ТИТУЛЬНИК

СОДЕРЖАНИЕ

[1 АЛГЕБРА МНОЖЕСТВ И АЛГЕБРА ОТНОШЕНИЙ](#_1_АЛГЕБРА_МНОЖЕСТВ)  4

[1.1 Введение в дискретную математику](#_1.1_Введение_в) 4

[1.2 Основные операции алгебры множеств](#_1.2_Основные_операции) 5

[1.3 Основные законы алгебры множеств](#_1.3_Основные_законы) 161617

[2 КОМБИНАТОРИКА](#_2_КОМБИНАТОРИКА) 8

[2.1 Элементы комбинаторики](#_2.1_Элементы_комбинаторики) 8

[2.2 Операции комбинаторики](#_2.2_Операции_комбинаторики) 9

[2.3 Теорема о включениях и исключениях](#_2.3_Теорема_о) 12

[2.4 Понятия об отношениях](#_2.4_Понятия_об) 1616113

[2.5 Свойства бинарных отношений](#_2.5_Свойства_бинарных) 1616117

[3 АЛГЕБРАИЧЕСКИЕ СИСТЕМЫ](#_3_АЛГЕБРАИЧЕСКИЕ_СИСТЕМЫ) 20

[3.1 Элементы алгебры Буля](#_3.1_Элементы_алгебры) 20

[3.2 Морфизм систем](#_3.2_Морфизм_систем) 22

[3.3 Эквивалентность формул](#_3.3_Эквивалентность_формул) 24

[3.4 Принцип двойственности](#_4.3_АВЛ) 1616129

[3.5 Базисные функции](#_3.5_Базисные_функции) 1616130

[4 ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ](#_4_ЭЛЕМЕНТЫ_ТЕОРИИ) 35

[4.1 Класс матриц инциденций](#_4.1_Класс_матриц) 40

[4.2 Операции над графами](#_4.2_Операции_над) 44

[4.3 Понятие о решетках](#_4.3_Понятие_о) 45

[4.4 Связность графов](#_4.4_Связность_графов) 1616149

[4.5 Способы задания графов](#_4.5_Способы_задания) 1616150

[4.6 Теорема Эйлера](#_4.6_Теорема_Эйлера) 52

[4.7 Цикломатика](#_3.7_Цикломатика) 54

[5 ДЕРЕВЬЯ](#_5_ДЕРЕВЬЯ)  57

[5.1 Понятие о деревьях](#_5.1_Понятие_о) 57

[5.2 Понятие о сетях](#_5.2_Понятие_о) 58

[5.3 АВЛ](#_5.3_АВЛ) 1616159

[5.4 Деревья Кантаровича](#_5.4_Деревья_Канторовича) 60

[5.5 Деревья с помеченными (размеченными) листьями](#_5.5_Деревья_с) 61

[5.6 Изоморфизм алгебры Буля и алгебры Кантора](#_5.6_Изоморфизм_алгебры) 62

[5.7 Минимизация булевых функций в классе ДНФ](#_5.7_Минимизация_булевых) 65

[5.8 Полнота и замкнутость систем булевых функций](#_5.8_Полнота_и) 67

[5.9 Принцип двойственности](#_5.9_Принцип_двойственности) 69

[6 ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ](#_6_ЭЛЕМЕНТЫ_МАТЕМАТИЧЕСКОЙ) 6870

[6.1 Понятия об индукции и дедукции](#_6.1_Понятия_об) 72

[6.2 Тождественно-истинные и тождественно-ложные высказывания](#_6.2_Тождественно-истинные_и) 72

[6.3 Тождественно – ложные высказывания (противоречия)](#_6.3_Тождественно_–) 73

[6.4 Аксиоматика Клини](#_6.4_Аксиоматика_Клини) 73

[6.5 Примеры кванторов](#_6.5_Примеры_кванторов) 75

[6.6 Исчисление кванторов и предикатов](#_6.6_Исчисление_кванторов) 76

[6.7 Элементы Аристотелевой логики](#_6.7_Элементы_Аристотелевой) 78

[6.8 Тенденции развития теории познания](#_6.8_Тенденции_развития) 81

[6.9 Задача минимизации представления множеств в алгебре Кантора](#_6.9_Задача_минимизации) 85

[7 КРИПТОГРАФИЯ](#_7_КРИПТОГРАФИЯ) 85

[7.1 Алгоритмы факторизации](#_7.1_Алгоритмы_факторизации) 88

[7.2 Выработка секретного ключа по Диффи - Хеллману](#_7.2_Выработка_секретного) 89

[7.3 Датчики М-последовательностей](#_7.3_Датчики_М) 1616190

[7.4 Конгруэнтные датчики](#_7.4_Конгруэнтные_датчики) 90

[7.5 Элементы алгебры вычетов](#_7.5_Элементы_алгебры) 91

[7.6 Элементы криптографии](#_7.6_Элементы_криптографии) 93

[8 МАШИННАЯ МАТЕМАТИКА](#_8_МАШИННАЯ_МАТЕМАТИКА) 93

[8.1 Элементы теории алгоритмов](#_8.1_Элементы_теории) 1616195

[8.2 Машина Тьюринга](#_8.2_Машина_Тьюринга) 97

[8.3 Нормальные алгоритмы Маркова](#_8.3_Нормальные_алгоритмы) 99

[8.4 Схема Поста](#_8.4_Схема_Поста) 100

[СЛОВАРЬ](#_СЛОВАРЬ_1) 102

# **1 АЛГЕБРА МНОЖЕСТВ И АЛГЕБРА ОТНОШЕНИЙ**

# **1.1 Введение в дискретную математику**

**Дискретная математика** – это наука, изучающая дискретные множества. **Множество** можно определить, как совокупность различаемых между собой объектов, объединенных общим свойством.

Существуют различные способы задания множеств:

1. перечисление всех элементов, т.е. формируется список, или таблица:

Применяется тогда, когда количество элементов не велико. Количество элементов называется **мощностью множества**:

**А = |n|; |B|=3**

1. универсальный, при помощи характеризующего свойства, или характеристических свойств элементов. Здесь задается распознающая процедура, которая устанавливает принадлежность объекта к данному множеству.

**D = {x /x – студенты группы}**

1. при помощи порождающей процедуры, т.е. имеется способ получения элементов множества из ранее полученных элементов, или других объектов.

Сюда можно отнести ряд формул. Например:

Эта процедура производит синтез элементов множества, а распознающая – анализ объектов с целью ведения из них элементов данного множества. Если можно построить распознающую процедуру, то можно построить и порождающую, и наоборот.

Существуют предельные понятия для каждого множества:

1. Пустое множество **Ø** – не содержит ни одного элемента;
2. Универсум **U** – все элементы предметной области.

Существует теорема Кантора, которая говорит о том, что множество всех действительных чисел отрезка **[0;1]** не является счетным (конечным). Его мощность называется **континуум**.

В любом множестве можно выделить ряд подмножеств. Множество всех подмножеств, класс множеств называется **булеаном**, обозначается заглавными буквами готического шрифта. Здесь существуют специальные символы, определяющие отношения включения :

, **В** является подмножеством **А**;

, В равносильно, или равномощно **А**, т.е. оба множества совпадают **, А=В**.

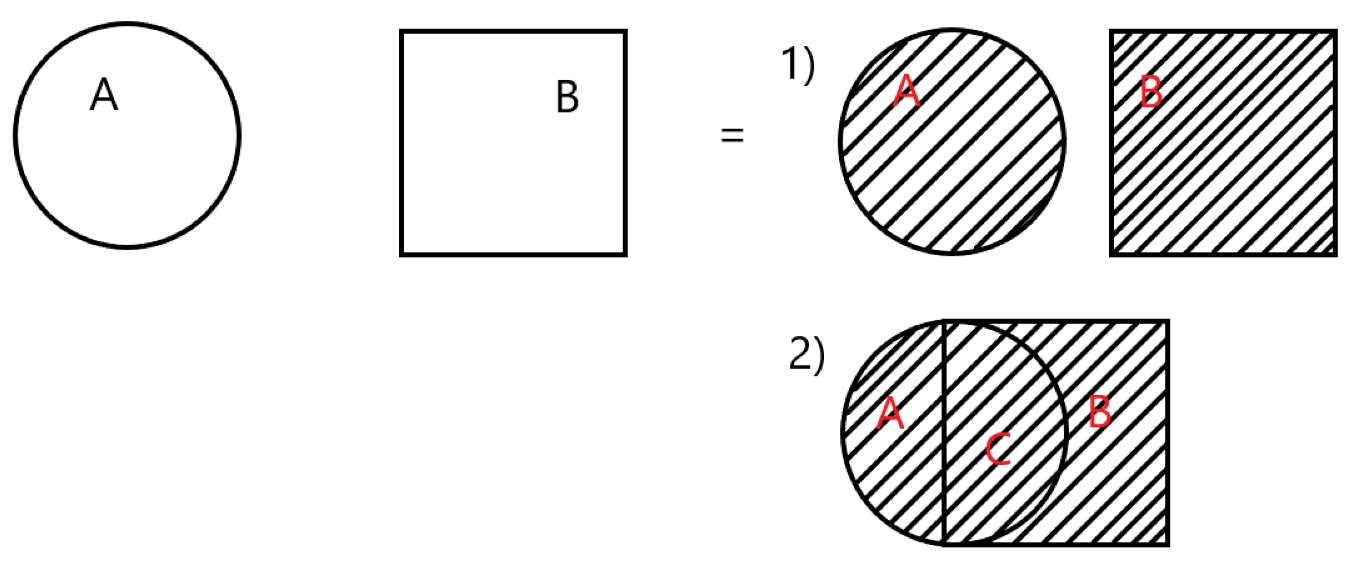
# **1.2 Основные операции алгебры множеств**

1. Объединение **(U)**

Формируется новое множество, содержащее все элементы, принадлежащие или одному, или другому или обоим сразу

Геометрическая интерпретация операций алгебры множеств представляет собой диаграммы Венна (1834 – 1923г.г). Он предложил каждое множество обозначать соответствующей геометрической фигурой.

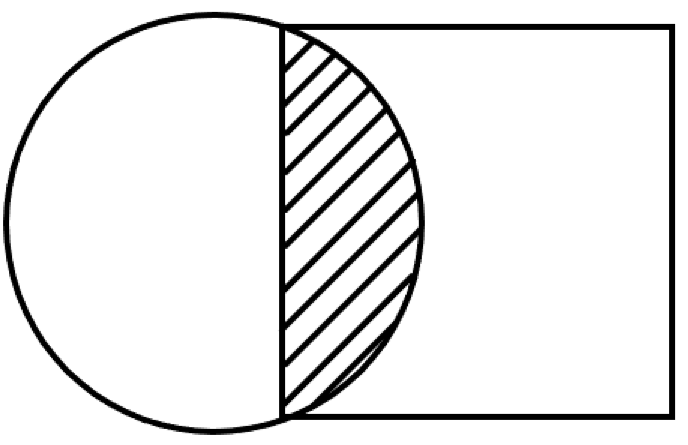
Пример:



1. Пересечения множеств (**∩**)

Формируется новое множество элементов, принадлежащих и первому и второму множеству:

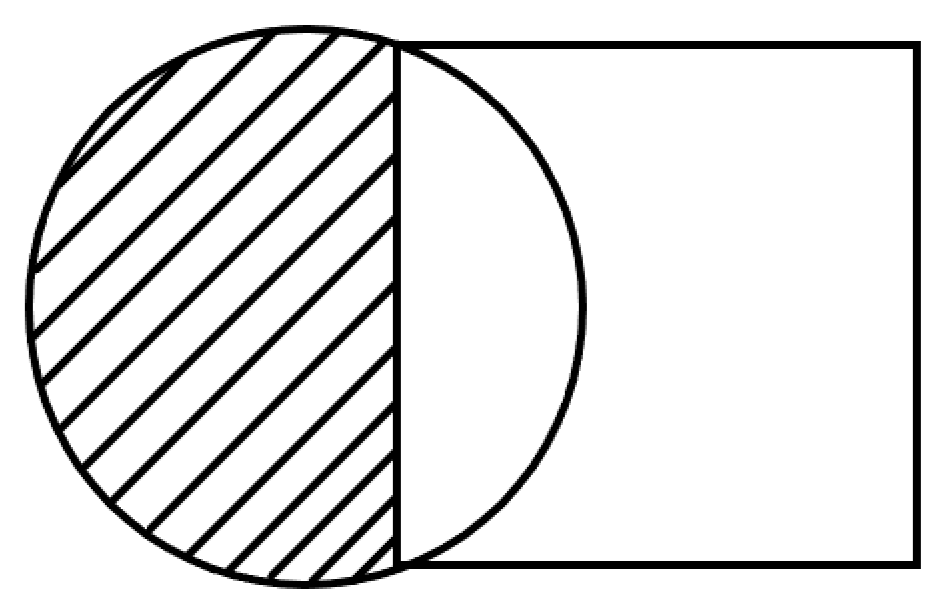
Пример:



1. Разность

Формируются множества таких элементов первого множества, которые не принадлежат второму множеству:

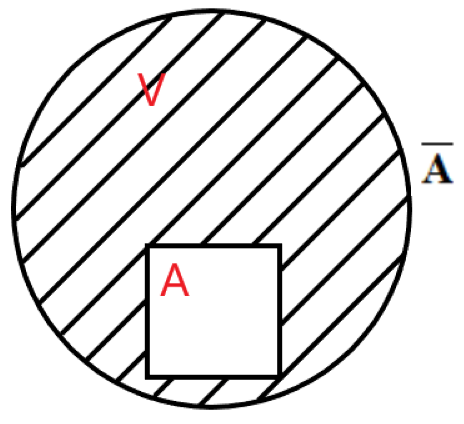
Пример:



1. Дополнение

Это разность между универсумом и данным множеством:

Пример:



1. Симметрическая разность, т.е. это новое множество полученное в результате объединения множеств **А** и **В** и множества, сформированного в результате пересечения этих же множеств.

# **1.3 Основные законы алгебры множеств**

1. **Коммутативный (переместительный)**
2. **Ассоциативный (сочетательный)**

Результат объединения двух множеств, а именно множества **А** и множества, полученного объединением множеств **В** и **С** эквивалентен результату, полученному объединением двух множеств: множества сформированного объединением **А** и **В** и множества **С**.

1. **Дистрибутивный (распределительный)**

Представленные примеры наглядно характеризуют принцип двойственности. Здесь символы , а также **U** и **Ø** являются двойственными друг к другу. Если в некоторое выражение или равенство алгебры множеств входят только операции и **\**, то заменив символы на двойственные можно получить выражение или равенство двойственное первоначальному.

1. **Закон идемпотентности**
2. **Закон поглощения**
3. **Закон дополнения, или законы де Моргана**
4. **Законы тождества, или пустого и универсального множества**

# **2 КОМБИНАТОРИКА**

# **2.1 Элементы комбинаторики**

**Комбинаторика** (от лат. соединять, сочетать) – это раздел дискретной математики, изучающий приемы вычисления количества различных соединений, или комбинаций. Т.е. здесь изучаются способы подсчета количества элементов различных конечных множеств.

Многие задачи решаются при помощи двух основных правил: правила суммы и произведения:

1. **Правило суммы**

Если некоторый объект А можно выбрать m способами, а другой объект В n способами, то выбор либо A, либо В можно осуществить **m + n** способами.

Пусть - это множество способов выбора объекта А, тогда МВ - множество способов выбора объекта В. Пусть - ни один из способов выбора объекта А не совпадает ни с одним из способов выбора объекта В. Иначе, имеется k общих способов и правило суммы принимает вид:

1. **Правило произведения**

Пусть объект А выбирается m способами, а объект В n способами, причем n не зависит от того, как был выбран объект А. Тогда число способов выбора пары элементов будет равно **m × n**.

Комбинация из k элементов называется **k – расстановкой**. Тогда общая задача принимает вид: сколько можно получить k – расстановок, если первый объект выбирается n1 способами, второй – способами, …, k-ый – nk способами. При этом две расстановки считаются различными, если хотя бы на одном месте в них находятся разные объекты:

Пример: сколько можно составить знаков из четырех геометрических фигур. Внутри знаков помещаются латинские буквы и цифры:

– буквы (26 штук), n2 – цифры (10 штук), n3 – фигуры (4 штуки),

# **2.2 Операции комбинаторики**

Существует три основных комбинации комбинаторики:

1. Размещение;
2. Перестановка;
3. Сочетание.

Каждая операция может быть с повторениями объектов или без повторений.

**Размещение с повторениями**

Дано n предметов разного вида, из которых нужно составить k-расстановки. При этом две комбинации считаются различными, если:

* Различен **порядок** предметов расстановки;
* Различен **состав** этих комбинаций;
* **Повторение** предметов комбинации **разрешено**.

Пример: , ,

Отсюда формируется теорема:

Количество k-расстановок из n элементов при размещении с повторениями равно:

**Размещение без повторения**

* Важен порядок;
* Важен состав;
* Повторения запрещены.

Пример:

Здесь общую формулу можно вывести, рассуждая так: Первый предмет в расстановке можно выбрать n способами и независимо от него выбирается второй предмет n – 1 способами и так далее. Для k предмета остается

:

**Перестановка без повторений**

* Состав не меняется;
* Важен порядок;
* Повторения запрещены.

Эту операцию можно считать предельным случаем размещения без повторения, когда количество предметов в расстановке равно количеству расстановок . Тогда формируется теорема:

Общее количество перестановок без повторений из n предметов равно:

**Перестановка с повторениями**

Всего n предметов, причем первый повторяется n1 раз, второй – n2 раз, …, k-ый – раз:

Формулу можно вывести следующим образом: на каждой перестановке из–за повторений предмета теряется расстановок. Т.к. перестановки разных групп можно осуществлять независимо друг от друга, то общее количество утерянных перестановок определяется по правилу произведения:

Если повторений не было, то было бы n! расстановок. Их количество корректируется в соответствии с ранее полученной формулой:

**Сочетание без повторений**

* Порядок не важен;
* Важен состав;
* Повторения запрещены.

Существует теорема о том, что число k сочетаний из n предметов равно:

**Сочетание с повторениями**

* Порядок не важен;
* Важен состав;
* Повторения разрешены.

Формула имеет вид:

Доказательство можно построить следующим образом: пусть предмет первого типа повторяется 2 раза, второго – 3 раза, третьего – 2 раза, …, n-го – 1 раз. Можно зашифровать комбинацию при помощи двух символов (0 и 1): 1 – соответствует объекту, 0 – разделяет группы предметов разного типа.

11 0 111 0 11 0 … 0 1

Тогда можно рассмотреть формулу перестановки из двух предметов с повторениями:

# **2.3 Теорема о включениях и исключениях**

Пусть имеется N предметов, некоторые из этих предметов обладают всеми свойствами, несколькими свойствами, одним свойством или не обладают ни одним свойством.

Пусть — это количество предметов, обладающих k свойствами. Если предмет не обладает каким – либо свойством, то свойство обозначается . Тогда формула включений и исключений примет вид:

Пример: В группе 13 студентов, каждый из которых занимается хотя бы одним видом спорта: 10 человек – легкой атлетикой, 7 – лыжами, 6 – плаванием, 5 – легкой атлетикой и лыжами, 4 – легкой атлетикой и плаванием, 3 – лыжами и плаванием. Сколько человек занимается тремя видами спорта, сколько двумя?

