , где  $V_2 = V_1 - v$  &  $E_2 = E_1 \setminus \{e = (v_1, v_2) \mid v_1 = v \vee v_2 = v\}$ .
- 5) **Удаление ребра**  $e$  из графа  $G_1(V_1, E_1)$  (обозначение —  $G_1(V_1, E_1) - e$  при условии  $e \in E_1$ ) даёт граф  $G_2(V_2, E_2)$ , где  $V_2 = V_1$ ,  $E_2 = E_1 - e$ .
- 6) **Добавление вершины**  $v$  в граф  $G_1(V_1, E_1)$  (обозначение —  $G_1(V_1, E_1) + v$  при условии  $v \notin V_1$ ) даёт граф  $G_2(V_2, E_2)$ , где  $V_2 = V_1 + v$ ,  $E_2 = E_1$ .
- 7) **Добавление ребра**  $e$  в граф  $G_1(V_1, E_1)$  (обозначение —  $G_1(V_1, E_1) + e$  при условии  $e \notin E_1$ ) даёт граф  $G_2(V_2, E_2)$ , где  $V_2 = V_1$ ,  $E_2 = E_1 + e$ .
- 8) **Стягивание** (правильного) подграфа  $A$  графа  $G_1(V_1, E_1)$  (обозначение —  $G_1(V_1, E_1) / A$  при условии  $A \subset V_1, v \notin V_1$ ) даёт граф  $G_2(V_2, E_2)$ , где  $V_2 = (V_1 \setminus A) + v$ ,  $E_2 = E_1 \setminus \{e = (u, w) \mid u \in A \vee w \in A\} \cup \{e = (u, v) \mid u \in \Gamma(A) \setminus A\}$ .
- 9) **Размножение вершины**  $v$  графа  $G_1(V_1, E_1)$  (обозначение —  $G_1(V_1, E_1) \uparrow v$  при условии  $v \in V_1, v' \notin V_1$ ) даёт граф  $G_2(V_2, E_2)$ , где  $V_2 = V_1 + v'$  &  $E_2 = E_1 \cup \{(v, v')\} \cup \{e = (u, v') \mid u \in \Gamma^+(v)\}$ .
- 10) **Отождествление вершин**. Пусть  $u$  и  $v$  – две вершины графа  $G = (V, E)$ . Удалим эти вершины из графа  $G$  и добавим новую вершину  $x$ , соединив ее ребром с каждой вершиной, входящей в объединение окружений вершин  $u$  и  $v$  в исходном графе  $G$ . Построенный граф получился из графа  $G$  отождествлением вершин  $u$  и  $v$ . Отождествление вершин  $u$  и  $v$  называется стягиванием ребра  $(u, v)$ , если  $(u, v) \in E(G)$ .

- 11) Расщепление вершины.** Пусть  $v$  – вершина графа  $G$ . Рассмотрим два множества  $N_1(v)$  и  $N_2(v)$ , объединение которых совпадает с окружением  $N(v)$  вершины  $v$ . Удалив вершину  $v$ , добавим новые вершины  $v_1, v_2$  и ребро  $(v_1, v_2)$ . Соединим  $v_1$  с каждой вершиной из  $N_1(v)$ , а  $v_2$  – с каждой вершиной из  $N_2(v)$ . Произведенная операция называется расщеплением вершины  $v$ , а полученный граф обозначается  $G_v^*$ .
- 12) Дублирование вершин.** При дублировании графа  $G$  получим новый граф  $G_2$  следующим образом: добавим вершину  $u'$  и соединим эту вершину со всеми вершинами, которые смежны с вершиной  $u$ .
- 13) Разбиение ребра.** При разбиении ребра  $l(u, v)$  графа  $G$  получаем новый граф следующим образом: удалим ребро  $e(u, v)$  из множества графа, добавим вершину  $w$ . Добавим рёбра  $(u, w)$ ,  $(w, v)$ .
- 14) Пересечение графов.** Пусть графов  $G_1(V_1, E_1)$  и  $G_2(V_2, E_2)$  – произвольные графы. Пересечением  $G_1(V_1, E_1) \cup G_2(V_2, E_2)$ , называется граф со множеством вершин  $V = V_1 \cap V_2$  и множеством ребер  $E = E_1 \cap E_2$ .
- 15) Композиция графов.** Пусть  $G_1(V_1, E_1)$  и  $G_2(V_2, E_2)$  – два ориентированных графа с одними и теми же множествами вершин  $V$ . Композицией (иногда произведением)  $G_1(G_2)$  графов  $G_1$  и  $G_2$  называется ориентированный граф с множеством вершин  $V$ , в котором существует дуга  $(v_i, v_j)$  тогда и только тогда, когда для некоторой вершины  $v \in V$  существуют дуги  $(v_i, v) \in E_1$  и  $(v, v_j) \in E_2$ . Операция композиции может быть выполнена в матричной форме.
- 16) Декартово произведение графов.**  $V(G_1 \times G_2) = V(G_1) \times V(G_2)$ .

## 9) Деревья, основные понятия, определения и теоремы

Связный граф без циклов называется (свободным) деревом. Деревья являются в некотором смысле простейшим классом графов.

**Ориентированные деревья.** Ориентированным деревом (или ордеревом, или корневым деревом) называется орграф со следующими свойствами.

1. Существует единственный узел  $r$ , полустепень захода которого равна 0,  $d^+(r) = 0$ . Он называется корнем ордерова.
2. Полустепень захода всех остальных узлов равна 1,  $d^+(v) = 1$ .
3. Каждый узел достижим из корня.

