Министерство образования и науки Российской Федерации

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-вычислительных систем (КИБЭВС)

Е.М. Давыдова, Р.В. Мещеряков

ДИСКРЕТНАЯ МАТЕМАТИКА

Учебное пособие

2004

Корректор: Осипова Е.А.

Давыдова Е.М., Мещеряков Р.В.
Дискретная математика: Учебное пособие. – Томск: Томский межвузовский центр дистанционного образования, 2004. – 181 с.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящее учебное пособие преследует несколько целей, а именно:

- ознакомить учащегося с широким кругом понятий дискретной математики;
- позволить овладеть методами дискретной математики, наиболее употребительными при решении практических задач;
- предоставит к изучению ряд алгоритмов для решения типовых задач;
- сформировать абстрактное мышление, без которого невозможно решение проблем информатизации.

Дискретная математика зародилась в глубокой древности и включает в себя многие разделы математики, такие как теория множеств, математическая логика, теория чисел, алгебра, теория графов и сетей и т.д. Наиболее интенсивно дискретная математика стала развиваться с появлением ЭВМ.

Пособие предназначено для студентов специальностей 2205 (проектирование и технология ЭВС) и 0755 (комплексная информационная безопасность автоматизированных систем).

1 ВВЕДЕНИЕ В ТЕОРИЮ МНОЖЕСТВ

Изучение дискретной математики начинается с изучения понятий «множество», «отношение», «функция», поскольку с помощью этих понятий строятся математические дисциплины. Любое понятие дискретной математики также можно определить с помощью множества.

Понятию «множество» сложно дать точное определение, поэтому строгого определения нет.

Под ***множеством*** понимается совокупность объектов, обладающих определенными свойствами. В данном случае определение «множество» дается через свойства его же элементов.

Бурбаки дают следующее ***определение понятия «множество»:*** множество строится из некоторых элементов, обладающих определенными свойствами и находящихся в каких-то отношениях между собой и с элементами других множеств.

Объекты, образующие множество, называются ***элементами*** множества и обозначаются малыми буквами латинского алфавита **a, b, c, d, …, x, y, z**. Большими буквами латинского алфавита обозначаются сами множества. Если элемент **m** принадлежит множеству **M**, то это записывают как $\mathbf{m} \in \mathbf{M}$, в противном случае $\mathbf{m} \notin \mathbf{M}$ (элемент **m** не принадлежит множеству **M).** Принадлежность нескольких элементов может быть записана $\mathbf{a} \in \mathbf{B}$, $\mathbf{b} \in \mathbf{B}$, $\mathbf{c} \in \mathbf{B}$ или $\mathbf{a,b,c} \in \mathbf{B}$. Множество может содержать любое количество элементов: счетное, бесконечное, конечное, один элемент, ни одного элемента.

Множество, содержащее конечное число элементов, называется ***конечным***. Если же множество не содержит ни одного элемента, то оно называется ***пустым*** множеством и обозначается $\varnothing$ или **{ }**.

Все элементы множества должны отличаться один от другого. Поэтому каждый элемент может входить в множество только один раз. Количество элементов множества называется ***мощностью***.

Если число элементов множества А конечно, то такое множество называют ***конечным*** множеством, в противном случае его называют ***бесконечным*** множеством.

Множество, имеющее мощность, равную единице, называют ***синглетоном.***

Остановимся на способе перечисления бесконечных множеств. Пусть N – множество натуральных чисел. Заметим, что способ перечисления его элементов очевиден: 1, 2, 3, …

Счетным множеством называется множество, равномощное с множеством натуральных чисел.

Бесконечное множество счетно, если его можно выстроить в цепочку. В качестве примера рассмотрим множество целых чисел. Цепочка будет выглядеть следующим образом: 0, +1, –1, +2, –2, +3, –3, …. Любой элемент из множества целых чисел попадает в эту последовательность и любому элементу последовательности можно поставить в соответствие его номер, т.о., множество целых чисел счетно.

Множество называется ***полностью определенным***, если о каждом предмете можно сказать, принадлежит он множеству или нет.

Множество может быть задано различными способами, но наибольшее распространение получили два способа задания.

1. Путем прямого перечисления его элементов, которые записываются через запятую внутри фигурных скобок. Например,

D = {понедельник, вторник, среда, четверг, пятница, суббота, воскресенье};

A = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

При этом порядок перечисления элементов множества не важен. Так, множества {1, 7, 3}и {1, 3, 7} совпадают. Таким способом можно задавать конечные множества, содержащие небольшое количество элементов.

2. С помощью характеристического свойства, которым должен обладать каждый объект, чтобы стать элементом множества. Записывается следующим образом:

A = {a | a обладает свойством Q}, т.е. **A** – множество элементов a таких, что они обладают свойством **Q**. Предыдущие примеры записываются:

D = {x | x – день недели }, A = {a | a целое, $0 \le a \le 9$}.

Этим способом можно задать множество, содержащее бесконечное количество элементов, т.е. бесконечное множество.

Иногда бесконечное множество можно задать просто перечислением нескольких первых элементов, и тогда характеристическое свойство оказывается заданным в неявном виде. Например, множество нечетных чисел можно задать так: A = {1, 3, 5, 7,…}.

Отношение равенства. Два множества равны, если все их элементы совпадают. Доказывается это утверждение в два этапа:

1. Каждый элемент множества **A** является элементом множества **B**, т.е., если $x \in A$, то $x \in B$.

2. Обратное утверждение: каждый элемент множества **B** является элементом множества **A**, т.е., если $x \in B$, то $x \in A$.

Отношение равенства обладает свойством транзитивности. Если **A=B** и **B=C**, то **A=C**.

Отношение включения. Множество **A** называется подмножеством **B** (**A** включено в **B**) тогда и только тогда, когда любой элемент множества **A** принадлежит множеству **B**:

$A \subset B \leftrightarrow (a \in A \rightarrow a \in B)$ или

$A \subset B \leftrightarrow A = \{ a \mid a \in B \}$;

Здесь $\subset$ – знак включения подмножества;

$a \rightarrow b$ означает: если **a** то **b**;

$a \leftrightarrow b$ означает: **b**, если и только если **a**.

Если хотя бы один элемент множества **A** не является элементом множества **B**, то множество **A** не является подмножеством **B** и это записывается следующим образом: $A \not\subset B$.

В соответствии с определением подмножества пустое множество является подмножеством любого множества, так как все его элементы (а у него их нет) являются элементами любого множества и любое множество является своим подмножеством: $A \subseteq A$ и $\varnothing \subseteq \varnothing$. $\varnothing \subset A$ для любого множества A.

Число всевозможных подмножеств любого конечного множества, содержащего **n** элементов, равно 2^n.

Например, пусть задано множество A = {1, 2, 3}, число подмножеств у него восемь:

∅, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}. Из них ∅ и само множество A – ***несобственные*** подмножества, все остальные подмножества – ***собственные***. Запишем элементы заданного множества A в каком-либо порядке.

№	1	2	3	подмножества
0	0	0	0	∅
1	0	0	1	{3}
2	0	1	0	{2}
3	0	1	1	{2, 3}
4	1	0	0	{1}
5	1	0	1	{1,3}
6	1	1	0	{1, 2}
7	1	1	1	{1, 2, 3}

Каждому элементу поставим в соответствие 0, если элемент отсутствует, и 1 – если элемент присутствует. Для каждого множества A существует множество, элементами которого являются подмножества множества A, и только они. Такое множество будем называть семейством множества A или ***булеаном*** этого множества и обозначать B(A). Множество A будем называть ***универсальным*** (универсумом) и обозначать I (в другой литературе его обозначают как T, U, 1).

Примеры правильных записей:

1) $A = \{1, 2, 3, 4\}$; $1 \in A$; $3 \in A$; $\{1\ 3\} \notin A$; $\{1,3\} \subset A$.
2) $B = \{1, 2, \{3\}\}$; $1 \in B$; $3 \notin B$; $\{3\} \in B$; $\{1, 2\} \subset B$; $\{2\} \subset B$; $\{3\} \not\subset B$; $\{\ \{3\}\ \} \subset B$.
3) $C = \{1, 2, \{1, 2\}\}$; $\{1, 2\} \in C$; $\{1, 2\} \subset C$.
4) $D = \{\varnothing, \{\varnothing\}\}$; $\varnothing \in D$; $\{\varnothing\} \in D$; $\varnothing \subset D$; $\{\varnothing\} \subset D$.

1.1 Операции над множествами

Основными операциями над множествами являются: объединение – $\cup$, пересечение – $\cap$, вычитание (разность) – \, сумма – $\oplus$ и унарная операция дополнение – $\neg$.

Операция объединения – $\cup$. Объединением двух множеств **A** и **B** называется такое множество **C**, элементы которого состоят из элементов множества **A** и из элементов множества **B**.

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

Пусть $C = A \cup B$, тогда, если $x \in C$, то $x \in A$ и/или $x \in B$.

Пример:

Пусть заданы множества A = {0, 1, 2, 3, 4} и B = {3, 4, 5, 6}. Тогда $A \cup B = C = \{0, 1, 2, 3, 4, 5, 6\}$.

Операция пересечения – ∩. Множество **C** есть пересечение множеств **A** и **B**, если каждый элемент множества **C** является элементом **A** и **B** одновременно.

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

Союз «и» заменяют часто знаком &.

Если $x \in C$, то $x \in A$ & $x \in B$.

Пример:

Если A = {0, 1, 2, 3, 4}, B = {3, 4, 5, 6}, то C = {3, 4}.

Операция разность – \. Разностью множеств **A** и **B** называется множество **C**, элементы которого принадлежат **A**, но не принадлежат **B**.

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

Если $x \in C$, то $x \in A$ и $x \notin B$.

Для предыдущего примера $C = A \setminus B = \{0, 1, 2\}$; $C' = B \setminus A = \{5, 6\}$.

Операция сумма – ⊕ (симметрическая разность).

$C = A \oplus B = (A \setminus B) \cup (B \setminus A)$.

Если $x \in C$, то x является элементом разности A и B или элементом разности B и A.

$C = A \oplus B = \{x \mid x \in A / B$ или $x \in B \setminus A\}$.

То есть, если A={1,2,3,4,5}, B={3,4,5,6,7}, тогда C=A⊕B= ={1,2,6,7}.

Операция дополнение (одноместная операция). Дополнение A обозначается **¬A**, **$\bar{A}$**, **A'**, содержит все те элементы универсального множества I, которые не принадлежат A.

$C = \bar{A} = I \setminus A$.

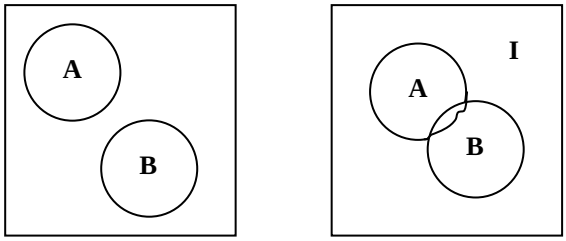
Пусть I = {0,1,2,3,4,5,6,7,8,9}. A={3,8,5,7,0}. Тогда $\bar{A}$={1, 2, 4, 6, 9}.

1.2 Диаграммы Эйлера-Венна

Возможно графическое представление множеств. Универсальное множество задается в виде квадрата, а множества A и B как множества точек плоскости, ограниченные соответствующи-

ми замкнутыми линиями. Например: на рисунке 1.1 изображены непересекающиеся (а) и пересекающиеся (б) множества A и B. На рисунке 1.2. показано отношение включения $A \subset B$.

Следующие рисунки демонстрируют результаты выполнения операций над множествами (показаны заштрихованной областью). Диаграммы, приведенные на рисунке 1.3, демонстрируют объединение множеств A и B.



а) б)

Рисунок 1.1 – Пример множеств

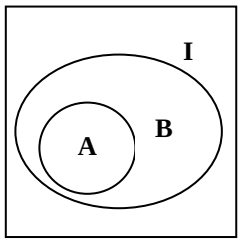


Рисунок 1.2 $A \subset B$

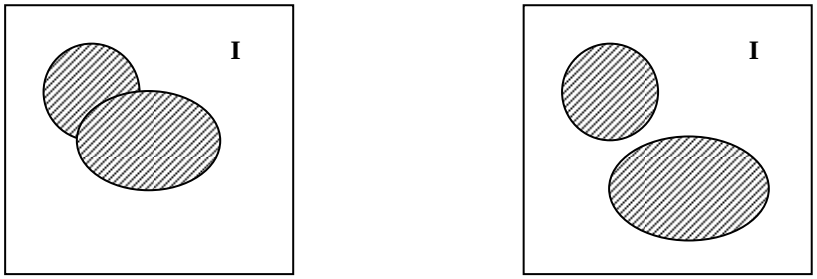
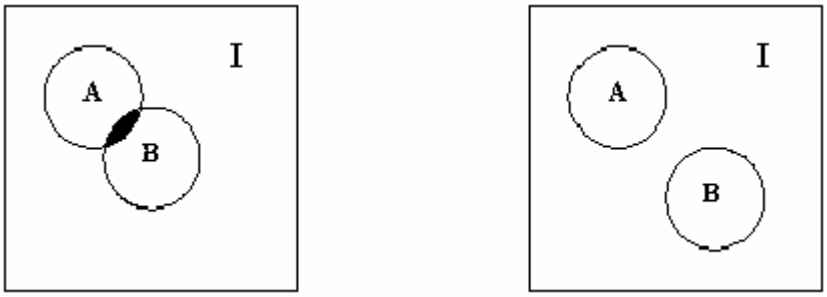


Рисунок 1.3 – Объединение множеств $A \cup B$

На рисунке 1.4 приведены примеры пересечения. На рисунке 1.4, а приведены множества, имеющие одинаковые элементы, их пересечение $A \cap B \neq \varnothing$ и случай 1.4 б множества не имеют общих элементов, и их пересечение $A \cap B = \varnothing$.



а) б)

Рисунок 1.4 – Пересечение $A \cap B$

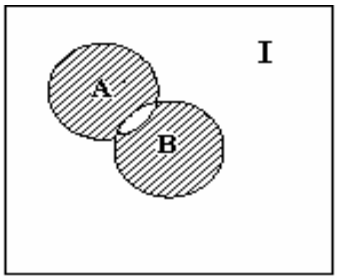


Рисунок 1.5 – Сумма A ⊕ B

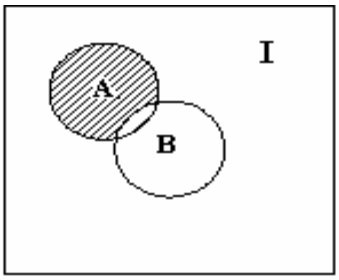


Рисунок 1.6 – Разность A \ B

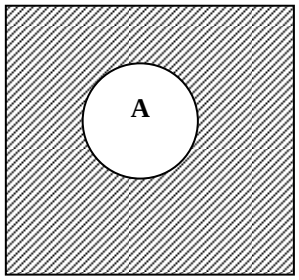


Рисунок 1.7 – Дополнение Ā

1.3 Понятие алгебры

Алгеброй A называется совокупность множества M с заданными в нем операциями:

$$S = \{f_1, f_2, \ldots, f_{m1}, f_{m2}, \ldots, f_{m,nm}\},$$

A = < M, S >, здесь множество M – носитель, S – сигнатура алгебры. Нижний индекс у идентификатора операции указывает ее местность.

Алгебра вида < M, f_2 > называется ***группоидом***.

Если f_2 – операция типа умножения, то группоид называется ***мультипликативным***, если f_2 – операция типа сложения, то ***аддитивным***.

Пусть A = <M, f_2> – группоид. Обозначим операцию f_2 как •. Тогда элемент ℓ, ℓ∈ M называется ***правым нейтральным элементом***, если m∈ M, и m • ℓ = m. Если ℓ • m = m – ***левым нейтральным элементом***. Если выполнены оба соотношения, ℓ называется ***двусторонним нейтральным элементом***, или просто ***нейтральным элементом***.

Если группоид <M, •> мультипликативный, то нейтральный элемент называется ***единицей*** и обозначается ***1***.

Если группоид <M, •> – аддитивный, то нейтральный элемент называется ***нулем*** и обозначается ***0***.

Группоид A = <M, •> называется ***идемпотентным***, если его сигнатура удовлетворяет закону идемпотентности:

$\forall m \in M,\ m \bullet m = m.$

Группоид A = <M,•>, сигнатура которого удовлетворяет закону коммутативности:

$\forall x,y \in M,\ x \bullet y = y \bullet x,$

называется ***коммутативным*** или ***абелевым***.

Группоид <M,•>, в котором выполняется закон ассоциативности:

$\forall x,y,z \in M \quad x \bullet (y \bullet z) = (x \bullet y) \bullet z,$

называется ***ассоциативным*** или ***полугруппой***.

Полугруппа <M,•>, в которой выполнимы обратные операции, т.е. для любых $a, b \in M$ каждое из уравнений $a \bullet x = b$, $y \bullet a = b$ обладает единственным решением, называется ***группой***.

Алгебра <M, *, +>, которая по умножению является мультипликативным группоидом, а по сложению – абелевой группой, причем умножение связано со сложением законами дистрибутивности

$a * (b+c) = a * b + a * c,\ (b+c) * a = b * a + c * a,$

называется ***кольцом***. Кольцо, в котором все отличные от нуля элементы составляют группу по умножению, называется ***телом***. Тело, у которого мультипликативная группа абелева, называется ***полем***.

Рассмотрим алгебру множеств

$A_к = < B(1), \cup, \cap, \bar{} >$

Носителем является булеан универсального множества 1, сигнатурой – операции $\cup$, $\cap$, $\bar{}$. Для операций алгебры множеств выполняются законы:

1. Коммутативности объединения и пересечения:

$A \cap B = B \cap A; \quad A \cup B = B \cup A.$

2. Закон ассоциативности:

$A \cup (B \cup C) = (A \cup B) \cup C;$

$A \cap (B \cap C) = (A \cap B) \cap C.$

3. Закон дистрибутивности пересечения относительно объединения и объединения относительно пересечения:

$A \cap (B \cup C) = A \cap B \cup A \cap C$;

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

4. Законы поглощения:

$A \cup A \cap B = A$; $A \cap (A \cup B) = A$.

5. Законы склеивания:

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

6. Законы Порецкого:

$A \cup \bar{A} \cap B = A \cup B$; $A \cap (\bar{A} \cup B) = A \cap B$.

7. Закон идемпотентности:

$A \cup A = A$; $A \cap A = A$.

8. Закон действия с универсальным и пустым множествами:

$M \cup \varnothing = M$, $M \cap \varnothing = \varnothing$, $M \cup 1 = 1$,

$M \cap 1 = M$, $M \cup \overline{M} = 1$, $M \cap \overline{M} = \varnothing$;

9. Законы де Моргана

$$\overline{A \cap B} = \bar{A} \cup \bar{B}; \quad \overline{A \cup B} = \bar{A} \cap \bar{B};$$

10. Закон двойного дополнения:

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

Алгебра множеств является абелевой полугруппой, но не является группой.

Докажем дистрибутивность $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$. Доказательство проходит в два этапа. Обозначим левую часть как Z, правую – D. Требуется доказать, что если $x \in Z$, то $x \in D$ и наоборот, если $x \in D$, то $x \in Z$.

1. Пусть $x \in A \cup (B \cap C)$, это значит, что $x \in A$ или $x \in (B \cap C)$.

Пусть $x \in A$, тогда $x \in A \cup B$ и $x \in A \cup C$, из чего следует $x \in (A \cup B) \cap (A \cup C)$.

Если $x \in (B \cap C)$, тогда $x \in B$ и $x \in C$; из чего следует $x \in A \cup B$ и $x \in A \cup C$, то есть $x \in (A \cup B) \cap (A \cup C)$.

Первая часть утверждения доказана.

2. Если $x \in (A \cup B) \cap (A \cup C)$, тогда $x \in A \cup B$ и $x \in A \cup C$,

Если $x \in A$, то $x \in A \cup (B \cap C)$;

Если $x \notin A$, то $x \in B$ и $x \in C$, тогда $x \in B \cap C$, из чего следует $x \in A \cup (B \cap C)$.

1.4 Упражнения

1.1. Опишите способы задания множеств.

1.2. В чем отличие между понятиями принадлежности множеству и включения в множество?

1.3. Справедливо ли, что $\{1, 2, 3\} \in \{\{1, 2, 3\}, \{1\}, \{2\}, \{3\}, \{1, 2\}\}$?

1.4. Верно ли, что $\{1, 2, 3\} \subset \{\{1, 2, 3\}, 1, 2, 3, \{1\}\}$?

1.5. Привести пример множеств A, B, C, D, таких что $A \subset B$ и $B \subset C$ и $C \not\subset D$ и $B \subset D$.

1.6. Доказать, что $(A \setminus B) \cup B = A$, $(A \cap B) \cap C = A \cap (B \cap C)$.

1.7. Доказать, что $A \cap (B \setminus A) = \varnothing$ (Доказательство от противного).

1.8. Укажите номера верных записей, если $A = \{2, 4, 6, 8, 9\}$.

а) $1 \in A$; б) $\{2\} \in A$; в) $\{6, 8, 9\} \in A$; г) $\varnothing \in A$?

1.9. Пусть $B = \{a, b, \{c\}\}$. Верно ли, что $a \in B$, $c \notin B$, $\{c\} \in B$, $\{a, b\} \subset B$; $\{b\} \subset B$; $\{c\} \subset B$; $\{\{c\}\} \subset B$?

1.10. Пусть $C = \{+, –, \{+, –\}\}$.

Укажите верные записи: $\varnothing \in C$; $+ \in C$; $\{+, –\} \subset C$; $\{\{+, –\}\} \subset C$; $\{+, –\} \in C$.

1.11. Заданы множества $A = \{1, 2, 9\}$, $B = \{2, 4, 8, 9\}$, $C = \{2, 7, 8, 1\}$ и универсальное множество $I = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Найти: а) $B \cup \overline{A}$; б) $\overline{C} \cup B$; в) $\overline{B} \cup A$; г) $A \cup B \cup C$.

1.12. Заданы множества $A = \{i, k, l\}$, $B = \{k, d, f, c\}$. Найти $A \cup B$, $B \cap A$, $A \setminus B$, $B \setminus A$, $A \oplus B$, а также все подмножества A.

1.13. Заданы множества $A = \{7, 6, 9, 4\}$, $B = \{3, 6, 7, 5, 8\}$. Найти $A \cup B$, $B \cap A$, $A \setminus B$, $B \setminus A$, $A \oplus B$, а также все несобственные подмножества множества A.

1.14. Задано $D = \{ \varnothing, \{ \varnothing \}\}$. Верно ли, что $\{ \varnothing \} \subset D$, $\varnothing \in D$, $\{ \varnothing \} \in D$, $\varnothing \subset D$?

1.15. Заданы множества $A = \{a, b, c\}$, $B = \{b, l, f, k\}$, $C = \{a, c, b, k, l, m, p\}$. Универсальное множество I – множество строчных букв латинского алфавита.

Найти $A \cap \overline{B}$, $C \setminus \overline{B}$, $\overline{A} \cap C \cup B$.

1.16. Задано множество $A = \{l, f, p\}$. Найти его булеан.

1.17. Дано множество S = {a, b, c, 1, 2, 3, 4}. Сколько существует подмножеств этого множества не содержащих букв? Сколько существует подмножеств, не содержащих цифр? Сколько существует подмножеств, не содержащих ни букв, ни цифр?

1.18. Какие из утверждений верны для любых A, B и C?

а) если $A \in B$ и $B \in C$, то $A \in C$;

б) если $A \cap B \subseteq C$ и $A \cup B \subseteq C$, то $A \cap C = \varnothing$;

в) если $A \neq B$ и $B \neq C$, то $A \neq C$;

г) если $A \subseteq \overline{B \cup C}$ и $B \subseteq \overline{A \cup C}$, то $B = \varnothing$

1.19. Нарисовать диаграммы Эйлера-Венна:

а) $(A \cup B) \cap (B \cup C) \cup (B \cap \bar{A})$,

б) $A \cap (B \cup I) \cap (C \cap \bar{I}) \cap (A \cup C)$,

в) $(A \cup B) \cap (\overline{C} \cap B) \cap (A \cap B) \cup (C \cap \bar{A})$.

1.20. Нарисовать диаграммы Эйлера-Венна:

а) $(A \cup B) \cap (\overline{A \cap C}) \cap (A \cup \overline{B})$,

б) $I \cap (A \cup C) \cap (\overline{B \cap C}) \cup (A \cap (B \cup C))$,

в) $(A \cap B) \cup (\overline{A \cap C}) \cup (B \cap C)$.

1.21. Нарисовать диаграммы Эйлера-Венна:

а) $(A \oplus B) \cap (A \oplus C)$,

б) $A \oplus \bar{A} \cap \overline{B} \oplus \bar{A} \cap B$,

с) $A \oplus B \oplus A \cap B$.

1.22. Записать формулу по диаграммам Эйлера-Венна.

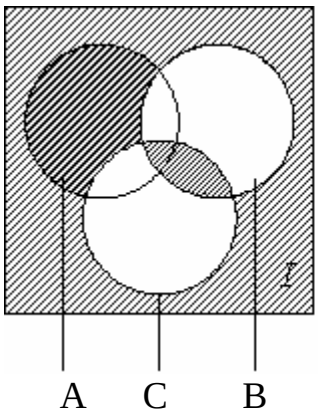


Рис. 1.8

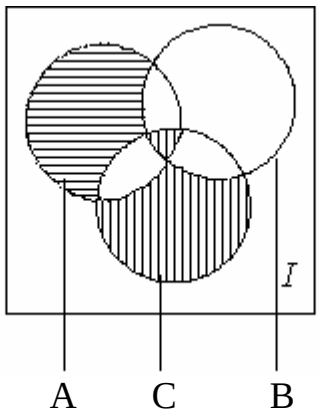


Рис. 1.9

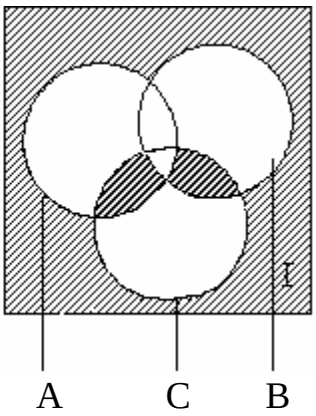


Рис. 1.10

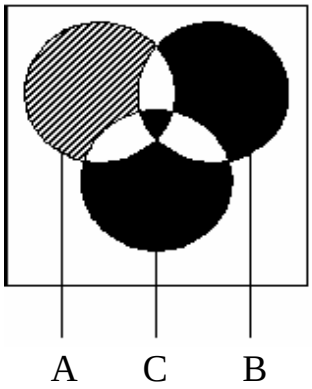


Рис. 1.11

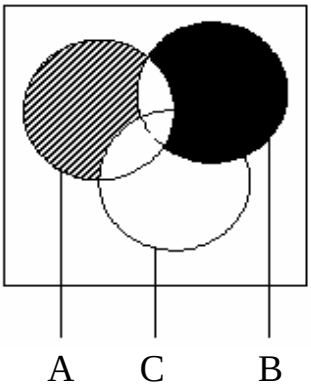


Рис. 1.12

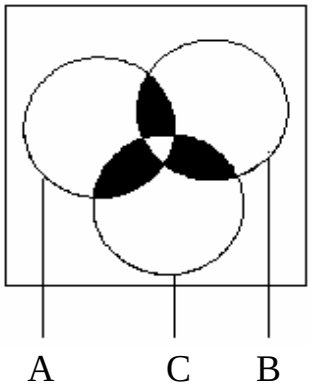


Рис. 1.13

1.23. Доказать, что $A \cap (B \setminus A) = \varnothing$.

1.24. Укажите пустые множества:

а) $A \cup \varnothing$; $\varnothing \cup \varnothing \cap A$;

б) $A \cap B \cap \varnothing$; $I \cup \varnothing \cap A$;

с) $(A \cup B) \cap I \cap \varnothing$; $I \cap \varnothing \cup \varnothing$.

1.25. Упростить выражения, если $B \supset A$.

а) $\overline{A \cup B}$; б) $\overline{\overline{A} \cup B}$; в) $\overline{\overline{A} \cup \overline{B}}$; г) $\overline{\overline{A} \cap \overline{B}}$.

1.26. Равны ли следующие выражения:

а) $A \cup \overline{B \cap C}$ и $A \cup \overline{B} \cup \overline{C}$;

б) $\overline{\overline{\overline{A} \cup I} \cup I}$ и $\overline{\overline{A} \cup \varnothing}$;

в) $\overline{\overline{A \cup A} \cup \overline{A \cup A}}$ и A;

г) $\overline{A \cap \varnothing} \cup \overline{B \cap I}$ и $\overline{I}$;

д) $\overline{\overline{A \cap \overline{A}} \cap \overline{\overline{B} \cap B}}$ и I.

1. 27. Упростить выражения:

а) $\overline{\overline{A \cap (\overline{B \cup C})} \cup (\overline{A \cup B}) \cap C}$;

б) $(C \cap (A \cup B \cup C)) \setminus B$;

в) $A \oplus A \oplus A \oplus A$;

г) $A \cap B \cap C \cup A \cap C \cap \overline{D} \cup A \cap \overline{B} \cap C \cap D$.

1.28. Указать верные выражения:

а) $A \cap (B \oplus C) = A \cap B \oplus A \cap C$;

б) $A \oplus B \oplus A \cap B = A \cup B$;

в) $(A \oplus I) \cap A = \varnothing$;

г) $(A \oplus I \oplus I) \cap A = \varnothing$.

1.29. Упростить выражения:

а) $\overline{A} \cap B \cap C \cup \overline{A} \cap B$;

б) $A \cap \overline{B} \cap \overline{C} \cap \overline{D} \cup \overline{C}$;

в) $A \cap C \cup A \cap B \cap C \cup A \cap C \cap D$;

г) $B \cap (\overline{A} \cap B \cup \overline{B} \cap B)$;

д) $(A \cup \overline{B}) \cap (A \cup \overline{B} \cup D)$;

е) $(\overline{A} \cup B) \cap B \cap (B \cup \overline{C})$;

ж) $A \cap B \cap C \cup A \cap C \cap \overline{D} \cup A \cap C \cup A \cap \overline{B} \cap C \cap D$.

1.30. Найти элементы множеств, если

$A = \{1, 2, 4, 5\}$; $B = \{1, 3, 6, 7\}$; $C = \{3, 2, 6, 7\}$.

а) $A \cap B \cap C \cup B \cap C \cup B \cap \overline{C}$;

б) $\overline{(A \oplus B)} \cap C$;

в) $(\overline{A \cap B \cup C}) \cap (A \cap B \cup \overline{C})$;

г) $(A \cup B) \cap (\overline{B \cap C}) \cap (A \cap B \cap \overline{C})$;

д) $(A \cup \overline{B} \cup C) \cap (\overline{A} \cup \overline{B} \cup C) \cap B$.