– студенты, занимающиеся легкой атлетикой;

- студенты, занимающиеся лыжами;

- студенты, занимающиеся плаванием;

– количество студентов, занимающихся легкой атлетикой;

– количество студентов, занимающихся лыжами;

– количество студентов, занимающихся плаванием;

– количество студентов, занимающихся легкой атлетикой и лыжами;

– количество студентов, занимающихся легкой атлетикой и плаванием;

– количество студентов, занимающихся лыжами и плаванием;

N =13;

- количество студентов, занимающихся тремя видами спорта

Ответ: 2 студента, занимаются тремя видами спорта; 6 студентов – тремя.

# **2.4 Понятия об отношениях**

Существуют разные виды отношений. В алгебре отношений рассматриваются отношения как подмножество декартового произведения. В любой алгебраической системе рассматриваются два множества: , где А – носитель системы, Σ – ее сигнатура. **Носитель** – это объект системы. **Сигнатура** – связи между объектами. В целом носитель включает в себя все объекты предметной области, или универсум. Сигнатура включает все операции над носителями.

В общем отношение понимается как свойства, присущие некоторому объекту. Если объект обладает одним свойством, то отношение называется **одноместным**, или **унарным**; **n=1**. Геометрически свойство интерпретируется как точка на оси координат.

Если объект имеет два свойства, то отношение является **бинарным**. Геометрически это точка (xi, yi), n=2.

Если формируется система трех показателей, то отношение является тернарным, или геометрически это координата точки в пространстве (x, y, z).

Рассмотренные примеры позволяют сформировать декартового произведение одинаковых множеств.

**R = R×R n = 2**

**R = R×R×R = R3 n = 3**

В алгебре отношений, кроме рассмотренных ранее операций над множествами, в сигнатуру включается декартово произведение. Алгебра отношений дополняется специальными операциями, в частности, операциями выбора, проекции, соединения.

Реляционная алгебра является основой современных **СУБД** – систем управления базами данных (с 1972г.). Основой таких СУБД являются таблицы, где количество строк неограниченно. Но количество и порядок столбцов строго определены. Строки называются **кортежами**, а столбцы – **доменами**. **P = D1×D2×…×Dn**, т.е. кортеж представляет собой декартово произведение n доменов. Носителем в реляционной алгебре является множество ее отношений:

**R = {P1, P2, …, Pn}**

Примером формирования отношений является таблица составления расписания экзаменов. В качестве доменов можно выбрать следующее:

D1 – группы;

D1 = {ПР – 226, ПР – 239к};

D2 – предметы;

D2 = {Математика, ОС, Программирование};

D3 – даты проведения экзаменов;

D3 = {8.01, 11.01, 14.01};

D4 – аудитории;

D4 = {364, 348}

Пример показывает, что каждый домен включает в себя множество объектов, характеризующих предметную область по какому – либо свойству. Тогда каждая строка будет соответствовать кортежу значений доменов, находящихся в отношении **R4**.

Можно в качестве примера предложить таблицу следующего вида:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **R4** | **D1** | **D2** | **D3** | **D4** |
| **1** | ПР-226 | Прогр. | 8.01 | 348 |
| **2** | ПР-239к | Матем. | 8.01 | 364 |
| **3** | ПР-239к | Прогр. | 11.01 | 348 |
| **4** | ПР-226 | ОС | 11.01 | 364 |
| **5** | ПР-226 | Матем. | 14.01 | 364 |
| **6** | ПР-239к | ОС | 14.01 | 348 |

В целом таблица может иметь любую арность, но количество строк ее определяется базовыми характеристиками.

1. **Операция выбора** – процедура построения «горизонтального» подмножества отношений, или подмножества кортежей, обладающих заданным свойством.

Например, нужно построить из отношения **R4** отношение, содержащее информацию расписания по ОС, т.е. осуществляется построение отношения. Формируется таблица **R'4** (арности 4), где строки представлены доменом **D2** со значением «ОС»:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **R4** | **D1** | **D2** | **D3** | **D4** |
| **1** | ПР-226 | ОС | 11.01 | 364 |
| **2** | ПР-239к | ОС | 14.01 | 348 |

1. **Операция проекции** – определяет построение «вертикального» подмножества отношений, т.е. из кортежей удаляются координаты, соответствующие не выделенным доменам.

Например, следует построить проекцию из отношения **R4**. Значит, получится таблица, содержащая 6 строк и 2 столбца:

|  |  |  |
| --- | --- | --- |
|  | **D2** | **D3** |
| **1** | Прогр. | 8.01 |
| **2** | Матем. | 8.01 |
| **3** | Прогр. | 11.01 |
| **4** | ОС | 11.01 |
| **5** | Матем. | 14.01 |
| **6** | ОС | 14.01 |

1. **Операция соединения** – предполагает работу с двумя таблицами, имеющими общий домен. В результате получается одна таблица, где каждая строка представляет собой соединение двух строк исходных таблиц с одним и тем же содержанием общего домена. В новой таблице общий домен занимает только один столбец. В примере должна получиться новая таблица **R7**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R7** | **D1** | **D2** | **D3** | **D4** | **D'2** | **D'3** | **D'4** |
| **1** | ПР-226 | Прогр. | 8.01 | 348 | Матем. | 14.01 | 364 |
| **2** | ПР-239к | Матем. | 8.01 | 364 | ОС | 14.01 | 348 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **R7** | **D1** | **D2** | **D3** | **D4** | **D'1** | **D'2** | **D'3** |
| **1** | ПР-226 | Прогр. | 8.01 | 348 | ПР-239к. | Прогр. | 11.01 |
| **2** | ПР-239к | Матем. | 8.01 | 364 | ПР-236 | ОС | 11.01 |

Рассмотренная операция предполагает соединение кортежей по условию равенства соответствующих координат. В общем случае соединения можно определить по любым правилам. Например, сортировка элементов соответствующих доменов.

**Вывод:** таким образом, любое отношение, определенное на множествах **D1, D2, …, Dn**, есть некоторое множество кортежей арности **n (Rn)**. При это домен **Di** может содержать любое количество объектов.

Декартово произведение множеств называется **прямым произведением**. В результате операции формируется множество элементов таких, что первый элемент принадлежит первому множеству, …, n – ый элемент – n- му.

Если все **,** то произведение называется **n - ой** степенью множества **А (An)**.

# **2.5 Свойства бинарных отношений**

Бинарные отношения играют особую роль в курсе дискретной математики, т.к. многие научные направления (например, теория графов) основаны на свойствах этих отношений.

Если формируется бинарное отношение вида **R = A2**, то такое отношение называется **полным**, или **универсальным**.

Отношение называется тождественным, или диагональным если:

**Транзитивное отношение** – это множество отношений, для которых справедливы высказывания: если и , то

Пример:

**Рефлексивность** – когда диагональ множества А полностью принадлежит отношению.

**Симметричность** – соответствуют утверждению: если , то .

**Эквивалентное отношение** – это одновременное выполнение свойств рефлексивности, транзитивности, симметричности.

Кроме того, бинарные отношения реализуются для простейших функций, или функций одного аргумента **y = f (x)**.

1. Область определения, т.е. δ (дельта)
2. Область значений
3. Обратное отношение
4. Образ множества (например, Х) относительно предиката P(R) – это:

для

1. Прообраз множества **Х** относительно предиката **P** является множество:, или образ множества **Х** относительно предиката **Р-1**.

Для функций стандартными являются три свойства:

1. **Инъекция**

Свойство, присущее функции, если при изменении аргумента функция меняет свое значение, т.е. при , .

1. **Сюрьекция**

Это свойство, когда полностью совпадает с множеством **А**.

1. **Биекция**

Когда функция одновременно и инъективна, и сюрьективна.

Отношения эквивалентности реализуются в различных системах. Например, минимальным отношением эквивалентности считается Е-равенство, где

Выделяются отношения **строгого** и **нестрогого** порядка.

Другие примеры отношений (стр.18)

1. Отношение равносильности — это любое тождество.

Или отношение эквивалентности

1. Отношение нестрогого порядка ≤ ≥ бывает:

2.1 Рефлексивно

* 1. Антисимметрично
  2. Транзитивно

3.1 Антирефлексивно;

3.2 Антисимметрично;

3.3 Транзитивно.

1. Отношение предшествования — это разновидность отношения порядка для анализа символьных объектов

Это отношение реализовано в программах сортировки символьных элементов по возрастанию (убыванию).

Отношение эквивалентности можно рассмотреть на примере алгебры множеств. Эквивалентность на множестве **А** можно обозначить как **Е**.

Отношение эквивалентности может быть отношением принадлежности к одной студенческой группе на множестве студентов. Другой пример на множестве **М**, где **М** множество программ, вычисляющих некоторые функции

**вычисление одной функции}**

Классом эквивалентности элемента **x** принадлежащего множеству **А**  называется множество значений **y**. Например, классом эквивалентности является множество студентов одной группы. Фактор–множество – множество студентов по отношению эквивалентности представляет собой множество студенческих групп.

К отношению упорядоченности относятся также отношение предшествования . Используется при изучении различных алфавитов. Любое слово можно представить в виде:

т.е. формируется алгоритм побуквенного сравнения, где **β** – эквивалентные элементы слов, **γ** и **ρ** – любые окончания. Отношения предшествования формируются при обработке алфавитов.

Пример: Лето

Легенда

β = «ле», a1 = «т», а2 = «г», γ = «о», ρ = «енда»:

Буква , а значит, слово α2 располагается раньше в словаре, чем слово **α1**.

# **3 АЛГЕБРАИЧЕСКИЕ СИСТЕМЫ**

# **3.1 Элементы алгебры Буля**

Джорж Буль (1815 – 1864г.г.) в середине девятнадцатого века предложил новую алгебру, где переменной является высказывание, или суждение. Оно принимает только одно из двух значений: **1 (true)**, если истинно

**0 (false)**, если ложно

В монографии «Законы мысли» Буль изложил основные законы новой алгебраической системы (булевой алгебры).

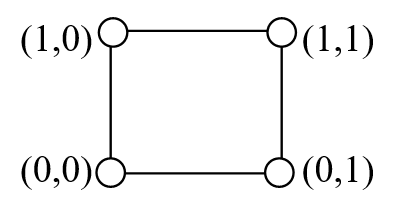
Функции, образы и прообразы которых являются элементами множества В, называются переключателями, или переключательными функциями.

Комбинации значений переменных являются аргументами таких функций и называются наборами*.* Множество всех наборов является областью определения функции . Количество таких наборов определяется как операция размещения с повторениями: , где **n** – арность функции.

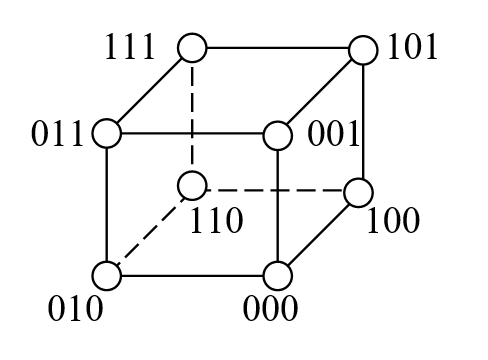
Если **n = 1**, то можно представить геометрически одномерный куб – отрезок



Если **n =2** - двумерный куб



Если **n = 3** - трехмерный куб



Две функции будут равносильными, если на одних и тех же наборах они имеют одни и те же значения. Алгоритм формирования множества функций можно проследить на примере функций двух переменных:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **х1** | 0 | 0 | 1 | 1 |
| **х2** | 0 | 1 | 0 | 1 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 1 |
|  | 0 | 0 | 1 | 0 |
|  | 0 | 0 | 1 | 1 |
|  | 0 | 1 | 0 | 0 |
|  | 0 | 1 | 0 | 1 |
|  | 0 | 1 | 1 | 0 |
|  | 0 | 1 | 1 | 1 |
|  | 1 | 0 | 0 | 0 |
|  | 1 | 0 | 0 | 1 |
|  | 1 | 0 | 1 | 0 |
|  | 1 | 0 | 1 | 1 |
|  | 1 | 1 | 0 | 0 |
|  | 1 | 1 | 0 | 1 |
|  | 1 | 1 | 1 | 0 |
|  | 1 | 1 | 1 | 1 |

- константа**Ø**,

– конъюнкция . В результате формируется высказывание, истинное, если оба высказывания истинны.

- запрет,

- повтор ,

- запрет ,

- повтор ,

- сложение по модулю два,

– дизъюнкция, или логическое сложение . В результате формируется высказывание, истинное, если хотя бы одно из высказываний истинно,

- стрелка Пиркса ,

- эквивалентность (~ тильда),

- отрицание ,

- обратная импликация,

- отрицание,

- прямая импликация, или логическое следование ,

- штрих Шеффера ,

- константа **1**.

Из таблицы следует, что некоторые значения аргумента не меняют значения функций. Пусть имеется два набора:

Они называются соседними, если тогда будет фиктивной переменной. Иначе - **х** - будет существенной.

# **3.2 Морфизм систем**

Две системы могут характеризоваться любой степенью морфизма. Рассмотренные раннее примеры показали, что могут быть предикаты различной местности **(P, R)**, а могут быть n-местные функции.

Алгебраическую систему формируют различные структуры:

* Множества с введенными на них бинарных отношений;
* Линейное пространство, связывающее геометрические объекты с операциями над числами;
* Поля обычной арифметики и т.д.

Можно выделить n-местную операцию на множестве **А** тогда **f** называется функцией, т.е. для любого набора аргументов результат применения операции однозначно определен. Т.к. – область значений операции определен множеством , то операция замкнута на множестве .

Сигнатура, или язык сигма – это совокупность предикатных и функциональных символов с указанием их местности *.*

Ноль местный функциональный символ – константный символ, или const. Пусть, **А** – функциональный или предикатный символ, его местность – **μ(а)**,**р(n)** *–* **n**-местный предикатный символ, f(n) – n-местный функциональный символ.

Алгебраическая система — это не пустое множество **A**, где каждому **n** – местному символу **(p(n)(f(n)))** из сигнатуры сигма ставится в соответствии **n** – местный предикат (операция), определенной на множестве **A**, где **А** – носитель, или универсум системы.

Предикаты и функции называются их интерпретациями соответствующих символов из сигма. Интерпретацией константного символа является константа **(const)** из множества **A**.

Мощность алгебраической системы **(U)** определяется мощностью носителя **А**.

Сигнатура сигма будет функциональной, если она не содержит предикатных символов – это алгебра. Иначе сигма предикатная, и система **U** называется моделью.

Пример:

Такая система называется алгеброй Кантора.

Существуют различные виды морфизмов, основных два:

1. **Изоморфизм** – строгое соответствие между системами;
2. **Гомоморфизм** – в системах совпадают некоторые существенные свойства.

Другие виды морфизмов: мономорфизм, эндоморфизм, автоморфизм.

Алгебра **U** сигнатуры **,** где функция двух переменных называется группоидом. Единственная операция сигнатуры обозначается как **◦**, тогда . Если эта операция ассоциативная, то группоид называется полугруппоидом:

Полугруппа **U** называется моноидом, если существует элемент **е**, называемый единицей, такой что . Полугруппоиды и моноиды используются в теории языков при обработке слов.

Пример: пусть дано множество слов алфавита **х W(x)**. Существует операция конкатенации **ˆ**, где результатом является слово, полученное соединением двух слов.

Эта операция является ассоциативной, поэтому система является полугруппой. В качестве **е** можно взять пустое слово (как пробел) , тогда **U**– моноид.

Моноид называется группоидом, если для любого существует элемент , или обратный к х такой что .

Группа **U** называется коммутативной, или абелевой, если для всех .

# **3.3 Эквивалентность формул**

Существует ряд тождеств, или эквивалентностей, которые характеризуют свойства элементарных функций. Эти свойства присущи конъюнкции, дизъюнкции и сложению по модулю два:

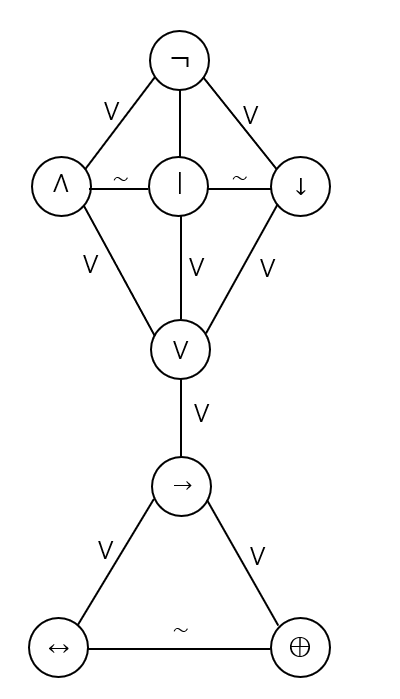
1. Коммутативности
2. Ассоциативности
3. Для **&***,*  справедливы дистрибутивные законы:
4. Соотношение между отрицанием,
5. Свойство

Если рассмотреть свойства 5, то можно сделать выводы:

1. Если в логическом произведении один из множителей равен нулю, то логическое произведение равно нулю.
2. Если в логическом произведении, содержащем не менее двух множителей, есть множитель, равный единице, то этот множитель можно удалить.
3. Если в логической сумме, содержащей не менее двух слагаемых, имеется слагаемое, равное нулю, то это слагаемое можно удалить.
4. Если в логической сумме одно из слагаемых равно единице, то и логическая сумма равна единице.

Система булевых функций будет функционально полной, если любая булева функция может быть записана в виде формулы через функции этой системы, то есть функция представима как суперпозиция указанных функций.

На множестве функций порядок их действий определяется диаграммой, в вершинах которой – функций, а на дугах – отношение их равносильности или преобладания.



**Разложение Шеннона**

Шеннон доказал теорему о том, что любая булева функция может быть представлена в виде:

где - либо 0 либо 1

Предельное разложение Шеннона (при **k=n**) имеет вид:

т.е. это разложение формируется для значений функции, равной единице, и называется совершенной дизъюнктивной нормальной формой (СДНФ).

Двойственное предельное разложение Шеннона называется совершенной конъюнктивной нормальной формой (СКНФ) и формируется по нулевым значениям булевой функции:

Эти формулы можно получить из таблицы истинности, используя следующие алгоритмы:

1. Для СДНФ

Нужно отметить в таблице строки, где функция равна единице.

* 1. Для каждой такой строки образовать логическое произведение

Нужно объединить полученные конъюнкции (конъюнкты) знаками дизъюнкции.

1. Для СКНФ

Нужно отметить в таблице строки, где функция равна нулю.

Сформировать дизъюнкции .

Объединить полученные дизъюнкции (дизъюнкты) знаками конъюнкции.

СДНФ аналогична - сумма произведений и называется дизъюнкцией конъюнктов.

СКНФ аналогична - конъюнкция дизъюнктов.

Пример: представить булеву функцию вида

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № |  |  |  | *f* |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |

СДНФ

СКНФ

**Базис** *–* это полная система, не содержащая лишних функций, т.к. удаление любой функции из базиса нарушает полноту системы.

Проблему полноты исследовал английский математик Пост. В СССР - Яблонский.

Выделяются сильная и слабая теоремы Поста.

1. **Сильная теорема Поста**

Для того, чтобы система была полной, необходимо и достаточно, чтобы для каждого класса функций Поста в системе нашлась хотя бы одна функция, не принадлежащая ему.