**Упорядоченные деревья.** Множества  $T_1, \dots, T_k$  в эквивалентном определении ордерова являются поддеревьями. Если относительный порядок поддеревьев  $T_1, \dots, T_k$  фиксирован, то ордерова называется упорядоченным.

**Последовательное дерево**, представляющее собой простую цепь, и звездное дерево (или куст), в котором одна из вершин (центр) смежна со всеми остальными вершинами. **Лесом** называют граф, связные компоненты которого являются деревьями.

Свойства: 1. Каждое дерево с  $n$  вершинами имеет в точности  $n-1$  ребро. 2. Граф является деревом тогда и только тогда, когда каждая пара различных вершин графа соединяется одной и только одной простой цепью. 3. У каждого дерева найдется висячая вершина. 4. При удалении любого ребра дерева оно распадается на связные компоненты, являющиеся либо изолированными вершинами, либо деревьями. При добавлении в дерево любого нового ребра в нем образуется простой цикл, и оно перестает быть деревом.

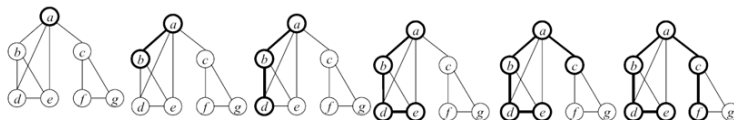
**Полные бинарные деревья.** Полным бинарным деревом будем называть такое дерево, в котором каждая вершина имеет не более двух «сыновей», а заполнение вершин осуществляется в порядке от верхних уровней к низшим, причем на одном уровне заполнение вершин производится слева направо. Верхним считается уровень с номером 1 (самый высокий).

## 10) Остовное дерево минимального веса и способы его построения

Подграф в  $G$ , являющийся деревом и включающий в себя все вершины  $G$ , называется остовным деревом. Остовное дерево в графе  $G$  строится просто: выбираем произвольное его ребро и последовательно добавляем другие ребра, не создавая при этом циклов, до тех пор, пока нельзя будет добавить никакого ребра, не получив при этом цикла. Для построения остовного дерева в графе из  $n$  вершин необходимо выбрать ровно  $n - 1$  ребро. На языке теории графов нам нужно в нагруженном графе найти остовное дерево наименьшего общего веса. Такое дерево принято называть минимальным остовным деревом или, сокращенно, МОД. **Алгоритм Прима:** берем любую вершину, всякий раз ищется минимальное по весу ребро, один конец которого — уже взятая в граф вершина, а другой конец — ещё не взятая, и это ребро добавляется в граф (если таких рёбер несколько, можно взять любое). Этот процесс повторяется до тех пор, пока граф не станет содержать все вершины (или, что то же самое, ребро).

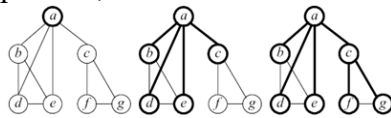
## 11) Обходы вершин графа: поиск в ширину и поиск в глубину

**В глубину:** когда возможные пути по *ребрам*, выходящим из вершин, разветвляются, нужно сначала полностью исследовать одну ветку и только потом переходить к другим.



Таким образом, основная идея **поиска в ширину** заключается в том, что сначала исследуются все вершины, смежные с начальной вершиной (*вершина с которой*

начинается обход).



## 12) Задача о кратчайшем пути в орграфе. Алгоритм Форда

Находит кратчайшие пути от одной вершины графа до всех остальных во взвешенном графе. Вес ребер может быть отрицательным.

### ЗАДАЧА

Дан ориентированный или неориентированный граф  $G$  со взвешенными рёбрами. Требуется найти кратчайшие пути от выделенной вершины  $s$  до всех вершин графа.

### АЛГОРИТМ

.Перед началом работы алгоритма, для всех вершин, кроме стартовой, расстояние полагается равным бесконечности.

.В цикле проверяется необходимость производить релаксацию для конкретного ребра (сравнение текущего пути, с заново посчитанным).

.Если текущая метка вершины больше чем метка нового пути, то она изменяется в его сторону, иначе остается неизменной.

.Алгоритм заканчивает свою работу, только если на одном из его очередных шагов не было не проведено ни одной релаксации.

$$a_{ij} = \begin{cases} 0, & \text{если вершины } v_i \text{ и } v_j \text{ совпадают,} \\ \infty, & \text{если вершины } v_i \text{ и } v_j \text{ не смежны,} \\ w, & \text{если вершины } v_i \text{ и } v_j \text{ смежны, и } w - \text{вес ребра } (v_i, v_j). \end{cases}$$

## 13) Отношение порядка между вершинами орграфа

В любом ориентированном графе без циклов можно установить отношение порядка между его вершинами. Рассмотрим сеть  $G=(J,U)$  с одним источником  $S$  и одним стоком  $T$ . В сети  $n$  вершин. Рангом вершины  $i$  назовем  $\max$  число дуг в пути, которое связывает источник – вершину  $S$ , с данной вершиной  $i$ .

Правильной считается такая нумерация вершин в сети, при которой все номера вершин  $g$ -го ранга будут меньше номеров вершин  $(t+1)$ -го ранга. Рассмотрим один из способов, который называется методом вычеркивания дуг.