1.31. Упростить выражения, если $B = I$, $A = \varnothing$:

$(A \cup B) \cap (C \cup D)$;

$(A \cap C) \cup (\overline{B} \cap C) \cup (A \cap B)$;

$\overline{A} \cap \overline{D} \cap C \cup B \cap C \cap D$;

$\overline{A} \cap (B \cup C \cup D) \cap B \cap C$;

$\overline{A} \cap (\cup B \cup C) \cap (C \cup D \cup B)$;

$(A \cup B \cup C) \cap (\overline{B} \cup D)$.

1.32. Доказать, что два множества равны тогда и только тогда, когда результаты их объединения и пересечения совпадают.

1.33. Известно, что из 100 студентов живописью увлекаются 28, спортом – 42, музыкой – 30, живописью и спортом – 10, живописью и музыкой – 8, спортом и музыкой – 5, живописью, спортом и музыкой – 3.

Определить количество студентов:

а) увлекающихся только спортом;

б) ничем не увлекающихся.

1.34. Экзамен по математике сдавали 250 абитуриентов, оценку ниже пяти баллов получили 180 человек, а выдержали этот экзамен 210 абитуриентов. Сколько человек получили оценки 3 и 4?

1.35. В школе 1400 учеников. Из них 1250 умеют кататься на лыжах,952 – на коньках. Ни на лыжах, ни на коньках не умеют кататься 60 учащихся. Сколько учащихся умеют кататься и на лыжах, и на коньках?

1.36. В группе из 100 туристов 70 человек знают английский язык, 45 знают французский язык и 23 человека знают оба языка. Сколько туристов в группе не знают ни английского, ни французского языка?

1.37. В олимпиаде по математике принимало участие 40 учащихся. Им было предложено решить одну задачу по математике, одну по геометрии и одну по тригонометрии. Результаты проверки:

Решены задачи	Кол-во решивших
По алгебре	20
По тригонометрии	18
По геометрии	18
По алгебре и геометрии	7
По алгебре и тригонометрии	8
По геометрии и тригонометрии	9

Известно также, что ни одной задачи не решили трое. Сколько учащихся решили все три задачи? Сколько учащихся решили две задачи?

1.38. Пусть А – подмножество множества натуральных чисел, каждый элемент множества А есть число, кратное или 2, или 3, или 5. Найти число элементов в множестве А, если среди них имеется: 70 чисел, кратных 2; 60 чисел, кратных 3; 80 чисел, кратных 5; 32 числа, кратных 6; 35 чисел, кратных 10; 38 чисел, кратных 15, и 20 чисел, кратных 30.

1.39. В штучном отделе магазина посетители обычно покупают либо один торт, либо одну коробку конфет, либо один торт и одну коробку конфет. В один из дней было продано 57 тортов и

36 коробок конфет. Сколько было покупателей, если 12 человек купили и торт и коробку конфет?

1.40. В спортивном лагере 65% ребят умеют играть в футбол, 70% – в волейбол и 75% – в баскетбол. Каково наименьшее число ребят, умеющих играть и в футбол, и в волейбол, и в баскетбол?

1.41. Каждый из учеников класса в зимние каникулы ровно два раза был в театре, при этом спектакли A, B и C видели соответственно 25, 12 и 23 ученика. Сколько учеников в классе? Сколько из них видели спектакли A и B, A и C, B и C?

1.42. В течение недели в кинотеатре демонстрировались фильмы A, B и C. Из 40 школьников, каждый из которых просмотрел либо все три фильма, либо один из трех, фильм A видели 13, фильм B – 16, фильм C – 19 школьников. Сколько учеников просмотрели все три фильма.

1.43. В отряде из 40 ребят 30 умеют плавать, 27 умеют играть в шахматы и только пятеро не умеют ни того ни другого. Сколько ребят умеют плавать и играть в шахматы?

1.44. На уроке литературы учитель решил узнать, кто из 40 учеников класса читал книги A, B и C. Результаты опроса оказались таковы: книгу A читали 25 учащихся, книгу B – 22, книгу C – также 22. Книги A или B читали 33 ученика, A или C – 32, B или C – 31; все три книги прочли 10 учащихся. Сколько учеников прочли только по одной книге? Сколько учащихся не читали ни одной из этих трех книг?

1.45. Среди абитуриентов, выдержавших приемные экзамены в вуз, оценку «отлично» получили: по математике – 48 абитуриентов, по физике – 37, по русскому языку – 42, по математике или физике – 75, по математике или русскому языку – 76, по физике или русскому языку – 66, по всем трем предметам – 4. Сколько абитуриентов получили хотя бы одну пятерку? Сколько среди них получивших только одну пятерку?

2 ОТНОШЕНИЯ

Понятие отношения используют для обозначения связи между объектами или понятиями.

Декартовым произведением множеств **A x B** называют третье множество **C,** элементами которого служат пары всех элементов множеств **A** и **B**, при этом первый элемент берется из множества **A**, второй – из множества **B**.

Пример:

$A = \{a_1, a_2, a_3\}$; $B = \{b_1, b_2, b_3\}$.

$A \times B = \{\langle a_1, b_1\rangle, \langle a_2, b_1\rangle, \langle a_3, b_1\rangle, \langle a_2, b_1\rangle, \langle a_2, b_2\rangle, \langle a_2, b_3\rangle, \langle a_3, b_1\rangle, \langle a_3, b_2\rangle, \langle a_3, b_3\rangle\}$.

Сопоставим с декартовым произведением двух множеств прямоугольную решетку, узлы которой взаимно однозначно соответствуют элементам декартова произведения.

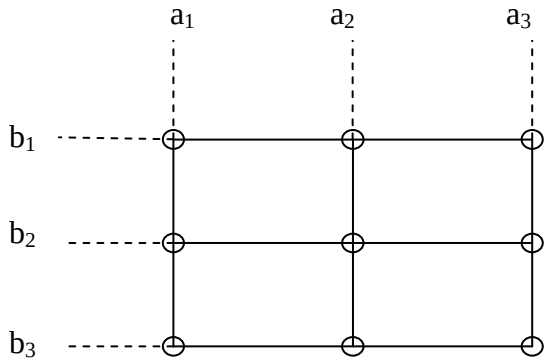


Рисунок 2.1 – Решетка декартова произведения

Декартовым произведением n множеств $\mathbf{A_1, A_2,\ldots, A_n}$ называют множество $\mathbf{C= A_1 x A_2 x\ldots x A_n}$.

Каждый элемент **C** рассматривается как упорядоченное множество

$$\mathbf{c_i=<a^i_1, a^i_2,\ldots,a^i_n>},$$

называемое ***кортежем***.

Кортеж состоит из компонент, для которых задается местоположение. Кортеж может иметь одинаковые компоненты. Число компонент кортежа называют его ***длиной***. Два кортежа считаются ***равными,*** если их длина одинакова и соответствующие компоненты равны между собой. Компонентами кортежа могут быть любые объекты, в том числе множества и кортежи.

Примеры.

1. Пусть Y = {1, 2, 3}, X = {3, 4};

Y x X = {<1, 3>, <1,4>, <2,3>, <2,4>, <3,3>, <3,4>};

X x Y = {<3,1>, <3,2>, <3,3>, <4,1>, <4,2>, <4,3>}.

Необходимо отметить, что Y x X ≠ X x Y.

2. Пусть заданы кортежи:

α = <1,2,3,2>, β = <1,3,2,2>, γ = <1,2,3,2>, δ = <2,1,3,2>.

Здесь α ≠ β, α = γ, α ≠ δ.

Степенью **S** множества **A** называется его прямое произведение самого на себя S раз.

$$A^s = \underbrace{A \times A \times \ldots \times A}_{s}$$

Любое подмножество **R ⊆ X x Y** декартова произведения множеств называется ***бинарным отношением*** из **X** в **Y.**

Если **R** есть некоторое отношение, и пара **<x,y>** принадлежит этому отношению, то наряду с записью **<x,y> ∈ R**, употребляется запись **xRy.**

Областью определения бинарного отношения R называется множество $\mathbf{A_R}$**={x| существует такое y, что xRy}.**

Областью значений бинарного отношения **R** называется множество

$\mathbf{B_R}$ **= {y| существует такое x, что xRy}.**

Пример.

Пусть даны множества: A = {1,2,3,4,5,6,7}, B = {1,2,3}. Построим отношение R⊂AxB, y∈ B есть делитель x∈ A(xRy).

R = {<1,1>, <2,1>, <3,1>, <4,1>, <5,1>, <6,1>, <7,1>, <2,2>, <4,2>, <6,2>, <3,3>, <6,3>}.

Используя решетку, отношение можно изобразить следующим образом.

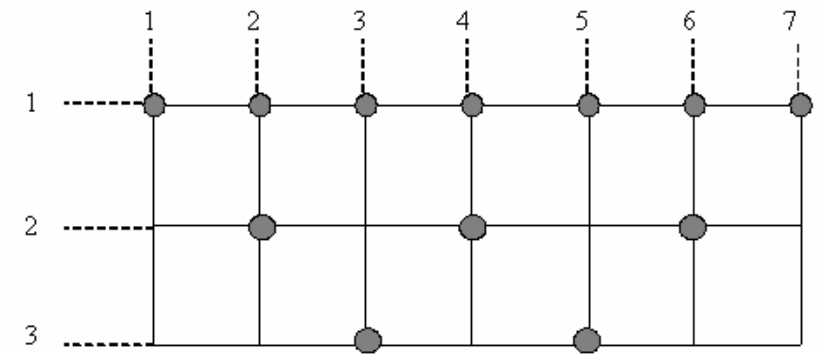


Рисунок 2.2 – Отношение «есть делитель»

Подмножество R обозначено зачернением соответствующих узлов решетки. Графическое представление данного отношения или представление графом (отношения представлены стрелками) показано на рисунке 2.3.

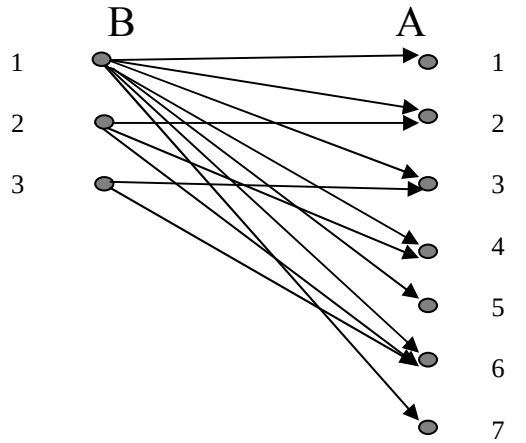


Рисунок 2.3 – Графическое изображение отношения

Так как элементы множества B ⊂ A, то можно показать отношение R способом, представленным на рисунке 2.4.

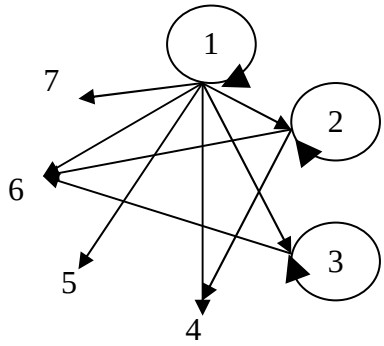


Рисунок 2.4 – Графическое изображение отношения

Квадратом множества **X** называется декартово произведение двух равных между собой множеств: $\mathbf{X \times X = X^2}$. ***Бинарным отношением T*** в множестве **X** называется подмножество его квадрата $\mathbf{T \subset X^2}$. Совокупность множества **X** с заданным в нем бинарным отношением $\mathbf{T \subset X^2}$ называется ***графом G***.

$$\mathbf{G = (X, T),}$$

где **X** – множество вершин (носитель графа);

T – множество дуг (сигнатура графа).

Пусть задано декартово произведение A x B. И пусть c ∈ A x B, c = <x, y>, x∈ A, y∈ B.

Проекцией элемента **c** на множество **A** назовем элемент **x.**

Сечением R ⊂ A x B по элементу x∈ A называется множество элементов y∈ B таких, что <x,y> ∈ R, R ⊂ A x B.

Вместо термина сечение часто употребляется термин ***окрестность единичного радиуса***.

Множество всех сечений, взятых для всех элементов множества **A** при задании на нем отношения **R ⊂ A x B**, называется ***факторможеством***.

Факторможество полностью определяет отношение R.

Рассмотрим предыдущий пример. Пусть X ⊂ A, и X = {2,7}. Тогда сечение R(X) = R(2) ∪ R(7) = {1,2} ∪ {7} = {1,2}. Факторможество определяется как множество всех сечений R по всем элементам из A. Зададим факторможество в виде двух строк, в первой из которых поместим элементы множества A, а во второй под каждым элементом запишем сечение по этому элементу.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \{1\} & \{1,2\} & \{1,3\} & \{1,2\} & \{1\} & \{1,2,3\} & \{1\} \end{bmatrix}$$

Вторая строка задает факторможество. Сечение и факторможество наглядно представлены решеткой на рисунке 2.2.

2.1 Операции над отношениями

Для бинарных отношений обычным образом определены теоретико-множественные операции объединения, пересечения и др.

Пусть $R_1 \subset A \times C$ отношение из A в C, а $R_2 \subset C \times B$ – отношение из C в B.

Композицией двух отношений $\mathbf{R_1}$ и $\mathbf{R_2}$ называется отношение **R ⊂ A x B** из **A** в **B**, определяемое следующим образом:

$R = R_1 \circ R_2$ = {<a,b>| существует C такое, что <a,c> ∈ R_1 и <c,b> ∈ R_2}

Обратным отношением для **R** называется отношение

R^1 = {<a,b>| <b,a>∈ R}.

Для любых бинарных отношений выполняются свойства:

Свойство 1:

$(R^{-1})^{-1} = R$

Свойство 2: Пусть S, R – отношения, тогда

$(S, R)^{-1} = R^{-1} \circ S^{-1}$

Свойство 3: Если $R \subset S$ и $T \subset Y$, то

$T\,R \subset Y\,S$

Функция f : A→B может быть определена как отношение, определенное на множестве **A x B**, обладающее свойством:

Если **(a,b) ∈ f** и **(a,c) ∈ f**, то **b = c.**

Область определения функции обозначается как $\mathbf{D_f}$, а **область ее значений** как $\mathbf{R_f}$.

Если $D_f = A$ и $R_f \subseteq B$, то говорят, что функция f задана на множестве A со значениями во множестве B и осуществляет отображение множества A во множество B. Вместо <a,b> ∈ f, пишут b = f(a), здесь b – значение функции, соответствующее аргументу a.

Пусть **f : A→B**. Функция называется ***инъективной***, если для любых $\mathbf{a_1}$, $\mathbf{a_2}$, **b** из $\mathbf{b=f(a_1)}$ и $\mathbf{b=f(a_2)}$ следует, что $\mathbf{a_1=a_2}$ (рисунок 2.6).

Функция **f** называется ***сюръективной***, если для любого элемента **b∈B** существует элемент **a∈A** такой, что **b=f(a)** (рисунок 2.7).

Функция f называется ***биективной***, если **f** одновременно сюръективна и инъективна (рисунок 2.8).

Если существует биективная функция **f : A →B**, то говорят, что f осуществляет взаимнооднозначное соответствие между множествами **A** и **B**.

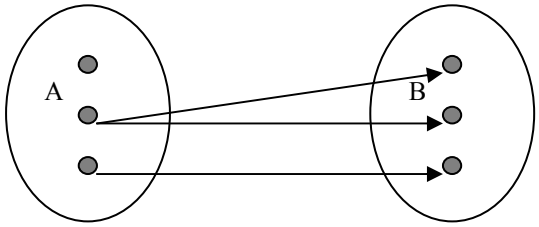


Рисунок 2.5 – Отношение, но не функция

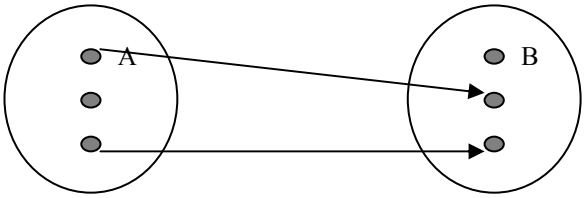


Рисунок 2.6 – Инъективная функция

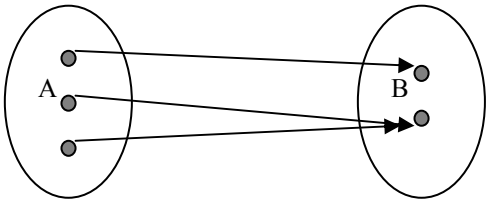


Рисунок 2.7 – Сюръективная функция

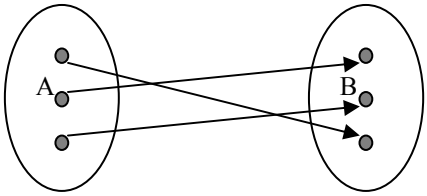


Рисунок 2.8 – Биективная функция

2.2 Свойства бинарных отношений

Рассмотрим наиболее важные свойства бинарных отношений. Пусть задано множество A, A x A=A^2 и отношение R на нем.

Отношение R в множестве A называется ***рефлексивным***, если для каждого элемента **a∈ A** справедливо утверждение **a R a.** Если изображать рефлексивность графически, то элемент имеет петлю.

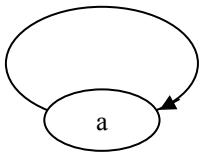


Рисунок 2.9 – Рефлексивное отношение

Отношение **R** в множестве **A** называется ***симметричным***, если для любых **a,b∈ A** выполнено: **aRb** следует **bRa**; или **(a,b) ∈ R ⇒ (b,a)∈ R a≠ b.**

Графически симметричность можно изобразить следующим образом:

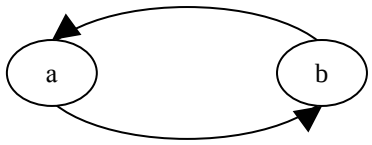


Рисунок 2.10 – Симметричное отношение

Отношение **R** называется ***транзитивным***, если для любых **a,b,c∈ A**, выполняется: **a R b** и **b R c ⇒ a R c**, при этом **a ≠ b, b ≠ c, a ≠ c**. Граф, представляющий транзитивное отношение R, выглядит следующим образом:

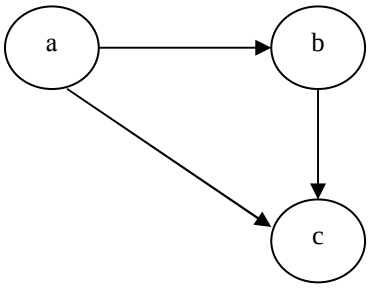


Рисунок 2.11 – Транзитивное отношение

При этом дуга **(a,c)** называется ***транзитивно замыкающей*** дугой.

Бинарное отношение **R** в множестве **A**, обладающее свойствами:

рефлексивности: для каждого **a∈ A, (a,a) ∈ R**;

транзитивности: для любых **a,b,c∈ A** следует **(a,b)∈ R, (b,c)∈ R⇒ (a,c)∈ R,**

называется отношением ***упорядоченности*** и обозначается **≤.**

Если любые два элемента **a, b** упорядоченного множества находятся в отношении упорядоченности **a≤b** или **b≤a**, то это множество называется ***линейноупорядоченным***, в противном случае – ***частично упорядоченным***.

Например, отношение a ≤ b на множестве действительных чисел является отношением упорядоченности. Во множестве подмножеств некоторого универсального множества U отношение A⊆B также есть отношение упорядоченности. Схема органи-

зации подчинения в учреждении есть отношение частичного порядка на множестве должностей.

Бинарное отношение в множестве А, обладающее свойствами антирефлексивности, антисимметричности и транзитивности, называется ***отношением строгой упорядоченности*** и обозначается <.

Рассмотрим отношение включения ⊂. Это отношение рефлексивно: $A_i \subset A_i$ (множество A_i включает само себя). Если $A_i \subset A_j$ и $A_j \subset A_i$, то $A_i = A_j$, следовательно отношение антисимметрично, если $A_i \subset A_j$ и $A_j \subset A_k$, то $A_i \subset A_k$, то есть отношение ⊂ транзитивно и является отношением упорядоченности ≤. Множество А с заданным на нем отношением упорядоченности ≤ называется упорядоченным этим отношением. Отношение включение является частично упорядоченным множеством.

Частично упорядоченное множество изображают в виде графов G = (X, U). Например, пусть А = {1, 2, 3}. Рассмотрим отношение быть подмножеством. Получаем следующую диаграмму (удалены петли и транзитивно замыкающие дуги), называемую **диаграммой Хассе Н.**

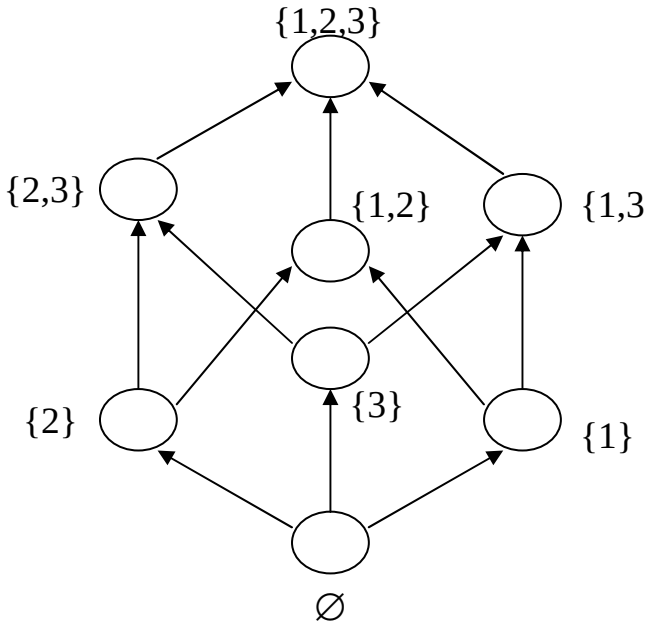


Рисунок 2.12 – Пример диаграммы Хассе

Понятие непосредственного старшего легко задается в частично упорядоченном множестве следующим определением: a_i покрывает a_j. Это означает, что $a_j < a_i$ и не найдется такого элемента a_k, что $a_j < a_k < a_i$. Если рассмотрим подмножество $A' \subset A$ и найдем такой элемент $a_L \in A$, что $a_i < a_L$ для любого элемента $a_i \in A'$, то этот элемент называется ***мажорантой*** подмножества A'. Аналогично, если найдется элемент $a_\beta \in A$, такой, что $a_\beta < a_j$ для любого элемента $a_j \in A'$, то элемент a_β называется ***минорантой*** подмножества A'.

Если отношение **R**, определенное на множестве **A**, является рефлексивным, симметричным и транзитивным, то его называют ***отношением эквивалентности***.

Классом эквивалентности, порожденным элементом **x**, называется подмножество множества **X,** состоящее из тех элементов **y ∈ X**, для которых **x R y**. Класс эквивалентности, порожденный через **x**, обозначается **[x].**

Например, дано множество A = {0, 1, 2, 3}. Определим отношение R, отвечающее общепринятому понятию «равно». E = ={<0, 0>, <1,1>, <2,2>, <3,3>}. Для заданного отношения эквивалентности E существуют четыре класса эквивалентности [0] = ={0}, [1] = {1}, [2] = {2}, [3] = {3}.

Отношение принадлежности к одной студенческой группе на множестве студентов института – это отношение эквивалентности, и классом эквивалентности является множество студентов одной группы.

Необходимо заметить, что класс эквивалентности порождается любым своим элементом. Классы эквивалентности, соответствующие отношению эквивалентности R, определенному на множестве A, разбивают множество A на конечное число непустых непересекающихся множеств.

Например, определим отношение R на множестве натуральных чисел следующим образом:

a R b справедливо, если и только если |a – b| делится на 5 без остатка. В этом случае множество N разбивается на пять бесконечных классов эквивалентности:

{{1,6,11,…}, {2,7,12,…}, {3,8,13,…}, {4,9,14,…}, {5,10,15}}.

Если **R** – произвольное отношение, определенное на множестве **A**, то его ***рефлексивным замыканием*** называется наименьшее рефлексивное отношение, определенное на множестве A, для которого отношение R является подмножеством.

Например, если R = {<0,1>, <1,1>, <1,2>} – отношение, определенное на множестве A = {0,1,2}, то его рефлексивным замыканием является множество {<0,0>, <0,1>, <1,1>, <1,2>, <2,2>}.

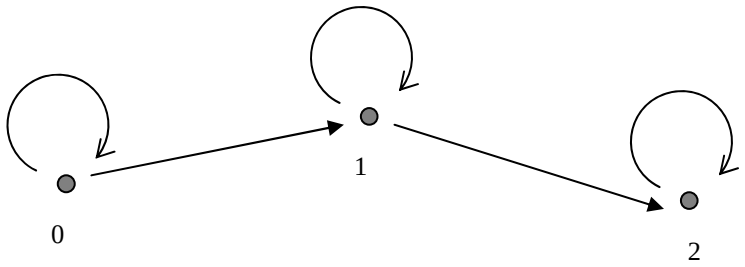


Рисунок 2.13 – Рефлексивное замыкание

Симметричным замыканием является множество {<0,1>, <1,0>, <1,1>, <1,2>, <2,1>}.

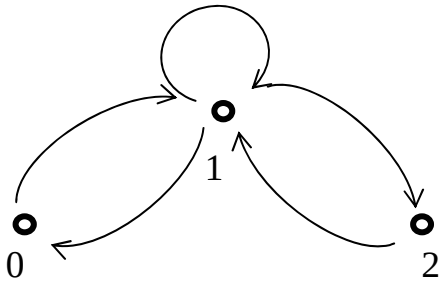


Рисунок 2.14 – Транзитивное замыкание {<0,1>, <0,2>, <1,1>, <1,2>}

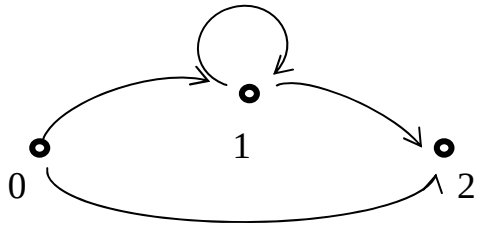


Рисунок 2.15 – Симметричное замыкание

Заметим, что рефлексивным замыканием отношения <, определенного на множестве целых чисел, является отношение ≤, его симметричным замыканием является отношение ≠, а транзитивным замыканием является само отношение <.

2.3 Задачи и упражнения

2.1. Укажите номера всех пар, являющихся элементами отношения:

$a - b = 2$, $a \in A$, $A = \{1, 2, 3, 4, 5\}$, $b \in B$, $B = \{6, 7, 8, 9, 10, 11, 12\}$.

1) 3,1; 2) 6,4; 3) 4,6; 4) 5,3; 5) 4,2; 6) 7,5; 7) 8,6.

2.2. Найдите элементы множества

$(A \times B) \oplus (B \cap A)$, $A = \{a, b\}$, $B = \{b, c\}$.

2.3. Найдите элементы множеств A и B, если

$A \times B = \{\langle a, 3\rangle, \langle a, 8\rangle, \langle b, 3\rangle, \langle b, 8\rangle, \langle k, 3\rangle, \langle k, 8\rangle\}$.

2.4. Декартово произведение множеств A и B содержит 12 элементов. Известно, что $A = \{a, k, f\}$ и $A \cap B = \varnothing$. Найдите число собственных подмножеств множества B.

2.5. Укажите рефлексивные отношения:

- точка a удалена от точки b на 4 см;
- $a \leq b$, где a и b – натуральные числа;
- $a \neq b$, где a и b – натуральные числа;
- a похоже на b (в множестве людей);
- Петров и Сидоров имеют одинаковый рост;
- Смирнов и Васильев живут на третьем этаже;
- число a не больше числа b;
- поезд a идет быстрее поезда b.

2.6. Укажите симметричные отношения:

- лесоруб спилил дерево;
- число a не больше числа b, где $a, b \in \{1, 2, 3, \ldots, 9\}$;
- a равно b;
- c старше, чем b;
- Таня – сестра Пети;
- $25 + 10 = 20 + 15$.

2.7. Укажите транзитивные отношения:

- быть южнее;
- не равно;
- быть врагом;
- являться матерью;
- дружить.

2.8. Укажите отношения эквивалентности:

- автомобиль a столкнулся с автомобилем b;
- высота горы a равна высоте горы b;
- Иванов задал вопрос Петрову;
- a + b = 100, где a, b∈ {1, 2, 3,…, 100}.
- прямая a перпендикулярна прямой b;
- A и b равновеликие треугольники;
- фраза a имеет тот же самый смысл, что и фраза b.

2.9. Привести примеры отношений:

- не рефлексивного, но симметричного и транзитивного;
- не симметричного, но рефлексивного и транзитивного;
- не транзитивного, но рефлексивного и симметричного.

2.10. На множестве A×A, где A – множество натуральных чисел {1, 2, 3,…} определено отношение R : <x, y> R<u,v>, такое, что x + v = y + v. Доказать, что R – отношение эквивалентности на этом множестве.

2.11. Доказать, что если отношения R_1 и R_2 рефлексивны, то рефлексивны отношения $R_1 \cup R_2$, $R_1 \cap R_2$.

2.12. На множестве A = {1, 2, 3, 4, 5} задано отношение R = ={<1,2>, <3,1>, <3,4>, <4,4>, <5,4>}. Построить рефлексивное, симметричное и транзитивное замыкания.

3 НЕЧЕТКИЕ МНОЖЕСТВА

Расширим понятия множества, введя свойство нечеткости. Принадлежность элемента **x** множеству **A** $x \in A$, $A \subset M$ будем задавать с помощью характеристической функции: $\mu_A(x)$, которая принимает значения на интервале **[0, 1]**. В соответствии с этим элемент может не принадлежать множеству **A** (в этом случае $\mu_A(x) = 0$), может принадлежать множеству **A** в какой-то степени, может быть элементом множества **A** ($\mu_A(x)=1$).

Нечеткие множества применяются в том случае, когда затруднительно использовать традиционный математический аппарат, то есть когда характеристики объекта размыты. Как правило, это качественные характеристики и они не могут быть однозначно интерпретированы.

Нечетким множеством **A** множества **M** назовем множество пар

$A=\{(\mu_A(x) \mid x)\}$, где $x \in M$, $\mu_A(x) \in [0, 1]$.

Функция $\mu_A : x \rightarrow [0, 1]$ называется ***функцией принадлежности*** нечеткого множества **A**. **M** называется ***базовым множеством***. Множество элементов $X \subset M$, для которых функция принадлежности не равна нулю, называется ***носителем нечеткого множества.***

Для каждого конкретного значения $x \in M$ величина $\mu_A(x)$ принимает определенное значение из заданного интервала **[0, 1].** Величина $\mu_A(x)$ называется ***степенью принадлежности*** элемента **x** к множеству **A**.