1. **Слабая теорема Поста**

Если система функций содержит тождественные 0 и 1, то для ее полноты достаточно, чтобы она содержала хотя бы одну монотонную и хотя бы одну не линейную функцию.

Теоремы основаны на анализе классов функции Поста. Всего пять классов функций:

1. **Класс функций, сохраняющий** **const 0**

**k0** – это множество функций вида:

**f(0,0,…,0)=0**

1. **Класс функций, сохраняющий** **const 1**

**k1** – это множество функций вида:

**f(1,1,…,1)=1**

1. **Класс самодвойственных булевых функций**

**k**- это множество функций вида:

то есть на любой паре противоположных наборов (наборов, сумма десятичных эквивалентов, которая равна **2n-1**) принимает противоположные значения.

1. **Класс монотонных функций**

Возрастание (убывание) значения набора приводит к не убыванию (не возрастанию) функций.

1. **Класс линейных функций**

Линейность или не линейность функции можно определить, используя теорему о том, что любая функция из может быть выражена при помощи полинома по модулю 2, или полинома И. И. Жегалкина.

Линейная функция в виде полинома Жегалкина.

В общем виде полином имеет вид:

Он включает в себя все возможные комбинации конъюнкции аргументов, которые называются взаимодействием элементов.

Пример: выразить дизъюнкцию в виде полинома Жегалкина:

|  |  |  |
| --- | --- | --- |
| x1 | x2 | f(x) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Вывод:** функция не линейная, т.к. в ее разложении присутствует взаимодействие аргументов.

# **3.4 Принцип двойственности**

Принцип двойственности применяется для получения тождеств.

Двойственной функцией к функции называется функция . Двойственную функцию можно получить из таблицы путем инвертирования столбца (заменой 0 на1, 1 на 0) и переворачиванием (первая строка становиться последней, вторая - предпоследней и т.д.). Из определения двойственности следует, что. Для получения двойственных формул нужно в исходной формуле заменить **0 на 1, 1 на 0, на &, & на .**

Существует теорема о том, что любая функция может иметь двойственную функцию. Из которой следует принцип двойственности:

Если формула реализует функцию , то формула , полученная из **U** заменой функции на двойственную будет реализовать двойственную функцию . Полученная формула называется двойственной к

Пример: cоставить двойственную формулу

# **3.5 Базисные функции**

Как доказал Пост в своих теоремах, в базис должна входить хотя бы одна функция, не принадлежащая классам функций Поста.

Существуют разные виды базисов, которые содержат то или иное количество функций. Существует теорема о том, что каждый базис содержит не более четырех булевых функций. Можно рассмотреть таблицу, где строки соответствуют 11 функциям двух переменных (кроме ). В столбцах указывается идентификатор строки, код функции (номер), классы функций Поста:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Идентификатор**  **строки** | **Функция**  **fi** | **Классы функций** | | | | |
| **К0** | **К1** | **Кл** | **Кс** | **Км** |
| a | 0 |  | 1 |  | 1 |  |
| b | 1 |  |  | 1 | 1 |  |
| c | 2 |  | 1 | 1 | 1 | 1 |
| d | 6 |  | 1 |  | 1 | 1 |
| e | 7 |  |  | 1 | 1 |  |
| g | 8 | 1 | 1 | 1 | 1 | 1 |
| k | 9 | 1 |  |  | 1 | 1 |
| m | 12 | 1 | 1 |  |  | 1 |
| n | 13 | 1 |  | 1 | 1 | 1 |
| p | 14 | 1 | 1 | 1 | 1 | 1 |
| r | 15 | 1 |  |  | 1 |  |

Элементы таблицы равны **1**, если **i-я** функция (на пересечение **i-й** строки,**i-го** столбца) не принадлежит **j-му** классу.

Существует метод Патрика, который позволяет преобразовать мультипликативно – аддитивную формулу в аддитивно – мультипликативную путем порождения всех покрытий столбцов строками таблицы:

Получено 17 покрытий (), каждый из которых порождает базис:

Рассмотренный метод Патрика реализуется в алгебре Кантора. Этот метод позволяет получить минимальную форму, а значит, минимизировать булеву функцию. Говорят, о минимизации булевой функции при помощи метода импликантных таблиц, или таблиц Квайна.

**Схемы Булевых функций**

Мультиграф представляет собой неориентированный граф, который является сетью.

Сеть отличается от графа тем, что в ней выделяют подмножество полюсов при чем один полюс, является входным, другой – выходным.

Формируется (1,1) – полюсник.

Мультиграф вида:

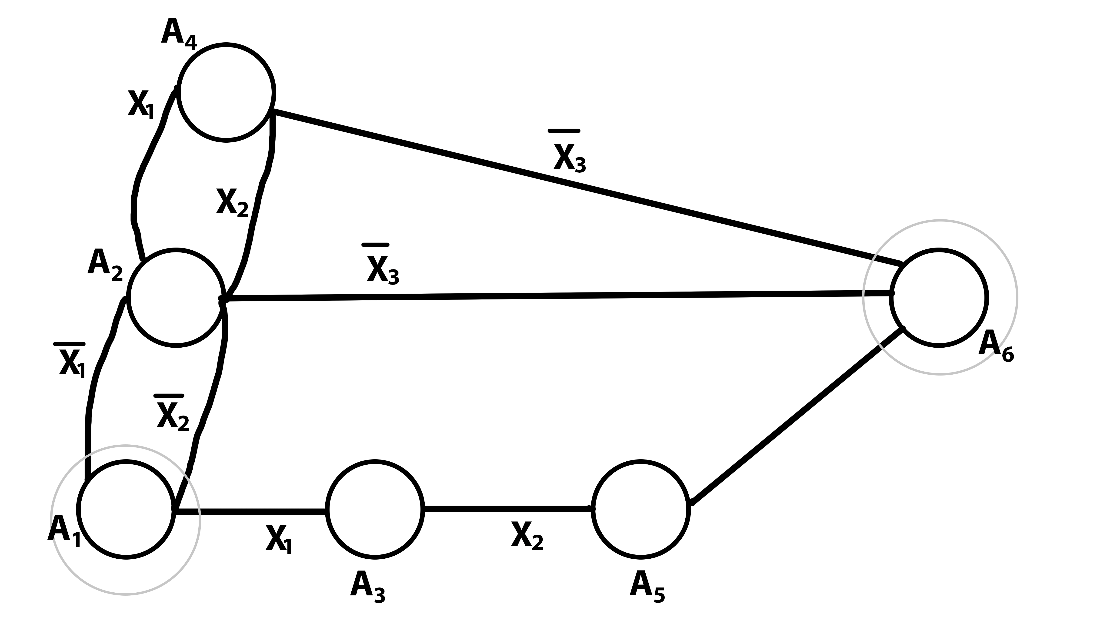
G = {M, U, K}, где М – множество вершин; U – множество дуг; K – множество полюсов.

Такой мультиграф называется K – полюсной сетью.

Сеть G, где каждое ребро помечено буквой из алфавита x = {} называется K – полюсной контанктной схемой.

Пример:

Дана двухполюсная контактная схема, где а – входной полюс, .



Бывает так, что один полюс выделен и является входным, а остальные выходные – равноправны, тогда (k+1) – полюсная схема называется (1, к)-полюсными.

Ребра контактной схемы называются контактными.

Контакт, соответствующий литере называется замыкающим

Без имени-1

Пропускает так, если

Контакт, соответствующий литере называется размыкающимся

Без имени-1222

Пропускает ток, при

Таким образом реализуется 2 состояния переключения.

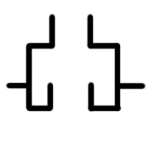
1. - Ток проходит; 0 – ток не проходит.

Здесь реализуется 2 функции:

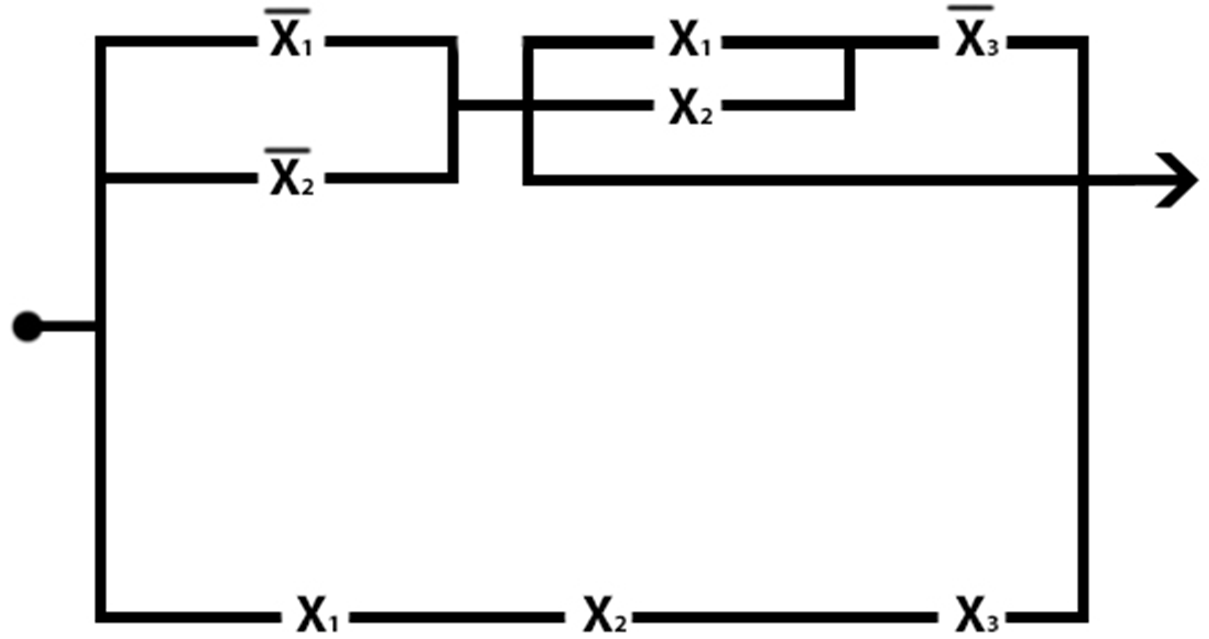
1 - Функция, которая соответствует последовательному соединению контактов.

23

2 соответствует параллельному соединению контактов



Представим электрическую схему



В этой схеме можно обозначить 2 полюса как a и b. Тогда – это некоторая цель из а и в b. Тогда между полюсами а, b – конъюнкция литер, приписанных ребрам цепи а, b.

Тогда булеву функцию можно представить формулой , где дизъюнкция по всем простым интервалам схемы соединяющими полюсы а и b.

Это функция производимости между полюсами а и b схемы сигма.

(1,1) – полюсники называются эквивалентными если они реализуют одну и ту же булеву функцию.

Сложность (1,1) – полюсника определяется числами контактов.

(1,1) полюсник имеющий минимальность нулю сложность среди эквивалентных ему схем называется минимальным.

Сложность минимального (1,1) полюсника реализующего функции называется сложностью функции b класса (1,1) полюсника и обозначается (ʄ) = 9.

Восстановим функцию:

() = ()(())

Эта функция не обладает минимальной сложностью.

Можно получить совершенную СДНФ в виде

Если реализовать алгоритм минимизации булевой функции, то можно получить две эквивалентные формулы

Рассмотренный пример показывает, что функция 3x переменных представлена различными формулами, которым в свою очередь соответствуют различные схемы.

Минимальная сложность этой функции равна 6.

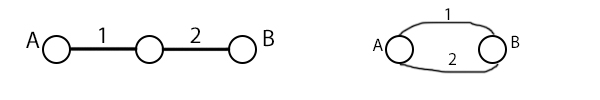
Символом П обозначают, что рассматривается функция в классе П – сетей.

П – сеть – это двухполюсная сеть, где входной полюс в, выходной b.

В таких сетях реализуются только операции последовательного и параллельного соединения.

1

В общем случае выполняется суперпозиция сетей, то есть в сети, выделяемой простейшие структуры.



Такие структуры называются двухполюсными сетями на двух объектных ребрах.

Это самые простые неразложимые сети.

**Схемы из функциональных элементов**

Это ориентированная бесконечная сеть, в которой выделяют входные и выходные полюсы.

Входные полюсы помечаются символами переменных – некоторые функциональным символом.

Здесь формируется понятие полустепень захода – это число двух входных в вершину.

Для входного полюса полустепень захода равна 0.

Для остальных вершин конечных n – местным функциональным символом = n, другие нумеруются от 1 до n.

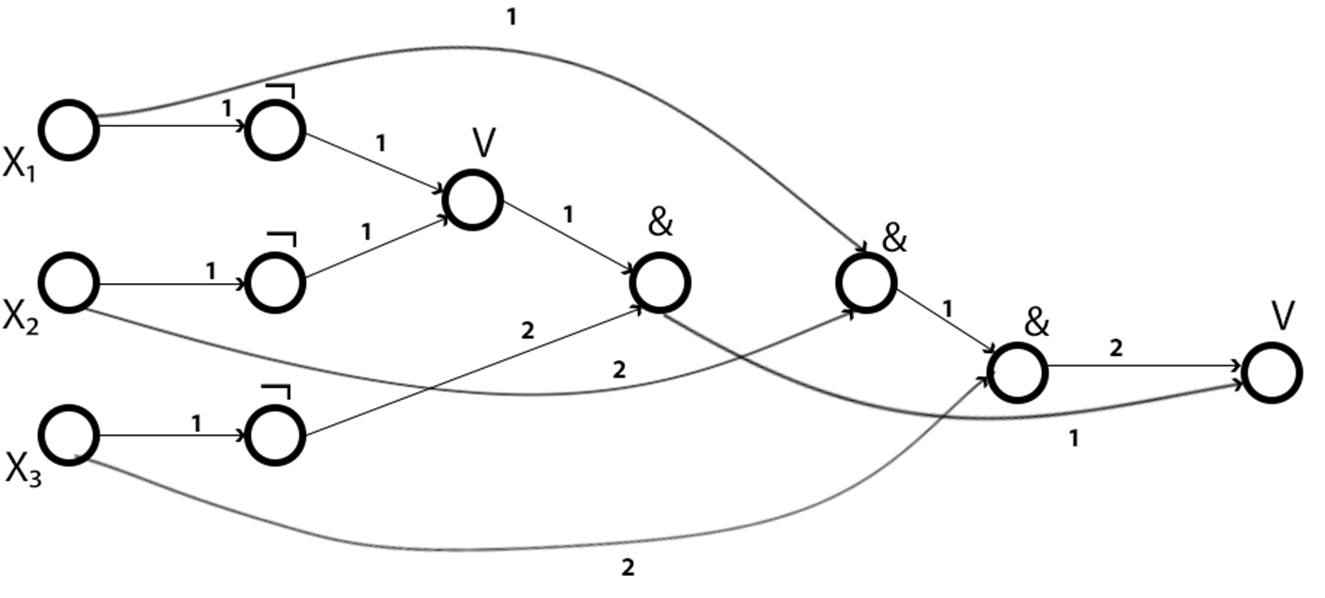
Функциональным элементом называется всякий подмультиграф схемы состоящий из невходного полюса, помеченного функциональным символом и вершин из которых исходят дуги в полюс.

- одноместный функциональный символ, соответствующий отрицанию.

**&** - двухместный функциональный символ, соответствующий операции конъюнкции.

**V** – двухместный символ, соответствующий операции дизъюнкции.

Составим схему функций:



Функция реализуется схемой сигма Z, если существует такой выход схемы сигма, эта функция соответствует терму выхода эквивалентна функции .

Здесь **Терма** – это любое функциональное выражение, составленное при помощи сигнатурных функциональных сигналов.

Схема из функциональных элементов с одним выходом, где входные полюсы помеченные символами , а остальные вершины символами , &,V, называются X – функциональными схемами.

Сложность схемы определяется числом ее вершин отличных от входных полюсов.

Сложность минимальной схемы, реализующей функцию называется сложностью функции в классе схем из функциональных элементов.

Пример:

Составить функциональную схему, состоящую из

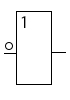
,,,))

Таким образом булева функция может быть представлена в виде любой формулы, в виде таблиц истинности, в видео СДНФ, СКНФ, полинома Жегалкина, в виде мультиграфа, электрической схемы и функциональной схемы.

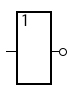
Существуют стандарты формирования схем, соответствующих булевым функциям.

Графически элементы обозначаются в виде прямоугольников, где инверсия входа и выхода обозначаются как пустые кружки.

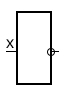
1. **Константа 0**



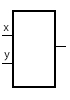
1. **Константа 1**



1. **Не х,**



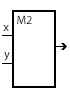
1. **X v Y**

****

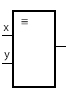
1. **X & Y**



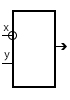
1. **Сложение по модулю 2**



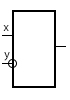
1. **Эквивалентность X Y**



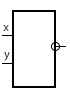
1. **Импликация**



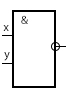
1. **Компенсация X &**



1. **Элемент Вебба**

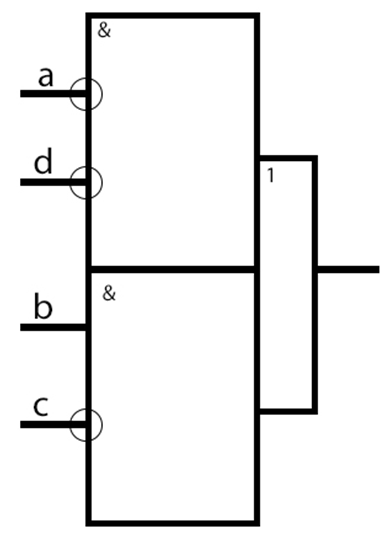


1. **Элемент Шиффера**



Пример:

Представить в виде схемы булеву функцию вида

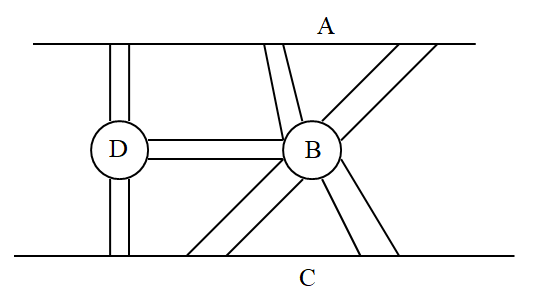


Как было замечено, техническая реализация базисных функций основывается на использовании различных физических явлений. Например: импликация и ко – импликация на магнитных явлениях; функции Вебба и Шеффера – на использовании явлений в полупроводниках.

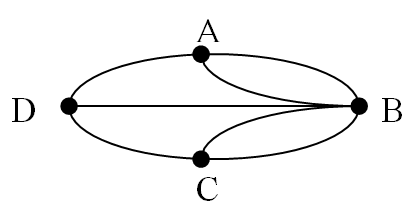
# **4 ЭЛЕМЕНТЫ ТЕОРИИ ГРАФОВ**

Одним из авторов, который считается родоначальником теории графов, был **Л. Эйлер (1707-1783)**. В 1736 году им была сформулирована задача о Кенигсбергских мостах. Постановка задачи:

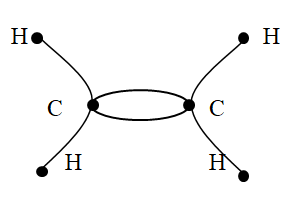
Через город течёт река Прегель, посередине реки имеется два острова. Мосты соединяют острова с берегами и друг с другом.