Исходной вершине сети, т.е. источнику присваиваем номер 1. Затем вычеркнем (удалим) все дуги выходящие из вершины 1. После этого в сети окажутся конечное число вершин без входящих дуг. Назовем эти вершины, вершины 1-го ранга и присвоим им в произвольном порядке номера  $2, \dots, k_1$ . Затем вычеркнем (удалим) все дуги, выходящие из вершин 1-го ранга. В сети опять окажется конечное число вершин без входящих дуг. Назовем их вершинами 2-го ранга и присвоим им в произвольном порядке номера  $k_1+1, \dots, k_2$ .

И т.д., процесс нумерации вершин прекращается тогда, когда будет пронумерована последняя вершина сети – сток.

#### 14) Задача о пути максимальной длины в орграфе

Задача ставится следующим образом.

Задан конечный ориентированный граф без контуров  $G(X, U)$ .

Каждой дуге графа “ $u$ ” ставится в соответствие длина дуги  $l(u)$ .

Требуется определить длиннейший путь, соединяющий две вершины графа  $x_0$  и  $x_n$ .

##### Алгоритм

Каждая вершина графа получает числовую метку, которая может меняться конечное число раз. Установившаяся метка – величина длиннейшего пути из вершины  $x_0$  в данную вершину  $x_j$ . В частности, установившаяся метка вершины  $x_n$  есть величина длиннейшего пути из  $x_0$  в  $x_n$ .

Чтобы определить искомый путь, нужно рассмотреть последовательность шагов, на каждом из которых ищется одна из дуг длиннейшего пути между  $x_0$  и  $x_n$ .

**Алгоритм** состоит в последовательном проведении следующих этапов:

1. Полагаем  $\lambda_0 = 0$ ;  $\lambda_i = -\infty$  ( $i = 1, \dots, n$ ).

2. Ищем дугу  $(x_i, x_j)$  такую, что  $\lambda_j - \lambda_i \leq l(x_i, x_j)$ . Если такой дуги нет, то не существует пути, соединяющего  $x_0$  и  $x_n$ . Если такая дуга найдется, то изменяем метку  $\lambda_j$  на  $\lambda'_j = \lambda_j + l(x_i, x_j)$ .

3. Продолжаем процедуру пункта 2 до тех пор, пока метки вершин  $x_i$  не перестанут меняться.

Установленные метки обозначим  $\lambda_i^*$ . При этом могут встретиться два случая:

1)  $\lambda_n^* = -\infty$ , это соответствует тому, что пути, соединяющего вершины  $x_0$  и  $x_n$ , не существует;

2)  $\lambda_n^*$  – конечное число. Оно равно длине пути максимальной длины из  $x_0$  в  $x_n$ .

Сам путь находим, отмечая вершины, по которым достигается максимум, т.е. те вершины,

для которых  $\lambda_j^* = \lambda_i^* + l(x_i, x_j)$

Если между вершинами графа-сети установлено отношение порядка, т.е. они “правильно” занумерованы, то решение задачи можно получить за один шаг, произведя подсчет меток с учетом следующей формулы:

$$\lambda_j = \max_i \{ \lambda_i + l(x_i, x_j) \}$$

Пример.

Определим длиннейший путь на графе, изображенном на рис.3.3.1, а также его длину.

Вначале полагаем для вершины  $x_0$   $\lambda_0 = 0$  и  $\lambda_j = -\infty$  для вершин  $x_i$  ( $i = 1, \dots, 5$ ).

Затем, т.к.  $\lambda_1 - \lambda_0 = -\infty < l(x_0, x_1)$ , меняем метку вершины  $x_1$ , т.е.  $\lambda_1$ , на

$$\lambda'_1 = \lambda_0 + l(x_0, x_1) = 2 \quad (x_0)$$

Аналогично  $\lambda'_2 = \lambda_0 + l(x_0, x_2) = 4$ .

Чтобы найти метку вершины  $x_3$ , пользуясь формулой (3.3.2)

$$\lambda'_3 = \max \{ [\lambda'_1 + l(x_1, x_3)], [\lambda_0 + l(x_0, x_3)] \} = \max \{ (2+4), (0+5) \} = 6 \quad (x_1)$$

Справа в скобках отмечаем вершины, по которым достигается максимум длины.

Аналогично

$$\lambda'_4 = \max \{ [\lambda'_1 + l(x_1, x_4)], [\lambda'_3 + l(x_3, x_4)] \} = \max \{ (2+3), (6+6) \} = 12 \quad (x_3)$$

$$\begin{aligned} \lambda'_5 &= \max \{ [\lambda'_3 + l(x_3, x_5)], [\lambda'_4 + l(x_4, x_5)], [\lambda'_2 + l(x_2, x_5)] \} = \\ &= \max \{ (6+4), (12+2), (4+7) \} = 14 \end{aligned} \quad (x_4)$$

Искомый путь имеет длину  $l(\square) = \lambda'_5 = 14$ . Причем в  $x_5$  он идет из вершины  $x_4$ , в  $x_4$  из  $x_3$ , в  $x_3$  из  $x_1$ , в  $x_1$  из  $x_0$ :  $x_5, x_4, x_3, x_1, x_0$ . Следовательно  $\square = (x_0, x_1, x_3, x_4, x_5)$ .

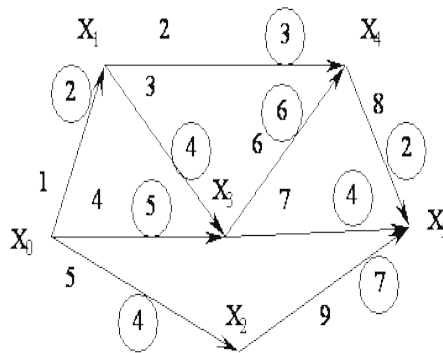
Путь максимальной длины называют критическим путем. Следовательно, критический путь в рассмотренном примере есть  $\square = (x_0, x_1, x_3, x_4, x_5)$ , а его длина  $l(\square) = 14$ .