Обозначим $P = \{\mu_A(x)\}$ – множество принадлежностей.

Пусть **M** – базовое множество, P – множество принадлежностей, **A** и **B** два нечетких множества.

Будем говорить, что **A** содержится в **B**, если $(\forall x \in M)$ $(\mu_A(x) \le \mu_B(x))$,

и обозначать $A \subset B$, если неравенство строгое, то обозначаем $A \subset\subset B$. Скажем, что **A** и **B** равны тогда и только тогда, когда $(\forall x \in M)$ $(\mu_A(x) = \mu_B(x))$, и будем обозначать **A=B**. Если найдется по крайней мере один такой элемент из **M**, что равенство

($\mu_A(x) = \mu_B(x)$) не удовлетворяется, то будем говорить, что **A** и **B** не равны и обозначать **$A \neq B$**.

Пример. Пусть X множество натуральных чисел. Тогда его нечеткое подмножество «очень малых чисел» может быть таким:

A={(1/1), (0.8/2), (0.7/3), (0.6/4), (0.5/5), (0.3/6), (0.1/7)}. Носителем нечеткого множества A является множество X={1, 2, 3, 4, 5, 6, 7}. Функция принадлежности определяется субъективно.

3.1 Операции над нечеткими множествами

Пусть M – базовое множество, P = [0,1] – множество принадлежности, A и B два нечетких подмножества. К нечетким множествам применимы те же операции, что и к обычным множествам.

Дополнение. **A** и **B** дополняют друг друга **$A = \bar{B}$** или **$A = \bar{B}$**, если

($\forall x \in M$) ($\mu_A(x) = 1 - \mu_B(x)$).

Пример.

M ={1, 2,3,4,5,6,7}. A = {(0.3/1), (0.7/3), (0.9/6)}.

Тогда $\bar{A}$ = {(0.7/1), (1/2), (0.3/3) (1/4), (1/5), (0.1/6), (1/7)}.

Пересечение. Пересечение **$A \cap B$** определяют как наибольшее нечеткое подмножество, содержащееся одновременно в **A** и **B**.

($\forall x \in M$) ($\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$).

Объединение. Определим объединение **$A \cup B$** как нечеткое множество, которое содержит как **A**, так и **B**.

($\forall x \in M$) ($\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$).

Введенные операции дополнения, объединения, пересечения удовлетворяют законам:

Коммутативности объединения и пересечения:

$A \cap B = B \cap A$, $A \cup B = BA$.

Закон ассоциативности:

$A \cup (B \cup C) = (A \cup B) \cup C$;

$A \cap (B \cap C) = (A \cap B) \cap C$.

Закон дистрибутивности пересечения относительно объединения и объединения относительно пересечения:

$A \cap (B \cup C) = A \cap B \cup A \cap C$;

$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

Закон идемпотентности:

$A \cup A = A; \qquad A \cap A = A.$

Законы де Моргана:

$\overline{A \cap B} = \bar{A} \cup \bar{B}, \qquad \overline{A \cup B} = \bar{A} \cap \bar{B};$

Закон двойного дополнения:

$\bar{\bar{A}} = A;$

Действия с универсальным и пустым множествами:

$A \cup \varnothing = A, \qquad A \cap \varnothing = \varnothing, \quad A \cup 1 = 1, \; A \cap 1 = A.$

$\varnothing$ – пустое множество. $\varnothing \leftrightarrow (\forall x \in A) \quad (\mu_{\varnothing}(x) = 0)$.

1 – универсальное множество $1 \leftrightarrow (\forall x \in A) \quad (\mu_{1}(x) = 1)$.

Необходимо заметить, что соотношения $A \cap \bar{A} = \varnothing$, $A \cup \bar{A} = 1$ с нечеткими множествами не выполняются. Рассмотрим пример. Пусть M = {1, 2, 3, 4, 5, 6}. A = {(0.3/1), (0.4/2), (0.5/3), (0.8/4), (0.9/5),(1/6)}.

Тогда $\bar{A}$ = {(0.7/1), (0.6/2), (0.5/3), (0.2/4), (0.1/5), (0/6)}.

$A \cap \bar{A}$ = {(0.3/1), (0.4/2), (0.5/3), (0.2/4), (0.1/5), (0/6)}.

$A \cup \bar{A}$ = {(0.7/1), (0.6/2), (0.5/3), (0.8/4), (0.9/5), (1/6)}.

В силу несправедливости выше приведенных соотношений не выполняются законы склеивания:

$B \cap A \cup B \cap \bar{A} \neq B, \quad B \cup A \cap B \cup \bar{A} \neq B.$

Так же не выполняются законы Порецкого:

$A \cup B \cap \bar{A} \neq B \cup A, \quad A \cap (B \cup \bar{A}) \neq A \cap B.$

Ряд задач информационной математики сводятся к определению «близости» нечеткого подмножества к подмножеству, выполняющему роль эталона. При решении этих задач используется понятие метрического пространства.

Метрическим пространством называется множество (M, D), состоящее из элементов множества **M** (точек) и определенного в нем расстояния $\mathbf{d(m_i, m_j) \in D}$ между любыми двумя точками $\mathbf{m_i, m_j}$, удовлетворяющего условиям:

Неотрицательности:

$$d(m_i, m_j) \begin{cases} 0, & \text{если } m_i = m_j \\ 1, & \text{если } m_i \neq m_j \end{cases}$$

Симметричности:

$$\mathbf{d(m_i, m_j) = d(m_j, m_i).}$$

Транзитивности:

$$\mathbf{d(m_i, m_j) + d(m_j, m_k) = d(m_i, m_k).}$$

Расстоянием Хемминга $d_h(A,B)$ между подмножествами **A, B** (обыкновенные детерминированные подмножества, в этом случае $\mu_A(x)$ принимает значения из {0,1}) называется число, равное

$$\sum_{i=1}^{n} |\mu_A(x_i) - \mu_B(x_i)|,$$

где n – размерность пространства.

Относительным расстоянием Хемминга $d_{ho}(A,B)$ между подмножествами A и B называется число, равное $\mathbf{n^{-1} \bullet d_h(A,B).}$

Рассмотрим пример. Пусть A ={10110}, B={01101}.

Расстояние Хемминга $d_h(A,B) = 4$. Относительное расстояние Хемминга $d_{ho}(A,B) = 5^{-1} \bullet 4 = 0.8$.

Обобщенное расстояние Хемминга (линейное расстояние) **$d_л(A, B)$** между нечеткими подмножествами A, B определяется значением

$$\sum_{i=1}^{n} |\mu_A(x_i) - \mu_B(x_i)|,$$

где n – размерность пространства.

Здесь $\mu_A(x)$ принимает значения на интервале [0, 1].

Пример.

A={(0.2/1), (0.8/2), (1.0/3), (0.0/4), (0.7/5), (0.8/6)},

B={(0.3/1), (0.9/2), (0.8/3), (0.6/4), (1.0/5), (0.0/6)}.

$d_л(A, B)$ =|0.2–0.3| + |0.8–0.9| +|1.0–0.8| +|0.0–0.6| +
+|0.7–1.0|+|0.8–0.0|=2.1.

Относительное линейное расстояние **$d_{ол}(A, B)$** определяется как **$d_{ол}(A, B) = n^{-1} \bullet d_{ол}(A,B).$**

Евклидовым (квадратным) расстоянием **$d_e(A,B)$** между нечеткими подмножествами A и B называется число

$$\sqrt{\sum_{i=1}^{n} (\mu_A(x_i) - \mu_B(x_i))^2}, \qquad 0 \le d_e(A,B) \le \sqrt{n},$$

где n – размерность пространства. Очевидно, что относительное евклидово расстояние **$d_{eo}(A, B)$** определяется как

$$d_{eo}(A,B) = \left(\sqrt{n}\right)^{-1} d_e(A,B),$$

в этом случае $0 \le d_{eo} \le 1$.

Рассмотрим еще два определения «расстояние»: – линейный индекс нечеткости λ (A'), вычисляемый через относительное линейное расстояние $d_{ол}$(A', A), и квадратичный индекс нечеткости X (A'), определяемый посредством относительного евклидова расстояния d_{eo}(A', A),

$$\lambda(A') = 2 \cdot (n)^{-1} \cdot \sum_{i=1}^{n} \left(\left| \mu_A(x_i) - \mu_A(x_i) \right| \right);$$

$$X(A') = 2 \cdot \left(\sqrt{n}\right)^{-1} \cdot \sqrt{\sum_{i=1}^{n} \left(\mu_{A'}(x_i) - \mu_A(x_i)\right)^2};$$

где n – размерность пространства,

2 – коэффициент, обеспечивающий соотношения

$$0 \le \lambda(A') \le 1,\ 0 \le X(A') \le 1.$$

3.2 Задачи и упражнения

1. Доказать закон де Моргана для нечетких множеств.

$\overline{A \cap B} = \overline{A} \cup \overline{B}$, $\overline{A \cup B} = \overline{A} \cap \overline{B}$;

2.Доказать закон поглощения для нечетких множеств.

$A \cup A \cap B = A$; $A \cap (A \cup B) = A$.

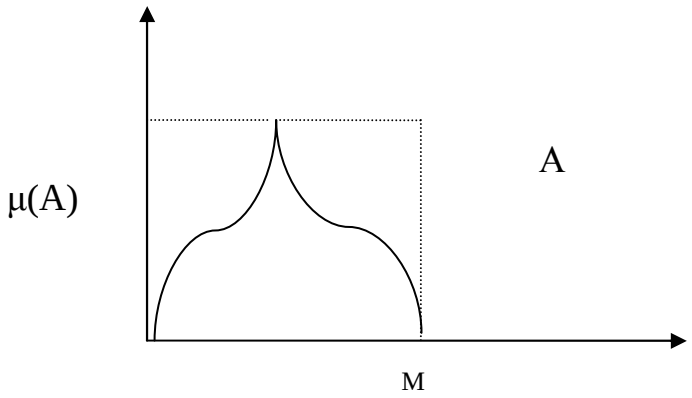
3. Задано два нечетких подмножества A и B множества M = ={1,2,3,4,5,6,7};

A={(0.3|2), (0.7|4), (0.9|6)};

B={(0.2|1), (0.4|3), (0.1|4), (0.9|6), (0.2|5), (0.5|7)}.

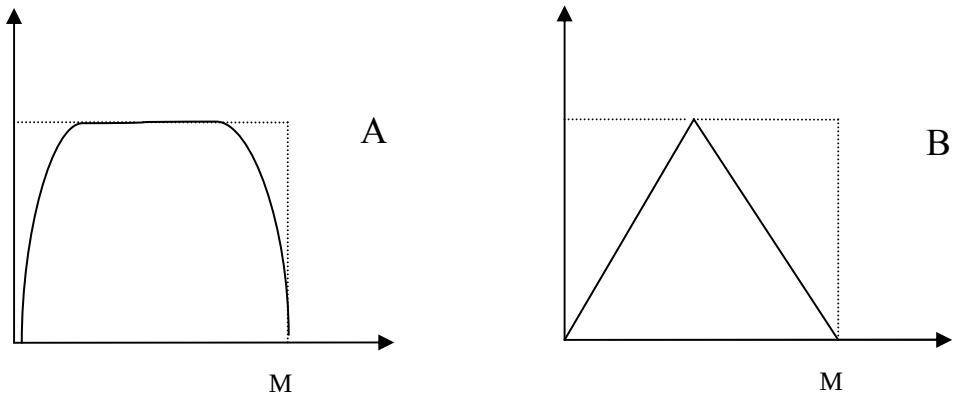
Найти: $\bar{A}$; $A \cup B$; $A \cup \bar{B}$; $A \cap B$; $\bar{A} \cap B$; $\bar{A} \cap \bar{B}$.

4. Задано нечеткое подмножество A. Найти $\bar{A}$.



5. Заданы два нечетких множества A и B.

Отобразить A∩B; A∪ $\bar{A}$ ∩B.



6. Заданы нечеткие подмножества A, B,C множества

M = {1, 2, 3, 4, 5, 6 }; A={(0.1|3), (0.6|4), (0.7|5)}; B={(0.3|2), (0.4|6), (0.1|1)}; C={(0.2|2), (0.3|4), (0.7|1), (0.1|5)}.

Найти дополнение их пересечения и объединения.

7. Даны два подмножества A и B множества M. M={1, 2, 3, 4, 5, 6, 7}, A={3, 6,2,1,7}, B={1, 2, 4, 6, 5}. Найти расстояние по Хеммингу, относительное расстояние по Хеммингу.

8. Заданы два нечетких подмножества A и B множества M = ={1, 2, 3, 4, 5, 6}. A={(0.1|1), (0.3|2), (0.4|3), (0.4|4), (0.3|5), (0.2|6)}, B={(0.2|1), (0.4|2), (0.8|3), (0.9|4), (0.1|5), (0.9|6)}. Найти расстояние по Хеммингу, относительное расстояние по Хеммингу между нечеткими подмножествами A и B.

4 K-ЗНАЧНАЯ ЛОГИКА

4.1 Элементарные функции k-значных логик и соотношение между ними

Всюду в этой главе число ***k*** предполагается натуральным большим 2. Через E_k обозначается множество **{0, 1, …, *k*–1}**. Функция $f(x_1, x_2, \ldots, x_n)$ называется функцией ***k-значной логики***, если на всяком наборе $\tilde{a}=(a_1,a_2,\ldots,a_n)$ значений переменных $x_1, x_2, \ldots, x_n$, где $a_i \in E_k$, значение $f(a_1, a_2, \ldots, a_n) \in E_k$. Совокупность всех функций **k**-значной логики обозначается через P_k.

Очевидно, что функция $f(x_1, x_2, \ldots, x_n)$ полностью определена, если задана ее таблица (см. табл.XXX). В этой таблице наборы суть разложения в **k**-ичной системе счисления чисел **0, 1, …, k^n–1**. Символ **f** здесь будет интерпретироваться как символ, обозначающий отображение, характеризуемое таблицей, а символы $x_1, x_2, \ldots, x_n$ – как названия столбцов.

x_1	x_2		x_i		x_{n-1}	x_n	$f(x_1, x_2, \ldots, x_n)$
0	0	…	0	…	0	0	f(0, 0, …, 0, 0)
0	0	…	0	…	0	1	f(0, 0, …, 0, 1)
			…				…
0	0	…	0	…	0	k–1	f(0, 0, …, 0, k–1)
0	0	…	0	…	1	0	f(0, 0, …,1, 0)
			…				…
k–1	k–1	…	k–1	…	k–1	k–2	f(k–1, k–1, …, k–1, k–2)
k–1	k–1	…	k–1	…	k–1	k–1	f(k–1, k–1, …, k–1, k–1)

Теорема. Число всех функций из P_k, зависящих от ***n*** переменных $x_1, x_2, \ldots, x_n$, равно k^{k^n}.

Из сказанного вытекает, что в P_k при $k \geq 3$ в значительной степени возрастают трудности по сравнению с P_2 как в возможности эффективного использования табличного задания функций, так и в возможности просмотра всех функций от **n** переменных. Уже в P_3 число функций от двух переменных равно $3^9=19683$, т.е. это множество практически необозримо. В P_k часто употребляют вместо табличного задания функций задание при помощи алгоритма вычислимости функций. Например, $\max(x_1, x_2, \ldots, x_n)$

можно рассматривать как алгоритм, который для любого набора $(a_1, a_2, \ldots, a_n)$ значений переменных выдает их максимум. Этот алгоритм определяет в P_k единственную функцию, которую будем обозначать тем же символом.

Понятия фиктивной и существенной переменных, равных функций, формулы над множеством функций (и связок), операций суперпозиции и замыкания, замкнутого класса, базиса и другие в **k**-значных логиках определяются так же, как существующие понятия в алгебре логики.

Следующие функции k-значной логики считаются элементарными:

Константы **0, 1, …, k–1;** эти функции будут рассматриваться как функции, зависящие от произвольного конечного числа переменных (включая и нуль переменных).

Отрицание Поста: $\overline{x} = x + 1 (\mathrm{mod}\ \ k)$. Здесь $\overline{x}$ представляет обобщение отрицания в смысле «циклического» сдвига значений.

Отрицание Лукасевича: $Nx = \sim x = k - 1 - x$. Здесь **~x** является обобщением отрицания в смысле «зеркального» отображения значений.

Характеристическая функция первого рода числа i: $\mathbf{j_i(x)}$ **(i=0, 1, …, k–1)**

$$j_i(x) = \begin{cases} 1, \text{ если } x = i, \\ 0, \text{ если } x \neq i. \end{cases}$$

Характеристическая функция второго рода числа i: $\mathbf{J_i(x)}$ **(i=0, 1, …, k–1)**

$$J_i(x) = \begin{cases} k-1, \text{ если } x = i, \\ 0, \text{ если } x \neq i. \end{cases}$$

Минимум x_1 и x_2: min (x_1, x_2) – обобщение конъюнкции.

Максимум x_1 и x_2: max (x_1, x_2) – обобщение дизъюнкции.

Сумма по модулю k: $\mathbf{x_1 + x_2 \ (mod\ k)}$, читается «$x_1$ плюс x_2 по модулю **k**».

Произведение по модулю k: $\mathbf{x_1 \cdot x_2 \ (mod\ k)}$, читается «произведение x_1 на x_2 по модулю **k**».

Усеченная разность:

$$x \div y = \begin{cases} 0, \text{если } 0 \le x < y \le k-1, \\ x - y, \text{если } 0 \le y \le x \le k-1. \end{cases}$$

Импликация

$$x \supset y = \begin{cases} k-1, \text{если } 0 \le x < y \le k-1, \\ (k-1) - x + y, \text{если } 0 \le y \le x \le k-1. \end{cases}$$

Функция Вебба: $v_k(x_1, x_2)=\max(x_1, x_2) + 1 \pmod k$.

Разность по модулю k:

$$x - y = \begin{cases} x - y, \text{если } 0 \le y \le x \le k-1; \\ k - (y - x), \text{если } 0 \le x < y \le k-1. \end{cases}$$

Опираясь на понятие эквивалентности, можно описать некоторые основные свойства элементарных функций. Пусть $(x_1 \circ x_2)$ обозначает любую из функций $\min(x_1,x_2)$, $x_1 \cdot x_2 \pmod k$, $\max(x_1, x_2)$, $x_1+x_2 \pmod k$.

1. Функция $(x_1 \circ x_2)$ обладает свойством **ассоциативности**:

$$((x_1 \circ x_2) \circ x_3)=(x_1 \circ (x_2 \circ x_3)).$$

2. Функция $(x_1 \circ x_2)$ обладает свойством **коммутативности**:

$$(x_1 \circ x_2)= (x_2 \circ x_1).$$

Кроме того, справедливы следующие соотношения:

1. **Дистрибутивность** умножения относительно с**ложения**:

$$(x_1+x_2) \cdot x_3=(x_1 \cdot x_3)+(x_2 \cdot x_3).$$

2. Д**истрибутивность** операции **max** относительно операции **min**:

$$\max(\min(x_1,x_2), x_3)=\min(\max(x_1,x_3),\max(x_2,x_3)).$$

3. **Дистрибутивность** операции **min** относительно операции **max**:

$$\min(\max(x_1,x_2), x_3)=\max(\min(x_1,x_3),\min(x_2,x_3)).$$

4. **Идемпотентности** операций **max** и **min**:

$$\max(x,x)=x; \ \min(x,x)=x.$$

5. Аналоги правил **де Моргана** в P_2:

$$\min(\sim x_1, \sim x_2)= \sim\max(x_1, x_2), \ \max(\sim x_1, \sim x_2)= \sim\min(x_1, x_2).$$

Следующие равенства вводятся по определению:

$$\max(x_1, x_2, \ldots, x_{n-1}, x_n) = \max(\max(x_1, x_2, \ldots, x_{n-1}), x_n)), \quad n \ge 3;$$

$$\min(x_1, x_2, \ldots, x_{n-1}, x_n) = \min(\min(x_1, x_2, \ldots, x_{n-1}), x_n)), \ n \ge 3;$$

$$-x=\begin{cases}0, & \text{если } x=0,\\ k-x, & \text{если } x\neq 0.\end{cases}$$

Рассмотрение свойств элементарных функций показывает, что не для всех обобщений булевых функций сохраняются соответствующие свойства. Например, **~(~x)=x**, но $\overline{\overline{x}}\neq x$ (при k≥3).

4.2 Разложение функций k-значных логик в первую и вторую формы

Любую функцию $\mathbf{f(x_1, x_2, \ldots, x_n)}$ из $\mathbf{P_k}$ **(n≥1)** можно представить в ***первой форме***, являющейся аналогом совершенной ДНФ для функции алгебры логики:

$$f(x_1,x_2,\ldots,x_n)=\max_{\tilde{\sigma}}\{\min(f(\sigma_1,\sigma_2,\ldots\sigma_n),J_{\sigma_1}(x_1),J_{\sigma 2}(x_2),\ldots,J_{\sigma_n}(x_n)\},$$

где максимум берется по всем наборам $\tilde{\sigma}=(\sigma_1,\sigma_2,\ldots,\sigma_n)$ значений переменных $\mathbf{x_1, x_2, \ldots, x_n}$.

Справедливо еще одно представление для функции *k*-значной логики, называемой ***второй формой***:

$$f(\tilde{x}^n)=\sum_{\tilde{\sigma}} f(\tilde{\sigma})\cdot j_{\sigma_1}(x_1)\cdot j_{\sigma 2}(x_2)\cdot\ldots\cdot j_{\sigma n}(x_n),$$

где суммирование ведется по всем наборам $\tilde{\sigma}=(\sigma_1,\sigma_2,\ldots,\sigma_n)$ значений переменных $\mathbf{x_1, x_2, \ldots, x_n}$ (сумма и произведение берутся по модулю **k**).

4.3 Замкнутые классы и полнота в k-значных логиках

Система **S** функций $\mathbf{f_1, f_2, \ldots, f_n}$, из $\mathbf{P_k}$ называется (функционально) ***полной***, если любая функция из $\mathbf{P_k}$ может быть записана в виде формулы через функции этой системы.

Приведем некоторые системы **S** полных систем.

1. Система $\mathbf{S=P_k}$ полна. Очевидно, что множество всех функций из $\mathbf{P_k}$ представляет полную систему.

2. Система

$\mathbf{S=\{0, 1, \ldots, k-1, J_0(x), \ldots, J_{k-1}(x), \min(x_1, x_2), \max(x_1, x_2)\}}$.

3. Система

$$S = \{\bar{x},\ \max(x_1, x_2)\}.$$

4. Система

$$S = \{ v_k(x_1, x_2)\}.$$

С понятием полноты связано понятие замыкания и замкнутого класса.

Пусть **M** – произвольное множество функций из P_k. **Замыканием M** называется множество **[M]** всех функций из P_k, представимых в виде формул через функции множества **M.**

Класс (множество) **M** называется (функционально) замкнутым, если **[M]=M**.

Функция $f(x_1, x_2, \ldots, x_n)$ из P_k $(n \geq 0)$ называется ***линейной***, если она представима в виде $a_0+a_1 \cdot x_1+\ldots+ a_n \cdot x_n$, где $a_j \in E_k$ (j=0, 1, …, n) и сумма, и произведение берутся по модулю **k**. Множество всех линейных функций из P_k образует замкнутый класс линейных функций, который обозначается через L_k (или **L**). Класс L_k отличен от P_k при всяком $k \geq 3$.

Полиномом (или многочленом) по модулю **k** от переменных $x_1, x_2, \ldots, x_n$ называется выражение вида $a_0+a_1 \cdot X_1+\ldots+ a_m \cdot X_m$, где коэффициенты a_i принадлежат множеству E_k и X_j – либо некоторая переменная из $\{x_1, x_2, \ldots, x_n\}$, либо произведение переменных этого множества (j=0, 1, …, n).

Говорят, что некоторая функция из P_k представима (или реализуется) полиномом по модулю **k,** если существует полином по модулю **k,** равный этой функции. Множество всех функций из P_k, представимых полиномами по модулю **k** (или, короче, множество всех полиномов по модулю **k**), является замкнутым классом в P_k.

Теорема (критерий полноты класса полиномов в P_k). Представление каждой функции из P_k полиномом по модулю **k** возможно в том и только том случае, когда **k** – простое число (иными словами, система полиномов по модулю **k** в P_k тогда и только тогда, когда **k** – простое число).

Необходимо отметить некоторые свойства, связанные с полной. Приведем их без доказательств:

1. Существует алгоритм для распознавания полноты.

2. Из всякой полной в P_k системы **S** можно выделить конечную подсистему, являющуюся также полной.

3. Класс **M** всех функций, сохраняющих **R,** является замкнутым. Функция $\mathbf{f(x_1, x_2, \ldots, x_n)}$ сохраняет множество **R**, если для любых функций $\mathbf{h_{i1}(y_1, y_2, \ldots, y_n)}$, $\mathbf{h_{i2}(y_1, y_2, \ldots, y_n)}$, …, $\mathbf{h_{in}(y_1, y_2, \ldots, y_n)}$ из **R**

$$\mathbf{f(h_{i1}(y_1, y_2, \ldots, y_n), h_{i2}(y_1, y_2, \ldots, y_n), \ldots, h_{in}(y_1, y_2, \ldots, y_n)) \in R.}$$

4. (О функциональной полноте). Можно построить систему замкнутых классов в $\mathbf{P_k}$ $\mathbf{M_1, M_2, \ldots, M_s}$, каждый из которых целиком не содержит ни одного из остальных классов, и такую, что подсистема функций из $\mathbf{P_k}$ полна тогда и только тогда, когда она целиком не содержится ни в одном из классов $\mathbf{M_1, M_2, \ldots, M_s}$.

5. Пусть система функций **S** функций из $\mathbf{P_k}$, где $k \geq 3$, содержит все функции одной переменной. Тогда для полноты системы **S** необходимо и достаточно, чтобы **S** содержала существенную функцию $\mathbf{f(x_1, x_2, \ldots, x_n)}$, принимающую все **k** значений.

6. Для всякого **k (k≥3)** существует в $\mathbf{P_k}$ замкнутый класс, не имеющий базиса.

7. Для всякого **k (k≥3)** существует в $\mathbf{P_k}$ замкнутый класс, со счетным базисом.

4.4 Задачи и упражнения

Пример 1.

Докажите справедливость неравенства

$$-(\overline{x}) = \sim x$$

Доказательство:

1) $\overline{x} = x + 1 (\mathrm{mod}\ k)$

2) $-(\overline{x}) = -(x + 1 (\mathrm{mod}\ k))$

получаем 2 случая:

а) получаемое число становится менее нуля,

из определения разности по модулю k:

$$-(\overline{x}) = k - ((x + 1 (\mathrm{mod} k)) - 0) = k - (x + 1) = k - x - 1.$$

б) получаемое число становится равным нулю,

$$-(\overline{x}) = -(0 + 1 (\mathrm{mod}\ k)) = -1 = k - 1.$$

С другой стороны: $\sim x = (k - 1) - x$.

Таким образом, в случаях а) и б) формулы приобретают одинаковые значения, что и требовалось доказать.

Пример 2.

Для k=3 представить функцию $f = \bar{x}$ в первой и второй формах (полученные выражения упростить)

Для представления функции в первой форме:

$$f(x_1, x_2, ..., x_n) = \max_{\tilde{\sigma}} \{\min(f(\sigma_1, \sigma_2, ...\sigma_n), J_{\sigma_1}(x_1), J_{\sigma 2}(x_2), ..., J_{\sigma_n}(x_n)\}$$

$$\bar{x} = \max\{\min(\bar{0}, J_0(x)), \min(\bar{1}, J_1(x)), \min(\bar{2}, J_2(x))\}$$

$$= \max\{\min(1, J_0(x)), \min(2, J_1(x)), \min(0, J_2(x))\}$$

Для представления функции во второй форме:

$$f(\tilde{x}^n) = \sum_{\tilde{\sigma}} f(\tilde{\sigma}) \cdot j_{\sigma_1}(x_1) \cdot j_{\sigma 2}(x_2) \cdot ... \cdot j_{\sigma n}(x_n)$$

$$\bar{x} = \bar{0} \cdot j_0(x) + \bar{1} \cdot j_1(x) + \bar{2} \cdot j_2(x) = 1 \cdot j_0(x) + 2 \cdot j_1(x) + 0 \cdot j_2(x) = j_0(x) + 2 \cdot j_1(x)$$

I. Докажите справедливость следующих неравенств

1) $x_1 \supset x_2 = \sim (x_1 \div x_2)$;

2) $x_1 \div (x_1 \div x_2) = \min(x_1, x_2)$;

3) $(x_1 \supset x_2) \supset x_2 = \max(x_1, x_2)$;

4) $(x_1 \supset x_2) + \bar{x}_1 = \min(x_1, x_2)$;

5) $x_1 \div x_2 = x_1 - \min(x_1, x_2)$;

6) $x_1 \div x_2 = \max(x_1, x_2) - x_2$;

7) $(\sim x_1) \div (x_2 \div x_1) = \sim \max(x_1, x_2)$;

8) $(\sim x_1) \div (\sim x_2) = x_2 \div x_1$;

9) $\sim (\bar{x}_1 + x_2) = (\sim x_1) + (\sim x_2)$;

10) $\overline{\sim (\bar{x}_1 \cdot \bar{x}_2)} = (\sim x_1) \cdot \bar{x}_2$;

11) $\max((x + 2) \div 1, J_{k-2}(x)) = \bar{x}$;

12) $\min(\sim J_{k-1}(x), (k-2) \supset x) = \bar{x}$;

13) $\bar{x}_1 \div \bar{x}_2 = (x_1 \div x_2) + \bar{x}_1 \cdot j_{k-1}(x_2) + \bar{x}_2 \cdot j_{k-1}(x_1)$;

14) $v_k(x_1, x_2) + \bar{x}_1 \cdot j_{k-1}(x_2) + \bar{x}_2 \cdot j_{k-1}(x_1) = \max(\bar{x}_1, \bar{x}_2)$;

15) $\max(x_1, x_2) + j_0(x_2 \div x_1) + j_k(x_1) \cdot x_2 = \max(\bar{x}_1, x_2)$;

16) $\overline{\min(x_1, x_2)} + J_0(x_2 \div x_1) - j_{k-1}(x_1) \cdot x_2 = \min(\bar{x}_1, x_2)$;

17) $J_0(\max(J_0(x), J_1(x), ..., J_{k-2}(x))) = J_{k-1}(x)$;

18) $J_1(\max(x, 1, J_1(x), J_2(x), ..., J_{k-2}(x))) = J_0(x)$;

19) $x \cdot j_0(j_1(x)) + j_0(x) \cdot \overline{j_1(x)} = x + j_0(x) - j_1(x)$;

20) $J_0(x \div i) \div J_0(x \div (i-1)) = J_i(x)$, *i*=1, 2, …, k–1.