Эйлер задался вопросом: «Можно ли найти непрерывный маршрут между берегами и островами, который проходил бы по каждому мосту только один раз?» Эйлер предложил символьную модель, или граф, где берега и острова обозначаются точками, или вершинами. А мосты – линиями, или дугами, соединяющими пару вершин. Для данной задачи граф может иметь вид:



Графы широко распространены в органической химии для представления изомеров химических соединений. Например, Этилен С2Н4:



Любой граф можно представить как совокупность двух множеств. Например, , где M – носитель графа, или множество вершин, а T – сигнатура графа, или множество дуг. Как правило, дуга – это ребро, имеющее стрелку, или ориентированное ребро. Например, формирование генеалогического дерева, где дуга показывает отношение родственности, или отношение происхождения, которое всегда является односторонним.

**Граф Неймана**

Любой граф можно представить, как совокупность двух множеств. Например, ,

где

M – носитель графа, или множество вершин,

T – сигнатура графа, или множество дуг,

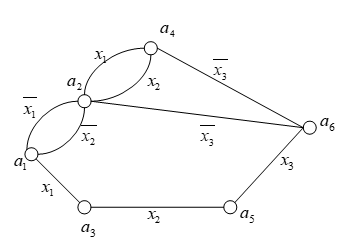
К – множество полюсов

Такой граф называется К- полюсной сетью.

Сеть G, где каждое ребро помечено буквой из алфавита

Как правило, дуга – это ребро, имеющее стрелку, или ориентированное ребро. Например, формирование генеалогического дерева, где дуга показывает отношение родственности, или отношение происхождения, которое всегда является односторонним.

Дана двухполюсная контактная схема, где -входной полюс, -выходной.

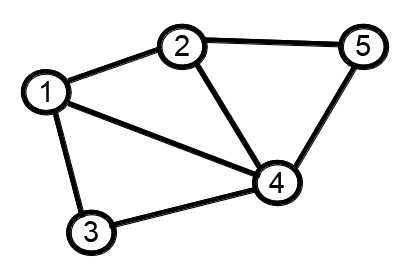


Бывает так, что один полюс выделен и является входным, а остальные- выходные – равноправны, тогда (к+1) – полюс схема называется (1, к) полюсным.

**Матрица смежности**

**Матрица смежности** — это квадратная матрица, в которой каждый элемент принимает одно из двух значений: **0** или **1**.

Рассмотрим следующий граф:



При представлении графа **матрицей смежности** информация о ребрах графа хранится в квадратной матрице (двумерном списке), где элемент равен **1**, если ребра **i** и **j** соединены ребром и равен 0 в противном случае. Для данного примера матрица смежности будет выглядеть так:

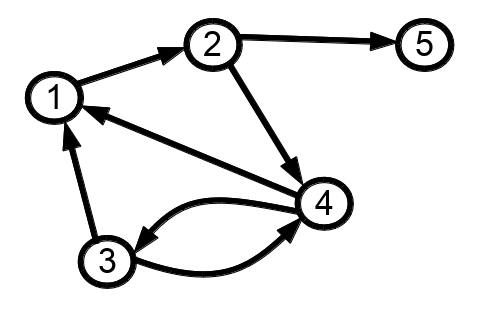
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 2 | 1 | 0 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 1 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |

Если граф неориентированный, то матрица смежности всегда симметрична относительно главной диагонали.

При использовании матрицы смежности удобно проверять соединены ли две вершины ребром – это просмотр одного элемента матрицы , но сложнее перебирать все ребра, исходящие из данной вершины (для этого необходимо перебрать все оставшиеся вершины и проверить, соединены ли они ребром). Также матрица смежности требует **O(n2)** памяти и может оказаться неэффективным способом хранения дерева или разреженных графов.

При помощи матриц смежности можно представлять и неориентированные графы. В случае матрицы смежности будет равно **1**, если есть ребро, начинающееся в вершине **i** и заканчивающееся в вершине **j**.

Например, для такого графа:



Матрица смежностей будет следующей:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 |

Очень часто рассматриваются графы, в которых каждому ребру приписана некоторая числовая характеристика – **вес**. Вес может означать длину дороги или стоимость проезда по данному маршруту. Соответствующие графы называются взвешенными.

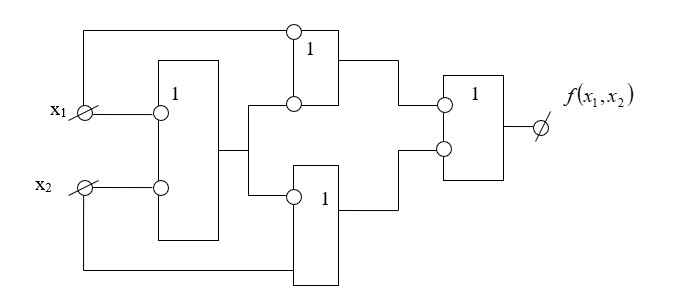
При представлении графа матрицей смежности вес ребра можно хранить в матрице, то есть в данном случае будет равно весу ребра из **i** в **j**.

# **4.1 Класс матриц инциденций**

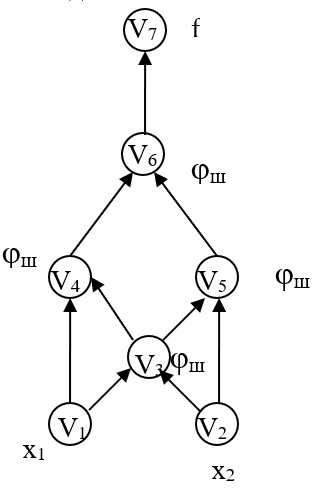
Здесь возникает проблема при задании графа, содержащего петли, т.к. одновременно элементы матрицы должны быть равны **±1**. В этом случае из матрицы инциденции выделяются две новые матрицы. Условно их называют  *и* :

Если граф не содержит петель, то:

Пример: логическая схема, реализующая сложение по модулю два в базисе Шеффера.



Этой схеме соответствует взвешенный граф вершины которого взвешены переменными **х1** и **х2**, функциональной переменой **φш** и функцией переменной **f***.* Граф будет иметь вид:



Вектор – столбец весов:

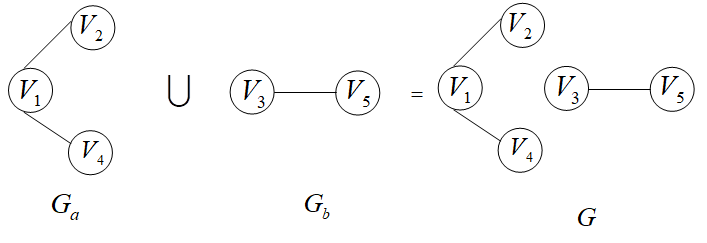
Матрица инциденции:



# **4.2 Операции над графами**

Существует ряд стандартных операций над графами, которые проводятся с различными целями. Например, можно представить графы, в виде композиции более простых графов выполнив следующие операции:

1. **Объединение графов**

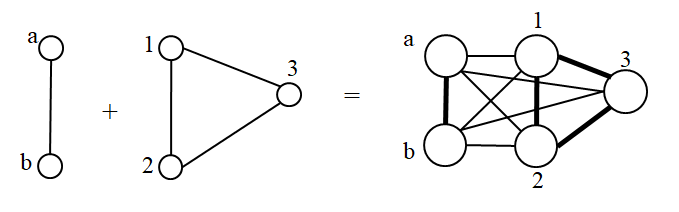
Формируется новый граф носитель и сигнатура которого являются теоретико-множественным объединением носителей и и сигнатур и исходных графов.

Новый граф является несвязным и имеет две компоненты связности.

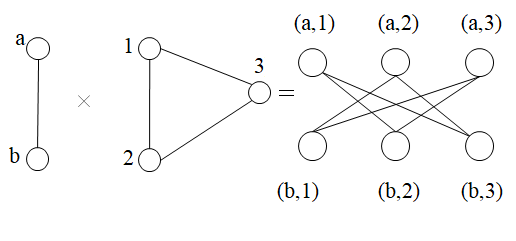
1. **Сумма графов**

Представляет собой объединение исходных графов и и двудольного полного графа , который построен на носителях, а именно каждая вершина, не вошедшая в пересечение, соединяется со всеми вершинами и наоборот, т.е. вершина **V** конусирует граф **j**, если она смежная со всеми вершинами графа **j**.

**Двудольный граф** – это граф, множество вершин которого можно разбить на два непересекающихся подмножества так, что каждое ребро графа соединяет две вершины из разных подмножеств.

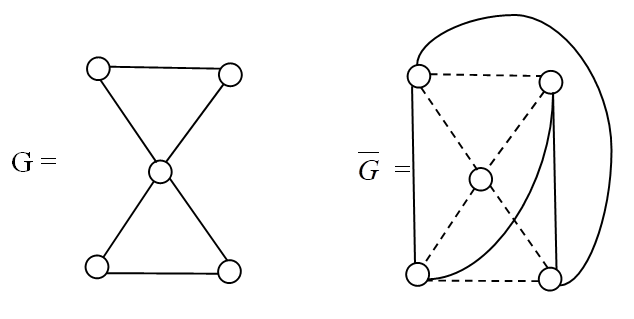


1. **Произведение (декартово) графов – это новый граф**:

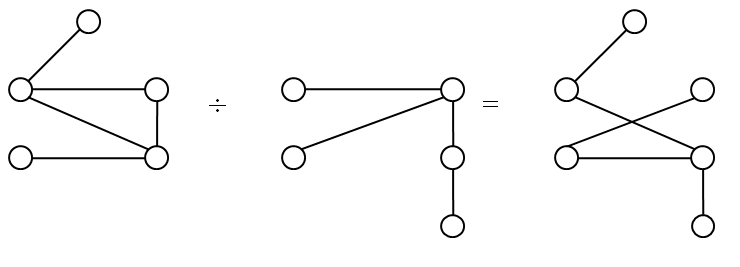


1. **Операция дополнения**

Граф дополняется до полного, где все вершины связаны между собой:

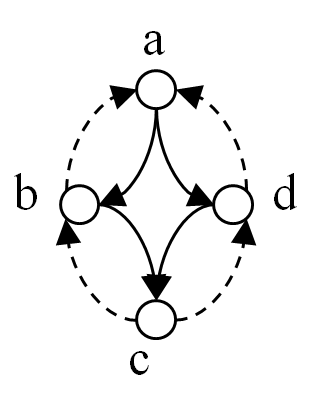


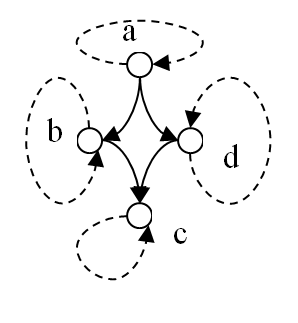
1. **Симметрическая разность**

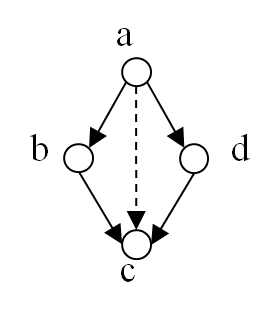
Формируется новый граф путем применения сложение по модулю два.

Кроме того, в графах применяются операции подразбиения ребра, т.е. удаляется одно ребро, добавляется вершина и два ребра. Подразбиение применяется для анализа морфизмов графов. Другими операциями являются удаление вершин (влечет удаление всех инцидентных ребер) и удаление ребер (только удаляются ребра без изменения количества вершин). Кроме того, часто необходимо добавлять ребра. Например, для оценки близости бинарного отношения к тому или иному свойству: , где - симметричность, - рефлексивность, - транзитивность.

Пример:



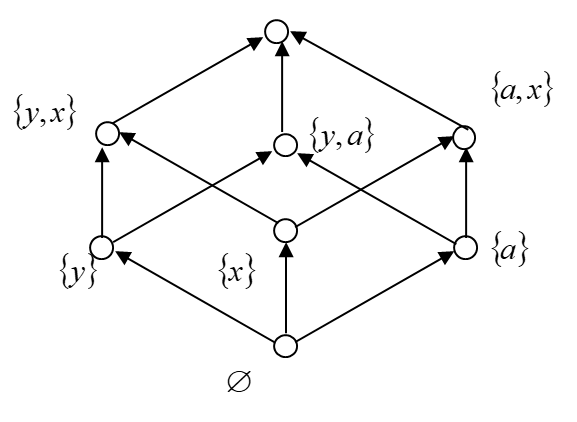




# **4.3 Понятие о решетках**

Графическая интерпретация графов обычно предполагает некоторую упорядоченность, которая ярче всего проявляется в графах и называется **решеткой** – это упорядоченное множество любые два элемента которого **,**  имеют наибольшую нижнюю грань, или пересечение и наименьшую верхнюю грань, или объединение. Множеству может быть частично упорядоченным, если в нем задается отношение упорядоченности. Если отношение упорядоченности или справедливы для любых двух элементов упорядоченного множества, то множество будет **линейно – упорядоченным***.*

Примером частично упорядоченного множества может быть граф вида:



Это стандартный граф . Если из этого графа удалить все петли и транзитивно – замыкающие дуги, то получится диаграмма Хассе. Ее применяли с конца 19 века в генеалогии для задания родства. Понятие непосредственного задается так: **mi** покрывает **mj**, т.е. и не найдется такого элемента **mx**, чтобы . Если найдется элемент , такой что для любого **mi** справедливо отношение , то называется **мажорантой** множества **М**. Если существует для любых , то отношение называется **минорантой**.

Мажоранта является максимальным элементом подмножества, а миноранта минимальным. Если множество мажорант имеет максимальный элемент, то он называется **верхней гранью множества** (**sup M**). Если множество минорант имеет минимальный элемент, то он называется **нижней гранью множества** (**inf М**). Упорядоченное множество M содержит не более одного максимального и одного минимального элемента.

Элемент, покрывающий ноль в частично упорядоченном множестве M, называется **атомом** или **точкой**. Если любое подмножество удовлетворяет этим условиям, то решетка будет полной.

Решетка может быть дистрибутивной, если она удовлетворяет следующим тождеству:

Решетка может быть дедекиндовой, или модулярной, т.е.:

Решетка будет дистрибутивной, если она дедекиндовая. Дистрибутивная решетка с дополнениями называется **булевой алгеброй**.

# **4.4 Связность графов**

**Связностью графа**  называется наименьшее число вершин, удаление которых делает граф несвязным, или тривиальным. Полный граф становится тривиальным, если удалить вершин, тогда .

Если , то граф называется **n - связным**.

Реберной связностью называется минимальное число ребер, удаление которых приводит к несвязному, или тривиальному графу:. Для любого графа существует зависимость: *,* где – это минимальная степень вершин.

Связность характеризуется наличием маршрутов, или цепей. Простые цепи называются **реберно-непересекающимися***,* если никакие две из них не имеют общего ребра. Если же у таких цепей нет и общих вершин, то они будут **вершинно-непересекающимися**.

Пусть в связном графе **G** выделены две различные вершины **U** и **V**. Множество ребер **Е** называется **U, V-разделяющим множеством** в графе **G**, если любая простая цепь из **U** в **V** содержит ребро из **Е**. Множество вершин **V** не содержащие **U** и **V** называется **UV-отделяющим множеством G**, если любая простая цепь из **U** в **V** проходит через вершину **V**.

Существуют теоремы, определяющие зависимость связности графа от числа непересекающихся цепей (теоремы Менгера):

1. Максимальное число реберно-непересекающихся простых цепей, соединяющих две вершины **U** и **V** равно минимальному числу рёбер в **U, V-разделяющем множестве***.*
2. Максимальное число вершинно-непересекающихся простых цепей между **U** и **V** равно минимальному числу вершин в **U, V***-***отделяющем множестве**.
3. Граф **n-связи** тогда, когда любая пара его вершин соединена, по крайней мере, **n****вершинно-непересекающимися цепями**.
4. Граф n-реберной связи тогда, когда любая пара его вершин соединена, по крайней мере, n реберно-непересекающимися цепями.

Для любых двух непересекающихся множеств вершин максимальное число непересекающихся цепей, соединяющих две вершины множеств равно минимальному числу вершин, отделяющих эти две вершины.

Анализ маршрутов в графе можно проводить при помощи матрицы достижимости, которая формируется на основе матрицы смежности вида:

Если получить матрицу **S2**, то ее ненулевые элементы будут указывать на существование цепей длины 2 и т.д. Причем, при сложении матриц **S+S2** и будут определены все маршруты длины 1 или 2.

Анализ матрицы достижимости позволяет сделать вывод о диаметре графа:

где **S(G)** – матрица смежности,

**D(G)** – матрица достижимости,

**d(G)** – диаметр графа

Минимальная длина цепи, соединяющая две его вершины, называется **расстоянием между вершинами**:

Максимальное расстояние между вершинами называется *диаметром графа:*

Функция расстояния называется метрикой, т.к. удовлетворяет трем аксиомам:

1. ,
2. ,
3. .

# **4.5 Способы задания графов**

Можно рассмотреть в качестве мультиграфа. Мультиграф представляет собой неориентированный граф, который является сетью. Сеть отличается от графо тем, что в ней выделяют подмножество полюсов причем один полюс, является входным, другие выходным.

Формируется (1,1)- полюсник.

Любой граф можно представить, как совокупность двух множеств. Например: ,

где

M – множество вершин,

T – множество дуг,

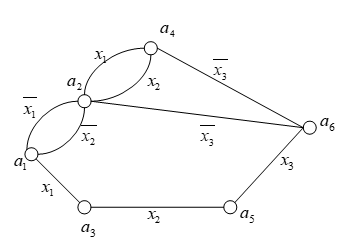
К – множество полюсов

Такой граф называется К- полюсной сетью.

Сеть G, где каждое ребро помечено буквой из алфавита

К-полюсная контактная схема

Дана двухполюсная контактная схема, где -входной полюс, -выходной.



Бывает так, что один полюс выделен и является входным, а остальные- выходные – равноправны, тогда (к+1) – полюс схема называется (1, к) полюсным.

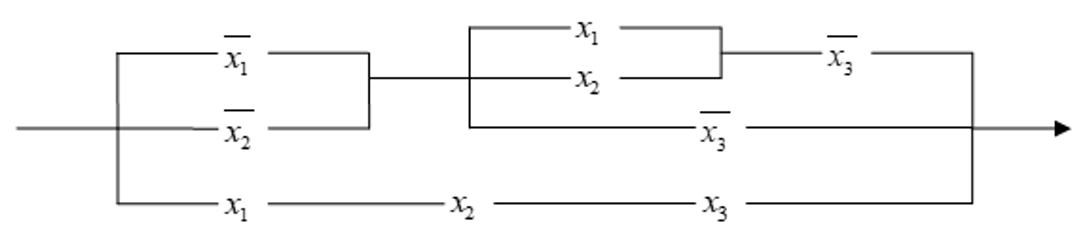
Ребра контактной схемы называются контактными.

Контактный, соответствующий , называют замыкающим, пропускает ток если =1.

Контакт, соответствующий литере, называют размыкающим, не пропускает ток, если =0.

Таким образом реализуется два состояния переключателя:1- ток проходит, 0- не проходит.