### 15) Сетевое планирование. Задача о скорейшем пути завершения проекта

Сетевое планирование — метод анализа сроков (ранних и поздних) начала и окончания нереализованных частей проекта, позволяет увязать выполнение различных работ и процессов во времени, получив прогноз общей продолжительности реализации всего проекта. Задача сетевого планирования состоит в том, чтобы графически, наглядно и системно отобразить и оптимизировать последовательность и взаимозависимость работ, действий или мероприятий, обеспечивающих своевременное и планомерное достижение конечных целей.

Рассмотрим некоторый проект – совокупность операций (работ), составляющий некоторый многошаговый процесс. Пусть данные о строительстве приведены в следующей таблице

Виды работ	Какие работы следуют за перечисленными	Продолжительность работ
1	2,3	2
2	8	3
3	6,7	4
4	6,7	5
5	9	4
6	8	6
7	–	4
8	–	2
9	–	7



Эту информацию о проекте представим в виде графа-сети. Дугами графа будем изображать работы, а вершинами графа – некоторые события. Назовем элементарными событиями начало и конец каждой работы, а некоторую совокупность элементарных событий – событием.

Вход графа – событие, заключающееся в начале всего проекта. Оно является событием, стоящим в начале одной или нескольких работ, а именно тех, которые не следуют ни за какими другими, т.е. работ, с которых может быть начато строительство. В нашем примере такими работами являются №1,4,5 (их нет во 2-ом столбце). Выходом графа будет являться событие, заключающееся в окончании работ, за которыми не следуют никакие другие работы, т.е. в окончании всего проекта. В данном примере – это работы №7,8,9. Все другие вершины графа есть события, заключающиеся в окончании одних и начале других работ. На графе номер работы обозначен числом вне кружка. Число, обведенное кружком, есть продолжительность данной работы.

Путь максимальной длины из вершины  $x_0$  в  $x_i$  есть скорейшее время наступления события  $x_i$ . В самом деле, событие  $x_3$ , например, соответствующее началу 6-й и 7-й работ, может произойти только после окончания 3-й и 4-й работ, а следовательно, и после окончания 1-й, т.к. для выполнения 3-й работы необходимо окончание 1-й работы. Следовательно, скорейшее время наступления события  $x_3$  есть  $\max\{5, (2+4)\}=6$ .

Скорейшее время наступления события 5 есть скорейшее время окончания проекта в целом и равно длине пути максимальной длины из вершины  $x_0$  в  $x_5$ .

Итак, если  $x_0$  и  $x_n$  есть вход и выход графа-сети, соответствующего данному проекту, то для определения наиболее раннего срока окончания всех работ нужно найти путь максимальной длины из  $x_0$  в  $x_n$ , т.е. критический путь, и определить его длину. Время, соответствующее скорейшему окончанию работ, т.е. скорейшему завершению проекта, называется критическим временем данного проекта. Оно численно совпадает с длиной критического пути из  $x_0$  в  $x_n$ .

В приведенном примере критический путь, проходящий через вершины  $x_0, x_1, x_3, x_4, x_5$ , имеет длину, равную 14  $l(m)=14$ , т.е. критическое время данного проекта равно 14.

Работы, составляющие критический путь, называются критическими работами (операциями). От своевременного выполнения критических операций зависит срок завершения проекта. Они не допускают запаздывания в исполнении в отличие от некритических операций.



## ТЕОРИЯ АВТОМАТОВ

### 1. Конечные автоматы, их реализация и применение.

Конечный автомат (Finite State Machine, FSM) - это абстрактная модель вычислений, которая может находиться в одном из конечного числа состояний в определенный момент времени. В зависимости от входных данных и текущего состояния автомат может переходить в другие состояния по определенным правилам. Конечные автоматы широко применяются в различных областях, включая программирование, цифровую логику, сетевые протоколы, компиляцию и многое другое.

Реализация конечного автомата:

1. Таблица переходов: В этом подходе каждое состояние представлено строкой таблицы, а входные символы - столбцами. В ячейках таблицы указывается состояние, в которое перейдет автомат при данном входе и текущем состоянии.

2. Диаграмма переходов (State Transition Diagram): Это графическое представление конечного автомата, где вершины представляют состояния, а дуги - переходы между ними. Каждая дуга помечена входным символом или условием перехода.

3. Кодирование в программе: Конечный автомат может быть реализован с помощью программного кода на различных языках программирования. Для каждого состояния и перехода может быть написан соответствующий код.

Применение конечных автоматов:

1. Синтаксический анализ и компиляция: Конечные автоматы часто используются в синтаксическом анализе для распознавания и разбора языков. Например, лексические анализаторы могут быть реализованы с

использованием конечных автоматов для разбора и классификации лексем.

2. Управление протоколами сетевого взаимодействия: Конечные автоматы широко применяются в сетевых протоколах для управления соединением, обменом сообщениями и управлением сеансом.

3. Управление состоянием в программном обеспечении: Конечные автоматы могут быть использованы для управления состоянием в различных приложениях и программных системах. Например, управление жизненным циклом объектов в GUI-приложениях.

4. Распознавание и управление последовательностями данных: Конечные автоматы могут использоваться для распознавания и обработки последовательностей данных, таких как звуки, изображения, текст и т.д.

Конечные автоматы являются мощным инструментом моделирования и управления состоянием и поведением различных систем и процессов. Их простота и эффективность делают их широко применимыми в различных областях.