II. При каких значениях k (k≥3) функции x^2, x^3 и x^4 попарно различны?

III. Для заданного k представить функцию f в первой и второй формах (полученные выражения упростить)

1) f=~ x, k=4;
2) f=–j_0(x), k=5;
3) f=2J_1(x), k=6;
4) f=J_2(x^2+x), k=5;
5) f=(~x)^2+x, k=4;
6) f=3j_1(x) – j_3(x), k=4;
7) f=x_1+2x_2, k=3;
8) f=max (x_1, x_2), k=3;
9) f= $x_1 \div x_2^2$, k=3;
10) f= $x_1^2 \cdot x_2$, k=3.

5 ЛОГИКА ВЫСКАЗЫВАНИЙ

Логика высказываний является разделом математической логики, в котором рассматриваются сложные предложения, получающиеся из предложений, принимаемых за элементарные высказывания, соединенных союзами «И», «ИЛИ», «ИЛИ…ИЛИ», «ЕСЛИ…, ТО», «ТОГДА И ТОЛЬКО ТОГДА, КОГДА» и присоединением к ним частицы «НЕ».

Высказывание – это предложение, которое может оцениваться по его истинности, а не с точки зрения его содержания.

Неделимое высказывание называется ***элементарным***.

Сложные высказывания соединяются логическими связями или связками «И», «ИЛИ», «ИЛИ…ИЛИ», «ЕСЛИ…, ТО», «ТОГДА И ТОЛЬКО ТОГДА, КОГДА» и частицей «НЕ».

Логика высказываний занимается не смыслом высказывания, а анализирует, истинно оно или ложно.

Про истинное предложение говорят, что его логическим значением является ***истина***, а про ложное – что его логическим значением является ***ложь***.

Примеры элементарных высказываний:

«пять – нечетное число», «трава голубая», «Томск – столица Сибири».

Следующие предложения не являются высказываниями: «уходя, гасите свет», «сколько Вам лет?» и т.п. Такого типа выражения в логике высказываний не рассматриваются.

В естественном языке союзы и частица «НЕ» имеют не вполне отчетливое значение, а некоторые из них могут употребляться в различных смыслах. Например, союз «ИЛИ» может быть разделительным (как в фразе «выбирай, он или я») или неразделительным (как в фразе «от шума или света я проснулся», в которой не исключается, что «я проснулся» от общих причин).

В логике высказываний принято ставить в соответствие высказываниям буквы и называть их логическими переменными.

Рассмотрим выражение: *Если в следующее воскресенье будет плохая погода и я не достану билет на концерт, то я схожу в кино или буду готовиться к зачету*. Разобьем это высказывание на элементарные высказывания и обозначим их буквами:

а – в следующее воскресенье будет плохая погода;

b – я достану билет на концерт;

c – я схожу в кино;

d – буду готовиться к зачету.

Выражение примет следующий вид:

Если a и не b, то c или d

Для сокращения письма связки обозначаются соответствующими знаками: ¬, ∧, ∨, ~, ⊗, →. Смысл операций (связок) устанавливается соответствующими таблицами, поскольку определить операцию – это значит определить истинность высказывания для каждого значения логических переменных.

1. ***Отрицание*** **¬a, ā, не a**; Ложь обозначим буквой Л, истину – И.

a	ā
Л	И
И	Л

2. ***Конъюнкция*** **a ∧ b**, (**a и b**, (**a & b**), (**a** конъюнкция **b**). Эту операцию называют логическим умножением.

a	b	a ∧ b
Л	Л	Л
Л	И	Л
И	Л	Л
И	И	И

Пример: 5 > 2 и 7 четное число. Оценим истинность данного высказывания. 5 > 2 – истина; 7 четное число – ложь; в результате исходное выражение ложно.

3. ***Дизъюнкция*** **a ∨ b**, (**a или b**). Операция логического сложения.

a	b	a ∨ b
Л	Л	Л
Л	И	И
И	Л	И
И	И	И

4. *Импликация* a → b (если a, то b).

a	b	a → b
Л	Л	И
Л	И	И
И	Л	Л
И	И	И

5. ***Эквивалентность a ~ b***, (**a** эквивалентно **b**), ***(a если и только если b)***.

a	b	a ~ b
Л	Л	И
Л	И	Л
И	Л	Л
И	И	И

6. ***Дизъюнкция с исключением ⊕; (или a или b)***.

a	b	a ⊗ b
Л	Л	Л
Л	И	И
И	Л	И
И	И	Л

Рассмотрим предыдущий пример. В соответствии с введенными операциями он будет выглядеть:

$$a \wedge \neg b \rightarrow c \vee d.$$

Вычислим истинность этого высказывания. Для этого необходимо построить таблицу истинности от четырех переменных, в которой будет 2^4 строк, и вычислить значение на каждом наборе переменных.

a	b	c	d	a ∧ ¬ b → c ∨ d
Л	Л	Л	Л	И
Л	Л	Л	И	И
Л	Л	И	Л	И
Л	Л	И	И	И
Л	И	Л	Л	И
Л	И	Л	И	И
Л	И	И	Л	И
Л	И	И	И	И
И	Л	Л	Л	Л
И	Л	Л	И	И
И	Л	И	Л	И
И	Л	И	И	И
И	И	Л	Л	И
И	И	Л	И	И
И	И	И	Л	И
И	И	И	И	И

Это высказывание ложно только в одном случае, когда высказывания a = И, b = Л, c = Л, d = Л.

Логика высказываний не дает методов вычисления истинности элементарных высказываний, но хорошо определяет истинность сложных высказываний.

Если высказывание сложное и стоит задача определить истинность высказывания, то необходимо построить формулу или систему формул. Далее необходимо определить множество правильных формул, затем выполнить действия и получить ответ.

1. Определим алфавит, т.е. символы, которыми можно пользоваться:

a, b, c,…, x, y, z, ∧, ∨, ¬, →, ⊕,), (.

2. Будем утверждать, что **a, b, c,…, x, y, z** – формулы.

3. Если **a, b** – формулы, то формулами являются выражения:

(¬a), (a∨b), (a∧b), (a→B), (a~b), (a⊕b).

4. Других формул нет.

Примеры:

((a → b) ∧ (¬(c ∨ a))), ((a ∨ (¬b)) ⊕ a) – формулы.

(a¬b)– не является формулой.

Введем правила упрощенного написания формул.

Прежде всего, определим приоритетность выполнения операций. Для этого разобьем их на группы и запишем в порядке уменьшения приоритета.

¬; ∧; ∨; ⊕; →; ~;

Если перед нами формула, то, прежде всего, выполняются операции в скобках и операция отрицания, затем – конъюнкция. Операции дизъюнкция и дизъюнкция с исключением имеют одинаковый приоритет. Если необходимо какую-либо из них выполнять первой, то надо уточнить, используя скобки. То же самое касается операций импликации и эквивалентности, которые имеют наиболее низкий приоритет.

Для упрощения написания можно опускать знак конъюнкции.

И последнее. Символ отрицания можно помещать над переменной, скобкой в виде черты.

Пример:

$$(a \to b)(c \vee a), \quad (a \vee \bar{b}) \oplus a.$$

Рассмотрим формулы А и В. Визуально формулы могут быть различны, но может оказаться, что для каждого набора значений переменных значения *ложь* и *истина* совпадают, тогда говорят, что формулы А и В ***равносильны***, т.е. А=В; А = И – формула истинна, если через И обозначить формулу, которая всегда истинна. А=Л – формула всегда ложна. Необходимо заметить, что под символом = понимается отношение равенства.

5.1 Тождества в алгебре высказываний

Пусть формула **А** зависит от списка переменных $\mathbf{x_1, x_2, \ldots, x_k}$. Формула **А** называется ***тавтологией*** (тождественно-истинной), если при любом значении переменных $\mathbf{x_1, x_2, \ldots, x_k}$ формула **А** принимает значение истина. То есть, тождества – это такие формулы, которые обращаются в **истину** при любой комбинации переменных. Рассмотрим основные тождества (законы).

1. ***Закон тождества.*** Всякое высказывание является логическим следствием самого себя:

$$\mathbf{x \to x}$$

2. ***Закон противоречия***. Для всякого высказывания x неверно, что истинно само высказывание и его отрицание:

$$\overline{(x \wedge \overline{x})}, \quad \neg(x \wedge \neg x)$$

3. ***Закон исключенного третьего***. Для каждого высказывания x истинно само высказывание или его отрицание:

$$\mathbf{x \vee \neg x}$$

4. ***Закон двойного отрицания***. Каково бы ни было высказывание x, отрицание его отрицания эквивалентно самому высказыванию:

$$\mathbf{\neg\neg x \sim x}$$

5. ***Истина из чего угодно***. Если **x** истина, то каково бы ни было y высказывание **y→x** – истина:

$$\mathbf{x \rightarrow (y \rightarrow x)}$$

6. ***Из ложного – что угодно***. Если **x** истина, то **¬x** – ложь. Ложь имплицирует все, что угодно:

$$\mathbf{\neg x \rightarrow (x \rightarrow y)}$$

7. ***Modus ponens*** (правило отделения). Если **x** истина и **x→y** – истина, то **y** – истина:

$$\mathbf{(x \wedge (x \rightarrow y)) \rightarrow y}$$

8. ***Modus tollens*** (правило устранения). Если **x** имплицирует **y** и **y** ложно, то **x** ложно:

$$\mathbf{((x \rightarrow y) \wedge \neg y) \rightarrow \neg x}$$

9. ***Закон силлогизма***. Если из **x** следует **y** и из **y** следует **z**, то из **x** следует **z:**

$$\mathbf{(x \rightarrow y) \wedge (y \rightarrow z) \rightarrow (x \rightarrow z)}$$

10. ***Тривиальные тождества***:

$$\mathbf{Л \rightarrow А, \quad А \rightarrow И.}$$

5.2 Булевы формулы

Булевыми формулами назовем такие формулы, в которых отсутствуют знаки операций **→; ~; ⊕.** Рассмотрим основные равносильности булевых формул. Эти равносильности носят название законов. Доказательство законов можно провести с помощью таблиц истинностей. Пусть A, B и C – формулы. Тогда для них справедливы следующие законы:

1. **Коммутативные**:
$A \vee B = B \vee A$,
$A \wedge B = B \wedge A$.
2. Ассоциативные:
$A \vee (B \vee C) = (A \vee B) \vee C$,
$A \wedge (B \wedge C) = (A \wedge B) \wedge C$.
3. **Идемпотентности**:
$A \vee A = A$,
$A \wedge A = A$.
4. Дистрибутивные:
$(A \vee B)C = AC \vee BC$,
$A \vee BC = (A \vee B)(A \vee C)$.
5. Де Моргана:
$\overline{A \vee B} = \overline{A} \wedge \overline{B}$,

$\overline{AB} = \overline{A} \vee \overline{B}$.

6. Двойного отрицания:
$\overline{\overline{A}} = A$.

7. $A \vee \bar{A} = И$, $A \wedge \bar{A} = Л$,
$A \vee Л = A$, $A \wedge Л = Л$,
$A \vee И = И$, $A \wedge И = A$.
8. $\overline{Л} = И$, $\overline{И} = Л$.

5.3 Интерпретации

Определим формальную систему, в которой заданы переменные **a, b, c,…;** операции над переменными **∨, ∧, ¬**; правила построения правильных формул; для придания более общего характера, заменим **Л** и **И** на **0** и **1**. В результате получим булеву алгебру.

Интерпретации:

1. *Булева алгебра высказываний*. Считается, что **a, b, c,… –** высказывания. Значения **0** и **1** кодируем значениями **Л** и **И**. Операции рассматриваются как логические связки **НЕ**, **ИЛИ**, **И**.

2. *Булева алгебра множеств*. Считаем, что **a, b, c,… –** множества, **0** и **1** интерпретируются как $\varnothing$ и **T**, а операции: как дополнение $\bar{}$, объединение $\cup$, пересечение $\cap$.

3. *Булева алгебра событий*. Переменные **a, b, c,… –** представляют события. Событие имеет место или нет. Несомненное событие обозначается **1**. Если событие не наступило – **0**. Операции представляются символами $\vee$, $\wedge$, $\neg$. Здесь $\neg$ – отрицание события, $\vee$ – сумма событий, $\wedge$ – произведение событий. Операциям придается определенный смысл. Сумма событий – это событие, которое наступает, когда, по крайней мере, наступает одно из этих событий **a**, **b**. Произведение событий – событие, которое наступает тогда, когда оба события имеют место. Алгебра событий является фундаментом теории вероятностей.

4. *Теория электрических цепей*. Используются те же самые булевы формулы. Переменные **a, b, c,…** ставятся в соответствие электрическим цепям. Интерпретация рассматривается с точки зрения проводит цепь ток или нет. Цепь может находиться в двух состояниях: проводимом и не проводимом

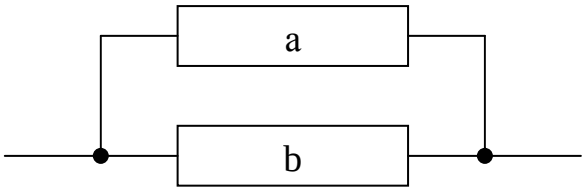


Рисунок 5.1 – дизъюнкция

a$\vee$b – означает параллельное соединение двух цепей, ток проходит, если проводит a или b.

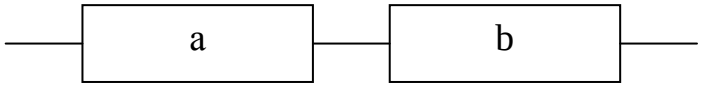


Рисунок 5.2 – Конъюнкция

Последовательное соединение цепей – **a $\wedge$ b.**

Операция отрицания $\neg$ – способ построения такой цепи, проводимость которой противоположна основной.

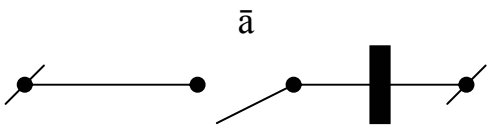


Рисунок 5.3 – Инвертирование цепи

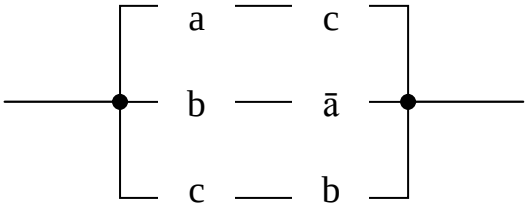


Рисунок 5.3 – $ac \vee b\bar{a} \vee cb$

6 БУЛЕВЫ ФУНКЦИИ

Двоичная функция двоичного аргумента называется ***булевой функцией.***

$y = f(x_1, x_2,\ldots, x_n)$, $x_i \in \{0,1\}$, $y \in \{0,1\}$, $i = \overline{1,n}$.

Система булевых функций задается следующим образом:

$$\begin{cases} y_1 = f_1(x_1,...,x_n), \\ y_k = f_k(x_1,...,x_n). \end{cases}$$

Будем говорить, что две функции равны, если их значения совпадают на любой комбинации значений переменных.

$$f(x_1,\ldots, x_n) = g(x_1,\ldots, x_n).$$

Различными считаются функции, не совпадающие хотя бы на одной комбинации переменных.

Число различных функций равно 2^{2^n}.

Пусть n = 0, тогда $2^{2^0} = 2$, т.е. функция принимает значение **0** или **1**.

При n=1, число различных функций равно 4.

x	функция			
	нуль	тождествен-на	отрицатель-на	едини-ца
0	0	0	1	1
1	0	1	0	1
	0	x	x', ¬x	1

При n=2, число различных функций равно 16.

Функции																	
x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Приведем эти булевы функции.

$f_1(x_1, x_2) = 0$ – константа 0;

$f_2(x_1, x_2) = x_1 \wedge x_2$ – конъюнкция;

$f_3(x_1, x_2) = x_1 \wedge \overline{x}_2 = \overline{\overline{x}_1 \vee x_2}$

$= x_1 \not\to x_2$ – левая коимпликация (читается «не если x_1, то x_2»);

$f_4(x_1, x_2) = \overline{x}_1 \wedge \overline{x}_2 \vee x_1 \wedge x_2 = x_1$;

$f_5(x_1, x_2) = \overline{x}_1 \wedge x_2 = x_1 \not\leftarrow x_2 = \overline{x_1 \leftarrow x_2} = \overline{x_1 \vee \overline{x}_2}$

правая коимпликация;

$f_6(x_1, x_2) = \overline{x}_1 \wedge x_2 \vee x_1 \wedge x_2 = x_2$;

$f_7(x_1, x_2) = \overline{x}_1 \wedge x_2 \vee x_1 \wedge \overline{x}_2 = x_1 \oplus x_2$ – сложение по модулю два, дизъюнкция с исключением;

$f_8(x_1, x_2) = x_1 \vee x_2$ – дизъюнкция;

$f_9(x_1, x_2) = \overline{x}_1 \wedge \overline{x}_2 = \overline{x_1 \vee x_2} = x_1 \circ x_2$ – функция Вебба;

$f_{10}(x_1, x_2) = \overline{x}_1 \wedge \overline{x}_2 = x_1 \sim x_2$ – функция эквивалентности;

$f_{11}(x_1, x_2) = \overline{x}_2$ – отрицание;

$f_{12}(x_1, x_2) = \overline{x}_1 \wedge \overline{x}_2 \vee x_1 \wedge \overline{x}_2 \vee x_1 \wedge x_2 = \overline{x}_2 \vee x_1 = x_1 \leftarrow x_2$ – правая импликация (читается «если x_2, то x_1»);

$f_{13}(x_1, x_2) = \overline{x}_1$ – отрицание;

$f_{14}(x_1, x_2) = \overline{x}_1 \wedge \overline{x}_1 \vee \overline{x}_2 \wedge x_2 \vee x_1 \wedge x_2 = \overline{x}_1 \vee x_2 = x_1 \to x_2$ – левая импликация (читается «если x_1, то x_2»);

$f_{15}(x_1, x_2) = \overline{x}_1 \wedge \overline{x}_2 \vee \overline{x}_1 \wedge x_1 \wedge \overline{x}_1 = \overline{x}_2 \vee \overline{x}_2 = x_1 | x_2$ – функция Шеффера;

$f_{16}(x_1, x_2) = 1$ – константа 1.

6.1 Способы задания булевой функции

1. Табличный способ

Область определения булевой функции – совокупность всевозможных наборов переменных, состоящих из нулей и единиц. Поэтому для задания булевой функции достаточно задать значение функции на всех наборах переменных.

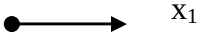
Естественным расположением наборов является расположение в порядке возрастания десятичного числа, соответствующего данному набору.

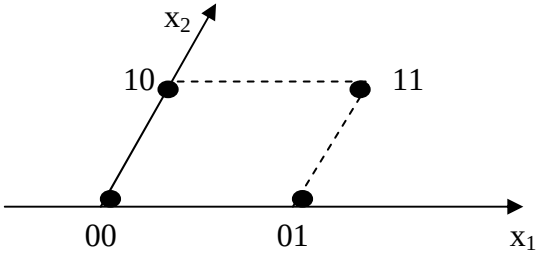
№ набора	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	0
8	1	1	1	1

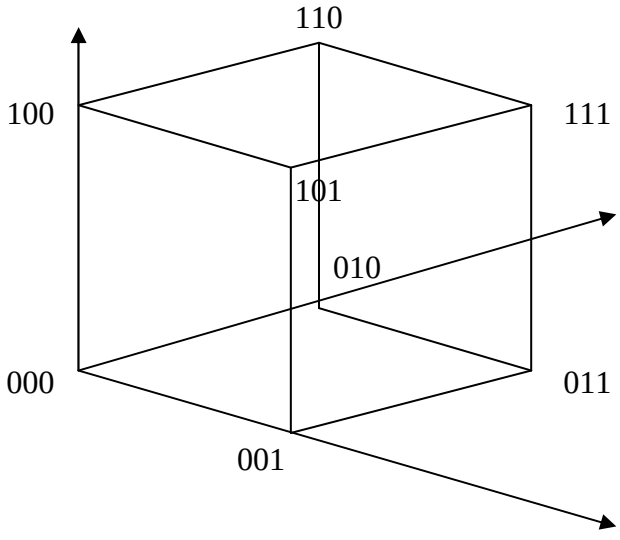
2. Представление вершинами n-мерного куба.

Множество значений вектора $x=(x_1, x_2,\ldots, x_n)$ составляет булево пространство. Каждому значению вектора сопоставлен элемент пространства. Число компонент вектора определяет размерность пространства.

n = 0 

n = 1 

n = 2 

n = 3 

Расстояние между вершинами n-мерного куба есть число компонент, значениями которых эти вершины отличаются. Также расстояние называется ***расстоянием по Хемингу***. Соседние вершины n-мерного куба различаются одной компонентой.

3. Задание булевой функции формулами.

Пусть F = {f_1, f_2, …, f_n} – множество булевых функций. Формулой над F называется выражение вида $F[F] = f(t_1,\ldots, t_n)$, где $f \in F$ и t_i, либо переменная, либо формула над F. Множество F называется ***базисом***, функция f называется ***главной (внешней) операцией (функцией)***, а t_i называются ***подформулами***.

Систему функций будем называть ***функционально полной***, если любая булева функция может быть представлена в виде суперпозиции функций этой системы. Функционально полная система называется ***базисом***. Базис называется ***безизбыточным***, если ни одну из функций базиса нельзя исключить так, чтобы оставшаяся система функций была функционально полной. Базис называется ***минимальным***, если он содержит наименьшее из возможных число функций.

Существует 17 базисов, в каждом из которых нельзя вычеркнуть ни одну функцию без потери полноты.

Базис Вебба – { 0 };

Базис Шеффера – { | };

{ $\not\rightarrow$, ~ };

Импликативный базис – {$\rightarrow$, 0};

{$\rightarrow$, $\not\rightarrow$};

{ $\not\rightarrow$, $-$} – коимпликативный базис;

{$\rightarrow$, $\oplus$};

{$\rightarrow$, $\bar{}$} – импликативный базис;

{&, $\bar{}$} – конъюнктивный базис Буля;

{$\vee$, $\bar{}$} – дизъюнктивный базис Буля;

{ $\not\rightarrow$, 1} – коимпликативный базис;

{~, &, 0};

{~, $\vee$, 0};

{$\oplus$, &, ~};

{$\oplus$, $\vee$, ~};

{$\oplus$, &, 1} – базис Жегалкина;

{$\oplus$, $\vee$, 1}.

Техническая реализация базисных функций может быть основана на использовании различных физических явлений, например, импликация и коимпликация может быть основана на использовании магнитных явлений, а функции Шеффера и Вебба – на использовании явлений в полупроводниках.

6.2 Равносильные преобразования формул

Две формулы, представляющие одну и ту же функцию, называются **равносильными.** Преобразования, приводящие некоторую формулу к равносильной ей формуле, называются ***равносильными***. Булева формула может быть представлена большим количеством равносильных формул. Некоторые представляют интерес. Например, формулы, содержащие наименьшее число букв, или формулы, содержащие только некоторые символы операций из множества элементарных операций.

Теория булевых функций занимается изучением специальных функций и равносильных преобразований, приводящих к этим функциям.

Основные свойства элементарных формул (основные равносильности):

1. закон идемпотентности:

$a \vee a = a$, $a \wedge a = a$;

2. закон коммутативности:

$a \vee b = b \vee a$, $a \wedge b = b \wedge a$;

3. закон ассоциативности:

$a \vee (b \vee c) = (a \vee b) \vee c$,

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

4. закон дистрибутивности:

$a \wedge (b \vee c) = a \wedge b \vee a \wedge c$,

$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$;

5. закон двойного отрицания:

$\bar{\bar{a}} = a$;

6. закон де Моргана:

$\bar{a} \vee b = \overline{a \wedge b}$, $\bar{a} \wedge \bar{b} = \overline{a \vee b}$;

7. законы склеивания:

$a \wedge b \vee a \wedge \bar{b} = a$, $(a \vee b) \wedge (a \vee \bar{b}) = a$;

8. законы поглощения:

$a \vee a \wedge b = a,\ a \wedge (a \vee b) = a;$

9. законы Порецкого:

$a \vee \bar{a} \wedge b = a \vee b,\ a \wedge (\bar{a} \vee b) = a \wedge b;$

10. законы, определяющие действия с константами 0 или 1:

$a \vee = a, \qquad a \wedge 0 = 0, \qquad a \vee 1 = 1,$

$a \wedge 1 = a, \qquad a \vee \bar{a} = 1, \qquad a \wedge \bar{a} = 0.$

Правило подстановки: если в равносильных формулах вместо всех вхождений некоторой переменной x подставить одну и ту же формулу, то получатся равносильные формулы.

Правило замены: если в формуле заменить некоторую подформулу на равносильную, то получится равносильная формула.

6.3 Нормальные формулы. Совершенные нормальные формулы

Обозначим $x^o = \bar{x}$, $x^1 = x$,

$$x^{\delta} = \begin{cases} x, \delta = 1; \\ \bar{x}, \delta = 0. \end{cases}$$

Любую конъюнкцию ранга n можно представить в виде:

Элементарной конъюнкцией называется конъюнкция переменных, некоторые из которых могут быть взяты со знаком отрицания, причем переменная в конъюнкции не должна встречаться более одного раза.

Или: элементарной конъюнкцией называется выражение вида $x_1^{\delta_1}, x_2^{\delta_2} \ldots x_n^{\delta_n}$, $x_1^{\delta_1}, x_2^{\delta_2} \ldots x_n^{\delta_n}$.

То есть элементарная конъюнкция есть логическое произведение переменных, взятых в некоторой степени δ, в которой ни одна переменная не употребляется два раза.

Число переменных в конъюнкции или дизъюнкции назовем ее ***рангом***.

Дизъюнкция различных элементарных конъюнкций называется ***дизъюнктивной нормальной формой*** (***ДНФ***).

Конъюнкция различных элементарных дизъюнкций называется ***конъюнктивной нормальной формой*** (***КНФ***).

Число конъюнкций в ДНФ называется длиной ДНФ.

Если мы возьмем любую формулу из символов a, b, c,…, x, y, z, ∧, ∨, ¬, →, ~, ⊕, (,), то можно ее заменить равносильной ДНФ.

$a \to b = \bar{a} \vee b$, $a \sim b = \bar{a}\,\bar{b} \vee a\,b$, $a \oplus b = a\bar{b} \vee \bar{a}\,b$.

Пример $(a \wedge \bar{b}) \to (c \vee d) = \overline{(a \wedge \bar{b})} \vee c \vee d = \bar{a} \vee b \vee c \vee d$;

ДНФ весьма просто получается из таблицы истинности. Достаточно поставить в соответствие каждому значению вектор аргумента, на котором функция принимает значение 1, элементарную конъюнкцию, называемую ***полной*** и состоящую из всех аргументов. Такая ДНФ, членами которой являются неповторяющиеся полные элементарные конъюнкции, называется ***совершенной ДНФ (СДНФ),*** а ее члены ***константуэнтами единицы***. С ДНФ любой булевой функции единственна с точностью до порядка следования членов и литералов (чего нельзя сказать о ДНФ) и является, как говорят, ***канонической формой***.

Пример. Пусть задана функция $(a \oplus b) \to (c \sim a)$.

Построим таблицу истинности.

a	b	c	a ⊕ b	c ~ a	(a ⊕ b) → (c ~ a)
0	0	0	0	1	1
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	1
0	1	0	0	0	1
1	1	1	0	1	1

Построим СДНФ.

$$\bar{a}\bar{b}\bar{c} \vee \bar{a}\bar{b}c \vee \bar{a}b\bar{c} \vee a\bar{b}c \vee ab\bar{c} \vee abc$$

КНФ так же удобна для представления нулей булевой функции, как ДНФ для представления единиц.

СКНФ для предыдущего примера запишется

$$(\bar{a} \vee b \vee c)(a \vee \bar{b} \vee \bar{c})$$

Каждая из составляющих ее полных элементарных дизъюнкций является ***конституэнтой нуля*** и определяет значение вектор аргумента, на котором функция обращается в нуль.

6.4 Разложение Шеннона. Декомпозиция булевых функций

Рассмотрим разложение булевой функции $f(x_1, x_2,\ldots, x_n)$ по k переменным $(x_1,\ldots, x_k)$ – разложение Шеннона.

Теорема 1. Любая функция $f(x_1, x_2,\ldots, x_n)$, не равная тождественно нулю, представлена в виде разложения Шеннона:

$$f(x_1,x_2,\ldots,x_n,x_{k+1},\ldots,x_n)=\bigvee_{\forall(\delta_1,\delta_2,\ldots,\delta_k)}\ \&_{i=1}^{k} x_i^{\delta_i}(\delta_1,\delta_2,\ldots,\delta_k,\delta_{k+1},\ldots,x_n).$$

Заметим, что $x_1^{\delta_1},x_2^{\delta_2},\ldots,x_k^{\delta_k}=1$, если $x_i = \delta_i$, для $\forall_i = \overline{1,k}$. Выберем набор $\delta_1, \delta_2,\ldots, \delta_k$ и положим, что $x_i = \delta_i$, $i = \overline{1,k}$. Тогда левая часть будет равна:

$f(x_1, x_2,\ldots, x_k, x_{k+1}, \ldots, x_n) = f(\delta_1,\ldots, \delta_k, x_{k+1}, \ldots, x_n)$,

а правая:

$$\bigvee x_1^{\delta_1},\ldots,x_k^{\delta_k} f(\delta_1,\delta_2,\ldots,\delta_k,x_{k+1},\ldots,x_n)=f(\delta_1,\delta_2,\ldots,\delta_k,x_{k+1},\ldots,x_n)\cdot$$

Здесь $x_1^{\delta_1},..,x_k^{\delta_k}$ разбиваются на единичные и нулевые наборы. Согласно закону $a\vee 0 = a$, получаем, что левая и правая части формул равны при любой подстановке переменных $x_1, x_2,\ldots,x_k$.

Теорема 2. Любая булева функция может быть представлена формулой, являющейся суперпозицией $\vee$, $\wedge$, $\neg$.

Доказательство.

1) В начале докажем для f=1 или f=0.

$f(x_1, x_2,\ldots, x_n) = 1 = \overline{x_i} \vee x_i$;

$f(x_1, x_2,\ldots, x_n) = 0 = \overline{x_i} \wedge x_i$;

2) Докажем для функции, не равной константе,

$f(x_1, x_2,\ldots, x_n)$ разложим по n переменным.

Получим

$$\bigvee_{\delta_1,\ldots,\delta_n} x_1^{\delta_1}\ldots x_n^{\delta_n} f(\delta_1\ldots\delta_n) = \bigvee x_1^{\delta_1}\ldots x_n^{\delta}.$$

Это совершенная ДНФ. Из этой формулы следует способ построения СДНФ.