Здесь реализуется две функции - которой соответствуют последовательному соединению контактов.



**Электронная схема**

В этой схеме можно обозначить 2 полюса как a и b, где - это некоторая цепь из a в b, тогда – конъюнкция.

**4.6 Теорема Эйлера**

При анализе графов следует применять теорему Эйлера, которая справедлива и для связных, и для несвязных плоских графов. Для связного плоского графа, имеющего **n** вершин, **m** ребер, **r** граней, справедлива формула Эйлера:

Выделяются внутренние и внешние грани. **Внутренняя грань**– это конечная область плоскости, ограниченная замкнутым маршрутом, не содержащая ни вершин, ни ребер этого графа. В теореме Эйлера учитывается и внешняя грань, находящаяся за границей внутренней грани. **Граница грани** – это минимальный маршрут. Существует две леммы:

1. для любого плоского графа , т.к. цикл имеет минимум три ребра. Если в графе все циклы имеют три ребра, то граф называется плоской триангуляцией.
2. для любого плоского графа при :

Теорема Эйлера для несвязных плоских графов:

где **k** – количество компонент связности.

Для проверки полноты графа используется соотношение:

Если вершин **n**, то ребер **m** будет .

Существуют стандартные свойства планарных графов:

1. в любом планарном графе есть вершина, степень которой не больше 5.
2. если у связного графа каждый простой цикл содержит не менее *k* ребер, то . Простой цикл подразумевает формирование маршрута, проходящего по каждому ребру только один раз.
3. если то хотя бы один из исходного графа, или дополнения не планарен.

# **4.7 Цикломатика**

Любой замкнутый маршрут называется **циклом**. Цикл называется **эйлеровым**, если каждое ребро графа участвует в его образовании только один раз. Граф содержащий такой цикл называется **эйлеровым**. Граф будет эйлеровым если он связен и степень каждой вершины четна.

Простой цикл называется **гамильтоновым**, если проходит через каждую вершину графа. Граф, содержащий такой цикл, называется **гамильтоновым**. В нем степень каждой вершины не меньше одной второй:

Циклы изучаются при помощи цикломатической матрицы . Каждому циклу графа сопоставляется вектор-строка матрицы. Для исследования циклов в графе используют цикломатическую матрицу

. Каждому циклу графа сопоставляется вектор-строка матрицы . При этом принимает одно из двух значений:

Число строк в матрице соответствует числу циклов, число столбцов – числу ребер. Множество всех векторов, каждый из которых соответствует одному циклу графа, образует **векторное пространство**, или **пространство циклов**. При этом выполняются следующие условия:

Выделяется базис векторного пространства - система линейно независимых векторов, порождающее векторное пространство. Множество векторов является базисным тогда, когда любой элемент этого пространства единственным образом может быть представлен в виде линейной комбинации векторов множества. Если базис состоит из n-векторов, то он называется **n-мерным пространством**. **Базис циклов** графа **G** — это базис пространства циклов графа G, состоящий из простых циклов.

Количество базисных циклов всегда известно и определяется цикломатическим числом **υ(G)** равному числу хорд любого остова в графе **G**. **Остов** — это остовный подграф, являющийся деревом. **Остовный подграф** - подграф, содержащий все вершины графа. **Дерево** - связный граф, не содержащий ни одного цикла. **Хорда остова дерева** в связном графе **G** — это ребро графа, не принадлежащее дереву.

Все циклы получаются как линейная комбинация (сложение по модулю два). Для связного графа с n вершинами и m ребрами цикломатическое число равно:

Для графа с **k**-компонентами связности:

Пример:

Пусть  – граф. Цикл графа может быть записан в виде

**, где**

Каждый цикл может быть представлен в качестве **двоичного вектора**. Множество циклов образует пространство двоичных векторов.

**Цикломатический базис** – совокупность линейно независимых циклов графа, с помощью которых могут быть получены все остальные циклы. **Цикломатическое число графа V(G)**- мощность базисной системы циклов графа **G**.

*Пример*

V(G) = 8-6+1=3

Граф, у которого цикломатическое число равно 0, называется **деревом** (или **ациклическим графом**). Многокомпонентный ациклический граф называется **деревом**. **Остов графа** – частичный граф исходного графа, в котором число вершин и число компонент связности совпадает с числом вершин и числом компонент связности исходного графа, но цикломатическое число равно 0.

# **5 ДЕРЕВЬЯ**

# **5.1 Понятие о деревьях**

Дерево представляет одну компоненту связности. **Лес** – это граф, не содержащий циклов и состоящий из нескольких компонент связности (деревьев). У дерева всегда есть одна вершина, называемая **корнем**. Существует понятие арности дерева, т.е. упорядоченной структуры из каждой вершины которой исходит **n-ребер**. Простейшее дерево бинарное.

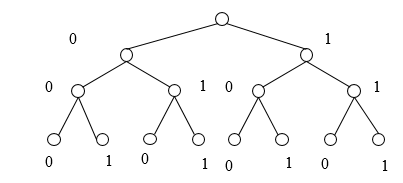
В теории кодирования схемы кодирования представляются кодовыми деревьями. В программировании левую ветвь бинарного дерева называют **левым сыном**, правую – **правым сыном**. Вершина может иметь двух сыновей, только одного или не иметь сыновей. Такая вершина называется висячей, или листом.

Базовым алгоритмом является сортировка бинарного дерева. Пусть нужно отсортировать n элементов. Здесь возникает проблема задания дерева. Можно задать в виде двух массивов: **l(N)** и **r(N)**, где **l(N)** указывает номер вершин, соответствующих левым сыновьям, **r(N)** – правым сыновьям, индекс **i** –номер вершины.

При сортировке реализуется попарное сравнение элементов, являющихся весами левого и правого сыновей. Более «легкий» («тяжелый») элемент поднимается к корню путем попарного сравнения весов вершин одного яруса. Этот элемент записывается как первый элемент выходного массива.

**Пример:** Задача о голосовании. Пусть существует устройство, фиксирующее принятие некоторой резолюции «комитетом трех». При одобрении резолюции нужно нажать кнопку.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | k |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



Здесь построено полное бинарное дерево, где вершины нижнего яруса показывают поле исходов. Вероятность принятия резолюции наблюдается в 4 исходах из 8 и вероятность непринятия равна 0,5.

# **5.2 Понятие о сетях**

**Сетью**называется структура, в которой выделяются следующие множества:

где является набором элементов из V, ,

**Е0** – полюсы сети, V – множество вершин.

Тогда сеть можно обозначить как:

Пример: **;**

**;**

**;**

**;**

**;**

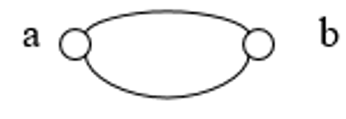
**;**

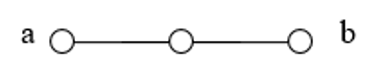
**;**

**.**

Графическая интерпретация выглядит следующим образом: сначала выделяются полюсы. Каждому набору сопоставляется круг (если содержит один объект - то ставится круг, если два - дуга). На периферии каждого выбираются попарно различные вершины, помеченные соответствующими символами, при этом круги попарно пересекаются. Все точки, помеченные одним и тем же символом, соединяются связной компонентой , которая с построенными ранее вершинами и кругами имеет общими только точки, помеченные символами .

**Особым классом сетей являются двухполюсные сети из двух объектных** наборов, т.е. ***,* .**В рамках этого класса выделяются **Р-сети**. Это сеть является суперпозицией сетей **.** Это два вида простейших неразложимых сетей:



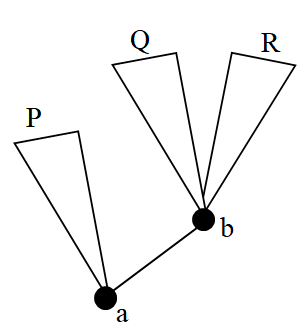


Поэтому булевы функции можно представить в виде **Р-сетей**. Каждой Р-сети соответствует множество укладок дерева, не концевым вершинам которых сопоставляются символы **Р** и **S**.

# **5.3 АВЛ**

Дерево называется сбалансированным, или АВЛ – деревом, в честь математиков Адельсона-Вельского и Линдаса, если для любой его вершины высоты левого и правого поддерева этой вершины отличаются не более чем на единицу.

В некоторых случаях можно восстановить сбалансированность деревьев.



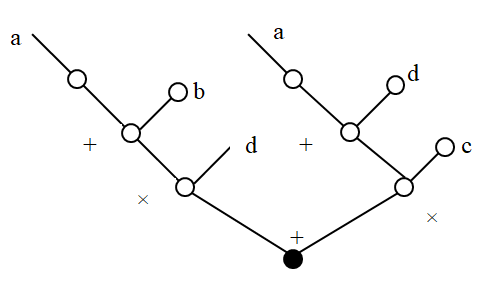
# **5.4 Деревья Канторовича**

В 1955г. Канторович предложил формировать схемы потоков данных с точки зрения теории графов. Дерево устанавливает взаимосвязь шагов вычисления и указывает место для проведения вычислений. А именно узлы для которых уже определены все операнды.

Пример: вычислить

Здесь вычисление можно организовать сразу по двум алгоритмам:

1. «Вычисление снизу-вверх», сначала, а×2, затем b×2 и т.д.
2. «Вычисление слева направо», по каждой ветви.



В 1975г. Эдсгер Дейкстра предложил идею «охраняемых команд». Он предложил использовать при создании программы только три структуры: sequence – исследование, selection – ветвление, iteration – цикл.

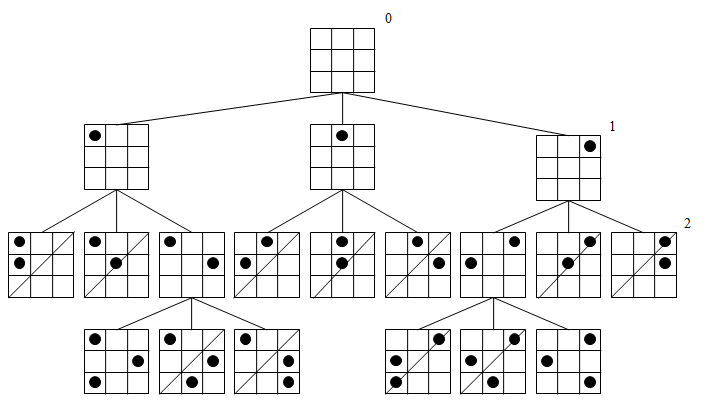
В общем случае перед каждой формулой ставится некоторое условие, которое называется охраняющим, или защищающим, или стражем. Он охраняет «свою» ветвь от ненужной обработки. Те ветви, стражи, которые выдали false, безоговорочно отключаются. Между всеми ветками стражи, которых выдали true производится свободный выбор.

Если ровно один страж выдает true, то выбора нет – это классическая детерминированная ситуация. Если ни один страж не выдает true, то результат вычисления не определен. Порядок, в котором вычисляются стражи и ветви ограничен лишь тем, что ветвь нельзя выполнять пока не будет вычислен ее страж и результат вычисления не окажется true.

**Задача обхода «дерева позиций»**

Перечислить все способы расстановки n ферзей на шахматной доске размерностью n×n, при которых они не бьют друг друга на соседних горизонталях.

Корнем дерева будет нулевая позиция, т.е. свободная доска. 1-ая позиция – ферзь располагается в 1-ой горизонтали. k-ая позиция – в k-ой горизонтали. Если в k-ой позиции ферзи бьют друг друга, то ставить новых ферзей нет смысла. Пусть n=3:



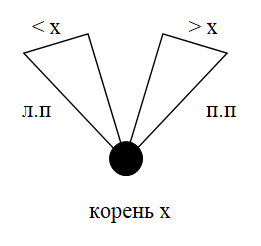
Всего исходных (количество висячих вершин последнего яруса) – 33=27. Благоприятных условий – 2. Вероятность равна .

Эта типовая задача условно разбивается на две части: обход произвольного дерева и реализация дерева допустимых позиций.

# **5.5 Деревья с помеченными (размеченными) листьями**

Пусть выбрано некоторое конечное множества вершин полного двоичного дерева с каждой вершиной всех ее предков. Пусть на каждой вершине существует некоторое значение Т, т.е. задано отображение множества вершин в множество значений типа Т. Такая структура называется Т-деревом.

Условно всякое не пустое Т-дерево можно разбить на корень, несущий пометку из Т и левое и правое поддеревья, которые могут быть пустыми. **Высотой поддерева** считается максимальная длина цепи, составленной из его вершин начиная от корня.



Упорядоченным Т-деревом называется такое дерево, для которого выполняется свойство: для любой вершины х все пометки в ее правом поддереве меньше пометки в х, а в правом – больше ее.

Для ровно подстриженных деревьев высота соответствует , где **n** – число вершин. При добавлении элемента в случайном порядке средняя высота не превышает .

# **5.6 Изоморфизм алгебры Буля и алгебры Кантора**

Множество вида

Множество вида

Общее число различных конституэнт не превышает **2n**. Каждой конституэнте можно сопоставить двоичный набор длины **n**. Если некоторые конституэнты равны , то их общее число меньше, чем **2n**, при этом среди подмножеств найдутся хотя бы две такие, которые можно выразить одно через другое, т.е. зависимые.

Пример: n=2 и , формируется 4 конституэнты (22):

Пересечение двух различных конституэнт пусто: .

Объединение всех конституэнт равно единице – универсум: .

Каждое не пустое множество, образованное из множеств с помощью операций является объединением некоторого числа конституэнт. Из n множеств в алгебре Кантора можно образовать не более чем функций.

Введение понятия конституэнты позволяет задавать множество **M** при фиксированных независимых подмножествах универсального множества в виде объединения конституэнт:

Каждое фиксированное множество разбивает пространство на две части: на, собственно, и на дополнение к нему . При n множествах пространство разбивается на **2n** областей. Каждая область является пересечением **n** множеств или . Можно сопоставить этой области двоичный вектор *,* в котором , если в конституэнту входит , и , если входит , а также десятичный эквивалент.

Любое множество **M** в пространстве 1 можно задать в виде объединения этих областей. Можно сопоставить множеству **M** двоичный вектор длины **2n**, в котором **i-му** разряду соответствует область с десятичным эквивалентом, равным **i**. Вектор, определяющий множество, можно представить в виде десятичного эквивалента:

Пример: есть трех мерное пространство , где выделено множество с десятичным эквивалентом d(M)=217

217=1\*27+1\*26+0\*25+1\*24+1\*23+0\*22+0\*21+1\*20

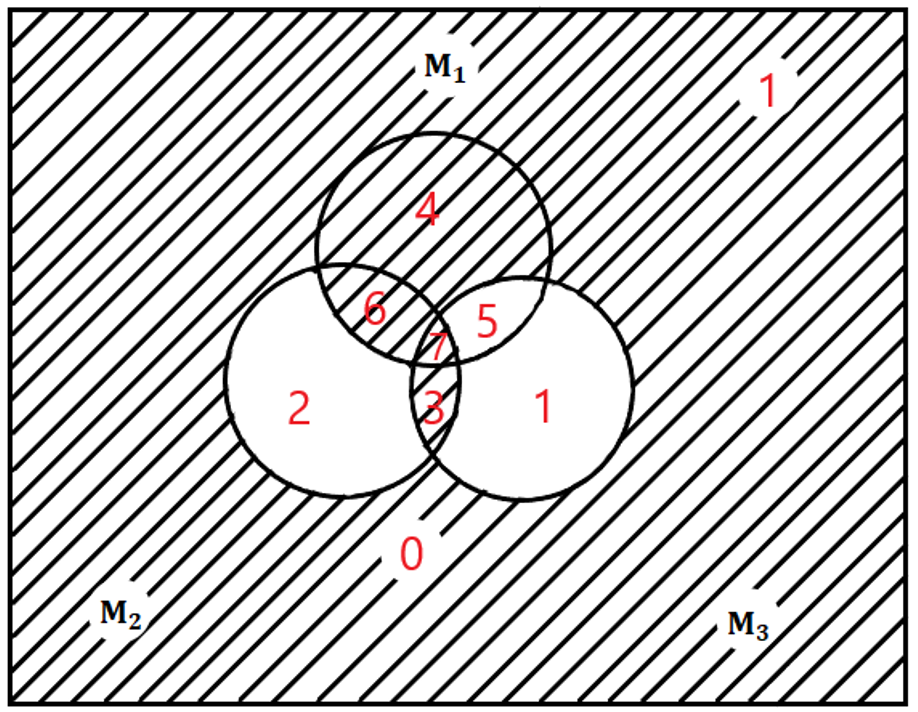
т.е. множеству M соответствует двоичный вектор (1,1,0,1,1,0,0,1), определяющий включение областей в множество M.

Множество M можно задать в виде двоичной таблицы, каждой строке которой взаимно однозначно соответствует конституэнта. Столбцам соответствуют множества, образующие пространство, последний столбец сопоставляется множеству M, и 1 указывает на вхождение соответствующей конституэнты в множество M.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | M |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 1 |

Аналитически множество **M** задается в виде:

Можно представить пространство в виде диаграмм Эйлера:



# **5.7 Минимизация булевых функций в классе ДНФ**

**Разложение Шеннона**

Шеннон доказал теорему о том, что любая функция может быть представлена в виде:

где - либо 0 либо 1

Предельное разложение Шеннона (при **k=n**) имеет вид:

т.е. это разложение формируется для значений функции, равной единице, и называется совершенной дизъюнктивной нормальной формой (СДНФ).

Двойственное предельное разложение Шеннона называется совершенной конъюнктивной нормальной формулой (СКНФ) и формируется по нулевым значением булевой функции:

Эти формулы можно получить из таблицы истинности, используя следующие алгоритмы:

1. **Для СДНФ**

Нужно отметить в таблице строки, где функция равна единице.

* 1. Для каждой такой строки образовать логическое произведение

Нужно объединить полученные конъюнкции (конъюнкты) знаками дизъюнкции.

1. **Для СКНФ**

Нужно отметить в таблице строки, где функция равна нулю.

Сформировать дизъюнкции .

Объединить полученные дизъюнкции (дизъюнкты) знаками конъюнкции.

СДНФ аналогична - сумма произведения и называется дизъюнкцией конъюнктов.

СКНФ аналогична - конъюнкция дизъюнктов.

Пример: представить булеву функцию вида

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № |  |  |  | *f* |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 |
| 5 | 1 | 0 | 1 | 0 |
| 6 | 1 | 1 | 0 | 1 |
| 7 | 1 | 1 | 1 | 0 |

СДНФ

СКНФ

# **5.8 Полнота и замкнутость систем булевых функций**

Система булевых функций будет функционально полной, если любая булева функция может быть записана в виде формулы через функции этой системы, то есть функция представима как суперпозиция указанных функций.

**Базис** *–* это полная система, не содержащая лишних функций, т.к. удаление любой функции из базиса нарушает полноту системы.

Проблему полноты исследовал Пост. В СССР Яблонский.

Выделяются сильная и слабая теоремы Поста.

1. **Сильная теорема Поста**

Для того чтобы система была полной необходимо и достаточно, чтобы для каждого класса функций Поста в системе нашлась хотя бы одна функция, не принадлежащая ему.

1. **Слабая теорема Поста**

Если система функций содержит тождественные 0 и 1, то для ее полноты достаточно, чтобы она содержала хотя бы одну монотонную и хотя бы одну не линейную функцию.