## 2. Классификация абстрактных автоматов.

Абстрактные автоматы могут быть классифицированы по различным критериям. Одним из основных способов классификации является их выразительная сила или способность выражать различные языки или функции. Вот несколько основных типов абстрактных автоматов:

### 1. Конечные автоматы (Finite State Machines, FSM):

- Детерминированные конечные автоматы (Deterministic Finite Automata, DFA)
- Недетерминированные конечные автоматы (Nondeterministic Finite Automata, NFA)
- Конечные автоматы с выводом (Finite State Transducers, FST)

## 2. Конечные автоматы с памятью:

- Конечные автоматы с ограниченной памятью (Finite State Machines with Limited Memory)
- Конечные автоматы с бесконечной памятью (Finite State Machines with Infinite Memory)

## 3. Конечные автоматы с выводом:

- Конечные автоматы с выводом (Finite State Transducers, FST)

## 4. Регулярные грамматики и регулярные выражения:

- Регулярные грамматики (Regular Grammars)
- Регулярные выражения (Regular Expressions)

## 5. Детерминированные и недетерминированные автоматы:

- Детерминированные автоматы
- Недетерминированные автоматы

## 6. Автоматы с конечной памятью:

- Машины Тьюринга (Turing Machines)
- Машины Маркова (Markov Machines)

## 7. Автоматы с бесконечной памятью:

- Машины Тьюринга (Turing Machines)
- Машины Маркова (Markov Machines)

### 3. Автоматное программирование. Графы переходов.

Автоматное программирование (Automata-based programming) - это подход к программированию, основанный на использовании автоматов для описания поведения программ или систем. Он предполагает моделирование поведения системы с помощью конечных автоматов или других типов автоматов, а затем преобразование этих моделей в код программы или в управляющие структуры.

Графы переходов (Transition Graphs) - это графическое представление структуры конечного автомата или другого автомата, показывающее его состояния, переходы между состояниями и условия переходов. Графы переходов являются интуитивно понятным способом визуализации поведения автомата и обеспечивают понимание его работы.

Процесс автоматного программирования с использованием графов переходов включает следующие шаги:

1. Моделирование: Определение состояний системы и условий переходов между этими состояниями. Это может включать в себя разработку конечного автомата, определение начального состояния, конечных состояний и условий переходов.
2. Построение графа переходов: Создание графического представления автомата в виде графа, где вершины представляют состояния, а дуги - переходы между ними.
3. Анализ и проверка: Проверка графа переходов на корректность и адекватность модели. Это включает в себя проверку полноты покрытия всех возможных переходов и состояний, а также анализ наличия недостижимых состояний или недопустимых переходов.
4. Генерация кода или управляющих структур: На основе графа переходов генерируется код программы или другие управляющие структуры, которые реализуют поведение системы. Это может быть

реализовано с помощью языков программирования, таких как C, C++, Python и т.д., или с использованием специализированных средств программирования, таких как Stateflow в MATLAB/Simulink.

Преимущества использования автоматного программирования с графами переходов включают упрощение анализа и проектирования систем, интуитивное понимание поведения системы, а также возможность автоматической генерации кода на основе моделей автоматов. Однако этот подход может быть ограничен в выразительной силе для некоторых видов систем или сложных алгоритмов.

#### 4. Теоретико-множественное определение автомата. Инициальные, синхронные и асинхронные автоматы.



**ChatGPT**

В теории автоматов автомат может быть определен как математическая модель, состоящая из конечного числа состояний и правил перехода между этими состояниями в ответ на входные данные. Давайте определим автоматы с теоретико-множественной точки зрения:

##### 1. Автоматы:

- Автомат  $A$  определяется как пятерка  $A = (Q, \Sigma, \delta, q_0, F)$ , где:
  - $Q$  - конечное множество состояний.
  - $\Sigma$  - конечный алфавит входных символов.
  - $\delta$  - функция перехода, определяющая переходы между состояниями. Это может быть частичная функция  $\delta : Q \times \Sigma \rightarrow Q$ , где  $Q \times \Sigma$  - множество всех возможных пар состояние-входной символ.
  - $q_0$  - начальное состояние,  $q_0 \in Q$ .
  - $F$  - множество конечных состояний,  $F \subseteq Q$ .

##### 2. Инициальные, синхронные и асинхронные автоматы:

- **Инициальные автоматы:** Это автоматы, в которых начальное состояние явно определено. То есть для каждого автомата  $A$  определено начальное состояние  $q_0$ .
- **Синхронные автоматы:** Это автоматы, в которых переход между состояниями происходит синхронно с входными символами. Каждый входной символ приводит к одному и только одному переходу. Функция перехода  $\delta$  определена для всех пар  $(q, \sigma)$ , где  $q$  - текущее состояние,  $\sigma$  - входной символ.
- **Асинхронные автоматы:** В асинхронных автоматах переходы могут происходить в любое время и независимо друг от друга. Это означает, что функция перехода  $\delta$  может быть частичной, а входные символы могут обрабатываться в любом порядке.

Эти различные типы автоматов могут иметь разные свойства и применения в зависимости от требований задачи. Например, синхронные автоматы часто используются в цифровой логике и сетевых протоколах, тогда как асинхронные автоматы могут быть более удобны для моделирования некоторых аспектов параллельных вычислений или взаимодействия между компонентами системы.

## 5. Автоматы Мили и Мура.



### ChatGPT

Автоматы Мили (Mealy Machine) и Мура (Moore Machine) - это два основных типа конечных автоматов, которые отличаются способом описания выходного поведения в зависимости от текущего состояния и входных данных.