Заметим, что $\bar{f}$ может быть разложена

$$\bar{f} = (x_1,...,x_n) = \bigvee_{\delta_1,...,\delta_n} x_1^{\delta_1},...,x_n^{\delta_n} \bar{f}(\delta_1,...,\delta_n)$$

(в булевой алгебре справедлив принцип двойственности).

Согласно закону двойного отрицания

$$f(x_1, x_2,...,x_n) = \bar{\bar{f}}(x_1, x_2,...,x_k, x_{k+1},...,x_n) = \overline{\bigvee_{\delta_1,...,\delta_n} x_1^{\delta_1},...,x_n^{\delta_n} \bar{f}(\delta_1,...,\delta_n)} =$$

$$= \bigwedge_{\delta_1,..,\delta_n} \left(x_1^{\bar{\delta}_1} \vee x_2^{\bar{\delta}_2} \vee ... \vee x_n^{\bar{\delta}_n} \vee f(\delta_1, \delta_2,...,\delta_n) \right) = \bigwedge_{\delta_1,..,\delta_n} \left(x_1^{\bar{\delta}_1} \vee x_2^{\bar{\delta}_2} \vee ... \vee x_n^{\bar{\delta}_n} \right). \quad (1)$$

Таким образом, любая булева функция $f(x_1, x_2,\ldots, x_n)$, не равная тождественно единице, представлена в виде выражения 1.

Покажем связь между разложением Шеннона и таблицами Вейча.

Представим пространство $P_n(X)$ в виде декартова произведения пространств $P_k(X_a)$ и $P_g(X_b)$, $X_a \cup X_b = X$, $X_a \cap X_b = \varnothing$, $k + g = n$:

$P_n(X) = P_k(X_a) \times P_g(X_b)$.

Каждой строке таблицы Вейча взаимно однозначно сопоставим точку пространства $P_k(X_a)$, столбцу – точку пространства $P_g(X_b)$ и рассмотрим разложение Шеннона булевой функции

$$f\left(x_{a_1}, x_{a_2},...,x_{a_k}, x_{b_1}, x_{b_2},...,x_{b_g}\right)$$

по первым k переменным. Тогда i-строка таблицы Вейча, идентифицируемая конъюнкцией $x_{a_1}^{\delta_1} \wedge x_{a_2}^{\delta_2} \wedge ... \wedge x_{a_k}^{\delta_k}$, соответствует остаточной функции

$$f\left(x_{a_1}, x_{a_2},..., x_{a_k}, x_{b_1}, x_{b_2},..., x_{b_g}\right).$$

Будем называть разложение Шеннона булевой функции f(X) строчным, если разложение осуществляется по переменным, соответствующим строкам таблицы Вейча.

6.5 Представление булевой функции картами Карно (Вейча)

Карта Карно – это диаграмма, состоящая из правильно расположенных квадратов, каждый из которых соответствует одной из 2^n полных конъюнкций соответствующих функции от n переменных. Значения данной функции f из таблицы истинностей вносят в нужные квадраты. Тогда функция f равна дизъюнкции всех полных конъюнкций, для которых в соответствующих квадратах стоит единица.

Рассмотрим построение карт Карно.

Пусть задана функция от двух переменных. Тогда карта Карно будет выглядеть

		x_1
	00	01
x_2	10	11

Чтобы наборы не мешали внутри клеток, их можно вынести за пределы таблицы

			x_1
		0	1
	0		
x_2	1		

В карте Карно соседние клетки различаются одной компонентой. Для кодировки используется код Грея. Код Грея получается из обыкновенного весового кода сложением по mod 2. Искомое число вычисляется следующим образом. Берем весовой код в качестве первого слагаемого, в качестве второго слагаемого берем тот же код, но сдвинутый на один разряд вправо. Производим сложение по mod 2. Тогда карта Карно для трех переменных будет выглядеть:

	0	0	1	1	x_1
	0	1	1	0	x_2
0					
1					
x_3					

Если вместо единиц поставить черту, а нули не писать, тогда эта же карта будет выглядеть следующим образом:

					x_1
					x_2
x_3					

Карты для четырех и пяти переменных представлены ниже.

						x_1
						x_2
x_3	x_4					

										x_1
										x_2
										x_3
x_4	x_5									

Необходимо отметить, что карта Карно обладает «зеркальной» симметрией и все соседние компоненты находятся на расстоянии один.

Представим функцию от четырех переменных картой Карно. Пусть задана функция от четырех переменных в табличной форме. Покажем наборы, на которых функция принимает единичное значение. Отметим их на карте точками.

x_1	x_2	x_3	x_4	F	Наборы
0	0	0	0	0	
0	0	0	1	0	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	$\neg x_1\, x_2\, \neg x_3\, x_4$
0	1	1	0	1	$\neg x_1\, x_2\, x_3\, \neg x_4$
0	1	1	1	1	$\neg x_1\, x_2\, x_3\, x_4$
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	0	
1	1	0	0	0	
1	1	0	1	1	$x_1\, x_2\, \neg x_3\, x_4$
1	1	1	0	0	
1	1	1	1	0	

						x_1
						x_2
			·	·		
			·			
			·			
x_3	x_4					

Зададим функцию в виде ДНФ, представим ее картой Карно.

$F(x_1\, x_2\, x_3\, x_4) = x_1\, x_2\, x_4 \vee x_1 \neg x_3\, x_4 \vee x_3\, \neg x_4 \vee x_1\, x_2\, \neg x_3$

						x_1
						x_2
				·		
				·	·	
				·		
		·	·	·	·	
x_3	x_4					

СДНФ вышеприведенной функции запишется

$F(x_1,x_2,x_3,x_4) = x_1\, x_2\, x_3\, x_4 \vee x_1\, x_2\, \neg x_3\, x_4 \vee x_1\, x_2 \neg x_3 \neg x_4 \vee x_1\, \neg x_2 \neg x_3\, x_4 \vee \neg x_1 \neg x_2\, x_3 \neg x_4 \vee \neg x_1 x_2\, x_3 \neg x_4 \vee x_1\, x_2 x_3 \neg x_4 \vee x_1 \neg\, x_2 x_3 \neg\, x_4$

6.6 Минимизация булевых функций

Импликантой функции f (x_1,...,x_n) называется такая элементарная конъюнкция К над множеством переменных { x_1,...,x_n }, что К $\vee$ f (x_1,...,x_n) = f (x_1,...,x_n). Импликанта называется ***простой***, если при отбрасывании любой буквы из К получается элементарная конъюнкция, не являющаяся импликантой функции f. Дизъюнкция всех простых импликант функции f называется ***сокращенной*** ДНФ функции f.

Дизъюнктивная нормальная форма называется:

минимальной, если она имеет наименьшее число букв среди всех эквивалентных ей ДНФ;

кратчайшей, если она имеет наименьшую длину среди всех эквивалентных ей ДНФ;

тупиковой, если отбрасывание любого слагаемого или буквы приводит к неэквивалентной ДНФ.

Тупиковая ДНФ функции f (x_1,...,x_n) получается из сокращенной ДНФ этой функции путем отбрасывания некоторых элементарных конъюнкций. Среди тупиковых ДНФ ищется минимальная и кратчайшая ДНФ функции.

Существует несколько методов получения сокращенной ДНФ (получения простых импликант).

Метод Квайна. Для того, чтобы можно было применить метод Квайна, необходимо, чтобы функция была приведена к виду СДНФ. Основой метода является закон склеивания.

$a \wedge b \vee a \wedge \bar{b} = a$, $(a \vee b) \wedge (a \vee \bar{b}) = a$;

1. Просматриваем поочередно пары конъюнкций, если возможно – производим склеивание и результат записываем отдельно. Пары, участвовавшие в склеивании, помечаем.

2. После выполнения всевозможных склеиваний в результат добавляем те конъюнкции, которые не участвовали в склеивании.

3. Если не было элементарных конъюнкций, которые можно было склеивать, то алгоритм закончен. В противном случае переходим к пункту1.

Пример.

Пусть задана функция $F(x_1,x_2,x_3,x_4) = x_1 x_2 x_3 x_4 \vee x_1 x_2 \neg x_3 x_4 \vee x_1 x_2 \neg x_3 \neg x_4 \vee x_1 \neg x_2 \neg x_3 x_4 \vee \neg x_1 \neg x_2 x_3 \neg x_4 \vee \neg x_1 x_2 x_3 \neg x_4 \vee x_1 x_2 x_3 \neg x_4 \vee x_1 \neg x_2 x_3 \neg x_4$

Произведем всевозможные склеивания, в результате получим:

$F(x_1,x_2,x_3,x_4) = x_1 x_2 x_4 \vee x_1 x_2 x_3 \vee x_1 x_2 \neg x_3 \vee x_1 \neg x_3 x_4 \vee x_1 x_2 \neg x_4 \vee \neg x_1 x_3 \neg x_4 \vee \neg x_2 x_3 \neg x_4 \vee x_2 x_3 \neg x_4 \vee x_1 x_3 \neg x_4$

Все конъюнкции исходной функции участвовали в склеивании.

Произведем склеивание еще раз. Конъюнкция $x_1 \neg x_3 x_4$ в склеивании не участвовала, поэтому записываем ее в результат.

$F(x_1,x_2,x_3,x_4) = x_1 x_2 \vee x_3 \neg x_4 \vee x_1 \neg x_3 x_4$

Метод Блейка

Метод Блейка позволяет получить сокращенную ДНФ из произвольной ДНФ и состоит в применении правил обобщенного склеивания ($xK_1 \vee \neg xK_2 = xK_1 \vee \neg xK_2 \vee K_1K_2$) и поглощения ($K_1 \vee K_1K_2 = K_1$).

На первом этапе производятся операции обобщенного склеивания до тех пор, пока это возможно. На втором – операции поглощения.

Пример.

Получить сокращенную ДНФ для функции

$F(x_1,x_2,x_3) = x_1 x_2 \vee \neg x_1 x_3 \vee \neg x_2 x_3$

После первого этапа получим:

$F(x_1,x_2,x_3) = x_1 x_2 \vee \neg x_1 x_3 \vee x_2 x_3 \vee \neg x_2 x_3 \vee x_1 x_3 \vee x_3$

После второго:

$F(x_1,x_2,x_3) = x_1 x_2 \vee x_3$

Если функция задана в КНФ, для получения сокращенной ДНФ нужно сначала раскрыть скобки, пользуясь законом дистрибутивности, затем из полученной ДНФ вычеркнуть буквы и слагаемые, используя законы: $\neg x_1 x_1 = 0$, $x_1 \vee x_1 = x_1$, $x_1 x_1 = x_1$, $K_1 \vee K_1K_2 = K_1$.

Для небольших значений n сокращенную ДНФ функции $f(x_1,...,x_n)$ можно находить с помощью карт Карно. Объединяя клетки, соответствующие единичным значениям функции f, в

максимальные интервалы и сопоставляя им элементарные конъюнкции, получим сокращенную ДНФ.

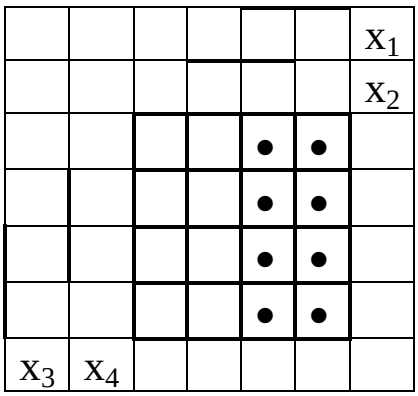
Рассмотрим на примере функции от четырех переменных объединение в максимальные интервалы.

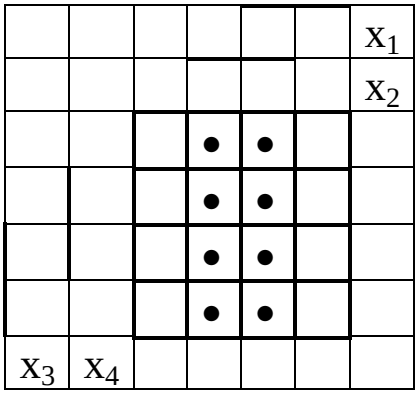
Пусть функция принимает единичное значение на всех наборах переменных. Функция тождественно равна единице.

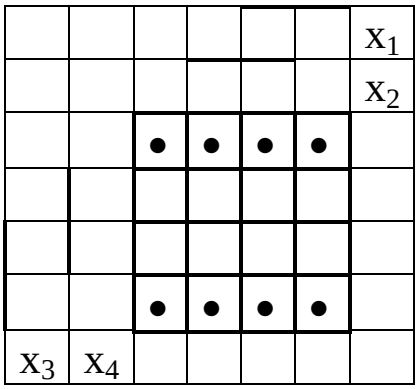
При объединении необходимо придерживаться правил:

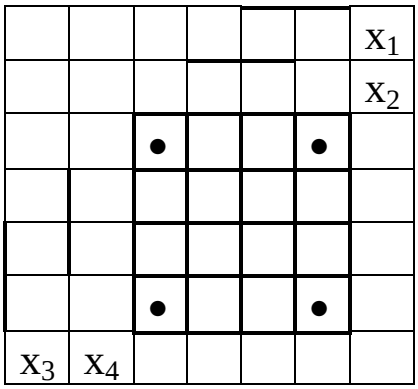
- в объединение включаются только клетки «зеркально» симметричные относительно какой-либо из осей;
- количество клеток соответствует степени 2 (2,4, 8, 16,...).

В первом случае результат объединения равен x_1, во втором – x_2.

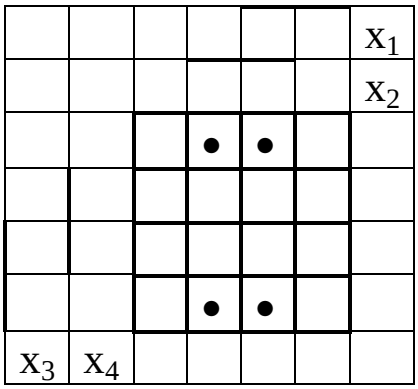




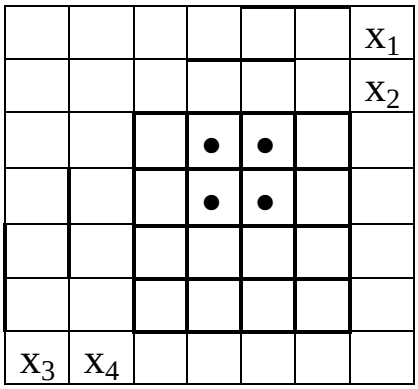




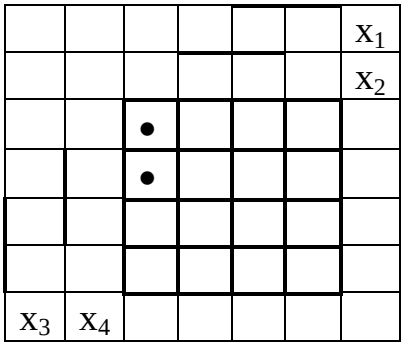
Результат – $\neg x_4$ $\neg x_2 \neg x_4$



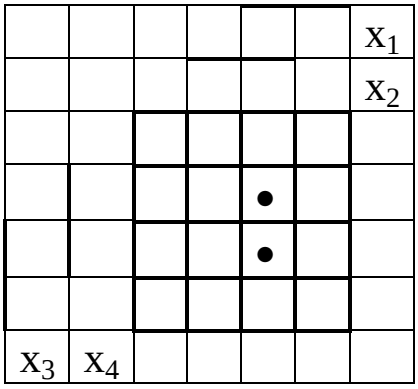
$x_2\neg x_4$



$x_2\neg x_3$



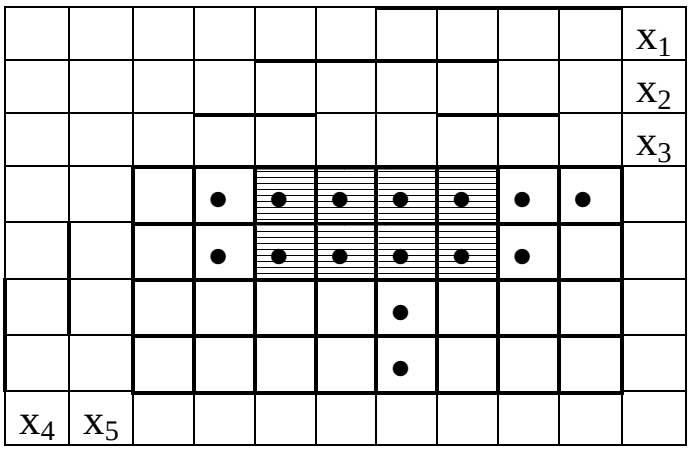
$\neg x_1\neg x_2\neg x_3$



$x_1x_2x_4$

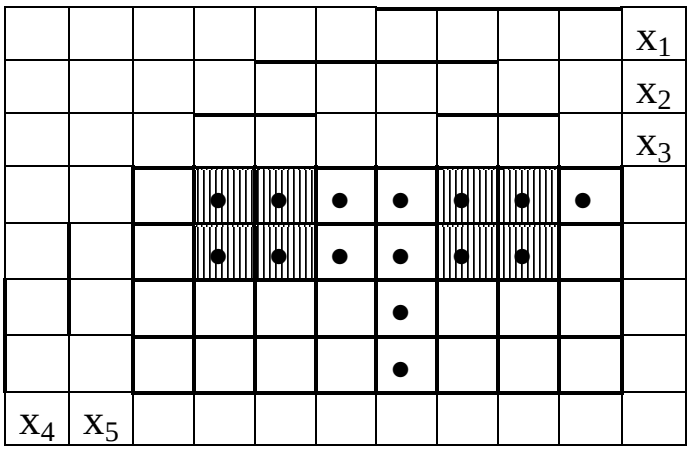
Пусть задана функция от пяти переменных.

$F(x_1,x_2,x_3,x_4,x_5) = x_3\neg x_4x_5 \vee \neg x_1x_2\neg x_3\neg x_4 \vee \neg x_1x_2\neg x_4\neg x_5 \vee x_1\neg x_4\neg x_5 \vee x_1x_2\neg x_3x_4 \vee x_1x_2\neg x_4x_5 \vee \neg x_1\neg x_2x_3\neg x_4\neg x_5$

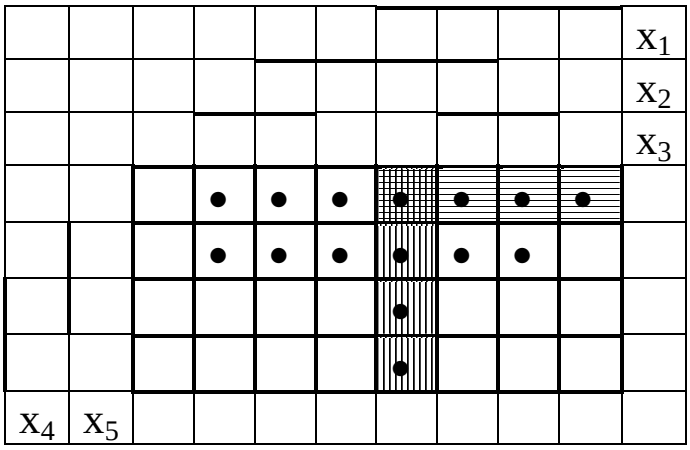


Заданная функция отображена на карте Карно. Получим следующие интервалы: $x_2\neg x_4$ (показан заштрихованной областью).

Интервал $x_3\neg x_4$ показан ниже вертикальными полосами. Ниже представлены еще два интервала $x_1\neg x_4\neg x_5$, $x_1x_2\neg x_3$.



В результате сокращенная функция, эквивалентная исходной, запишется: $F(x_1,x_2,x_3,x_4,x_5) = x_2\neg x_4 \vee x_3\neg x_4 \vee x_1\neg x_4\neg x_5 \vee x_1x_2\neg x_3$



Простая импликанта называется ***ядерной***, если удаление ее из сокращенной ДНФ приводит к ДНФ, которая не эквивалентна исходной. Для каждой ядерной импликанты элементарной конъюнкции существует такой набор значений переменных, который обращает конъюнкцию в единицу, а остальные слагаемые сокращенной ДНФ в ноль. Простая импликанта входит во все тупиковые ДНФ тогда и только тогда, когда она входит в ядро функции.

Для поиска импликант, входящих в ядро функции, используется условие, состоящее в том, что импликанта, которая обращается в единицу на тех же наборах переменных, что и дизъюнкция ядерных импликант, в тупиковую ДНФ не входит.

Метод поиска минимальной ДНФ состоит в нахождении минимального покрытия булевой матрицы. Булева матрица строится следующим образом. В качестве строк берутся простые импликанты. В качестве столбцов – полные элементарные конъюнкции соответствующие СДНФ исходной функции. На пересе-

чении строки и столбца ставится единица, если на наборе, обращающем полную элементарную конъюнкцию в единицу, простая импликанта также обращается в единицу.

Далее подсчитывается количество единиц в столбце, выбирается столбец, количество единиц в котором минимально, и строка с максимальным количеством единиц. Простая импликанта, соответствующая выбранной строке, входит в покрытие (в ядро функции). Далее она не рассматривается. Из рассмотрения удаляются также столбцы, в которых присутствует единица в выбранной строке.

Выбор осуществляется до тех пор, пока все столбцы не будут покрыты.

Пример.

Пусть задана функция:

$F(x_1,x_2,x_3,x_4) = x_1 x_3 \vee x_2x_3 \vee x_1 x_2 \vee \neg x_1 \neg x_2x_4 \vee \neg x_2 x_3x_4 \vee x_1 x_3x_4$

Найдем все полные конъюнкции и перенумеруем их/

x_1x_3	1	$\neg x_1x_2x_3x_4$
	2	$\neg x_1x_2x_3\neg x_4$
	3	$\neg x_1\neg x_2x_3x_4$
	4	$\neg x_1\neg x_2x_3\neg x_4$

x_2x_3	5	$x_1x_2x_3x_4$
	6	$x_1x_2x_3\neg x_4$
	1	$\neg x_1x_2x_3x_4$
	2	$\neg x_1x_2x_3\neg x_4$

x_1x_2	5	$x_1x_2x_3x_4$
	6	$x_1x_2x_3\neg x_4$
	7	$x_1x_2\neg x_3x_4$
	8	$x_1x_2\neg x_3\neg x_4$

$\neg x_1\neg x_2x_4$	3	$\neg x_1\neg x_2x_3x_4$
	9	$\neg x_1\neg x_2\neg x_3x_4$

$\neg x_2\ x_3x_4$	10	$x_1\neg x_2x_3x_4$
	3	$\neg x_1\neg x_2x_3x_4$

$x_1\ x_3x_4$	5	$x_1x_2x_3x_4$
	10	$x_1\neg x_2x_3x_4$

Построим булеву матрицу.

	1	2	3	4	5	6	7	8	9	10
$x_1\ x_3$	1	1	1	1						
x_2x_3	1	1			1	1				
x_1x_2					1	1	1	1		
$\neg x_1\neg x_2x_4$			1						1	
$\neg x_2\ x_3x_4$			1							1
$x_1\ x_3x_4$					1					1
	2	2	3	1	3	2	1	1	1	2

В покрытие обязательно войдет четвертый столбец и импликанта $x_1 x_3$. Удалим первую строку и с первого по пятый столбцы. В результате получим матрицу:

	5	6	7	8	9	10
x_2x_3	1	1				
x_1x_2	1	1	1	1		
$\neg x_1\neg x_2x_4$					1	
$\neg x_2\, x_3x_4$						1
$x_1\, x_3x_4$	1					1
	3	2	1	1	1	2

В оставшейся части матрицы выберем 7 столбец и импликату x_1x_2. Удалим выбранную строку и столбцы с 5 по 8. В полученной матрице выберем 9 столбец.

$\neg x_1\neg x_2x_4$.

	9	10
x_2x_3		
$\neg x_1\neg x_2x_4$	1	
$\neg x_2\, x_3x_4$		1
$x_1\, x_3x_4$		1
	1	2

В оставшейся части можно выбрать любую из строк.

В покрытие войдут импликанты:

$x_1\, x_3 \vee\ x_1x_2\ \vee \neg x_1\neg x_2x_4 \vee x_1\, x_3x_4$

Полученное решение и есть минимальная ДНФ.

	10
x_2x_3	
$\neg x_2\, x_3x_4$	1
$x_1\, x_3x_4$	1
	2

6.7 Классы булевых функций

Суперпозицией системы $S=\{\varphi_1(x_1, x_2, \ldots, x_{k1}), \varphi_2(x_1, x_2, \ldots, x_{k2}), \ldots, \varphi_l(x_1, x_2, \ldots, x_{kl})\}$ называется любая функция f, полученная:

1) из $\varphi_j(x_1, x_2, \ldots, x_{ki})$ переименованием переменных, $\varphi_1 \in S$;

2) подстановкой вместо некоторых переменных функции $\varphi_a(x_1, x_2, \ldots, x_{ka})$, функций $\varphi_j(x_1, x_2, \ldots, x_{kj})$, $\varphi_a, \varphi_j \in S$;

3) с помощью многократного применения п.1) и 2).

Система S называется *полной в* P_k, если любая функция f, $f\in P_k$ представима в виде суперпозиции этой системы, и *базисом*,

если теряется полнота S при удалении хотя бы одной функции, где P_k – k– значная логика.

Классы

1. *Классом* K_0 *булевых функций* f_i $(x_1,x_2, \ldots, x_n)$, *сохраняющих константу* 0, *называется множество функций вида*

$\{f_i (x_1,x_2, \ldots, x_n)/ f_i (0,0, \ldots, 0)=0\}$.

2. *Классом* K_1 *булевых функций* f_i $(x_1,x_2, \ldots, x_n)$, *сохраняющих константу* 1, *называется множество функций вида*

$\{f_i (x_1,x_2, \ldots, x_n)/ f_i (1,1, \ldots, 1)=1\}$.

3. *Классом* $K_л$ *линейных булевых функций* f_i $(x_1,x_2, \ldots, x_n)$, *называется множество функций вида*

$\{f_i (x_1,x_2, \ldots, x_n)/ f_i (x_1,x_2, \ldots, x_n) = c_0 \oplus \Sigma\, c_i x_i\}$,

$c_0, c_i = 0,1; i= 1,2, \ldots n$,

где $\oplus\, \Sigma$ – знаки операции «сложение по модулю два».

4. *Классом* $K_с$ *самодвойственных булевых функций* f_i $(x_1,x_2, \ldots, x_n)$, *называется множество функций вида*

$$\left\{ f_i(x_1, x_2, ..., x_n) / f_i(x_1, x_2, ..., x_n) = \overline{f_i}\left(\overline{x_1}, \overline{x_2}, ..., \overline{x_n}\right) \right\}.$$

5. *Классом* $K_м$ *монотонных булевых функций* f_i $(x_1,x_2, \ldots, x_n)$ *называется множество функций вида*

$$\left\{ \begin{array}{l} f_i(x_1, x_2, ..., x_n) / (\sigma_1^*, \sigma_2^*, ..., \sigma_n^*) \geq (\sigma_1, \sigma_2, ..., \sigma_n) \leftrightarrow (\sigma_i^* \geq \sigma_i, i = 1,2, ..., n) \rightarrow \\ f_i(\sigma_1^*, \sigma_2^*, ..., \sigma_n^*) \geq f_i(\sigma_1, \sigma_2, ..., \sigma_n) \end{array} \right\}.$$

Критерий полноты. Система S булевых функций f_i является полной тогда и только тогда, когда выполняются пять условий: существуют:

- функция $f_i \in S$, не сохраняющая константу нуль: $f_i \notin K_0$;
- функция $f_i \in S$, не сохраняющая константу единицу: $f_i \notin K_1$;
- нелинейная функция в системе S;
- несамодвойственная функция в системе S;
- немонотонная функция в системе S.

Производная первого порядка $\frac{\partial f}{\partial x_i}$ от булевой функции f по переменной x_i

$$\frac{\partial f}{\partial x_i} = f(x_1, x_2, ..., x_{i-1}, 1, ..., x_n) \oplus f(x_1, x_2, ..., x_{i-1}, 0, ..., x_n),$$

где $f(x_1, x_2, \ldots, x_{i-1}, 1, \ldots, x_n)$ – единичная остаточная функция; $f(x_1, x_2, \ldots, x_{i-1}, 0, \ldots, x_n)$ – нулевая остаточная функция; $\oplus$ – сложение по модулю два.

В общем случае:

$$\frac{\partial^k f}{\partial(x_{i_1}, x_{i_2}, ..., x_{i_k})} = \sum_i \frac{\partial f}{\partial x_i} \oplus \sum_{i,j;i\neq j} \frac{\partial^2 f}{\partial x_i \partial x_j} \oplus \sum_{\substack{i,j,s \\ i\neq j, i\neq s, j\neq s}} \frac{\partial^3 f}{\partial x_i \partial x_j \partial x_s} \oplus ... \oplus \frac{\partial^k f}{\partial x_{i_1} \partial x_{i_{12}} ... \partial x_{i_k}},$$

$i, j, s, \ldots = i_1, i_2, \ldots, i_k$.

7 КОМБИНАТОРИКА

Введение

Комбинаторика является разделом дискретной математики, в котором рассматриваются исследование дискретных конечных математических структур. Задачи обычно оцениваются с точки зрения *размера*, то есть общего количества различных вариантов, среди которых нужно найти решение, а алгоритмы оцениваются с точки зрения *сложности*. При этом различают *сложность по времени* (или *временную сложность*), то есть количество необходимых шагов алгоритма, и *сложность по памяти* (или *емкостную сложность*), то есть объем памяти, необходимый для работы алгоритма.

Во многих случаях возникает необходимость подсчитать количество возможных комбинаций объектов, удовлетворяющих определенным условиям. Такие задачи называют *комбинаторными*. Разнообразие комбинаторных задач не поддается исчерпывающему описанию, но среди них есть целый ряд особенно часто встречающихся, для которых известны способы подсчета.

Прежде всего, необходимо ввести понятие факториала, определенного на множестве целых положительных чисел.

$n!=1\cdot 2\cdot 3\cdot \ldots (n-1)\cdot n=n\cdot(n-1)\cdot \ldots 2\cdot 1$

или зададим рекурсивно:

$$\begin{cases} f(0)=1, \\ f(1)=1, \\ f(x)=x\cdot f(x-1). \end{cases}$$

Для формулировки и решения комбинаторных задач используются различные модели *комбинаторных конфигураций*. Рассмотрим следующие две наиболее популярные.

1. Дано n предметов. Их нужно разместить по m ящикам так, чтобы выполнялись заданные ограничения. Сколькими способами это можно сделать?