Теоремы основаны на анализе классов функции Поста. Всего пять классов функций:

1. Класс функций, сохраняющий **const 0**

**k0** – это множество функций вида:

**f(0,0,…,0)=0**

1. Класс функций, сохраняющий **const 1**

**k1** – это множество функций вида:

**f(1,1,…,1)=1**

1. Класс самодвойственных булевых функций

**k**- это множество функций вида:

то есть на любой паре противоположных наборов (наборов, сумма десятичных эквивалентов, которая равна **2n-1**) принимает противоположные значения.

1. Класс монотонных функций.

Возрастание (убывание) значения набора приводит к не убыванию (не возрастанию) функций.

1. Класс линейных функций.

Линейность или не линейность функции можно определить, используя теорему, о том, что любая функция из может быть выражена при помощи полинома по модулю 2, или полинома И. И. Жегалкина.

В общем виде полином можно представить в виде:

Он включает в себя все возможные комбинации конъюнкции аргументов, которые называются взаимодействием элементов.

Пример: выразить дизъюнкцию в виде полинома Жегалкина:

|  |  |  |
| --- | --- | --- |
| x1 | x2 | f(x) |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**Вывод:** функция не линейная, т.к. в ее разложении присутствует взаимодействие аргументов.

# **5.9 Принцип двойственности**

Применяется для получения тождеств.

Двойственной функцией к функции называется функция . Двойственную функцию можно получить из таблицы путем инвертирования столбца (заменой 0 на1, 1 на 0) и переворачиванием (первая строка становиться последней, вторая - предпоследней и т.д.). Из определения двойственности следует, что. Для получения двойственных формул нужно в исходной формуле заменить **0 на 1, 1 на 0, на &, & на .**

Существует теорема о том, что любая функция может иметь двойственную функцию. Из которой следует принцип двойственности:

Если формула реализует функцию , то формула , полученная из **U** заменой функции на двойственную будет реализовать двойственную функцию . Полученная формула называется двойственной к

Пример: cоставить двойственную формулу

# **6 ЭЛЕМЕНТЫ МАТЕМАТИЧЕСКОЙ ЛОГИКИ**

Математическая логика есть анализ методов рассуждений с подключением математического аппарата. При этом исследователей не интересует, истинны или ложны посылки (причины) и заключения из них. Но нужно ответить на вопрос: вытекает ли истинность заключений из истинности посылок. Систематическая классификация и формализация правильных способов рассуждения – **основная задача****математической логики***.* В более узком смысле цель математической логики дать точное определение понятия математического доказательства.

До XIX века математики недостаточно обращали внимание на математическую строгость доказательств теорем. Поэтому изложение теорем называлось неявно аксиоматизированным:

1. Аксиомы и основные понятия имеют определенный содержательный смысл. Аксиомы истинны в силу своей очевидности.
2. Теоремы истинны, так как они выводятся путем логических рассуждений из истинных посылок. При этом не указывается, какие способы рассуждений являются логически правильными, то есть отсутствует понятие правила вывода.

Эта форма аксиоматического построения теории называется **содержательной***.* Однако развитие математики в XIX – XX вв. привело к созданию некоторого нового уровня дедуктивно–аксиоматических построений, которые стали называться **формальными***.*

**Математика**– дедуктивная наука, то есть истинность математических рассуждений не может быть проверена путем опыта или наблюдений, а выводится с помощью логических рассуждений, исходя из небольшого количества исходных утверждений.

**Высказывание** *–* это некоторое предложение, которому уместно задать вопрос: истинно оно или ложно**.**Высказывания могут быть **простыми** *–* такие, которое нельзя разделить на части. **Сложное высказывание**состоит из простых, соединенных между собой логическими связками (пропозициональные связки). Высказывания обозначаются заглавными латинскими буквами.

При разработке любой теории возникают следующие проблемы:

1. **Проблема непротиворечивости**

Состоит в том, чтобы выяснить непротиворечивость методов, которые содержатся в аксиомах. Этого можно достичь, если показать, что из данной системы аксиом нельзя вывести **А** - истинное или - истинное.

1. **Проблема полноты аксиом**

Требуется установить, достаточна ли система аксиом, чтобы из нее в качестве логических рассуждений выводились все тождественно–истинные утверждения теории, то есть теоремы.

1. **Проблема разрешимости**

Требуется установить, существует или нет эффективная процедура, которая для любого утверждения теории устанавливает, выводимо или не выводимо это утверждение из заданных аксиом.

Формальная, или аксиоматическая теория считается определенной, если выполняются следующие условия:

1. Задано множество символов, или алфавит теории. Конечные последовательности символов теории формирует слова ивыражения;
2. Имеется подмножество выражений теории, которые называются **формулами**. Обычно имеется процедура, которая позволяет по данному выражению определить, является она формулой или нет;
3. Выделено некоторое множество формул, или аксиом. Если имеется процедура, с помощью которой можно по данной формуле установить, является ли она аксиомой или нет, то формальная теория называется **эффективно***–***аксиоматизированной**;
4. Выделено конечное множество отношений между формулами данной теории, которые называются правилами вывода.

**Выводом**называется всякая последовательность формул , где является либо аксиомой, либо непосредственным следствием из предыдущих формул по одному из правил вывода.

Формула **А** называется **теоремой** некоторой теории, если существует вывод, в котором последней формулой является - формула выводима.

# **6.1 Понятия об индукции и дедукции**

Выделяют два метода рассуждений – дедукцию и индукцию.

Дедукция – от лат. выведение – цепь умозаключений (рассуждение), звенья которой (высказывания) связаны отношением логического следования. Началом (посылками) дедукции являются аксиомы, постулаты или просто гипотезы, имеющие характер общих утверждений («общее»), а концом – следствия из посылок, теоремы («частное»). Если посылки дедукции истинны, то истинны и её следствия.

Индукция – от лат. наведение – умозаключение от фактов к некоторой гипотезе (общему утверждению). Полная индукция – обобщение относится к конечно – обозримой области фактов. Неполная – относится к бесконечно – необозримой или конечно – необозримой области фактов. Математическая индукция – это общий способ определения некоторого свойства А для всех натуральных n, основанный на заключении от n к n+1.

# **6.2 Тождественно-истинные и тождественно-ложные высказывания**

Примеры тождественно–истинных высказываний (тавтология) – эти высказывания истинны при любых значениях аргументов.

1. **Закон тождества (всякое высказывание является следствием самого себя):**
2. **Закон противоречия:**
3. **Закон исключения третьего:**

**(А или не А)**

1. **Закон двойного отрицания:**
2. **Закон Modus ponens (правило заключения, или правило отделимости)**

**(правило заключения, или правило отделимости):**

1. **Закон Modus Tollens (доказательство от противного):**
2. **Правило силлогизма (транзитивности):**

# **6.3 Тождественно – ложные высказывания (противоречия)**

Формула вида:

Называется «из ложного что угодно»

# **6.4 Аксиоматика Клини**

Логические формулы реализуются при помощи логических символов. Аксиоматика теории высказываний основана на аксиоматике Клини, включающей в себя 10 аксиом:

# **6.5 Примеры кванторов**

Примеры кванторов в логике высказывания. **А** и **В** – простые высказывания.

**А ~ B**

1. если **А**, то **В** и обратно;
2. для необходимо и достаточно **В**;
3. **А** равносильно **В**;
4. **А** тогда и только тогда, когда **В**;
5. **А** суть **В**.
6. если **А**, то **В**;
7. в случае **А** имеет место **В**;
8. для **В** достаточно **А**;
9. для **А** необходимо **В**;
10. **А** только, если **В**;
11. **А** имплицирует **В**.
12. не только **А**, но и **В**;
13. как **А**, так и **В**;
14. **А** вместе с **В**;
15. **А** в то время, как **В**.
16. **А** или **В**, или оба;
17. **А** или **В**.
18. неверно **А**;
19. **А** не имеет место.

Пример 1:

Составить высказывания: я заплачу за ремонт телевизора только тогда, когда он будет работать.

Я заплачу за ремонт телевизора – А

Он будет работать – B

B Ↄ A, ├

Пример 2:

Если бы он не сказал, она бы и не узнала. Не спросила бы она, он бы не сказал.

Сказал – C

Узнала – Y

Спросила – B

Ↄ , ├B

# **6.6 Исчисление кванторов и предикатов**

Исчисления высказываний является самым простым элементом при формировании логических рассуждений, так как используется только двоичная логика: **1** или **0**. В рамках **k** – значной логики можно построить предикат, а значит формируется научное направление, называемое исчислением предикатов. Тогда можно определить наличие того или иного свойства на конечном множестве элементов.

Функция **Р**, принимающая одно из значений, 0 или 1, аргументы которой принимают значения из произвольного множества М, называется **предикатом Р**в**предметной области М**. Число аргументов предиката называется его **порядком**. Этот предикат определяет **n–арное** отношение **R** в множестве **М**.

Если , то находятся в отношении R, определяемом этим предикатом. Если , то эти элементы не находятся в отношении **R** .

Для упрощения структуры сложных логических рассуждений можно ввести встречающихся выражений:

1. для всякого элемента свойство **R** выполнено - , где - квантор всеобщности;
2. существует, по крайней мере, один элемент , обладающий свойством **R** - , где - - квантор существования.

То есть в рамках исчисления предикатов существует исчисление кванторов. Другие кванторы применяются в исчислении суждений и позволяют получать тождественно-истинные и тождественно-ложные формулы логики высказываний.

Пример: пусть формируется выражение «x делит y» на множестве натуральных чисел **N**. Тогда применение кванторных опций к предикату приводит к 8 возможным высказываниям:

1. - «Для всякого y и для всякого x, y является делителем x».
2. - «Существует y, которое является делителем всякого x».
3. - «Для всякого y существует x такое, что x делится на y».
4. - «Существует y и x такие, что х делит y».
5. - «Для всякого x и всякого y, y является делителем x».
6. - «Для всякого x существует такое y, что x делится на y».
7. - «Существуют x и y такие, что y является делителем x».
8. - «Существует x такое, что для всякого y, x делится на y».

Из множества высказываний можно выделить ложные и истинные. Ложные - 1,5,8, остальные истинные. Ноль местным предикатом и является высказывание, или некоторое предложение, позволяющее в том или ином контексте задать вопрос истинно оно или ложно.

Для аксиоматики Клини добавляются еще две аксиомы:

1. (all схема)
2. (exist схема)
3. не для каждого х верно **А(х)**;
4. не все суть **А**.

Составить высказывание: x бабушка y, используя предикаты:

Ж(х) – женщина,

Р(х,y) – y родитель x,

х бабушка y = Ж(х)

# **6.7 Элементы Аристотелевой логики**

Аристотель (384 – 322г.г до н.э) - автор оригинальной логической системы, которая называется силлогистикой. Она изучает высказывания о **характеристическом классе**– это множество предметов, на которых предикат принимает значение true. Обозначаются классы как множества, но под ними понимаются одноместные предикаты. Поэтому **силлогистика** называется исчислением одноместных предикатов, или исчислением классов, или теорий категорических суждений. Любое высказывание о классе называется **категорическим суждением**. Выделяется четыре типа высказывания:

А – общеутвердительное

Все Р суть Q -

Е – общеотрицательное

Никакие Р не суть Q -

I – частноутвердительное

Некоторые Р суть Q -

О – частноотрицательное

Некоторые Р не суть Q -

Высказывания связаны между собой законами двух типов:

1. **Законы обратимости**:
2. **Законы логики**:

Выделяется конструкция, состоящая из трех суждений о трех классах, в которой третье суждение (заключение) следует из двух первых. Это категорический силлогизм, или модус. Силлогизм является правильным, если истинность посылок всегда вызывает истинность заключений независимо от содержания высказывания. Силлогизмы делятся на четыре группы:

**PQ** посылка **QP PQ QP**

**SP** посылка **SP PS PS**

**SQ** заключение **SQ SQ SQ**

Посылки условно делятся на большие – относятся ко всему множеству объектов и малые – относятся к некоторой группе объектов.

Пример: Все люди смертны

Сократ - человек

Следовательно, Сократ смертен.

Если класс не применяется в заключении, то он называется промежуточным.

Таким образом, логика Аристотеля отличается от вывода традиционной логики.

Традиционный силлогизм может быть правильным или неправильным, но не может быть истинным или ложным, т.к. силлогизм формируется не из одного предложения, а из ряда предложений. Таким образом, большинство логических теорий в настоящее время базируются на Аристотелевой логике, т.к. в основе любых логических построений лежит высказывание, или суждение, которое может принимать только одно из двух значений: **false** или **true**.

Выделяют новый метод рассуждений – абдуктивный, основанный на редукции. Это вывод (заключение), который получается, исходя из одной посылки. Вторую посылку восстанавливают по смыслу. Применяется в эмпирических исследованиях.

С развитием вычислительной техники и научных направлений, связанных с информационными технологиями, появилась необходимость развития новых направлений, связанных с мышлением.

# **6.8** **Тенденции развития теории познания**

Современные теории производства ЭВМ почти достигли своего предела, поэтому ученые отыскивают другие пути производства ЭВМ. В частности, по созданию квантового компьютера. Он позволяет неограниченно распараллеливать один и тот же процесс с вариацией начальных условий. Обрабатываться будут **Q-биты** (квантовые биты), он может заменить 2600 параллельно соединенных процессоров.

Идею квантового компьютера выдвинул в 1982г. Фейнман. Если взять систему с 40 частицами в квантовой механике, то для ее изучения нужно умножать и брать экспоненты матриц размером 240240, что технически слабо разрешимо, то есть квантовые системы используются как «черный ящик».

В 20-30г.г. двадцатого века было доказано, что самая высокая скорость вычислений может быть достигнута, если основанием системы счисления будет выбрано число е.

Эти рассуждения продолжил физик и математик сэр Роджер Пенроуз. Он отметил, что алгоритмическое мышление, присущее любой ЭВМ, не совершенно, а значит, в природе используется не только оно. Он написал ряд книг: «Новый мозг короля» и «Тени ума», где пытался доказать, что мозг – это квантовый компьютер, поэтому аристотелевое мышление чуждо.

Но если Фейнман решать алгоритмически разрешимые задачи, техническая реализация которых не возможна, то Пенроуз утверждает, что можно решать и алгоритмически неразрешимые задачи.

Квантовая гравитация, т.е. общая теория относительности на расстоянии близких к квантовым, целиком не построена. Работу квантового компьютера нельзя проверить или обосновать детерминистской логикой (причинно-обусловленной), поэтому Пенроуз утверждает, что человеческий интеллект использует квантовую гравитацию как основу для интуитивных озарений, проверяемых или нет аристотелевой логикой. В рамках парадигмы (образец, пример для доказательства) интересны идеи, основанные на немеханической, трансцендентной природе мышления, приходящие как озарение, а не в результате дедуктивных логических построений.

Аргументы Пенроуза основаны на теореме Геделя о неполноте: при любой системе аксиом найдутся такие высказывания, относительно которых нельзя сказать, истинны они или ложны. Чтобы определиться, нужно расширять систему аксиом, но тогда обнаружатся новые неконкретные высказывания. Случайно отбирая некоторую систему аксиом, можно автоматически породить виртуальный (логически возможный) новый мир. Например, геометрия Лобачевского и Евклида, Римана.

Такими рассуждениями можно доказать или опровергнуть любое математическое утверждение, хотя доказательство не будет формальным и никак не выводится алгоритмически. С другой стороны, аристотелево мышление может существовать только в рамках формальной системы аксиом.

Используя теорему Гёделя, Пенроуз доказывает, что существует инструкция, способная сломать (остановить) любой классический компьютер достаточно сложной структуры. Если мышление детерминировано логикой, то человечество в целом следует видеть, как аристотелев (детерминистский) компьютер, который может существовать вечно (потенциально, если не в реальности), и поэтому будет абсолютно безошибочен.

Логика работает только с вербальным (словесным) предоставлением информации.

Другое предоставление информации – обратное. Законы манипулирования образами пока ещё не изучены. Компьютер накапливает просто множество данных, как истинных, так и ложных, или базу мнений. Что происходит в действительности, нужно определять логическим путем и (или) с помощью практики. Это будет уже база знаний (база мнений, данных + некие правила). Логический пусть – это пусть выбора аксиом и следствий из них. Но человечество уже недетерминировано логикой, т. к. реализуются нелогичные поступки.

Таким образом, в настоящее время человечество решает две проблемы: создание новой технической базы и нового мышления будущих ЭВМ. Сокращаются размеры физических компонентов ЭВМ и используются всё более тонкие (глубокие) физические эффекты, в том числе квантовые. Есть доказательства, что вполне достижимы скорости. Превышающие скорость света. Чем меньше детали или эффекты, тем выше скорость обработки и меньше габариты вычислительных машин.

Вторая тенденция – последовательное расширение функциональных возможностей вычислительной техники, использование новых методов обработки информации, в т.ч. интуитивных.

Такие запуски первых спутников в 1957г., 1959 г.г. американцы не поверили, что это возможно. При нашей вычислительной технике нельзя было рассчитать траекторию, навигацию и т.д. Наши математики просто придумали другие методы решения задач, не требующие сложных вычислений, которые можно было произвести даже на обычных счётах. Поиск вариантов присущих человеку.

Стремление моделировать интуитивные озарения (скачки мысли) реализовали великие умы (Пуанкаре, Менделеева). В 60-е годы использовались датчики случайных чисел, методы Монте Карло, метод оврагов.

Механизм озарения следующих: количество данных ограничивается, и происходит поиск решения на этом пространстве данных и делается вывод о предполагаемом глобальном решении.

**Метод оврагов** – метод поиска экстремального решения (min). Градиент – это направление уменьшения функции. «Залезли в овраг и нашли дно. Но где гарантия, что рядом нет другого оврага с более глубоким дном? Бросок в сторону – в иное пространство. Скачок – и мы в другом овраге».

Это математическая модель метода мозгового штурма. Он не дает гарантии, что найден глобальный оптимум, но позволят хотя бы расширять горизонты.

На квантовом уровне очень большое влияние начинают оказывать помехи. Они пронизывают весь наш мир. Но на тела с большой массой, как правило, они не влияют. Но мелочи действуют и на крупное тело (бабочка села на штангу волку, и он упал в «Ну, погоди!».

При достижения системной критической точки её развитие может пойти по нескольким совершенно различным (дискретным) направлениям. И выбор его сильно зависим от незначительных факторов, т.е. любая бабочка может нарушить хрупкое равновесие. Такие явления изучает математическая теория катастроф, или большие системы, поверженные воздействию мелочей. Чем меньше тело, тем больше влияние на него оказывают случайные явления. Таким образом возникает проблема обеспечения устойчивой работы квантовых компьютеров, в т.ч. и при выполнении логических операций.

Современная информатика всё более отходит от своего первоначального узкого понимания как науки об автоматизированных (машинных) информационных системах. Теперь она рассматривает любые информационные системы – людей, животных, растения. А также все их сообщества.

# **6.9 Задача минимизации представления множеств в алгебре Кантора**

Пересечение попарно различных множеств называется элементарным. Выражение, задающее множество в виде объединения различных элементарных пересечений, называется нормальной формой Кантора (НФК) множества **M**. Объединение конституэнт множества **M** называется совершенной НФК множества **M**. Минимальной НФК множества **M** называется НФК этого множества, имеющая минимальную сложность.