#### 1. Автомат Мили (Mealy Machine):

- В автомате Мили выходные данные (выходной символ) зависят от текущего состояния и входного символа.
- Каждый переход между состояниями сопровождается выходной символ.
- Формально, автомат Мили определяется как пятерка  $M = (Q, \Sigma, \Gamma, \delta, \lambda)$ , где:
  - $Q$  - множество состояний.
  - $\Sigma$  - входной алфавит.
  - $\Gamma$  - выходной алфавит.
  - $\delta$  - функция перехода, определяющая переходы между состояниями:  $\delta : Q \times \Sigma \rightarrow Q$ .
  - $\lambda$  - функция выхода, определяющая выходной символ для каждого перехода:  $\lambda : Q \times \Sigma \rightarrow \Gamma$ .

#### 2. Автомат Мура (Moore Machine):

- В автомате Мура выходные данные (выходной символ) зависят только от текущего состояния.
- Выходные символы связаны с состояниями, а не с переходами.
- Формально, автомат Мура также определяется как пятерка  $M = (Q, \Sigma, \Gamma, \delta, \lambda)$ , где:
  - $Q$  - множество состояний.
  - $\Sigma$  - входной алфавит.
  - $\Gamma$  - выходной алфавит.
  - $\delta$  - функция перехода:  $\delta : Q \times \Sigma \rightarrow Q$ .
  - $\lambda$  - функция выхода, определяющая выходной символ для каждого состояния:  $\lambda : Q \rightarrow \Gamma$ .



## 6. Табличная форма задания автоматов.

Табличная форма задания автоматов - это удобный способ представления конечных автоматов с помощью таблицы, где строки представляют состояния, столбцы - входные символы или условия перехода, а ячейки содержат информацию о переходах и выходных данных. Вот как может выглядеть таблица для автомата Мили и Мура:

Табличная форма для автомата Мили:

...

Состояние	Входной символ 1	Входной символ 2	...	Выходной символ
q0	q1 / Y1	q2 / Y2	...	Z1
q1	q3 / Y3	q4 / Y4	...	Z2
q2	q5 / Y5	q6 / Y6	...	Z3
...	...	...	...	...

В этой таблице каждая строка представляет состояние автомата, каждый столбец - входной символ, а содержимое ячейки указывает на следующее состояние (например, q1) и выходной символ (например, Y1) для данного состояния и входного символа.

Табличная форма для автомата Мура:

Состояние	Входной символ 1	Входной символ 2	...	Выходной символ
q0	q1	q2	...	Z1
q1	q3	q4	...	Z2
q2	q5	q6	...	Z3
...	...	...	...	...

В этой таблице каждая строка представляет состояние автомата, каждый столбец - входной символ, а содержимое ячейки указывает на следующее состояние для данного состояния и входного символа. Выходной символ связан с каждым состоянием и указывается в последнем столбце.

Табличная форма предоставляет компактный и удобный способ описания структуры и поведения конечных автоматов, что делает ее полезной для анализа, проектирования и реализации автоматов в программном коде.

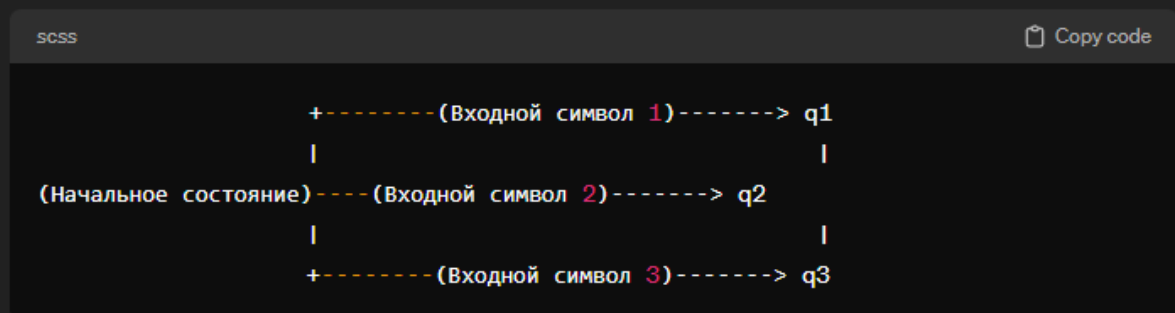


## 7. Графовая форма задания автоматов.

Графовая форма задания автоматов представляет собой графическое изображение структуры и поведения конечного автомата. Обычно в этой форме вершины графа представляют состояния, а дуги между вершинами представляют переходы между состояниями, обусловленные входными символами или условиями.

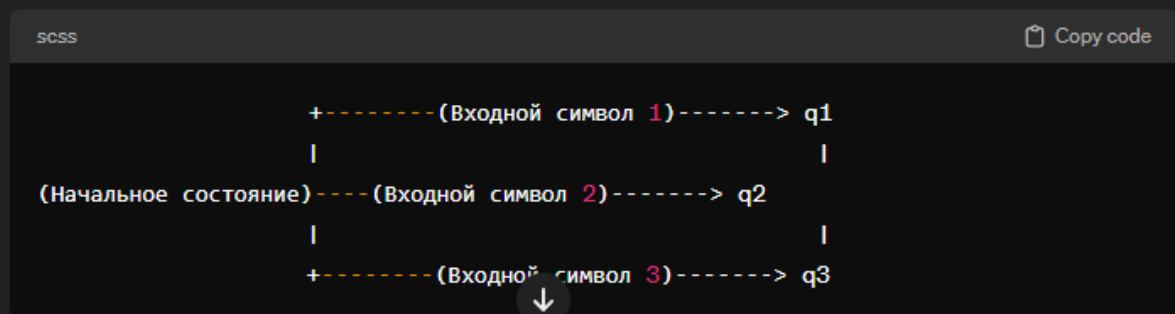
Вот как может выглядеть графовая форма для автомата Мили и Мура:

### Графовая форма для автомата Мили:



В этом графе каждая вершина представляет состояние автомата, а дуги – переходы между состояниями в ответ на определенные входные символы. Начальное состояние обозначено как стартовая вершина, а выходные символы обычно не отображаются непосредственно на графе, но могут быть указаны отдельно.