2. Рассмотрим множество функций

$F: X\to Y$, где $|X|=n$, $|Y|=m$, $X=\{1,2,\ldots,n\}$

Не ограничивая общности, можно считать, что

$Y=\{1,2, \ldots, m\}$, $F=<F(1), F(2), \ldots, F(n)>$, $1 \le F(i) \le m$.

Сколько существует функций F, удовлетворяющим заданным ограничениям?

Размещения

Число всех функций (при отсутствии ограничений), или число всех возможных способов разместить n предметов по m ящикам, называется *числом размещений* и обозначается $U(m, n)$

$U(m, n) = m^n$.

Размещения без повторений

Число инъективных функций, или число всех возможных способов разместить n предметов по m ящикам, не более чем по одному в ящик, называется *числом размещений без повторений* и обозначается $A(m, n)$ или $[m]_n$, или $(m)_n$.

$$A(m,n) = \frac{m!}{(m-n)!}$$

Перестановки

Число взаимнооднозначных функций, или *число перестановок* n предметов, обозначается $P(n)$.

$P(n) = n!$

Сочетания

Число строго монотонных функций, или число размещений n неразличимых предметов по m ящикам, не более чем по одному в ящик, то есть число способов выбрать из m ящиков n ящиков с предметами, называется *числом сочетаний* и обозначается $C(m, n)$ или C_m^n или $\binom{n}{m}$

$$C(m,n) = \frac{m!}{n!(m-n)!}.$$

Сочетания с повторениями

Число монотонных функций, или число размещений n неразличимых предметов по n ящикам, называется *числом сочетаний с повторениями* и обозначается $V(m, n)$.

$V(m, n)= C(\mathrm{n} + \mathrm{m} - 1, n)$.

Подстановки

Подстановки и перестановки являются равнообъемными понятиями. Для вычисления перестановок установлена очень простая формула: $P(n) = n!$ При решении практических задач не следует забывать, что факториал – это *очень* быстро растущая функция, в частности, факториал растет быстрее экспоненты. Действительно, используя известную *формулу Стирлинга*

$$n! \approx \sqrt{2\pi n} n^n e^{-n},$$

или более точно

$$\sqrt{2\pi n} n^n e^{-n} < n! < \sqrt{2\pi n} n^n e^{-n+1/(12n)},$$

нетрудно показать, что

$$\lim_{n\to+\infty} \frac{n!}{2^n} = +\infty.$$

Группа подстановок

Взаимнооднозначная функция f: $X \to X$ называется *подстановкой* на X.

Замечание. Если множество X конечно ($|X|=n$), то, не ограничивая общности, можно считать, что X=1.. n. В этом случае подстановку f: 1.. n $\to$ 1.. n удобно задавать таблицей из двух строк. В первой строке – значения аргументов, во второй – соответствующие значения функции.

Пример

$$f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 1 & 4 & 3 \end{vmatrix}, \quad g = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 2 & 3 & 5 \end{vmatrix}.$$

Произведением подстановок f и g называется их суперпозиция $f \circ g$.

Пример

$$fg = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 1 & 4 & 3 & 2 \end{vmatrix}.$$

Тождественная подстановка – это подстановка e такая, что $e(x)= x$.

Пример

$$e = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 & 5 \end{vmatrix}.$$

Обратная подстановка – это обратная функция, которая всегда существует, поскольку подстановка является биекцией. Таблицу обратной подстановки можно получить, если просто поменять местами строки таблицы исходной подстановки.

Пример

$$f = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 1 & 4 & 3 \end{vmatrix}, \quad f^{-1} = \begin{vmatrix} 5 & 2 & 1 & 4 & 3 \\ 1 & 2 & 3 & 4 & 5 \end{vmatrix} = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 5 & 4 & 1 \end{vmatrix}.$$

Таким образом, множество подстановок образует группу относительно операции суперпозиции. Эта группа называется *симметрической* степени n.

Циклы

Цикл – это такая последовательность элементов x_0, x_1, …, x_k такая, что

$$f(x_i) = \begin{cases} x_{i+1}, & o \le i \le k, \\ x_0, & i = k. \end{cases}$$

Цикл длины 2 называется *транспозицией*.

Подстановки и перестановки

В таблице подстановки нижняя строка (значения функции) является перестановкой элементов верхней строки (значения аргументов). Если принять соглашение, что элементы верхней строки (аргументы) всегда располагаются в определенном порядке (например, по возрастанию), то верхнюю строку можно не указывать – подстановка определяется одной нижней строкой. Таким образом, подстановки взаимно однозначно соответствуют перестановкам.

Перестановку (и соответствующую ей подстановку) элементов 1, 2, …, n будем обозначать $<a_1, a_2, \ldots, a_n>$, где все a_i – различные числа из диапазона 1.. n.

Инверсии

Если в перестановке $f = <a_1, a_2, \ldots, a_n>$ для элементов a_i и a_j имеет место неравенство $a_i > a_j$ при $i < j$, то пара (a_i, a_j) называется *инверсией*. Обозначим число $I(f)$ – *число инверсий* в перестановке f.

Произвольную подстановку f можно представить в виде суперпозиции $I(f)$ транспозиций соседних элементов. Всякая сортировка может быть выполнена перестановкой соседних элементов.

Генерация перестановок

На множестве перестановок естественным образом можно определить упорядоченность элементов. А именно, говорят, что перестановка $<a_1, a_2, \ldots, a_n>$ *лексикографически* предшествует перестановке $<b_1, b_2, \ldots, b_n>$, если $\exists\, k \le n\ a_k < b_k\ \&\ \forall i < k\ a_i = b_i$. Аналогично, говорят, что перестановка $<a_1, a_2, \ldots, a_n>$ *антилексикографически* предшествует перестановке $<b_1, b_2, \ldots, b_n>$, если $\exists\, k \le n\ a_k > b_k\ \&\ \forall i > k\ a_i = b_i$.

Биномиальные коэффициенты

Число сочетаний $C(m, n)$ – это число различных n-элементарных подмножеств m-элементарного множества. Числа

$C(m, n)$ встречаются в формулах решения многих комбинаторных задач. Действительно, рассмотрим следующую типовую схему рассуждений при решении комбинаторной задачи. Пусть нужно определить число подмножеств m-элементарного множества, удовлетворяющих некоторому условию. Разобьем задачу на подзадачи: рассмотрим отдельно 1-элементные подмножества, 2-элементные и т.д., а затем сложим полученные результаты. К счастью, числа $C(m, n)$ обладают целым рядом свойств. Элементарные тождества:

$C(m, n) = C(m, m - n)$.

$C(m, n) = C(m - 1, n) + C(m - 1, n - 1)$.

$C(n, i)\, C(i, m) = C(n, m)\, C(n - m, i - m)$.

Бином Ньютона

Числа сочетаний $C(m, n)$ называют также *биномиальными коэффициентами*. Смысл этого названия устанавливается следующей теоремой, известной также как формула *бинома Ньютона*.

$$(x+y)^m = \sum_{b=0}^{m} C(m,n) x^n y^{m-n}\,.$$

Свойства:

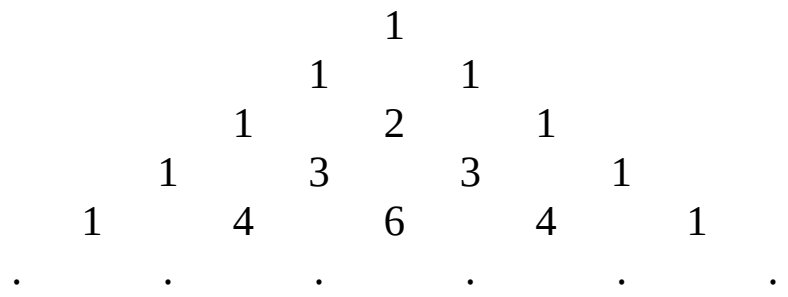
$$\sum_{n=0}^{m} C(m,n) = 2^m;$$

$$\sum_{n=0}^{m} (-1)^n C(m,n) = 0;$$

$$\sum_{n=0}^{m} nC(m,n) = m2^{m-1};$$

$$C(m+n,k) = \sum_{i=0}^{k} C(m,i)C(n,k-i)\,.$$

Треугольник Паскаля



В данном равнобедренном треугольнике каждое число (кроме единиц на боковых сторонах) является суммой двух чисел, стоящих над ним. Число сочетаний $C(m, n)$ находится в $(m + 1)$-м ряду на $(n + 1)$-м месте.

8 КОДИРОВАНИЕ

Вопросы кодирования издавна играли заметную роль в математике.

Пример

1. *Десятичная позиционная система счисления* – это способ кодирования натуральных чисел. Римские цифры – другой способ кодирования натуральных чисел, причем гораздо более наглядный и естественный: палец – I, пятерня – V, две пятерни – X. Однако при этом способе кодирования труднее выполнять арифметические операции над большими числами, поэтому он был вытеснен позиционной десятичной системой.

2. *Декартовы координаты* – способ кодирования геометрических объектов числами.

Ранее средства кодирования играли вспомогательную роль и не рассматривались как отдельный предмет математического изучения, но с появлением компьютеров ситуация радикально изменилась. Кодирование буквально пронизывает информационные технологии и является центральным вопросом при решении самых разных (практически всех) задач программирования:

- представление данных произвольной природы (например, чисел, текста, графики) в памяти компьютера;
- защита информации от несанкционированного доступа;
- обеспечение помехоустойчивости при передаче данных по каналам связи;
- сжатие информации в базах данных.
- составление текста программы.

Не ограничивая общности, задачу кодирования можно сформулировать следующим образом. Пусть заданы алфавиты $A = \{a_i,...,a_n\}$, $B = \{b_i,...,b_m\}$ и функция $F: S \to B^*$, где S – некоторое множество слов в алфавите A, $S \subset A^*$. Тогда функция F называется *кодированием*, элементы множества S – *сообщениями*, а элементы $\beta = F(\alpha)$, $\alpha \in S$, $\beta \in B^*$ – *кодами* (соответствующих со-

общений). Обратная функция F^{-1} (если она существует!) называется *декодированием*.

Если $|B| = m$, то F называется *m-ичным кодированием*. Наиболее распространенный случай $B = \{0,1\}$ – *двоичное* кодирование. Именно этот случай рассматривается в последующих разделах; слово «двоичное» опускается.

Типичная задача теории кодирования формулируется следующим образом: при заданных алфавитах A, B и множестве сообщений S найти такое кодирование F, которое обладает определенными свойствами (то есть удовлетворяет заданным ограничениям) и оптимально в некотором смысле. Критерий оптимальности, как правило, связан с минимизацией длин кодов. Свойства, которые требуются от кодирования, бывают самой разнообразной природы:

- существование декодирования – это очень естественное свойство, однако даже оно требуется не всегда. Например, трансляция программы на языке высокого уровня в машинные команды – это кодирование, для которого не требуется однозначного декодирования;
- помехоустойчивость, или исправление ошибок: функция декодирования F^{-1} обладает таким свойством, что $F^{-1}(\beta) = F^{-1}(\beta')$, если β' в определенном смысле близко к β;
- заданная сложность (или простота) кодирования и декодирования. Например, в криптографии изучаются такие способы кодирования, при которых имеется просто вычисляемая функция F, но определение функции $F\sim^{J}$ требует очень сложных вычислений.

Большое значение для задач кодирования имеет природа множества сообщений S. При одних и тех же алфавитах A, B и требуемых свойствах кодирования F оптимальные решения могут кардинально отличаться для разных S. Для описания множества S (как правило, очень большого или бесконечного) применяются различные методы:

- теоретико-множественное описание, например $S = \{\alpha \mid \alpha \in A^* \ \& \ |a| = n\}$;
- вероятностное описание, например $S = A^*$, и заданы вероятности p_i появления букв в сообщении, $\sum_{i=1}^{n} p_i = 1$;

● логико-комбинаторное описание, например, *S* задано порождающей формальной грамматикой.

8.1 Алфавитное кодирование

Кодирование *F* может сопоставлять код всему сообщению из множества *S* как единому целому или же строить код сообщения из кодов его частей. Элементарной частью сообщения является одна буква алфавита *A*. Этот простейший случай рассматривается в этом и следующих двух разделах.

Префикс и постфикс слова

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

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

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

Таблица кодов

Алфавитное (или *побуквенное)* кодирование задается *схемой* (или *таблицей кодов)* σ.

$\sigma := < a_1 \to \beta_1, \ldots, a_n \to \beta_n >$, $a_i \in A$, $\beta_i \in B^*$.

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

$F: A^* \to B^*$, $a_{i1} \ldots a_{ik} = a \in A^*$, $F(a): = \beta_{i1} \ldots \beta_{ik}$.

Пример

Рассмотрим алфавиты *A:* ={0,1,2,3,4,5,б,7,8,9}, *B:* ={0,1} и схему

δ*:* =(0 → 0, 1→ 1, 2 → 10, 3 → 11, 4 → 100, 5 → 101, 6 → 110, 7 → 111, 8 → 1000, 9 → 1001).

Эта схема однозначна, но кодирование не является взаимно однозначным:

$$F_\delta(333) = 111111 = F_\delta(77),$$

а значит, невозможно декодирование. С другой стороны, схема

δ*:* =(0 → 0000, 1→ 0001, 2 → 0010, 3 → 0011, 4 → 0100, 5 → 0101, 6 → 0110, 7 → 0111, 8 → 1000, 9 → 1001),

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

Разделимые схемы

Рассмотрим схему алфавитного кодирования σ и различные слова, составленные из элементарных кодов. Схема σ называется *разделимой,* если

$$\beta_{i1} \ldots \beta_{ik} = \beta_{i1} \ldots \beta_{ik} \Rightarrow k = l \,\&\, \forall t \in 1..k\ i_t = j_t,$$

то есть любое слово, составленное из элементарных кодов, единственным образом разлагается на элементарные коды. Алфавитное кодирование с разделимой схемой допускает декодирование.

Если таблица кодов содержит одинаковые элементарные коды, то есть, если

$$\exists\ i,j\ i \neq j\ \&\ \beta_i = \beta_j,$$

где β_i, $\beta_i \in V$, то схема заведомо не является разделимой. Такие схемы далее не рассматриваются, то есть

$$\forall\ i \neq j\ \beta_i, \beta_i \in V \Rightarrow \beta_i \neq \beta_i.$$

Префиксные схемы

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

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

Префиксная схема является разделимой. Свойство быть префиксной является достаточным, но не является необходимым для разделимости схемы.

Пример. Разделимая, но не префиксная схема: A = {a, b}, B = {0,1}, δ = {a → O, b → 01}.

Неравенство Макмиллана

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

ТЕОРЕМА. *Если схема* $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$ *разделима, то*

$$\sum_{i=1}^{n} \frac{1}{2^{l_i}} \le 1, \text{ где } l_i = |\beta_i|.$$

Неравенство Макмиллана является не только необходимым, но и в некотором смысле достаточным условием разделимости схемы алфавитного кодирования.

ТЕОРЕМА. *Если числа* $l_1, \ldots, l_n$ *удовлетворяют неравенству*

$$\sum_{i=1}^{n} \frac{1}{2^{l_i}} \le 1,$$

то существует разделимая схема алфавитного кодирования $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$, *где* $\forall i \; l_i = |\beta_i|$.

Пример

Азбука Морзе – это схема алфавитного кодирования

($A \to 01$, $B \to 1000$, $C \to 1010$, $D \to 100$, E → 0, F → 0010, G → 110, $H \to 0000$, $I \to 00$, $J \to 0111$, $K \to 101$, $L \to 0100$, $M \to$ ll, $N \to$ 10, $O \to 111$, $P \to 0110$, $Q \to 1101$, $R \to 010$, $S \to 000$, $T \to 1$, $U \to$ 001, $V \to 0001$, $W \to 011$, $X \to 1001$, $Y \to 1011$, $Z \to 1100$),

где по историческим и техническим причинам 0 называется *точкой* и обозначается знаком «•», а 1 называется *тире* и обозначается знаком «–». Имеем:

1/4 + 1/16 + 1/16 + 1/8 + 1/2 + 1/16 + 1/8 + 1/16 + 1/4 + 1/16 + 1/8 + 1/16 + 1/4 + 1/4 + 1/8 + 1/16 + 1/16 + 1/8 + 1/8 + 1/2 + 1/8 + 1/16 + 1/8 + 1/16 + 1/16 + 1/16 = 2/2 + 4/4 + 7/8 + 12/16 = 3 + 5/8 > 1.

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

8.2 Кодирование с минимальной избыточностью

Для практики важно, чтобы коды сообщений имели по возможности наименьшую длину. Алфавитное кодирование пригодно для любых сообщений, то есть $S = A^*$. Если больше про множество S ничего не известно, то точно сформулировать задачу оптимизации затруднительно. Однако на практике часто доступна дополнительная информация. Например, для текстов на естественных языках известно распределение вероятности появления букв в сообщении. Использование такой информации позволяет строго поставить и решить задачу построения оптимального алфавитного кодирования.

Минимизация длины кода сообщения

Если задана разделимая схема алфавитного кодирования $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$, то любая схема $\sigma' = \langle a_i \to \beta_i' \rangle_{i=1}^{n}$, где $\langle \beta'_1,..., \beta'_n \rangle$ является перестановкой $\langle \beta_1,..., \beta_n \rangle$, также будет разделимой. Если длины элементарных кодов равны, то перестановка элементарных кодов в схеме не влияет на длину кода сообщения. Но если длины элементарных кодов различны, то длина кода сообщения зависит от состава букв в сообщении и от того, какие элементарные коды каким буквам назначены. Если заданы конкретное сообщение и конкретная схема кодирования, то нетрудно подобрать

такую перестановку элементарных кодов, при которой длина кода сообщения будет минимальна.

Пусть $k_1,...,k_n$ – количества вхождений букв $a_1,...,a_n$ в сообщение *S*, а $l_1,...,l_n$ – длины элементарных кодов $\beta_1,...,\beta_n$, соответственно. Тогда, если $k_i \le k_j$ и $l_i \ge l_j$, то $k_i l_i + k_j l_j \le k_i l_j + k_j l_i$. Действительно, пусть $k_j = k + a$, $k_i = k$ и $l_i = l$, $l_i = l + b$, где a, $b \ge 0$. Тогда

$(k_i l_j + k_j l_i) - (k_i l_i + k_j l_j) = (kl + (k + a)(l + b)) - (k(l + b) + l(k + a)) =$

$= (kl + al + bk + ab + kl) - (kl + al + kl + bk) = ab \ge 0.$

Отсюда вытекает алгоритм назначения элементарных кодов, при котором длина кода конкретного сообщения *S* будет минимальна: нужно отсортировать буквы в порядке убывания количества вхождений, элементарные коды отсортировать в порядке возрастания длины и назначить коды буквам в этом порядке. Этот простой метод решает задачу минимизации длины кода только для фиксированного сообщения *S* и фиксированной схемы σ.

Цена кодирования

Пусть заданы алфавит $A = \{a_1,...,a_n\}$ и вероятности появления букв в сообщении $P = \langle p_1,...,p_n \rangle$ (p_i – вероятность появления буквы a_i). Не ограничивая общности, можно считать, что $p_1 + \ldots + p_n = 1$ и $p_1 \ge \ldots \ge p_n > 0$ (то есть можно сразу исключить буквы, которые не могут появиться в сообщении, и упорядочить буквы по убыванию вероятности их появления).

Для каждой (разделимой) схемы $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$ алфавитного кодирования математическое ожидание коэффициента увеличения длины сообщения при кодировании (обозначается 1_σ) определяется следующим образом:

$$l_\sigma(P) := \sum_{i=1}^{n} p_i l_i \text{, где } 1_i := |\beta_i|$$

и называется средней *ценой* (или *длиной)* кодирования σ при распределении вероятностей P.

Пример

Для разделимой схемы $A = \{a, b\}$, $B = \{0,1\}$, $\delta = \{a \to 0, b \to 01\}$ при распределении вероятностей <0.5, 0.5> цена кодирования

составляет 0.5 * 1 + 0.5 * 2 = 1.5, а при распределении вероятностей <0.9, 0.1> она равна 0.9 * 1 + 0.1 * 2 = 1.1.

Обозначим

$$l_*(P) := \inf_{\sigma} l_\sigma(P),\ p_* = \min_{i-1} p_i,\ L := \lceil \log_2(n-1) \rceil + 1.$$

Очевидно, что всегда существует разделимая схема $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$, такая что $\forall\ i\ |\beta_i| = L$. Такая схема называется схемой *равномерного* кодирования. Следовательно, $1 \le l_*(P) \le L$ и достаточно учитывать только такие схемы, для которых $\forall i\ p_i l_i \le$ L, где l_i – целое и $l_i \le$ L/p$_*$. Таким образом, имеется лишь конечное число схем σ, для которых $l_*(P) \le l_\sigma(P) \le L$. Следовательно, существует схема σ_*, на которой инфимум достигается: $1_{\sigma^*}(P) = 1_*(P)$.

Алфавитное (разделимое) кодирование и σ_*, для которого $l_{\sigma^*}(P) = l_*(P)$, называется кодированием с *минимальной избыточностью*, или *оптимальным* кодированием, для распределения вероятностей P.

Алгоритм Фано

Следующий рекурсивный алгоритм строит разделимую префиксную схему алфавитного кодирования, близкого к оптимальному.

Алгоритм 1. Построение кодирования, близкого к оптимальному

Вход: P : **array** [l..n] **of real** – массив вероятностей появления букв в сообщении, упорядоченный по невозрастанию; 1 ≥ P[1] ≥ …≥ P[n] > 0, P[1] +…+ P[п] = 1.

Выход: C : **array** [l..n, 1..L] **of** 0..1 – массив элементарных кодов.

Fano(l,n, 0) { вызов рекурсивной процедуры Рано }

Основная работа по построению элементарных кодов выполняется следующей рекурсивной процедурой Fano.

Вход: *b* – индекс начала обрабатываемой части массива *P*, *e* – индекс конца обрабатываемой части массива *P*, *k* – длина уже построенных кодов в обрабатываемой части массива *C*.

Выход: заполненный массив *C*.

if *e* > b **then**

k: = *k* + 1 { место для очередного разряда в коде }

m: = Med(b, *e)* { деление массива на две части }

for i **from** *b* ***to*** *e* ***do***

C[i,k]: =i > *m* { в первой части добавляем 0, во второй – 1 }

end for

Fano(*b*, *m*, *k)* { обработка первой части }

Fano (*m* + 1, *e*, *k)* { обработка второй части }

end if

Функция Med находит *медиану* указанной части массива *P*[*b*..*e*], то есть определяет такой индекс *m (b* ≤ *m* < *e)*, что сумма элементов *P*[*b*..*m*] наиболее близка к сумме элементов *P*[*m* + 1..*e*].

Вход: *b* – индекс начала обрабатываемой части массива *P*, *e* – индекс конца обрабатываемой части массива P.

Выход: m – индекс медианы, то есть $\min\limits_{m\in b..e-1}\left|\sum\limits_{i=b}^{m}P[i]-\sum\limits_{i=m+1}^{e}P[i]\right|$.

S_b: = 0 { сумма элементов первой части }

for *i* **from** *b* **to** *e* – 1 **do**

S_b : = S_b + *P*[*i*] { вначале все, кроме последнего }

end for

S_e : = *P*[*e*] { сумма элементов второй части }

m: = *e* { начинаем искать медиану с конца }

repeat

d: = S_b – S_e { разность сумм первой и второй части }

m : = *m* – 1 { сдвигаем границу медианы вниз }

S_b: = S_b – *P*[*m*]; S_e: = *Se* + *P*[*m*]

until $|S_b–S_e| \geq d$

return *m.*

Обоснование

При каждом удлинении кодов в одной части коды удлиняются нулями, а в другой – единицами. Таким образом, коды од-

ной части не могут быть префиксами другой. Удлинение кода заканчивается тогда и только тогда, когда длина части равна 1, то есть остается единственный код. Таким образом, схема по построению префиксная, а потому разделимая.

Пример

Коды, построенные алгоритмом Фано для заданного распределения вероятностей (n = 7).

p_i	$C[i]$	l_i
0.20	00	2
0.20	010	3
0.19	011	3
0.12	100	3
0.11	101	3
0.09	110	3
0.09	111	3
$l_\sigma(P)$		2.80

Оптимальное кодирование

Оптимальное кодирование обладает определенными свойствами, которые можно использовать для его построения.

ЛЕММА. *Пусть* $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$ *– схема оптимального кодирования для распределения вероятностей* $P = p_1 \geq \ldots \geq p_n > 0$. *Тогда, если* $p_i > p_j$, *то* $l_i \leq l_j$.

Таким образом, не ограничивая общности, можно считать, что $l_1 \leq \ldots \leq 1_n$.

ЛЕММА. *Если* $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$ *– схема оптимального префиксного кодирования для распределения вероятностей* $P = p_1 \geq \ldots \geq p_n > 0$, *то среди элементарных кодов, имеющих максимальную длину, имеются два, которые различаются только в последнем разряде.*

ТЕОРЕМА. *Если* $\sigma_{n-1} = \langle a_i \to \beta_i \rangle_{i=1}^{n-1}$ *– схема оптимального префиксного кодирования для распределения вероятностей* $P = p_1 \geq \ldots \geq p_{n-1} > 0$. *и* $p_i = q' + q''$, *причем*

$p_1 \geq \ldots \geq p_{i-1} \geq p_{i+1} \geq \ldots \geq p_{n-1} \geq q' \geq q'' > 0$,

то кодирование со схемой

$$\sigma_n = \langle a_1 \to \beta_1, \ldots, a_{j-1} \to \beta_{j-1}, a_{j=1} \to \beta_{j=1}, \ldots, a_{n-1} \to \beta_{n-1}, a_j \to \beta_j 0, a_n \to \beta_j 1 \rangle$$

является оптимальным префиксным кодированием для распределения вероятностей $P_n = p_1, \ldots, p_{j-1}, p_{j+1}, \ldots, p_{n-1}, q', q''$.

Алгоритм Хаффмена

Следующий рекурсивный алгоритм строит схему оптимального префиксного алфавитного кодирования для заданного распределения вероятностей появления букв.

Алгоритм 2. Построение оптимальной схемы – рекурсивная процедура Huffman.

Вход: *n* – количество букв, *P :* **array** [l..n] **of real** – массив вероятностей букв, упорядоченный по убыванию.

Выход: *C :* **array** [l..*n*, 1..*L*] **of** 0..1 – массив элементарных кодов, *d* : **array** [l..*n*] of 1..*L* – массив длин элементарных кодов схемы оптимального префиксного кодирования.

if *n* = 2 **then**
C[1,1]: = 0; *d*[1]: = 1{ первый элемент }
C[2, 1]: = 1; *d*[2]: = 1{ второй элемент }
else
q: = P[n – 1] + *P[n]* { сумма двух последних вероятностей }
j : = Up(n, *q*) { поиск места и вставка суммы }
Huffman (P, n – 1) { рекурсивный вызов }
Down (n, j) { достраивание кодов }
end if

Функция Up находит в массиве *P* место, в котором должно находиться число *q* (см. предыдущий алгоритм) и вставляет это число, сдвигая вниз остальные элементы.

Вход: n – длина обрабатываемой части массива *P*, *q* – вставляемая сумма.

Выход: измененный массив *P*.

for *i* **from** n – 1 **downto** 2 **do**
if *P*[*i* – 1] ≤ *q* **then**
P[*i*]*: = P*[*i* – 1]{ сдвиг элемента массива }
else
j : =i – l { определение места вставляемого элемента }
exit for *i* { все сделано – цикл не нужно продолжать }
end if
end for
P[*j*] := *q* { запись вставляемого элемента }
return *j*

Процедура Down строит оптимальный код для *n* букв на основе построенного оптимального кода для *n* – 1 буквы. Для этого код буквы с номером *j* временно исключается из массива *C* путем сдвига вверх кодов букв с номерами, большими j, а затем в конец обрабатываемой части массива *C* добавляется пара кодов, полученных из кода буквы с номером *j* удлинением на 0 и 1, соответственно. Здесь C [*i*,*] означает вырезку из массива, то есть *i*-ю строку массива *C*.

Вход: *n* – длина обрабатываемой части массива *P*, *j* – номер «разделяемой» буквы.

Выход: оптимальные коды в первых *n* элементах массивов *C* и *d*.

c: = C[*j,* *]{ запоминание кода буквы *j* }
l: = d[*j*] { и длины этого кода }
for г **from** *j* **to** *n* – 2 **do**
C [*i,* *]: = *C*[*i* + 1, *]{ сдвиг кода }
d [*i*] := *d* [*i* + 1] { и его длины }
end for
C[*n* – 1, *]: = c; *C*[*n,* *]: = *c* { копирование кода буквы *j* }
C[*n* – 1,*l* + 1]: = 0; *C*[*n, l* + 1]: = 1{ наращивание кодов }
d[*n* – 1]: = / + 1; d[*n*] : = *l* + 1{ и увеличение длин }

Обоснование

Для пары букв при любом распределении вероятностей оптимальное кодирование очевидно: первой букве нужно назначить код 0, а второй – 1. Именно это и делается в первой части опера-

тора **if** основной процедуры Huffman. Рекурсивная часть алгоритма в точности следует доказательству теоремы предыдущего подраздела. С помощью функции Up в исходном упорядоченном массиве *P* отбрасываются две последние (наименьшие) вероятности, и их сумма вставляется в массив *P*, так чтобы массив (на единицу меньшей длины) остался упорядоченным. Заметим, что при этом место вставки сохраняется в локальной переменной *j*. Так происходит до тех пор, пока не останется массив из двух элементов, для которого оптимальный код известен. После этого в обратном порядке строятся оптимальные коды для трех, четырех и т. д. элементов. Заметим, что при этом массив вероятностей *P* уже не нужен – нужна только последовательность номеров кодов, которые должны быть изъяты из массива кодов и продублированы в конце с добавлением разряда. А эта последовательность хранится в экземплярах локальной переменной j, соответствующих рекурсивным вызовам процедуры Huffman.

Пример

Построение оптимального кода Хаффмена для $n = 7$. В левой части таблицы показано изменение массива *P*, а в правой части – массива *C*. Позиция, соответствующая текущему значению переменной *j*, выделена полужирным начертанием.

0.20	0.20	**0.23**	**0.37**	**0.40**	**0.60**	**0**	**1**	**00**	**01**	10	10
0.20	0.20	0.20	0.23	0.37	0.40	1	00	01	10	11	11
0.19	0.19	0.20	0.20	0.23.			01	10	11	000	000
0.12	**0.18**	0.19	0.20					11	000	**001**	010
0.11	0.12	0.18							001	010	011
0.09	0.11									011	0100
0.09											0011

Цена кодирования составляет

0.20 x 2 + 0.20 x 2 + 0.19 x 3 + 0.12 x 3 + 0.11 x 3 + 0.09 x 4 + +0.09 x 4 = 2.78, что несколько лучше, чем в кодировании, полученном алгоритмом Фано.