Метод Квайна, используемый для получения минимальной НФК множества **M**, заключается в последовательном выполнении этапов:

1. **Выделение максимальных интервалов**. Интервалом множества **M** называется множество конституэнт множества **M**, образующих гиперкуб (некоторой размерности). Мощность интервала равна степени 2 (20,21 и т.д.).

Множество интервалов для примера:

{000,100,110,011,111, -00.1, -0.11, -11}

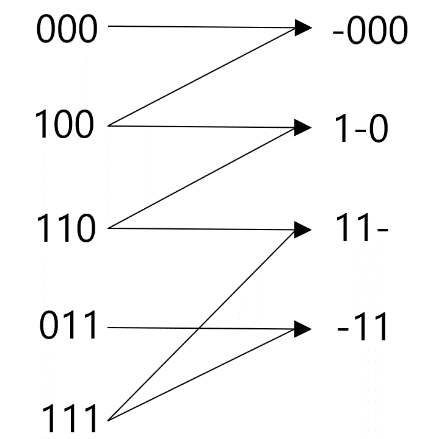
**«-»** означает, что множество, соответствующее этому разряду, в пересечении отсутствует, т.е. по этому множеству произошло склеивание. Например, интервал -00, соответствующий множеству конституэнт 000 и 100, получается в результате преобразования:

Интервал называется максимальным интервалом множества ***M***, если он не содержится ни в каком другом множестве.

В примере четыре максимальных интервала:

**-00, 1-0, 11-, -11**; каждый из них образует гиперкуб размерности 1(т.е. ребро).

Пересечение , соответствующее максимальному интервалу множества **M**, называется простой импликантой этого множества. Объединение простых импликант множества **M** называется сокращенной НФК множества **M**. Количество первичных термов, образующих простую импликанту, называется рангом простой импликанты, а элементарное пересечение – рангом соответствующего интервала.

При выделении максимальных интервалов множество интервалов, имеющих один и тот же ранг, разбивают на пояса, причем **i-й** пояс содержит интервалы, которым соответствуют наборы с **i** единицами в каждом. Тогда выделение максимальных интервалов сводится к сравнению элементов только соседних поясов, номера которых отличаются на единицу. Если построенные интервалы не являются максимальными, то процесс сравнения продолжают.

Сокращенная НФК множества М имеет вид:

1. **Нужно получить тупиковую НФК множества M покрытием двумерной таблицы**. Это такая НФК этого множества, которая при вычеркивании хотя бы одного первичного терма не определяет **M**. Минимальная НФК множества **M** состоит из простых импликант этого множества. Тупиковая НФК множества **M**, в т.ч. и минимальная НФК, содержится в сокращенной НФК этого множества.

Покрытием столбцов строками в двумерной таблице называется такое множество срок, при котором для каждого столбца найдется хотя бы одна строка из этого множества, на пересечении с которой этот столбец имеет 1, причем при вычеркивании хотя бы одного элемента из этого множества строк указанное свойство не выполняется.

Построение и покрытие таблицы Квайна.

**Таблица Квайна** – это двумерная таблица, каждой строке которой взаимно однозначно соответствует максимальный интервал, столбцу конституента, а на пересечении **i-й** строки и **j-го** столбца находится **1**, если **j-я** конституэнта водит в **i-й** максимальный интервал, иначе клетку **(i, j)** не заполняют или ставят в ней **Ø**.

Для примера:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Max интервал | Конституэнта | | | | |
| 000 | 100 | 110 | 011 | 111 |
| -00 | 1 | 1 |  |  |  |
| 1-0 |  | 1 | 1 |  |  |
| 11- |  |  | 1 |  | 1 |
| -11 |  |  |  | 1 | 1 |

Максимальный интервал называется обязательным, если найдется конституэнта принадлежащая только ему. Множество обязательных интервалов образует ядро покрытия. Ядром в примере является множество {-00,-11}, которое покрывает первый, второй, третий, четвертый и пятый столбцы. Для образования покрытия можно взять либо вторую, либо третью строку. В результате получится два покрытия: **{-00,-11,1-0}**,

**{-00,-11,11-}**, каждое из которых является максимальным и имеет сложность шесть. Выберем первое из покрытий, которое соответствует минимальной НФК, задающей множество:

.

Минимальная НФК находится в результате перебора всех покрытий, осуществляемого с помощью преобразования мультипликативно – аддитивной формы в аддитивно – мультипликативную форму.

Для примера: обозначим строки, тогда можно создать множество **Аj**, каждый элемент которого покрывает **j-й** столбец:

Нужно получить объединение этих множеств и найти их пересечения:

Каждое покрытие является минимальной НФК. Таким образом, полученные покрытия изоморфны базисным в булевой алгебре. Техническая реализация базисных функций может быть основана на использовании различных физических явлений, например, импликация и коимпликация – магнитных явлений, функций Шеффера и Вебба – явлений в полупроводниках. Базисные элементы графически изображают в виде прямоугольников ([ГОСТ 2.743 - 72](file:///F:\Учебники\ГОСТ%202.743-72.pdf)):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Элементы** | **Константа нуля** | **Отрицание x** | **Дизъюнкция** | **Конъюнкция** |
| Обозначения | |  | | --- | | 0 |  |  | | --- | | 1 | | |  | | --- | | x |  |  | | --- | | 1 | | |  | | --- | | x |  |  | | --- | | y |  |  | | --- | | 1 | | |  | | --- | | x |  |  | | --- | | y |  |  | | --- | | & | |
| **Элемент Вебба** | **Эл – нт Шеффера** | **Импликация** | **Коимпликация** | **Сложение по mod 2** |
| |  | | --- | | x |  |  | | --- | | y |  |  | | --- | | 1 | | |  | | --- | | x |  |  | | --- | | y |  |  | | --- | | & | | |  | | --- | | x |  |  | | --- | | y |  |  | | --- | | 1 | | |  | | --- | | x |  |  | | --- | | y |  |  | | --- | | & | | |  | | --- | | x |  |  | | --- | | y |  |  | | --- | | M2 | |
| **Эквивалентность** | **Константа 1** |
| |  | | --- | | x |  |  | | --- | | y | | |  | | --- | | 1 | |

# **7 КРИПТОГРАФИЯ**

# **7.1 Алгоритмы факторизации**

1. **Алгоритм Евклида**

Является одним из первых алгоритмов, который был предложен математиком. Он позволяет вычислять НОД двух натуральных чисел. Его суть в том, что нужно вычитать из большего числа меньшее, занося результат на место большего. Действует до тех пор, пока не станут равны между собой. Таким образом при рассмотрении алгоритма Евклида, а именно НОД (а1, а2), порождает следующие равенства:

– остаток, который на последнем шаге равен нулю. В общем случае алгоритм Евклида требует меньше, чем целочисленных делений.

1. **Прямой перебор**

В трудах Чебышева, Эрмита и других была получена оценка для количества простых чисел, предшествующих числу x:

,

где - количество простых чисел.

Наименьший делитель числа n не превосходит **.**

При факторизации методом прямого перебора нужно делить на первых чисел, что практически невыполнимо. Этот алгоритм обычно является частью других, более сложных алгоритмов.

1. **Алгоритм Полларда**

Пусть p – минимальный простой сомножитель факторизуемого числа n. Пусть в результате случайного генерирования чисел удалось получить a и b такие, что , тогда p является делителем a-b и может быть получено значение р вычислением НОД (a-b, n). То есть факторизацию n можно рассматривать как получение двух чисел, a и b, дающих при делении на p одинаковые остатки.

По принципу Дирихле — это всегда можно сделать, породив разных чисел. Если задать некоторое число и оценить вероятность того, что случайно порождены r разных чисел, можно получить их все с разными остатками при делении на p. Эта вероятность зависит от r и p: (Феллер, 1964 г.)

От произведения нужно перейти к сумме:

1. **Алгоритм генерации псевдослучайных чисел**
2. В датчик вводится любое число
3. На k – том шаге, где k=1, 2, …, порождается число , где **p** – полином. По формуле порождается следующая последовательность: x1, x2,…,xk. Последнее полученное число xk нужно проверить на выполнение условия для всех ранее полученных **.**

Для выполнения проверки надо осуществить квадратичный перебор. Обойти его можно, воспользовавшись результатами теоремы:

Пусть порождены полиномиальным датчиком с и . Тогда для любого справедливо сравнение

**.**

Если , то **.** То есть для факторизации числа достаточно найти пару с нужной разностью индексов. Тогда – алгоритм Полларда можно представить так:

1. Порождается случайно ;
2. Сумматор **;**
3. **;**
4. Если k – нечетное, то выполняется пункт 2. Иначе пункт 5;
5. , **;**
6. Если НОД = 1, идти к 2;
7. Если НОД <n, то р=НОД. Конец;
8. Если НОД делится не только на Р, но и на n, то идти к пункту 1.

В 4 и 5 пункте алгоритма х2 сравнивается с х1, разность индексов 1; х4 сравнивается с х2, разность индексов 2; х6 сравнивается сх3, разность индексов 3. Алгоритм назван греческой буквой **ρ**, потому что, хотя последовательность хk бесконечная, но по модулю p она является периодической, образуя петлю в виде буквы . Данный алгоритм применяется при анализе донорской крови на редкий вирус проверяют не каждую порцию, а смесь, образованную по капле из каждой порции. Обычно результат анализа отрицательный, а если он положительный, то тогда уже индивидуально анализируются все представленные в смеси порции.

Аналогично можно поступить с разностями , получаемыми в ρ- алгоритме. Вместо того, чтобы для каждой разности считать НОД, формируется произведение . После того, как наберется v сомножителей, считается НОД (d, n). Обычным результатом будет НОД = 1, после чего нужно продолжать алгоритм и формировать новое d. При удачном исходе НОД = n, то нужно проанализировать все сомножители d, так как среди них может быть один, делящийся на p, и другой, делящийся на q.

Пример: Разложит 209, используя и

# **7.2 Выработка секретного ключа по Диффи – Хеллману**

В 1976 г. – выработка секретного ключа путем обмена информацией по открытому каналу без предварительного сговора. Это новый принцип шифровки, где умение зашифровывать не связано с умением расшифровывать, то есть шифрование открытое. Абонент электронной сети открыто сообщал о том, как зашифровать к нему сообщения, но расшифровать их он мог только сам. Это операция one – way functions (слить – разделить жидкости). В этом случае прямая операция достаточно проста, а обратная – очень сложна:

1. **Прямая операция**

Перемножаются два больших числа .

Обратная операция: разложить n на множители p и q

1. **Прямая операция**

Возвести основание **a** в степень **p** и взять остаток по модулю **m**:

.

Обратная операция: найти p, зная L, a, m.

Эти задачи почти не решаются, если p – очень велико.

**Алгоритм Диффи - Хеллмана**

1. Отправитель вырабатывает случайное число , вычисляет и посылает получателю;
2. Получатель получает , вырабатывает случайное число , вычисляет и посылает  отправителю;
3. Отправитель получает и вычисляет ;
4. Получатель вычисляет **.**

— это число является общим секретным ключом. Враг перехватил и, но не смог узнать x и y.

Пример: a=2, m=601, х=178

# **7.3 Датчики М – последовательностей**

**М – последовательности** – это линейные рекуррентные последовательности max периода, формируемые k – разрядными генераторами на основе регистров сдвига.

На каждом такте поступивший бит сдвигает k предыдущих, и к нему добавляется их сумма по модулю 2. Вытесняемый бит добавляется к гамме. Это можно представить в виде отношений:

где - k однобитных регистров,

- коэффициенты неприводимого двоичного полинома степени k-1,

- i–ое значение выходной гаммы.

Исходя из свойств М – последовательностей ее период равен 2k-1. Количество различных М-последовательностей для заданного k называется объемом**ансамбля**:

|  |  |
| --- | --- |
| k | Объем ансамбля |
| 5 | 6 |
| 6 | 8 |
| 7 | 18 |
| 8 | 16 |
| 9 | 48 |
| 10 | 60 |
| 16 | 2048 |

Такие объемы ансамблей на практике неприемлемы. Поэтому разрабатываются последовательности, которые получаются суммированием нескольких М–последовательностей. Например, при k = 10, V =60, тогда объем ансамбля резко возрастает (по Голду).

Таким образом существуют различные виды криптографических систем и различные алгоритмы их реализации. В частности, в линейном конгруэнтном датчике присутствует , т.е. речь идет об алгебре вычетов, входящей в теорию сравнений.

# **7.4 Конгруэнтные датчики**

Конгруэнтные генераторы псевдослучайных последовательностей (ПСП) максимально доступны и эффективны, при этом можно сделать математически строгое заключение о том, какими свойствами обладают выходные сигналы этих генераторов с точки зрения периодичности и случайности.

Одним из простых генераторов является линейный конгруэнтный датчик псевдослучайных чисел (ПСЧ). Он вырабатывает последовательности псевдослучайных чисел, описываемые соотношением где **А** и **С – const**; **T (0)** – исходная величина, выбранная в качестве порождающего числа. Эти три величины и образуют ключ. Такой датчик ПСЧ генерирует псевдослучайные числа с определенным периодом повторения, зависящим от выбранных значений **А** и **С**. Значение **m**, обычно устанавливается равным **2n**, где **n** – длина машинного слова в битах. Датчик имеет максимальный период **М** до того, как генерируемая последовательность начнет повторяться.

Дональд Кнут доказал, что лучший датчик ПСЧ имеет максимальную длину **М** тогда, когда **С** – нечетное, а Для шифрования данных с помощью датчика ПСЧ может быть выбран ключ любого размера.

Например, пусть ключ состоит из набора чисел **хj** размерностью **b**, где **j** меняется от **1** до **n**. Тогда создаваемую гамму шифра **G** можно представить как объединение непересекающихся множеств **H(j)**. **Гамма шифров** – это некоторая последовательность символов, используемых для шифрования.

# **7.5 Элементы алгебры вычетов**

Алгебра вычетов реализуется в ЭВМ и является предметной областью компьютерной алгебры, которая в свою очередь базируется на теории чисел. Алгебра вычетов была реализована в простейшем цикле Цезаря.

Пусть **m** – целое число **(m>1)** и дано — отношение эквивалентности на множестве **Z**. Говорят, что для некоторого . Можно говорить о классе эквивалентности элемента **а**. Это множество:

Получившееся множество можно обозначить через или Пусть в качестве примера выбирается **m=5**, тогда при , . Таким образом множество **Z** разбивается на непересекающиеся подмножества: Фактор-множество – это , т.е. каждый класс эквивалентности содержит бесконечно много элементов, а множество классов эквивалентности содержит всего 5элементов. В общем случае . — это множество вычетов, или множество целых чисел по модулю **m**.

# **7.6 Элементы криптографии**

Криптография, как наука, стала развиваться еще в древние времена, так как нужно было защищать информацию от прочтения. В начале н.э. криптография возникла как самостоятельная наука. До наших дней дошел систематический шифр, получивший имя Цезаря. Цезарь Гай Юлий (102 или 100 – 44г.г. до н.э.) – римский диктатор, который ввел с 1.01.45г. юлианский календарь. Он отстает от природных явлений примерно на одни сутки за 128 лет.

Развитие вычислительных средств после Второй мировой войны ускорило разработку и совершенствование криптографических методов. Проблемой защиты информации путем ее преобразования занимается **криптология**(от греческого криптос – тайный; логос – наука). Криптология разделяется на два направления:

1. Криптография;
2. Криптоанализ.

**Криптография** – занимается поиском и исследованием методов преобразования информации с целью скрытия ее содержания.

**Криптоанализ** – исследует возможность расшифровывания информации без знания ключей.

Криптографические методы делятся на четыре класса:

1. Симметричные криптосистемы;
2. Криптосистемы с открытым ключом;
3. Системы электронной цифровой подписи (ЭЦП);
4. Системы управления ключами.

Все методы используются для:

1. Передачи конфиденциальной информации по каналам связи (например, электронная почта);
2. Установления подлинности передаваемых сообщений;
3. Хранения информации на носителях в зашифрованном виде.

Криптография позволяет преобразовать информацию так, чтобы прочитать (восстановить) ее можно было бы только при знании ключа.

**Информация** – это тексты, построенные на некотором алфавите.

**Шифрование** – процесс преобразования исходного текста (открытого) в шифрованный. Обратный процесс – **расшифрование***,* то есть на основе ключа шифрованный текст преобразуется в исходный.

**Криптографические системы** *–* это семейство преобразований Т открытого текста. Элементы этого семейства k называются **ключами**. Преобразование Т определяется соответствующим алгоритмом и значением ключа k.

**Ключ** – информация для беспрепятственного шифрования и расшифрования текстов.

**Пространство ключей К** *–* это набор возможных значений ключа. Обычно ключ – это последовательный ряд букв алфавита.

В симметричных криптосистемах для шифрования и для расшифрования используется один и тот же ключ. До начала использования системы нужно получить этот ключ так, чтобы исключить к нему доступ потенциального злоумышленника.

В системах с открытым ключом используется два ключа – открытый и закрытый (секретный). Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

**ЭЦП** – это присоединяемое к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователям проверить авторство и подлинность сообщения.

Процесс криптографического закрытия данных осуществляется как программно, так и аппаратно. Аппаратная реализация значительно дороже, но отличается высокой производительностью, защищенностью, простотой. Программная реализацияболее практична, допускает гибкость в использовании.

Для реализации эффективности криптографических систем недостаточно взять за основу хорошо изученный криптографический алгоритм и написать программу, или построить аппаратную реализацию. Нужно установить критерии эффективности, которыми, в частности, могут быть минимизация вычислительных ресурсов для реализации алгоритма, отсутствие уязвимых мест в реализации.

В реализации криптосистем одной из базовых проблем является проблема генерации случайных чисел, так, как только абсолютно случайное число может являться надежным ключом. Например, в ряде ЭВМ она реализована аппаратно и проводится соответствующей машинной командой. В других случаях используются генераторы псевдослучайных чисел, реализованные функциями в языках высокого уровня.

Рекуррентные генераторы формируют последовательности чисел, где каждый элемент зависит от одного или нескольких предыдущих, то есть вся последовательность зависит от начального числа, которое должно быть случайным. Если злоумышленник узнал исходное число и рекуррентное соотношение, то он может получить любой элемент – это недостаток таких генераторов. Достоинство состоит в том, что, синхронизировав исходные значения генераторов один раз, абоненты могут менять ключи, которые будут общими для абонентов, но недоступными для постороннего наблюдателя.

# **8 МАШИННАЯ МАТЕМАТИКА**

# **8.1 Элементы теории алгоритмов**

Понятие алгоритма является одним из основных понятий математики. Название произошло от латинизированного написания фамилий ученого Аль – Хорезми. Он работал более 12 веков назад и стал известен в Европе благодаря исследованиям в таких областях как алгебра, астрономия, естествознание, история.

На протяжение многих веков люди пользовались интуитивным понятием алгоритма, которое можно сформировать так: **алгоритм** – это строгая и четкая конечная система правил, которая определяет последовательность действий над некоторыми объектами и после конечного числа шагов приводит к достижению поставленной цели. **Система правил** – это инструкция, по которой разные люди будут действовать одинаково.

Одним из первых алгоритмов считается алгоритм древнегреческого математика Евклида, для вычисления НОД двух натуральных чисел.