### Графовая форма для автомата Мура:



## 8. Матричная форма задания автоматов.

### матричная форма для автомата Мили.

Для автомата Мили мы можем использовать две матрицы: матрицу переходов и матрицу выходов.

#### 1. Матрица переходов $T$ :

- Эта матрица определяет переходы между состояниями в ответ на входные символы.
- Элемент  $T_{ij}$  указывает на состояние, в которое перейдет автомат, находясь в состоянии  $q_i$  и получив входной символ  $\sigma_j$ .

#### 2. Матрица выходов $O$ :

- Эта матрица определяет выходные символы для каждого перехода.
- Элемент  $O_{ij}$  указывает на выходной символ, который будет выдан, когда автомат перейдет в состояние  $q_i$  и получит входной символ  $\sigma_j$ .

Пример:

Матрица переходов:

$$T = \begin{pmatrix} q_1 & q_2 \\ q_2 & q_3 \end{pmatrix}$$

Матрица выходов:

$$O = \begin{pmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{pmatrix}$$

### Матричная форма для автомата Мура:

Для автомата Мура мы можем использовать одну матрицу - матрицу выходов.

#### 1. Матрица выходов $O$ :

- Эта матрица определяет выходные символы для каждого состояния.
- Элемент  $O_{ij}$  указывает на выходной символ, который будет выдан, когда автомат находится в состоянии  $q_i$ .

Пример:

Матрица выходов:

$$O = \begin{pmatrix} Z_1 \\ Z_2 \\ Z_3 \end{pmatrix}$$

Эти матричные формы представления конечных автоматов облегчают анализ и реализацию автоматов с помощью матричных операций  $\downarrow$  и алгоритмов.

## 9. Понятие частичного автомата. Реакция автомата.

Частичный автомат - это конечный автомат, у которого функция перехода  $\delta$  определена не для всех пар состояние-входной символ, а только для некоторых из них. Другими словами, функция перехода может быть частичной, а не всюду определенной. Это означает, что для некоторых состояний и входных символов не существует определенного следующего состояния.

В частичном автомате могут возникать ситуации, когда автомат не может определить следующее состояние в ответ на некоторый входной символ. В таких случаях обычно принимается одно из следующих решений:

1. Неопределенное состояние (Undefined State): Автомат остается в текущем состоянии, и возвращается специальный код, указывающий на отсутствие перехода для данного входного символа.
2. Задание дополнительного состояния (Additional State): Создается дополнительное специальное состояние, которое является целевым состоянием для всех неопределенных переходов.
3. Отказ (Reject): Автомат переходит в отказное состояние, указывающее на то, что входная последовательность не распознана.

Реакция автомата - это действие, которое выполняется автоматом в ответ на входной символ и текущее состояние. Реакция автомата может быть различной в зависимости от типа автомата и его конкретной реализации. Например, для автомата Мили реакция может включать в себя выдачу выходного символа, основанного на текущем состоянии и входном символе, в то время как для автомата Мура реакция может быть просто переходом в следующее состояние без выдачи выходного символа.

В случае частичного автомата реакция также может включать обработку неопределенных переходов и принятие соответствующего решения, как описано выше.

## 10. Переход от автомата Мили к эквивалентному автомату Мура



### ChatGPT

Для перехода от автомата Мили к эквивалентному автомату Мура необходимо выполнить следующие шаги:

1. Определить выходные символы для каждого состояния в автомате Мура.
2. Преобразовать функцию перехода и выходные символы в соответствии с новой структурой.

Предположим, у нас есть автомат Мили с функцией перехода  $\delta$  и функцией выхода  $\lambda$ . Мы хотим преобразовать его в эквивалентный автомат Мура с функцией перехода  $\delta'$  и выходными символами  $\lambda'$ .

1. **Определение выходных символов для каждого состояния в автомате Мура:**

Для каждого состояния  $q$  в автомате Мили определяем выходной символ  $\lambda(q)$ , который соответствует этому состоянию.

2. **Преобразование функции перехода и выходных символов:**

- Функция перехода  $\delta'$  для автомата Мура будет такой же, как и для автомата Мили.
- Выходные символы  $\lambda'$  для автомата Мура будут зависеть только от текущего состояния, а не от текущего состояния и входного символа, как в автомате Мили.

Пример:

Пусть у нас есть автомат Мили с двумя состояниями  $q_0$  и  $q_1$ , входными символами  $a$  и  $b$ , и функциями перехода и выхода:

- $\delta(q_0, a) = q_1, \delta(q_1, a) = q_0, \delta(q_0, b) = q_0, \delta(q_1, b) = q_1$
- $\lambda(q_0, a) = Y_0, \lambda(q_1, a) = Y_1, \lambda(q_0, b) = Y_2, \lambda(q_1, b) = Y_3$

Преобразуем его в эквивалентный автомат Мура:

- Состояния:  $q_0, q_1$
- Входные символы:  $a, b$
- Функция перехода: такая же как в автомате Мили
- Выходные символы:  $Y_0$  для состояния  $q_0$ ,  $Y_1$  для состояния  $q_1$



Таким образом, мы преобразовали автомат Мили в эквивалентный автомат Мура.