8.3 Помехоустойчивое кодирование

Надежность электронных устройств по мере их совершенствования все время возрастает, но, тем не менее, в их работе возможны ошибки, как систематические, так и случайные. Сигнал в канале связи может быть искажен помехой, поверхность магнитного носителя может быть повреждена, в разъеме может быть потерян контакт. Ошибки аппаратуры ведут к искажению или потере передаваемых или хранимых данных. При определенных условиях, некоторые из которых рассматриваются в этом разделе, можно применять методы кодирования, позволяющие правильно декодировать исходное сообщение, несмотря на ошибки в данных кода. В качестве исследуемой модели достаточно рассмотреть канал связи с помехами, потому что к этому случаю легко сводятся остальные. Например, запись на диск можно рассматривать как передачу данных в канал, а чтение с диска – как прием данных из канала.

Кодирование с исправлением ошибок

Пусть имеется канал связи *C*, содержащий источник помех:

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

где *S* – множество переданных, а S’ – соответствующее множество принятых по каналу сообщений. Кодирование *F*, обладающее таким свойством, что

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

называется *помехоустойчивым*, или *самокорректирующимся*, или кодированием *с исправлением ошибок.*

Без ограничения общности можно считать, что $A = B = \{0,1\}$, и что содержательное кодирование выполняется на устройстве, свободном от помех.

Классификация ошибок

Ошибки в канале могут быть следующих типов:

- $0 \to 1$, $1 \to 0$ – ошибка типа замещения разряда;
- $0 \to \Lambda$, $1 \to \Lambda$ – ошибка типа выпадения разряда;

- $\Lambda \to 1$, $\Lambda \to 0$ – ошибка типа вставки разряда.

Канал характеризуется верхними оценками количества ошибок каждого типа, которые возможны при передаче через канал сообщения определенной длины. Общая характеристика ошибок канала (то есть их количество и типы) обозначается δ.

Пример

Допустим, что имеется канал с характеристикой δ = <1,0,0>, то есть в канале возможна одна ошибка типа замещения разряда при передаче сообщения длины *n*. Рассмотрим следующее кодирование: $F(a) := aaa$ (то есть каждый разряд в сообщении утраивается) и декодирование $F^{-1}(abc) := a + b + c > 1$ (то есть разряд восстанавливается методом «голосования»). Это кодирование кажется помехоустойчивым для данного канала, однако на самом деле это не так. Дело в том, что при передаче сообщения длины 3*n* возможно не более 3 ошибок типа замещения разряда, но места этих ошибок совершенно не обязательно распределены равномерно по всему сообщению. Ошибки замещения могут произойти в соседних разрядах, и метод голосования восстановит разряд неверно.

Возможность исправления всех ошибок

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

ТЕОРЕМА. *Чтобы существовало помехоустойчивое кодирование с исправлением всех ошибок, необходимо и достаточно, чтобы* $\forall\ s_1,s_2 \in S\ E_{S1} \cap E_{S2} = \varnothing$, *то есть необходимо и достаточно, чтобы существовало разбиение множества* B^* *на множества* B_s ($\cup B_s = B^*$, $\cap B_s = \varnothing$), *такое что* $\forall s \in S\ E_s \subset B_s$.

Пример

Рассмотрим канал, в котором в любом передаваемом разряде происходит ошибка типа замещения с вероятностью p ($0 < p < 1/2$), причем замещения различных разрядов статистически независимы. Такой канал называется *двоичным симметричным*. В этом случае любое слово $s \in E_2^{\delta}$ может быть преобразовано в любое другое слово $s' \in E_2^{\delta}$ замещениями разрядов. Таким образом, $\forall s\ E_s = E_2^{\delta}$, и исправить все ошибки в двоичном симметричном канале невозможно.

Кодовое расстояние

Неотрицательная функция $d(x, y)$: $M \times M \to \mathrm{R}_+$ называется *расстоянием* (или *метрикой)* на множестве M, если выполнены следующие условия (аксиомы метрики):

1. $d(x, y) = 0 \Leftrightarrow x = y$;
2. $d(x, y) = d(y, x)$;
3. $d(x, y) \le d(x, z) + d(y, z)$.

Пусть

$$d_{\delta}(\beta', \beta'') := \begin{cases} \min\limits_{\{E^{\delta}\langle \beta', \beta'' \rangle\}} \left| E^{\delta}\langle \beta', \beta'' \rangle \right|, & \beta'' \in E_{\beta'}^{\delta}; \\ +\infty & \beta'' \notin E_{\beta'}^{\delta}. \end{cases}$$

Эта функция называется *расстоянием Хэмминга*.

В данном случае рассматриваем симметричные ошибки, то есть, если в канале допустима ошибка $0 \to 1$, то допустима и ошибка $1 \to 0$.

Введенная функция dg является расстоянием. Действительно:

1. $d_{\delta}(\beta',\beta'') = 0 \Leftrightarrow \beta'=\beta''$, поскольку ошибки симметричны, и из последовательности $E<\beta',\beta''>$ можно получить последовательность $E<\beta'',\beta'>$, применяя обратные ошибки в обратном порядке.

2. $d_{\delta}(\beta',\beta'') = d_{\delta}(\beta'',\beta')$ по той же причине.

3. $d_{\delta}(\beta',\beta'') \le d_{\delta}(\beta',\beta''') + d_{\delta}(\beta'',\beta'')$, поскольку $E<\beta',\beta''>$ U $E<\beta''',\beta''>$ является некоторой последовательностью, преобра-

зующей β' в β'', а $d_\delta(\beta',\beta'')$ является кратчайшей из таких последовательностей.

Пусть $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$ – схема некоторого алфавитного кодирования, а d – некоторая метрика на B^*. Тогда минимальное расстояние между элементарными кодами

$$d(\sigma) := \min_{1 \le i \le j \le n} d(\beta_i, \beta_j)$$

называется *кодовым расстоянием* схемы σ.

ТЕОРЕМА. *Алфавитное кодирование со схемой* $\sigma = \langle a_i \to \beta_i \rangle_{i=1}^{n}$ *и с кодовым расстоянием*

$$d_\delta(\sigma) := \min_{\beta', \beta'' \in V} d_\delta(\beta', \beta'')$$

является кодированием с исправлением р ошибок типа δ *тогда и только тогда, когда* $d_\delta(\sigma) > 2p$.

Пример

Расстояние Хэмминга в $E_2^n : d(\beta', \beta'') := \sum_{i=1}^{n} (\beta_i' \neq \beta_i'')$.

Код Хэмминга для исправления одного замещения

Рассмотрим построение *кода Хэмминга*, который позволяет исправлять одиночные ошибки типа замещения.

Очевидно, что для исправления ошибки вместе с основным сообщением нужно передавать какую-то дополнительную информацию.

Пусть сообщение $a = a_1 \dots a_m$ кодируется словом $\beta = b_1 \dots b_n$, $n > m$. Обозначим $k := n - m$. Пусть канал допускает не более одной ошибки типа замещения в слове длины n.

Рассматриваемый случай простейший, но одновременно практически очень важный. Таким свойством, как правило, обладают внутренние шины передачи данных в современных компьютерах.

При заданном n количество дополнительных разрядов k подбирается таким образом, чтобы $2^k \geq n + 1$. Имеем:

$$2^k \geq n+1 \Rightarrow \frac{2^n}{n+1} \geq 2^{n-k} \Rightarrow \frac{2^n}{n+1} \geq 2^m.$$

Пример

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

Определим последовательности натуральных чисел в соответствии с их представлениями в двоичной системе счисления следующим образом:

- V_1 : = 1, 3, 5, 7, 9, 11, … – все числа, у которых разряд №1 равен 1;
- V_2 : = 2, 3, 6, 7, 10, … – все числа, у которых разряд №2 равен 1;
- V_3 : = 4, 5, 6, 7, 12, … – все числа, у которых разряд №3 равен 1,
- и т.д. По определению последовательность V_k начинается с числа 2^{k-1}.

Рассмотрим в слове $b_1 ... b_n$ k разрядов с номерами $2^0 = 1$, $2^1 = 2$, $2^2 = 4$,... 2^{k-1}. Эти разряды назовем *контрольными*. Остальные разряды, а их ровно m, назовем *информационными*. Поместим в информационные разряды все разряды слова $a_1 ... a_n$ как они есть. Контрольные разряды определим следующим образом:

b_1: = $b_3 \oplus b_5 \oplus b_7 \oplus$ …, – то есть все разряды с номерами из V_1, кроме первого;

b_2 : = $b_3 \oplus b_6 \oplus b_7 \oplus$ …, – то есть все разряды с номерами из V_2, кроме первого;

b_4 : = $b_5 \oplus b_6 \oplus b_7 \oplus$ …, – то есть все разряды с номерами из V_3, кроме первого;

и вообще,

$$b_{2^{j-1}} := \bigoplus_{i \in V_j \setminus \{2^{j-1}\}} b_i.$$

Пусть после прохождения через канал получен код c_1 … c_n, то есть c_1 … c_n : = $C(b_1 \ldots b_n)$, причем

$$\exists I\, c_I = \begin{cases} b_I \\ \overline{b_I} \end{cases}, \forall\ i \neq I\ c_i = b_i.$$

Здесь I – номер разряда, в котором, возможно, произошла ошибка замещения. Пусть это число имеет следующее двоичное представление: $I = i_l \ldots i_1$.

Определим число $J = j_l \ldots j_1$ следующим образом:

$j_1 := c_1 \oplus c_3 \oplus c_5 \oplus c_7 \oplus \ldots$, – то есть все разряды с номерами из V_1;

$j_2 := c_2 \oplus c_3 \oplus c_6 \oplus c_7 \oplus \ldots$, – то есть все разряды с номерами из V_2;

$j_3 := c_4 \oplus c_5 \oplus c_6 \oplus c_7 \oplus \ldots$, – то есть все разряды с номерами из V_3;

и вообще, $j_p := \bigoplus_{q \in V_p} c_q$.

ТЕОРЕМА. *I = J*.

8.4 Сжатие данных

Материал раздела кодирования с минимальной избыточностью показывает, что при кодировании наблюдается некоторый баланс между временем и памятью. Затрачивая дополнительные усилия при кодировании и декодировании, можно сэкономить память, и, наоборот, пренебрегая оптимальным использованием памяти, можно существенно выиграть во времени кодирования и декодирования. Конечно, этот баланс имеет место только в определенных пределах, и нельзя сократить расход памяти до нуля или построить мгновенно работающие алгоритмы кодирования. Для достижения прогресса нужно рассмотреть неалфавитное кодирование.

Сжатие текстов

Допустим, что имеется некоторое сообщение, которое закодировано каким-то общепринятым способом (для текстов это, например, код ASCII) и хранится в памяти компьютера. Заметим, что равномерное кодирование (в частности, ASCII) не является

оптимальным для текстов. Действительно, в текстах обычно используется существенно меньше, чем 256 символов (в зависимости от языка – примерно 60–80 с учетом знаков препинания, цифр, строчных и прописных букв). Кроме того, вероятности появления букв различны и для каждого естественного языка известны (с некоторой точностью) частоты появления букв в тексте. Таким образом, можно задаться некоторым набором букв и частотами их появления в тексте и с помощью алгоритма Хаффмена построить оптимальное алфавитное кодирование текстов (для заданного алфавита и языка). Простые расчеты показывают, что такое кодирование для распространенных естественных языков будет иметь цену кодирования несколько меньше 6, то есть даст выигрыш по сравнению с кодом ASCII на 25% или несколько больше.

Методы кодирования, которые позволяют построить (без потери информации!) коды сообщений, имеющие меньшую длину по сравнению с исходным сообщением, называются методами *сжатия* (или *упаковки)* данных. Качество сжатия определяется *коэффицентом сжатия*, который обычно измеряется в процентах и показывает, на сколько процентов кодированное сообщение короче исходного.

Известно, что практические программы сжатия (arj, zip и другие) имеют гораздо лучшие показатели: при сжатии текстовых файлов коэффициент сжатия достигает 70% и более. Это означает, что в таких программах используется *не* алфавитное кодирование.

Предварительное построение словаря

Рассмотрим следующий способ кодирования.

1. Исходное сообщение по некоторому алгоритму разбивается на последовательности символов, называемые *словами* (слово может иметь одно или несколько вхождений в исходный текст сообщения).

2. Полученное множество слов считается буквами нового алфавита. Для этого алфавита строится разделимая схема алфавитного кодирования (равномерного кодирования или оптималь-

ного кодирования, если для каждого слова подсчитать число вхождений в текст). Полученная схема обычно называется *словарем,* так как она сопоставляет слову код.

3. Далее код сообщения строится как пара – код словаря и последовательность кодов слов из данного словаря.

4. При декодировании исходное сообщение восстанавливается путем замены кодов слов на слова из словаря.

Пример

Допустим, что требуется кодировать тексты на русском языке. В качестве алгоритма деления на слова примем естественные правила языка: слова отделяются друг от друга пробелами или знаками препинания. Можно принять допущение, что в каждом конкретном тексте имеется не более 2^{16} различных слов (обычно гораздо меньше). Таким образом, каждому слову можно сопоставить номер – целое число из двух байтов (равномерное кодирование). Поскольку в среднем слова русского языка состоят более чем из двух букв, такое кодирование дает существенное сжатие текста (около 75% для обычных текстов на русском языке). Если текст достаточно велик (сотни тысяч или миллионы букв), то дополнительные затраты на хранение словаря оказываются сравнительно небольшими.

Данный метод попутно позволяет решить задачу *полнотекстового поиска*, то есть определить, содержится ли заданное слово (или слова) в данном тексте, причем для этого не нужно просматривать весь текст (достаточно просмотреть только словарь).

Указанный метод можно усовершенствовать следующим образом. На шаге 2 следует применить алгоритм Хаффмена для построения оптимального кода, а на шаге 1 – решить экстремальную задачу разбиения текста на слова таким образом, чтобы среди всех возможных разбиений выбрать то, которое дает наименьшую цену кодирования на шаге 2. Такое кодирование будет «абсолютно» оптимальным. К сожалению, указанная экстремальная задача очень трудоемка, поэтому на практике не используется – время на предварительную обработку большого текста оказывается чрезмерно велико.

Алгоритм Лемпела-Зива

На практике используется следующая идея, которая известна также как *адаптивное сжатие.* За один проход по тексту одновременно динамически строится словарь и кодируется текст. При этом словарь не хранится – за счет того, что при декодировании используется тот же самый алгоритм построения словаря, словарь динамически восстанавливается.

Здесь приведена простейшая реализация этой идеи, известная как *алгоритм Лемпела-Зива.* Вначале словарь *D :* **array** [**int**] **of string** содержит пустое слово, имеющее код 0. Далее в тексте последовательно выделяются слова. Выделяемое слово – это максимально длинное слово из уже имеющихся в словаре плюс еще один символ. В сжатое представление записывается найденный код слова и расширяющая буква, а словарь пополняется расширенной комбинацией.

Алгоритм 3. Упаковка по методу Лемпела-Зива.

Вход: исходный текст, заданный массивом кодов символов *f*: **array** [l..n] **of char**.

Выход: сжатый текст, представленный последовательностью пар <p, q>, где *p* – номер слова в словаре, *q* – код дополняющей буквы.

D[0]: = ««; *d:* = 0 { начальное состояние словаря }
k: = 1 { номер текущей буквы в исходном тексте }
while *k* ≤ n **do**
p: = FD(*k*) { *p* – индекс найденного слова в словаре }
l: = Length(*D*[*p*[) {*lI* – длина найденного слова в словаре }
yield <*p*, *f*[*k* + *l*]) { код найденного слова и еще одна буква }
d: = *d*+ 1; *D*[*d*]: = *D*[*p*] U *f*[*k* +*lI*
{ пополнение словаря, здесь U – это конкатенация }
k: = *k* + *l* + 1 { продвижение вперед по исходному тексту }
end while

Слово в словаре ищется с помощью несложной функции FD.

Вход: *k* – номер символа в исходном тексте, начиная с которого нужно искать в тексте слова из словаря.

Выход: *p* – индекс самого длинного слова в словаре, совпадающего с символами *f*[*k*]..*f*[*k* + *l*]. Если такого слова в словаре нет, то *p* = 0.

l: = 0; *p*: = 0 { начальное состояние }
for *i* **from** 1 **to** *d* **do**
if *D*[*i*] = *f*[*k*..*k* + Length(*D*[*i*]) – l]&Length(*D*[*i*]) > *l* **then**
p: = i;l: = Length(*D*[*i*]) { нашли более подходящее слово }
end if
end for
return *p*

Распаковка осуществляется следующим алгоритмом.

Алгоритм 4. Распаковка по методу Лемпела-Зива

Вход: сжатый текст, представленный массивом пар *g :* **array** [l..*m*] **of record** *p* **: int;** *q* **: char endrecord,** где *p* – номер слова в словаре, *q* – код дополняющей буквы.

Выход: исходный текст, заданный последовательностью строк и символов.

D[0]: = ««; *d:* = 0 { начальное состояние словаря }
for *k* ≤ *n* **do**
p: = *g*[*k*].*p* { *p* – индекс слова в словаре }
q:= *g*[*k*].*q* { *q* – дополнительная буква }
yield *p* U *q* { вывод слова и еще одной буквы }
d: = d + 1; D[*d*]*: = D*[*p*] *U q*
{ пополнение словаря, здесь U – это конкатенация }
end for

На практике применяют различные усовершенствования этой схемы.

1. Словарь можно сразу инициализировать, например, кодами символов (то есть считать, что однобуквенные слова уже известны).

2. В текстах часто встречаются регулярные последовательности: пробелы и табуляции в таблицах и т.п. Сопоставлять каждой подпоследовательности такой последовательности отдельное слово в словаре нерационально. В таких случаях лучше применить специальный прием, например, закодировать последовательность

пробелов парой <*k*,*s*>, где *k* – количество пробелов, а *s* – код пробела.

8.5 Шифрование

Защита компьютерных данных от несанкционированного доступа, искажения и уничтожения в настоящее время является серьезной социальной проблемой. Применяются различные подходы к решению этой проблемы.

● Поставить между злоумышленником и данными в компьютере непреодолимый барьер, то есть исключить саму возможность доступа к данным путем физической изоляции компьютера с данными, применения аппаратных ключей защиты и т.п. Такой подход надежен, но он затрудняет доступ к данным и легальным пользователям, а потому постепенно уходит в прошлое.

● Поставить между злоумышленником и данными в компьютере логический барьер, то есть проверять наличие прав на доступ к данным и блокировать доступ при отсутствии таких прав. Для этого применяются различные системы паролей, регистрация и идентификация пользователей, разграничения прав доступа и т.п. Практика показывает, что борьба между «хакерами» и модулями защиты операционных систем идет с переменным успехом.

● Хранить данные таким образом, чтобы они могли «сами за себя постоять». Другими словами, так закодировать данные, чтобы даже получив их, злоумышленник не смог бы нанести ущерба.

Этот раздел посвящен обсуждению методов кодирования, применяемых в последнем случае.

8.6 Криптография

Шифрование – это кодирование данных с целью защиты от несанкционированного доступа.

Процесс кодирования сообщения называется *шифрованием* (или *зашифровкой),* а процесс декодирования – *расшифровыванием* (или *расшифровкой).* Само кодированное сообщение назы-

вается *шифрованным* (или просто *шифровкой),* а применяемый метод называется *шифром.*

Основное требование к шифру состоит в том, чтобы расшифровка (и, может быть, зашифровка) была возможна только при наличии санкции, то есть некоторой дополнительной информации (или устройства), которая называется *ключом* шифра. Процесс декодирования шифровки *без* ключа называется *дешифрованием* (или *дешифрацией,* или просто *раскрытием* шифра).

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

Свойство шифра противостоять раскрытию называется *криптостойкостью* (или *надежностью)* и обычно измеряется сложностью алгоритма дешифрации.

В практической криптографии криптостойкость шифра оценивается из экономических соображений. Если раскрытие шифра стоит (в денежном выражении, включая необходимые компьютерные ресурсы; специальные устройства и т.п.) больше, чем сама зашифрованная информация, то шифр считается достаточно надежным.

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

Шифрование с помощью случайных чисел

Пусть имеется датчик *псевдослучайных* чисел, работающий по некоторому определенному алгоритму. Часто используют следующий алгоритм:

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

где T_i – предыдущее псевдослучайное число, T_{i+1} – следующее псевдослучайное число, а коэффициенты a, b, c постоянны и хорошо известны. Обычно $c = 2^n$, где n – разрядность процессора, $a \bmod 4 = 1$, а b – нечетное.

В этом случае последовательность псевдослучайных чисел имеет период c. Процесс шифрования определяется следующим

образом. Шифруемое сообщение представляется в виде последовательности слов S_0, S_1,..., каждое длины n, которые складываются по модулю 2 со словами последовательности T_0,T_1..., то есть

$$C_i := S_i \oplus T_i.$$

Последовательность T_0, T_1,… называется *гаммой шифра*.

Процесс расшифровывания заключается в том, чтобы еще раз сложить шифрованную последовательность с той же самой гаммой шифра:

$$S_i := C_i \oplus T_i.$$

Ключом шифра является начальное значение T_0, которое является секретным и должно быть известно только отправителю и получателю шифрованного сообщения.

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

Если период последовательности псевдослучайных чисел достаточно велик, чтобы гамма шифра была длиннее сообщения, то дешифровать сообщение можно только подбором ключа. При увеличении n экспоненциально увеличивается криптостойкость шифра.

Это очень простой и эффективный метод часто применяют «внутри» программных систем, например, для защиты данных на локальном диске. Для защиты данных, передаваемых по открытым каналам связи, особенно в случае многостороннего обмена сообщениями, этот метод применяют не так часто, поскольку возникают трудности с надежной передачей секретного ключа многим пользователям.

Криптостойкость

Описанный в предыдущем подразделе метод шифрования обладает существенным недостатком. Если известна хотя бы часть исходного сообщения, то все сообщение может быть легко дешифровано. Действительно, пусть известно одно исходное слово S_i. Тогда

$$T_i := C_i \oplus S_i$$

и далее вся правая часть гаммы шифра определяется по указанной формуле датчика псевдослучайных чисел.

На практике часть сообщения вполне может быть известна злоумышленнику. Например, многие текстовые редакторы помещают в начало файла документа одну и ту же служебную информацию. Если злоумышленнику известно, что исходное сообщение подготовлено в данном редакторе, то он сможет легко дешифровать сообщение.

Для повышения криптостойкости симметричных шифров применяют различные приемы:

1) вычисление гаммы шифра по ключу более сложным (или секретным) способом;

2) применение вместо ф более сложной (но обратимой) операции для вычисления шифровки;

3) предварительное перемешивание битов исходного сообщения по фиксированному алгоритму.

В настоящее время широкое распространение получили шифры *с открытым ключом*. Эти шифры не являются симметричными – для зашифровки и расшифровки используются разные ключи. При этом ключ, используемый для зашифровки, является открытым (не секретным) и может быть сообщен всем желающим отправить шифрованное сообщение, а *ключ*, используемый для расшифровки, является закрытым и хранится в секрете получателем шифрованных сообщений. Даже знание всего зашифрованного сообщения и открытого ключа, с помощью которого оно было зашифровано, не позволяет дешифровать сообщение (без знания закрытого ключа).

Для описания метода шифрования с открытым ключом нужны некоторые факты из теории чисел, изложенные (без доказательств) в следующем подразделе.

Модулярная арифметика

В этом подразделе все числа целые. Говорят, что число a *сравнимо по модулю n* с числом b (обозначение: $a \equiv b \pmod n$), если a и b при делении на n дают один и тот же остаток:

$a \equiv b \pmod n := a \bmod n = b \bmod n.$

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

эквивалентности по отношению сравнимости (по модулю n) называются *вычетами* (по модулю n). Множество вычетов по модулю n обозначается Z_n. Обычно из каждого вычета выбирают одного представителя – неотрицательное число, которое при делении на n дает частное 0. Это позволяет считать, что $Z_n = \{0,1,2,\ldots, n-1\}$, и упростить обозначения. Над вычетами (по модулю n) определены операции сложения и умножения по модулю n, обозначаемые, соответственно, $+_n$ и $\bullet_n$ и определяемые следующим образом:

$a +_n b := (a + b) \bmod n$, $a \bullet_n b := (a \bullet b) \bmod n$.

Если из контекста ясно, что подразумеваются операции по модулю n, то индекс n опускается.

Легко видеть, что $\langle Z_n; +_n \rangle$ образует абелеву группу, а $\langle Z_n; +_n, \bullet_n \rangle$ – коммутативное кольцо с единицей.

Рассмотрим Z^*_n – подмножество Z_n чисел, взаимно простых с n.

Можно показать, что $(Z^*_n; \bullet_n)$ – абелева группа. Таким образом, для чисел из множества Z^*_n существуют обратные по умножению по модулю n.

Функция $\varphi(n) := |Z^*_n|$ называется *функцией Эйлера*.

Если p – простое число, то $\varphi(p) = p - 1$, и вообще, $\varphi(n) < n$.

ТЕОРЕМА (Эйлера). *Если* $n > 1$, *то*

$\forall a \in Z^*_n; a^{\varphi(n)} \equiv 1 \pmod n$.

Отсюда непосредственно выводима

ТЕОРЕМА (малая теорема Ферма). *Если* $p > 1$ – *простое число, то*

$\forall a \in Z^*_p; a^{p-1} \equiv 1 \pmod p$.

Имеет место следующее утверждение.

ТЕОРЕМА. *Если числа* $n_1,\ldots, n_k$ *попарно взаимно простые, число* $n = n_1 n_2 \ldots n_k$ – *их произведение,* x *и* a – *целые числа, то*

$x \equiv a \pmod n \Leftrightarrow \forall i \in 1..k \; x \equiv a \pmod{n_i}$.

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

Шифрование с открытым ключом

Шифрование с открытым ключом производится следующим образом.

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

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

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

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

C_i : =$(S_i)^e$ mod *n* и отправляет получателю.

3. Получатель расшифровывает сообщение с помощью закрытого ключа *d:*

P_i: =$(C_i)^d$ mod *n.*

ТЕОРЕМА. *Шифрование с открытым ключом корректно, то есть в предыдущих обозначениях* $P_i = Si.$

Пример

Генерация ключей:

1. *p*: = 3, *q*: = 11;
2. *n*:=*pq* = 3* 11 = 33;
3. (*p* – l)(*q* – 1) = 2 * 10 = 20, *e*: = 7;
4. *d*:= 7^{-1} mod 20 = 3, (7*3 mod 20 = 1).

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

1. C_1 : = 3^7 mod 33 = 2187 mod 33 = 9;
2. C_2 : = 1^7 mod 33 = 1 mod 33 = 1;
3. C_3 : =2^7 mod 33 = 128 mod 33 = 29.

При расшифровке имеем:

1. P_1 : = 9^3 mod 33 = 729 mod 33 = 3;

2. $P_2 := l^3 \bmod 33 = 1 \bmod 33 = 1$;

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

Шифры с открытым ключом сравнительно просты в реализации, очень практичны (поскольку нет необходимости пересылать по каналам связи закрытый ключ и можно безопасно хранить его в одном месте) и в то же время обладают высочайшей криптостойкостью. Кажется, что дешифровать сообщение несложно: достаточно разложить открыто опубликованное число *n* на множители, восстановив числа *p* и *q*, и далее можно легко вычислить секретный ключ *d*. Однако дело заключается в следующем. В настоящее время известны эффективные алгоритмы определения простоты чисел, которые позволяют за несколько минут подобрать пару очень больших простых чисел (по 100 и больше цифр в десятичной записи). В то же время неизвестны эффективные алгоритмы разложения очень больших чисел на множители. Разложение на множители числа в 200 и больше цифр потребовало бы сотен лет работы самого лучшего суперкомпьютера. При практическом применении шифров с открытым ключом используют действительно большие простые числа (не менее 100 цифр в десятичной записи, а обычно значительно больше). В результате вскрыть этот шифр оказывается невозможно, если не существует эффективных алгоритмов разложения на множители (что очень вероятно, хотя и не доказано строго).

8.7 Цифровая подпись

Шифр с открытым ключом позволяет выполнять и многие другие полезные операции, помимо шифрования и посылки сообщений в одну сторону. Прежде всего, для организации многосторонней секретной связи каждому из участников достаточно сгенерировать свою пару ключей (открытый и закрытый), а затем сообщить всем партнерам свой открытый ключ.

Заметим, что операции зашифровки и расшифровки по существу одинаковы и различаются только показателем степени, а потому коммутируют:

$M = (M^e)^d \bmod n = M^{ed} \bmod n = M^{de} \bmod n = (M^e)^d \bmod n = M$.

Это обстоятельство позволяет применять различные приемы, известные как *цифровая* (или *электронная)* подпись.

Рассмотрим следующую схему взаимодействия корреспондентов *X* и *Y*. Отправитель *X* кодирует сообщение *S* своим закрытым ключом *(C: = M*^{d} mod n) и посылает получателю *Y* пару <*S*, *C*>, то есть подписанное сообщение. Получатель *Y*, получив такое сообщение, кодирует подпись сообщения открытым ключом *X*, то есть вычисляет *S': = C*^{e} mod *n*. Если оказывается, что *S* = S'', то это означает, что (нешифрованное!) сообщение *S* действительно было отправлено корреспондентом *X*. Если же $S \neq S'$, то сообщение было искажено при передаче или фальсифицировано.

В подобного рода схемах возможны различные проблемы, которые носят уже не математический, а социальный характер. Например, допустим, что злоумышленник *Z* имеет техническую возможность контролировать всю входящую корреспонденцию *Y* незаметно для последнего. Тогда, перехватив сообщение *X*, в котором сообщался открытый ключ *e*, злоумышленник *Z* может подменить открытый ключ *X* своим собственным открытым ключом. После этого злоумышленник сможет фальсифицировать все сообщения *X*, подписывая их своей цифровой подписью, и, таким образом, действовать от имени *X*. Другими словами, цифровая подпись удостоверяет, что сообщение S пришло из того же источника, из которого был получен открытый ключ е, но не более того.

Можно подписывать и шифрованные сообщения. Для этого отправитель *X* сначала кодирует своим закрытым ключом сообщение 5, получая цифровую подпись *C*, а затем кодирует полученную пару <*S, C*> открытым ключом получателя У. Получив такое сообщение, У сначала расшифровывает его своим закрытым ключом, а потом убеждается в подлинности полученного сообщения, сравнив его с результатом применения открытого ключа *X* к подписи *C*.

К сожалению, даже эти меры не смогут защитить от злоумышленника *Z*, сумевшего подменить открытый ключ *X*. Конечно, в этом случае *Z* не сможет дешифровать исходное сообщение, но он сможет подменить исходное сообщение фальсифицированным.