**Алгоритм вычисления НОД двух натуральных чисел**

Алгоритм формулируется так:

Нужно вычитать из большего числа меньшее, занося результат на место большего до тех пор, пока числа не станут равны между собой – это и будет НОД.

Ниже представлена блок схема алгоритма вычисления НОД.

**Любая блок схема** – это модель алгоритма, которая определяет и его структуру, и функции, так как каждый блок означает соответствующее действие.

В блоке процесса указаны операторы присвоения, которые соответствуют типовым алгоритмическим структурам, а именно сумматором на вычитание.

**Сумматор** – это оператор присвоения, в левой и правой части которого находится одна и та же переменная.

Алгоритм поиска НОД реализован во многих криптографических системах.

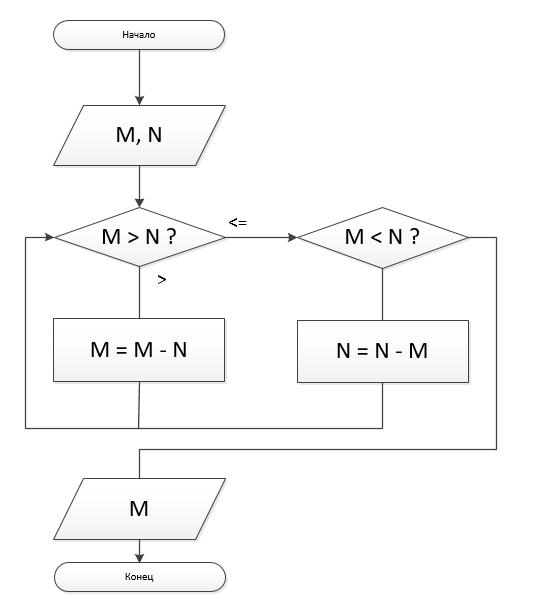


Рисунок 1 - Блок схема алгоритма

Лейбниц в XVII веке сформулировал свою идею нахождения общего алгоритма решения любых математических задач. В XX в. его идея была сформулирована так:

Найти алгоритм проверки правильности любой теоремы при любой системе аксиом, т.е. алгоритм который отвечал бы на вопрос: Верна ли теорема? И если да – то давал бы вывод её доказательства.

Доказать невозможность решения таких задач было нельзя, так как не было строгого определения алгоритма.

Поэтому возникла проблема построения формального определения алгоритма. Для чего нужно было определить понятие объекта.

**Алгоритм сложения**

15 + 31 = 52

Он работает над числовыми объектами 15 и 37 и вырабатывает числовой результат 52.

Можно считать, что объектом алгоритма является изображение, состоящее из пяти знаков: 1, 5, +, 3, 7.

Результат: 5, 2 – изображение, состоящее из двух знаков.

**Знаки** — это буквы, а набор знаков — это алфавит:

**Алфавит** – это конечная совокупность различных букв. Любая конечная последовательность букв называется, **словом***,* в данном алфавите. **Знаки** – это буквы, **набор** – алфавит.

Таким образом алгоритм сложения перерабатывает слово, состоящее из изображений слагаемых, разделенных знаком плюс, в слово изображающее сумму.

Слово, к которому применяется алгоритм, называется **входным**. Слово, вырабатываемое в результате применения алгоритма, называется **выходным**.

Если в алфавит включить символ пробел, а исключить знаки (**. , ?, !** ), то словом будет слово в соответствующем языке. А если включить в алфавит и эти знаки, то, словом, будет текст.

Формальное определение алгоритма стали вырабатывать различные ученые в 30-40 гг. 20 века.

Алгоритм – это четкая, конечная система правил, для преобразования слов из некоторого алфавита в слова из этого же алфавита.

Замена любого алфавита другим называется кодировкой.

Количество букв в слове называется его длиной. Если в слове нет букв, то это пустое слово **λ**.

Совокупность слов, к которым алгоритм применяется называется **областью применимости этого алгоритма**.

# **8.2 Машина Тьюринга**

Формальные определения алгоритма появились в 30-40г.г. ХХ века. В 1936 году английский математик Алан Тьюринг (годы жизни: 1912-1954) описал схему гипотетической (абстрактной) машины, и предложил назвать алгоритмом то, что умеет делать такая машина.

В машине Тьюринга реализуется бесконечная лента, поэтому она считается абстрактной или гипотетической, где ЗУ (запоминающее устройство) бесконечно. В ленте, в каждой ее ячейке, может находиться только одна буква, какого-либо алфавита. В пустой – специальная буква **λ**.

Для машины Тьюринга программа представляет собой таблицу, где строки указывают на состояние автомата, а столбцы соответствую буквам.

Автомат за каждый шаг видит одну букву. Входное слово с обеих сторон заключается в пустые ячейки.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| λ | λ | **о** | **т** | **в** | **е** | **т** | λ | λ |

автомат

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| λ | **з** | **а** | **д** | **а** | **н** | **и** | **е** | λ |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | λ | S1 | S2 | … | Si | … | Sl |
| q1 |  |  |  |  |  |  |  |
| q2 |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |
| qj |  |  |  |  |  |  |  |
| … |  |  |  |  |  |  |  |
| qk |  |  |  |  |  |  |  |

Автомат, находясь в состоянии qj и видя букву Si записывает вместо нее букву Sl, которая в частном случае может совпасть с Si.

Далее автомат смещается влево (если л), вправо (если п), или остается неподвижным (если н). Затем переходит в состояние qk, т.е. следующую команду нужно искать в к-той строке таблицы.

Кроме ленты, у машины Тьюринга есть автомат, который может двигаться вдоль ленты и по очереди «обозревать» содержимое ячеек. Входное слово размещается на ленте по одной букве в расположенных подряд ячейках и занимает конечное число ячеек. Слева и справа от входного слова на ленте находятся только пустые слова:

Тьюринг доказал, что алгоритмом можно считать все то, что может реализовать машина Тьюринга.

В процессе работы машина будет «перескакивать» из одной клетки программы в другую, в соответствии с информацией на ленте и указаниями команды, пока не дойдет до клетки, где будет указана **н**. Тогда алгоритм завершит свою работу.

Здесь, на ленте, каждый раз одна буква заменяется на другую, какие-то буквы стираются, а в каких-то пустых ячейках появляются новые буквы, т.е. меняется длина выходного слова.

Если в программе нет клеток останова, то машина никогда не прекратит работу, но даже если такая команда есть машина может не дойти до нее, так как переходы зависят от программы и от входного слова.

Если машина Тьюринга никогда не остановится, то считается, что **она неприменима** к данному входному слову.

Даже при наличии клеток останова в нижних строках программы машина неприменима ни к одному слову, если входная строка имеет вид:

|  |  |  |  |
| --- | --- | --- | --- |
|  | λ | 0 | 1 |
| q1 | λ, п, q1 | о, п, q1 | 1, п, q1 |

Здесь автомат ничего не меняет, чтобы он не увидел, и все время смещается вправо, оставаясь в состоянии q1. А так как лента бесконечна, он никогда не остановится. Алгоритм неприменим.

Алгоритм работает, когда автомат находится против самой левой непустой ячейки и шаг за шагом стирает содержимое непустых клеток.

Таким образом машина Тьюринга будет применима ко всем словам этого алфавита.

Описывая различные алгоритмы и доказывая реализуемость всевозможных композиций алгоритмов Тьюринг показал разнообразие возможностей предложенных им конструкций.

Тьюринг выдвинул тезис: **Всякий алгоритм может быть реализован соответствующей машиной Тьюринга**.

Это основная гипотеза теории алгоритмов в форме Тьюринга. А также этот тезис является формальным определением алгоритма.

Доказать тезис Тьюринга нельзя, так как в его формулировке не определено понятие «всякий алгоритм», т.е. левая часть тождества. Его можно только обосновать, представляя известные алгоритмы в виде машин Тьюринга.

Позднее было предложено еще несколько общих определений алгоритма и каждый раз доказывалось, что эти схемы эквиваленты машинам Тьюринга.

# **8.3 Нормальные алгоритмы Маркова**

В 1954г. советский математик Марков предложил свою алгоритмическую схему преобразования слов. Здесь нет ленты, а есть непосредственный доступ к различным частям преобразуемого слова. Марков назвал эту схему нормальным алгоритмом. Это упорядоченный набор пар слов, соединенных между собой формулой подстановки .

Каждый такт алгоритма включает в себя поиск первой по порядку применимой формулы подстановки и применение этой формулы. Например:

Первый такт:

Проверка есть ли вхождение слова **А1** в исходное слово. Если есть, то оно заменяется на слово **B1**.

Второй такт определяется вхождением левой части **А1** уже в измененное слово. Если нет вхождения, то первый такт продолжается рассмотрением второй пары. Иначе, заменяется **А1** на **B1** слева на право.

Пример: составить нормальную формулу алгоритма кодировки слов из алфавита алфавитом .

Частный случай, это слово из одной буквы.

а = 101

b = 1001

с = 10001

Слово: c a a b

1. c 101 a b
2. c 101 101 b
3. c 101 101 1001
4. 10001 101 101 1001

Процесс выполнения нормального алгоритма заканчивается, если:

1. Все формулы оказались не применимыми, т.е. в обрабатываемом слове нет вхождения ни одной левой части в какой-либо формуле подстановке.
2. Применилась завершающая формула.

Если в процессе выполнения нормального алгоритма бесконечное число раз применяются не завершающие формулы, то алгоритм неприменим к данному входному слову.

Здесь нет обозначения пустого слова, нет ячеек, не зафиксирован носитель информации, а преобразуемое слово свободно меняет свои размеры.

**Итоги**

Переход от других способов описания алгоритмов к эквивалентным нормальным алгоритма называется **представлением в нормальной форме**, или **нормализацией**.

Основная гипотеза теории алгоритмов в форме Маркова: **Всякий алгоритм нормализуем**.

Алгоритмы Маркова и Тьюринга эквивалентны в том смысле, что все алгоритмы одной из схем могут быть описаны в другой схеме.

У машин Тьюринга развито управление, при слабых возможностях преобразования информации (замена буквы на букву).

В схеме Маркова больше возможностей преобразования, слово меняется на слово, при менее развитом управлении.

# **8.4 Схема Поста**

Почти одновременно с Аланом Тьюрингом английский математик Пост предложил свою алгоритмическую схему. В ячейке бесконечной ленты можно записывать только два знака **0** и **1**. Каждому состоянию соответствует не строка программы с ячейками (клетками), а некоторая команда из шести возможных:

1. записать **1**, перейти к **i**;
2. записать **0,** перейти к **i**;

Они устанавливают новое содержимое обозреваемой ячейки, оставляя ее в «поле зрения» машины.

1. сдвиг влево, перейти к **i**;
2. сдвиг вправо, перейти к **i**;
3. остановка;
4. если **1**, то перейти к **i**, иначе – перейти к **j**.

Анализ этой схемы показывает, что в рамках бинарного алфавита каждое состояние машины Тьюринга описывается несколькими состояниями машины Поста.

В теории алгоритмов известны задачи, о которых доказано, что для их решения не существует алгоритма, т.е. они алгоритмически неразрешимы.

# 

# **СЛОВАРЬ**

**Дискретная математика** – это наука, изучающая дискретные множества.

**Множество** можно определить как совокупность различаемых между собой объектов, объединенных общим свойством.

Количество элементов называется **мощностью множества**.

Множество всех подмножеств, класс множеств называется **булеаном**.

**Носитель** – это объект системы.

**Сигнатура** – связи между объектами.

**СУБД** – система управления базами данных.

Строки называются **кортежами**.

Столбцы – **доменами**.

**Операция выбора** – процедура построения «горизонтального» подмножества отношений, или подмножества кортежей, обладающих заданным свойством.

**Операция проекции** – определяет построение «вертикального» подмножества отношений, т.е. Из кортежей удаляются координаты, соответствующие не выделенным доменам.

**Операция соединения** – предполагает работу с двумя таблицами, имеющими общий домен.

Декартово произведение множеств называется **прямым произведением**.

Если формируется бинарное отношение вида **R = A2**, то такое отношение называется **полным, или универсальным**.

**Комбинаторика** (от лат. Соединять, сочетать) – это раздел дискретной математики, изучающий приемы вычисления количества различных соединений, или комбинаций.

Комбинация из k элементов называется **k –** **расстановкой**

**Сигнатура, или язык сигма** – это совокупность предикатных **(Р)** и функциональных символов с указанием их местности **(А0 = Ø)**.

**Ноль местный функциональный символ** – константный символ, или const.

**Предикаты и функции** называются их интерпретациями соответствующих символов из сигмы.

**Мощность алгебраической системы** **(U)** определяется мощностью носителя .

**Изоморфизм** – строгое соответствие между системами;

**Гомоморфизм** – в системах совпадают некоторые существенные свойства.

**Базис** – это полная система, не содержащая лишних функций, т.к. Удаление любой функции из базиса нарушает полноту системы.

Множество вида , называется первичным термом.

Множество вида Называется **конституэнтой**.

Пересечение попарно различных множеств Называется **элементарным**.

Выражение, задающее множество В виде объединения различных элементарных пересечений, называется **нормальной формой Кантора** (НФК) множества M.

Объединение конституэнт множества **M** называется совершенной НФК множества M.

Пересечение, соответствующее максимальному интервалу множества M, называется простой **импликантой** этого множества.

Объединение простых импликант множества **M** называется сокращенной НФК множества M.

Количество первичных термов, образующих простую импликанту, называется **рангом простой** **импликанты**.

Элементарное пересечение – **рангом соответствующего интервала**.

Функция **Р**, принимающая одно из значений, 0 или 1, аргументы которой принимают значения из произвольного множества **М**, называется **предикатом Р в предметной области М**.

Число аргументов предиката называется его порядком.

Систематическая классификация и формализация правильных способов рассуждения **– основная задача математической логики**.

**Математика** – дедуктивная наука, то есть истинность математических рассуждений не может быть проверена путем опыта или наблюдений, а выводится с помощью логических рассуждений, исходя из небольшого количества исходных утверждений.

**Высказывание** – это некоторое предложение, которому уместно задать вопрос: истинно оно или ложно**.**

Имеется подмножество выражений теории, которые называются формулами

**Выводом** называется всякая последовательность формул , где Является либо аксиомой, либо непосредственным следствием из предыдущих формул по одному из правил вывода.

**Характеристическом классе** – это множество предметов, на которых предикат принимает значение **true**.

Поэтому **силлогистика** называется исчислением одноместных предикатов, или исчислением классов, или теорий категорических суждений.

Любое высказывание о классе называется **категорическим** **суждением**.

Множество сечений, взятых для всех элементов множества М при задании в нем отношения Т называется **фактор-множеством множества М** по отношению к **Т**.

Если ребра и вершины имеют некоторые значения функций или значения некоторых характеристик: (длина, время, ресурсы), то такие графы называются взвешенными и называются в моделировании **сетями**.

**Степень** — это число ребер, которым инцидентна вершина **V**.

**Род поверхности** – это максимальное число простых замкнутых кривых на поверхности, которые не разъединяют эту поверхность.

**Род графа** – это минимальный род среди всех поверхностей, на которых граф можно изобразить так, чтобы его ребра пересекались только в вершинах.

Таким образом **планарным** называется граф, который можно представить на плоскости.

**Полным** называется граф, содержащий все возможные ребра, т.е. Любая вершина достижима из любой другой вершины, т.е. Имеются маршруты(цепи), соединяющие любую пару вершин

**Двудольный граф** – это граф, множество вершин которого можно разбить на два непересекающихся подмножества так, что каждое ребро графа соединяет две вершины из разных подмножеств.

**Внутренняя грань** – это конечная область плоскости, ограниченная замкнутым маршрутом, не содержащая ни вершин, ни ребер этого графа.

**Граница грани** – это минимальный маршрут.

**Связностью графа** әе(G) называется наименьшее число вершин, удаление которых делает граф несвязным, или тривиальным.

Простые цепи называются **реберно-непересекающимися**, если никакие две из них не имеют общего ребра.

Минимальная длина цепи, соединяющая две его вершины, называется **расстоянием** **между** **вершинами**:

Максимальное расстояние между вершинами называется **диаметром графа**

Любой замкнутый маршрут называется **циклом**.

Цикл называется **эйлеровым**, если каждое ребро графа участвует в его образовании только один раз

Простой цикл называется **гамильтоновым**, если проходит через каждую вершину графа.

Множество всех векторов, каждый из которых соответствует одному циклу графа, образует **векторное** **пространство**, или **пространство** **циклов**.

**Остов** — это остовный подграф, являющийся деревом.

**Остовный** **подграф** - подграф, содержащий все вершины графа.

Дерево - связный граф, не содержащий ни одного цикла.

Хорда остова дерева в связном графе G — это ребро графа, не принадлежащее дереву.

**Решеткой** – это упорядоченное множество любые два элемента которого **mi, mj** имеют наибольшую нижнюю грань, или пересечение и наименьшую верхнюю грань, или объединение.

Элемент, покрывающий ноль в частично упорядоченном множестве **M**, называется **атомом** или **точкой**.

Если множество мажорант имеет максимальный элемент, то он называется **верхней гранью множества** (sup M).

Если множество минорант имеет минимальный элемент, то он называется **нижней гранью множества** (inf М).

**Лес** – это граф, не содержащий циклов и состоящий из нескольких компонент связности (деревьев). У дерева всегда есть одна вершина, называемая корнем.

В программировании левую ветвь бинарного дерева называют левым сыном, правую – **правым** **сыном**.

**Криптография** – занимается поиском и исследованием методов преобразования информации с целью скрытия ее содержания.

**Криптоанализ** – исследует возможность расшифровывания информации без знания ключей.

**Информация** – это тексты, построенные на некотором алфавите.

**Шифрование** – процесс преобразования исходного текста (открытого) в шифрованный.

Обратный процесс – **расшифрование**, то есть на основе ключа шифрованный текст преобразуется в исходный.

**Криптографические** **системы** – это семейство преобразований Т открытого текста.

**Ключ** – информация для беспрепятственного шифрования и расшифрования текстов.

**Пространство ключей К** – это набор возможных значений ключа.

**ЭЦП** – это присоединяемое к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователям проверить авторство и подлинность сообщения.

**Гамма шифров** – это некоторая последовательность символов, используемых для шифрования.

**М – последовательности** – это линейные рекуррентные последовательности max периода, формируемые **k** – разрядными генераторами на основе регистров сдвига.

Количество различных **М**-последовательностей для заданного **k** называется объемом **ансамбля**.

**Алгоритм** – это строгая и четкая конечная система правил, которая определяет последовательность действий над некоторыми объектами.

**Система** **правил** – это инструкция, по которой разные люди будут действовать одинаково.

**Алфавит** – это конечная совокупность различных букв.

Любая конечная последовательность букв называется, **словом,** в данном алфавите. **Знаки** – это буквы.

**Набор** – алфавит.

Слово, к которому применяется алгоритм, называется **входным**.

Слово, вырабатываемое в результате применения алгоритма, называется **выходным**.

Множество может не содержать ни одного элемента, тогда оно называется **пустым** **множеством**.

Множества могут быть **равномерными**, т.е. Состоять из одних и тех же элементов. Множества могут быть **непересекающимися**, т.е. Не иметь общих элементов.

**Объединение (U)** - Формируется новое множество, содержащее все элементы, принадлежащие или одному, или другому или обоим сразу.