## 11. Переход от автомата Мура к эквивалентному автомату Мили



### ChatGPT

Для перехода от автомата Мура к эквивалентному автомату Мили необходимо выполнить следующие шаги:

1. Определить функцию перехода и функцию выхода для автомата Мили на основе состояний и выходных символов автомата Мура.
2. Преобразовать функции перехода и выхода в соответствии с новой структурой.

Предположим, у нас есть автомат Мура с функцией перехода  $\delta$  и выходными символами  $\lambda$ . Мы хотим преобразовать его в эквивалентный автомат Мили с функцией перехода  $\delta'$  и функцией выхода  $\lambda'$ .

1. **Определение функции перехода и функции выхода для автомата Мили:**
  - Функция перехода  $\delta'$  будет содержать переходы, определенные в автомате Мура, с учетом выходных символов.
  - Функция выхода  $\lambda'$  будет возвращать выходной символ, соответствующий новому состоянию.
2. **Преобразование функций перехода и выхода:**
  - Функция перехода  $\delta'$  для автомата Мили будет зависеть от текущего состояния и входного символа, как в автомате Мура.
  - Функция выхода  $\lambda'$  для автомата Мили будет возвращать выходной символ для каждого состояния, как в автомате Мура.

Пример:

Пусть у нас есть автомат Мура с двумя состояниями  $q_0$  и  $q_1$ , входными символами  $a$  и  $b$ , и функциями перехода и выхода:

- $\delta(q_0, a) = q_1, \delta(q_0, b) = q_0, \delta(q_1, a) = q_1, \delta(q_1, b) = q_0$
- $\lambda(q_0) = Y_0, \lambda(q_1) = Y_1$

Преобразуем его в эквивалентный автомат Мили:

- Состояния:  $q_0, q_1$
- Входные символы:  $a, b$
- Функция перехода:
  - $\delta'(q_0, a) = q_1, \delta'(q_0, b) = q_0$
  - $\delta'(q_1, a) = q_1, \delta'(q_1, b) = q_0$
- Функция выхода:
  - $\lambda'(q_0, a) = Y_1, \lambda'(q_0, b) = Y_0$
  - $\lambda'(q_1, a) = Y_1, \lambda'(q_1, b) = Y_0$



Таким образом, мы преобразовали автомат Мура в эквивалентный автомат Мили.

## 12. Минимизация автоматов.



### ChatGPT

Минимизация автоматов - это процесс сокращения количества состояний и переходов в конечном автомате, сохраняя его функциональность. Это важный шаг в проектировании автоматов, так как он помогает сделать автоматы более компактными, эффективными и легче понимаемыми.

Существует несколько методов минимизации конечных автоматов, включая:

1. **Метод эквивалентности состояний (таблица или алгоритм):** Этот метод использует таблицы или алгоритмы для определения эквивалентности состояний и их объединения, если они эквивалентны по функциональности.
2. **Метод удаления недостижимых состояний:** В некоторых случаях некоторые состояния могут быть недостижимыми из начального состояния. Удаление этих состояний помогает уменьшить размер автомата.
3. **Метод поглощения (абсорбции):** Этот метод позволяет объединять эквивалентные состояния в автомате путем итеративного сравнения состояний и их переходов.
4. **Метод двойного разбиения:** Этот метод разбивает множество состояний на классы эквивалентности и итеративно сокращает число классов до минимума.
5. **Метод квинтезии (пять минут):** Этот метод является улучшенной версией метода двойного разбиения и может быть эффективным для больших автоматов.

Минимизация автоматов имеет большое значение в теории формальных языков и автоматов, а также в практических задачах, таких как синтез цифровых схем, разработка компиляторов и т. д. Эффективная минимизация может существенно улучшить производительность и уменьшить затраты ресурсов.



### 13. Распознающие автоматы.



#### ChatGPT

Распознающий автомат (RA) - это модель вычислений, которая используется для распознавания строк или языков. RA состоит из конечного числа состояний и правил перехода между этими состояниями в зависимости от символов входной строки. Он может принимать или отвергать входную строку в зависимости от того, достигнуто ли конечное состояние в конце обработки строки.

RA может быть двух типов:

1. **Конечный автомат (Deterministic Finite Automaton - DFA):** В DFA для каждой пары состояние-входной символ существует ровно один переход, что делает его детерминированным. DFA подходит для распознавания регулярных языков.
2. **Недетерминированный конечный автомат (Nondeterministic Finite Automaton - NFA):** В NFA для каждой пары состояние-входной символ может быть несколько возможных переходов. В результате, NFA более гибок в выражении определенных классов языков, но требует дополнительной обработки для преобразования в эквивалентный DFA.

RA может использоваться в различных областях, включая:

- Распознавание и анализ языков программирования и регулярных выражений.
- Проверка синтаксиса и семантики входных данных.
- Моделирование и анализ протоколов и автоматическое управление.
- Распознавание и анализ естественных языков.
- Различные задачи в области информационной безопасности, такие как обнаружение вредоносных программ и фильтрация спама.

Распознающие автоматы представляют собой мощный инструмент для формального моделирования и анализа различных видов систем и языковых конструкций.