Вопросы, затронутые в этой главе, очень существенны для практических информационных технологий, которые невозмож-

ны без кодирования, сжатия данных и шифрования. Разумеется, в реальных современных программах применяются более изощренные, по сравнению с описанными здесь простейшими вариантами, методы.

Упражнения

1. Является ли схема алфавитного кодирования

$$<a \to 0,\ b \to 10,\ c \to 011,\ d \to 1011,\ e \to 1111>$$

префиксной? разделимой?

6.2. Построить оптимальное префиксное алфавитное кодирование для алфавита {*a, b*, c, *d*} со следующим распределением вероятностей появления букв:

$$p_a = 1/2,\ p_b = 1/4,\ p_c = 1/8,\ p_d = 1/8.$$

6.3. Показать, что для несимметричных ошибок функция

$$d_\delta(\beta', \beta'') = 2 \min_{\{\beta''' \in B^*\}} \max\left(\min_{\left\{E^\delta\langle \beta', \beta'''\rangle\right\}} \left|E^\delta\langle \beta', \beta'''\rangle\right|, \min_{\left\{E^\delta\langle \beta''', \beta''\rangle\right\}} \left|E^\delta\langle \beta''', \beta''\rangle\right| \right)$$

является расстоянием.

6.4. Проследить работу алгоритма сжатия Лемпела-Зива на примере следующего исходного текста: abaabaaab.

6.5. Пусть в системе программирования имеются процедура Randomize, которая получает целочисленный параметр и инициализирует датчик псевдослучайных чисел, и функция без параметров Rnd, которая выдает следующее псевдослучайное число в интервале [0,1]. Составить алгоритмы шифровки и расшифровки с закрытым ключом.

9 ГРАФЫ

9.1 Определение графа

Понятие графа опирается на понятие множества. Графически задается множество и элементы множества, находящиеся между собой в некотором отношении.

При проектировании конструкций пользователю удобнее иметь дело с моделями, которые легко образуются, если элементы конструкций принять за точки, а связи между ними принять за линии.

Объект, состоящий из двух множеств (множества точек и множества линий), которые находятся между собой в некотором отношении, называется ***графом***.

Точки обозначают $X = \{x_1, x_2, \ldots, x_n\}$, $|X| = n$ и называют ***вершинами*** графа.

Множество линий, соединяющих пары вершин (x_i, x_j), где $x_i, x_j \in X$, называется ***множеством ребер или дуг,*** и обозначается $U = \{u_1, u_2, \ldots, u_m\}$, $|U| = m$.

Графом можно считать объект, который обозначается как $G = (X, U)$.

В общем случае множество линий U можно представить в виде

$$U = \tilde{U} \cup \bar{U} \cup \mathring{U},$$

где $\tilde{U}$ – подмножество неориентированных линий (***ребер***), в котором каждое ребро $u_i \approx \in \tilde{U}$ определяется неупорядоченной парой вершин x_i, x_j, которые оно соединяет, и записывается $u_k = (x_i, x_j)$, или $u_k = (x_j, x_i)$.

$\bar{U}$ – подмножество ориентированных линий (***дуг***).

Существенно направление соединений. Каждая дуга $u_i \in \bar{U}$ определяется упорядоченной парой вершин x_i, x_j, которые u_k соединяет, и записывается $u_k = < x_i, x_j >$.

$\mathring{U}$ – подмножество линий (петель), каждая из которых выходит и входит в одну и ту же соответствующую этой линии вершину. Каждая петля определяется упорядоченной или неупорядоченной парой $u_i = (x_k, x_k)$ или $u_i = < x_k, x_k >$.

Граф G = (X, U), у которого Ū, Ũ, Ů ≠ ∅, называется ***смешанным***.

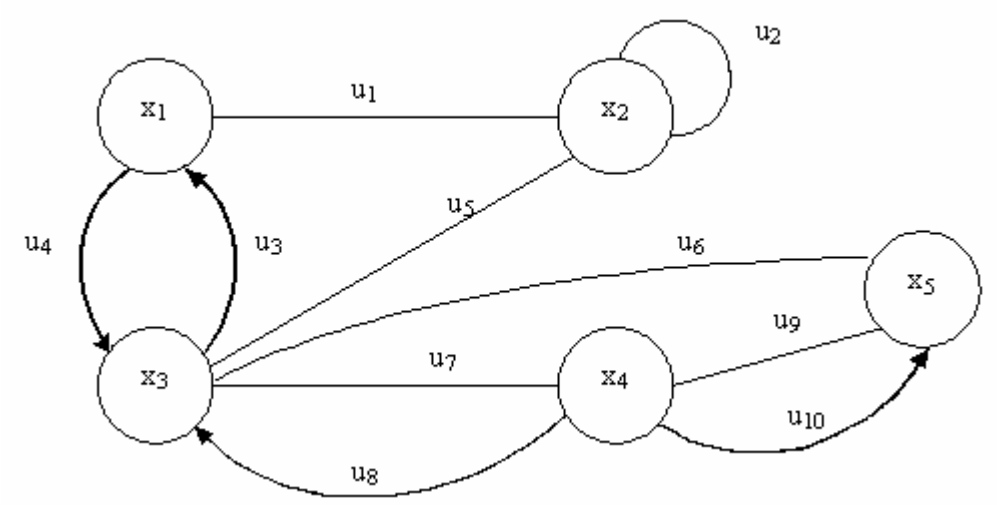


Рисунок 9.1

Здесь |X| = 5; |U| = 13; U = Ũ∪Ū∪Ů;

Ū = {u_3, u_4, u_7, u_8, u_{10}}; Ũ = {u_1, u_5, u_6, u_9}; Ů = {u_2}.

Подмножество U можно представить как множество кортежей длины 2.

Граф G = (X, U), у которого U = Ů∪Ū, а Ũ = Ø, называется ***ориентированным*** графом, или ***орграфом***.

Граф G = (X, U), у которого U = Ũ∪Ů, называется ***неориентированным***, или ***неорграфом***.

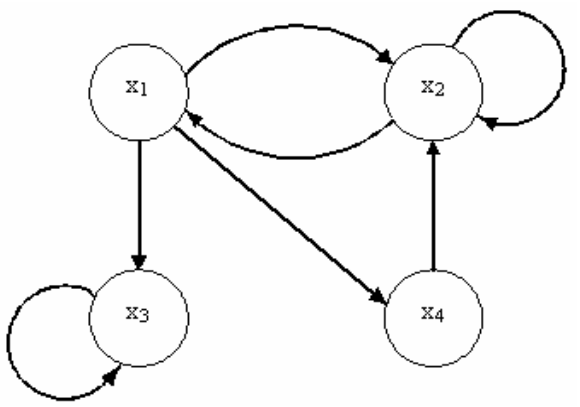


Рисунок 9.2 – Орграф с петлями

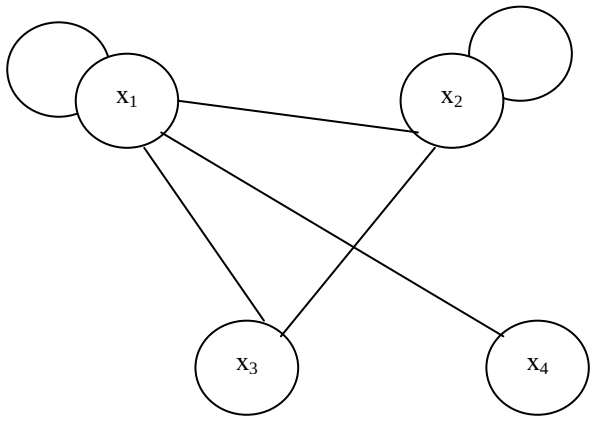


Рисунок 9.3 – Неорграф с петлями

На рисунке 9.2 показан орграф с петлями, где u = {<x_1, x_2>, <x_2, x_1>, <x_2, x_2>, <x_1, x_4>, <x_1, x_3>, <x_4, x_2>, <x_3, x_3>}. Каждая дуга $u_i \in U_z$ представляется парой соединяемых вершин, причем первой в кортеже стоит вершина, из которой дуга выходит, а второй – вершина, в которую дуга входит.

На рисунке 9.3 приведен пример неорграфа с петлями. В дальнейшем неорграфы будем называть просто графами.

Граф G = (X, U), у которого существует хотя бы одна пара вершин, соединяемых m ребрами (m>1) $u_i \in U$, называется ***мультиграфом***, а максимальное значение m называется ***мультиграфическим числом*** графа G. Ребра, соединяющие одну и ту же пару вершин, называются кратными. Пример мультиграфа приведен на рисунке 9.4.

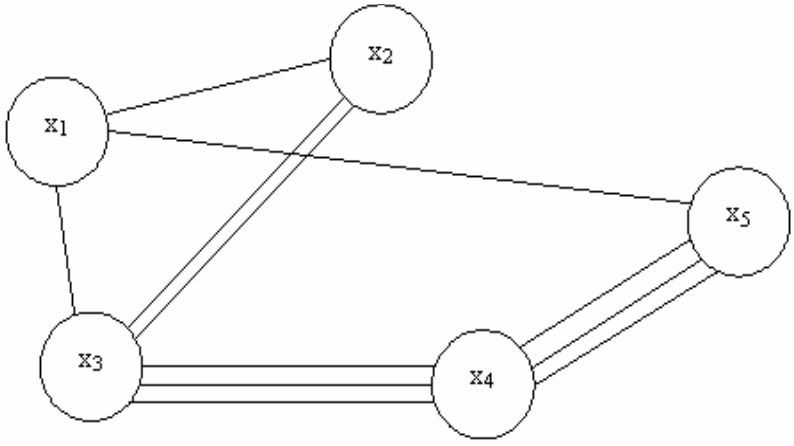


Рисунок 9.4 – Мультиграф

Мультиграфическое число графа, приведенного в качестве примера, равно m = 3.

9.2 Задание графов

Если ребро $u_k \in U$ графа G = (X, U) соединяет вершины x_i, $x_j \in X$, т.е. $u_k = (x_i, x_j)$, то говорят, что ребро u_k ***инцидентно*** вершинам x_i, x_j. Вершины x_i, x_j называют инцидентными ребру u_k.

Любые две вершины x_i, $x_j \in X$ графа G = (X, U) называют смежными, если существует соединяющее эти вершины ребро $u_k \in U$, т.е. $u_k = (x_i, x_j)$. Если два ребра инцидентны одной и той же вершине, то их называют смежными.

Отношения смежности и инцидентности могут иметь место как на множестве X, так и на множестве U.

Основными способами задания графов являются геометрический, аналитический и матричный.

Граф называется ***помеченным***, если его вершины отличаются одна от другой метками. Например, x_1, x_2, x_3,…, x_n.

Говорят, что ***задан граф***, если заданы множество вершин X, множество ребер U и инцидентор F, определяющий, какую пару вершин x_i, $x_j \in X$ соединяет ребро $u_k = (x_i, x_j)$.

Большинство задач автоматизации конструирования решается при помощи матричного задания графа.

Квадратную таблицу $R = ||r_{ij}||_{nyn}$ называют матрицей смежности, если ее элементы образуются по правилу:

$$r_{ij} = \begin{cases} 1, \text{ если вершина } x_i \text{ смежна с } x_j; \\ 0, \text{ в противном случае.} \end{cases}$$

Для мультиграфа запишется:

$$r_{ij} = \begin{cases} m, \text{ если вершина } x_i \text{ соединена с вершиной } x_j \text{ m ребрами;} \\ 0, \text{ в противном случае.} \end{cases}$$

При таком задании очевидно, что матрица будет симметричной.

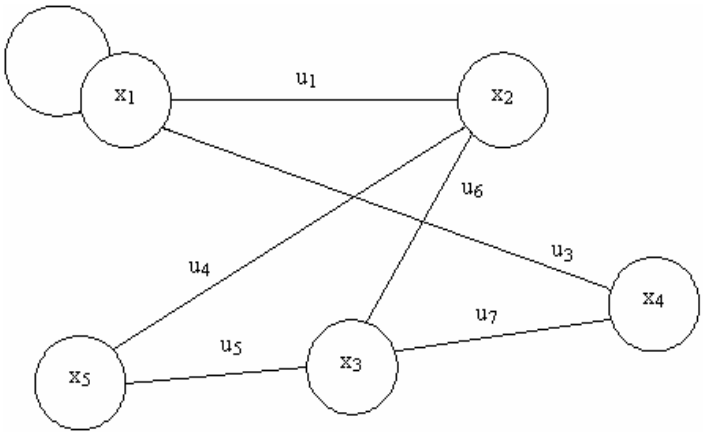


Рисунок 9.5 Пример графа

Матрица смежности R будет выглядеть:

	x_1	x_2	x_3	x_4	x_5
x_1	1	1	0	1	0
x_2	1	0	1	0	1
x_3	0	1	0	1	1
x_4	1	0	1	0	0
x_5	0	1	1	0	0

Недостаток этого представления состоит в том, что объем занимаемой памяти составляет n^2. Объем памяти можно сократить, если хранить треугольную матрицу:

	x_1	x_2	x_3	x_4	x_5
x_1	1	1	0	1	0
x_2		0	1	0	1
x_3			0	1	1
x_4				0	0
x_5					0

в том случае, когда в графе G = (X, U) |X| = n, |U| = m m<<n, его задают с помощью списка пар, соответствующих его ребрам или с помощью списков смежности. Для рисунка 7.5 список пар выглядит: 1 1, 1 2, 1 4, 2 5, 2 3, 3 4, 3 5. Объем памяти составит 2m. При таком представлении нетрудно найти все ребра, ведущие из одной вершины.

При представлении графа списком смежности для каждой вершины $x_i \in X$ составляется список вершин x_j, таких, что $(x_i, x_j) \in U$.

Для графа, приведенного на рисунке 7.5, списки смежности выглядят:

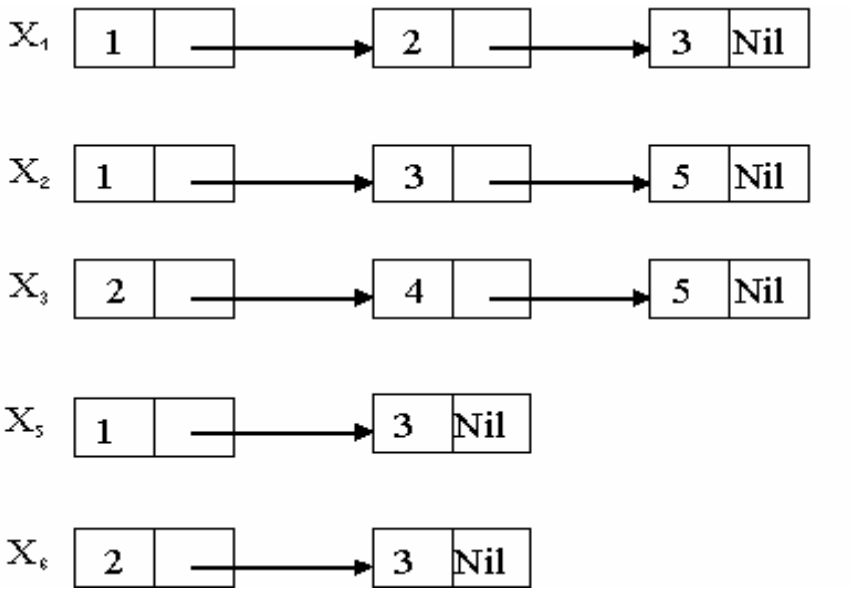


Рисунок 9.6 – Задание графа списком смежности

При таком представлении каждое ребро в списке записано дважды.

Прямоугольная таблица вида $I = \| i_{k,\ell} \|_{nym}$ называется матрицей инциденций, если ее элементы образуются по правилу:

$$i_{k,\ell} = \begin{cases} 1, \text{ если вершина } x_k \text{ инцидентна ребру } u_\ell; \\ 0, \text{ в противном случае.} \end{cases}$$

Матрица инциденций I для графа, изображенного на рисунке 7.5, приведена ниже.

	u_1	u_2	u_3	u_4	u_5	u_6	u_7
x_1	1	1	1	0	0	0	0
x_2	1	0	0	1	0	1	0
x_3	0	0	0	0	1	1	1
x_4	0	0	1	0	0	0	1
x_5	0	0	0	1	1	0	0

Строки таблицы соответствуют вершинам графа, а столбцы – ребрам. В каждом столбце не более двух единиц. Две единицы в случае, если u_i ребро, и одна – если петля.

Матрицы R и I однозначно задают информацию о графе. Можно переходить от одной матрицы к другой. При переходе от I к R теряется нумерация ребер. При переходе от R к I нумеруем единичные элементы $r_{i,j}$ соответствующими значениями $u_k \in U$.

Определим граф $G_s = (U, V)$, двойственный графу $G = (X, U)$.

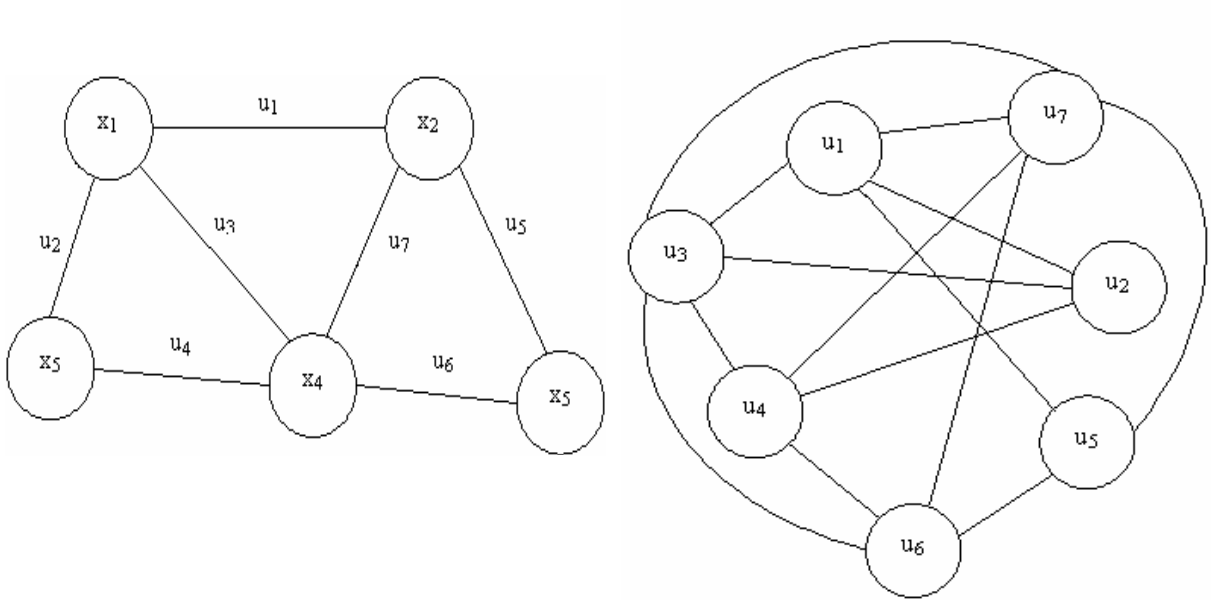


Рисунок 9.7 Граф $G = (X, U)$ и двойственный ему граф $G_s = (U, V)$

Вершинами графа G_s являются ребра графа G, а ребрами – пары (u_i, u_j), причем ребро $v_k = (u_i, u_j)$ соединяет вершины $u_i, u_j \in G_s$, если в графе $G = (X, U)$ ребра $u_i, u_j \in G_s$ смежны.

Граф G называется ***конечным***, если конечны множества его вершин и ребер.

Граф, у которого множество вершин $X \neq \emptyset$ и множество ребер $U = \emptyset$, называется ***нуль-графом***, а вершины его – ***изолированными***. Нуль-граф обозначается через G_0.

Граф $G = (X, U)$ $|X| = n$ называется ***полным***, если между любой парой вершин $x_i, x_j \in X$ имеется ребро $u_k \in U$, $i \neq j$. Полный граф обозначается K_n.

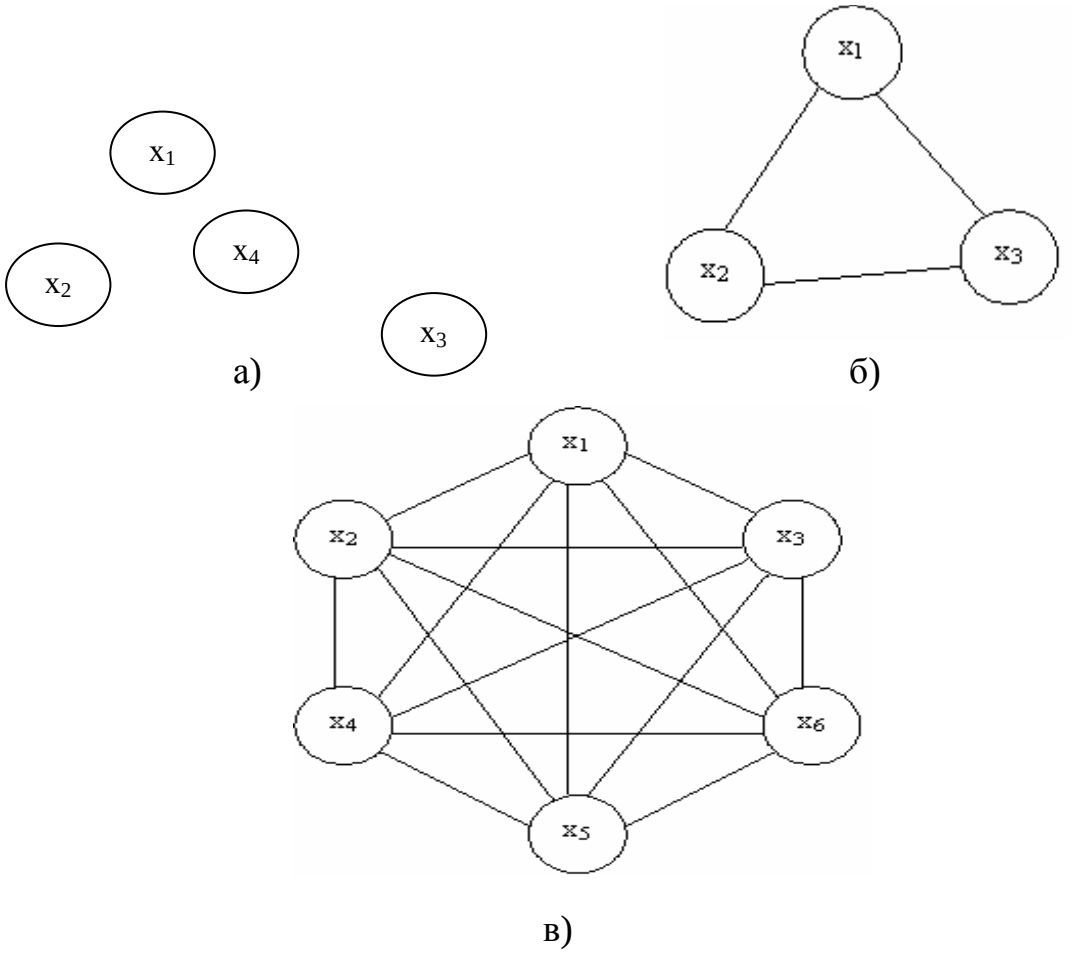


Рисунок 9.8 – Пример нуль-графа (а) и полных графов (б, в)

Число ребер, инцидентных вершине $x_i \in X$ графа, называется ***локальной степенью*** вершины и обозначается $\rho(x_i)$. Степень изолированной вершины равна нулю. Вершина называется ***висячей***, если степень ее равна единице.

Число ребер графа G = (X, U) без петель | x | = n, | u | = m равно

$$m = \sum_{i-1}^{n} \frac{\rho(x_i)}{2}.$$

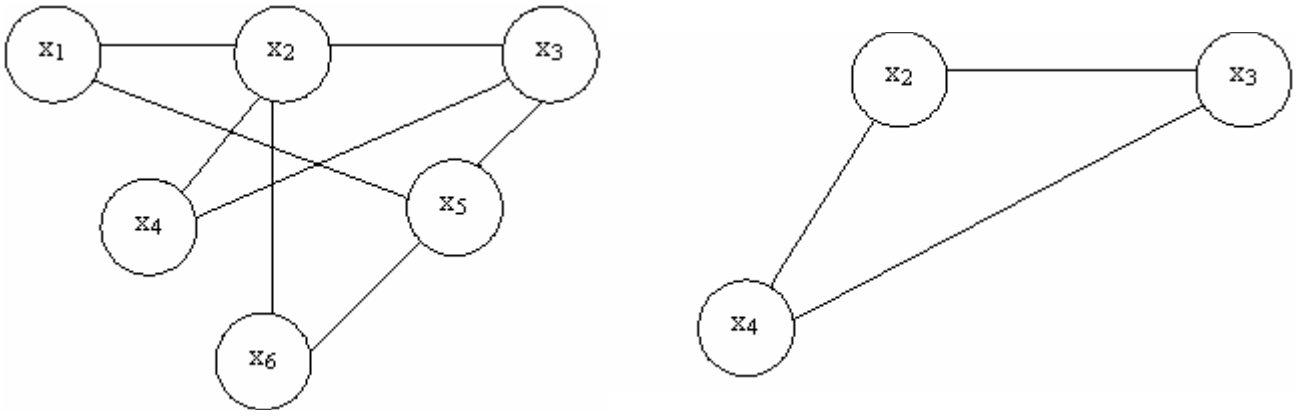
Если в графе имеются петли, то каждую из них нужно считать дважды.

Вершина называется ***четной***, если ее степень есть четное число, и ***нечетной*** – если ее степень нечетное число.

Степень любой вершины полного графа равна n–1, где n – количество его вершин, поскольку каждая вершина соединена с n–1 остальными вершинами графа.

Графы, у которых все вершины имеют одинаковую локальную степень, называются ***регулярными (однородными).*** Очевидно, что всякий полный граф является регулярным. Число ребер регулярного графа с локальной степенью r равно $m=n\cdot r/2$.

Подграфом графа G называется граф, у которого все ребра и вершины принадлежат графу G, т.е. $G''=(X', U')$ подграф графа G, если $X' \subseteq X$, $U' \subseteq U$ и ребра U' соединяют только вершины X'.



а) б)

Рисунок 9.9 – Граф G=(X, U) (а) и его подграф (б)

Суграфом $G'=(X', U')$ графа G=(X, U) называют граф, у которого $X'=X$, $U' \subseteq U$. Пример приведен на рисунке 9.10.

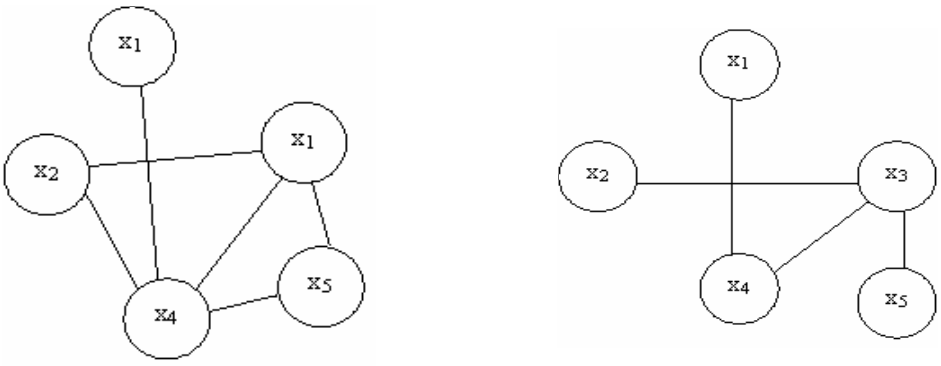
Граф $\overline{G}$ называется дополнением графа G до полного, если множество его ребер состоит из ребер полного графа K_n, не принадлежащих G. $\overline{G}=(X, \bar{U})$, $\bar{U}=U_k \backslash U$, $K_n=(X, U_k)$ (рис. 9.10).

Операции над графами.

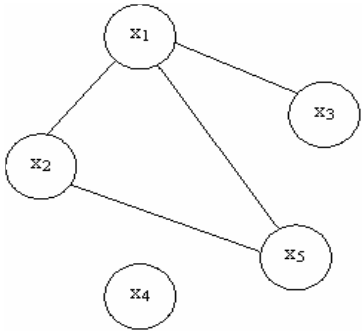
Объединением графов $G_1 = (X_1, U_1)$, $G_2 = (X_2, U_2)$ называют граф $G=G_1 \cup G_2=(X, U)$, где $X=X_1 \cup X_2$ и $U=U_1 \cup U_2$; если $X_1=X_2$ и $U_1 \subset U_2$, то $G=G_1 \cup G_2=G_2$; если $X_1=X_2$ и $U_1=U_2$, то $G=G_1 \cup G_2=G_1=G_2$. Пример объединения приведен на рисунке 7.11.

Пересечением двух графов G_1 и G_2 называется граф G=(X, U), где $X=X_1 \cap X_2$ и $U=\overline{U_2} \cap \overline{U_1}$.

$G=G_1 \cap G_2 = \varnothing$, если $X_1 \cap X_2 = \varnothing$, а $\overline{U_1} \cap \overline{U_2} = \varnothing$, то $G=G_1 \cap G_2$ нуль-граф.



а) б)



в)

Рисунок 9.10 – Граф G=(X, U) (а), его суграф (б), дополнение (в)

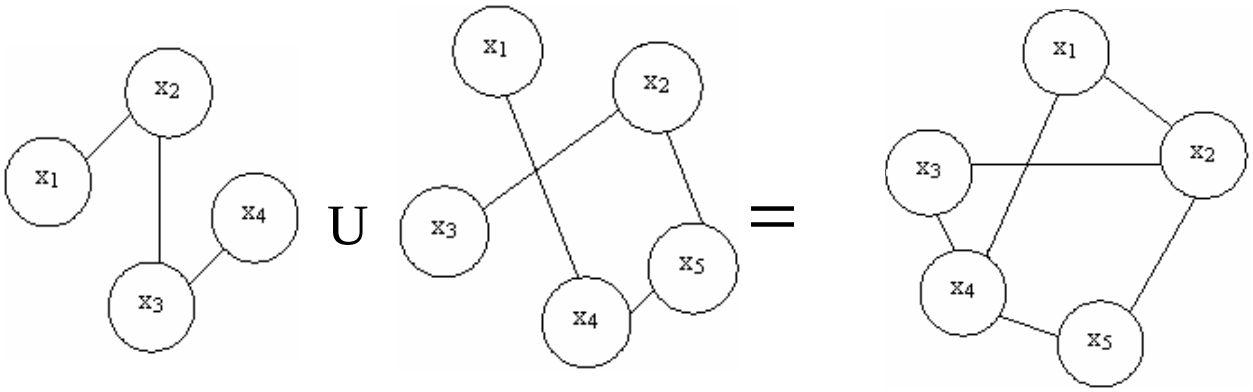


Рисунок 9.11 – Пример объединения

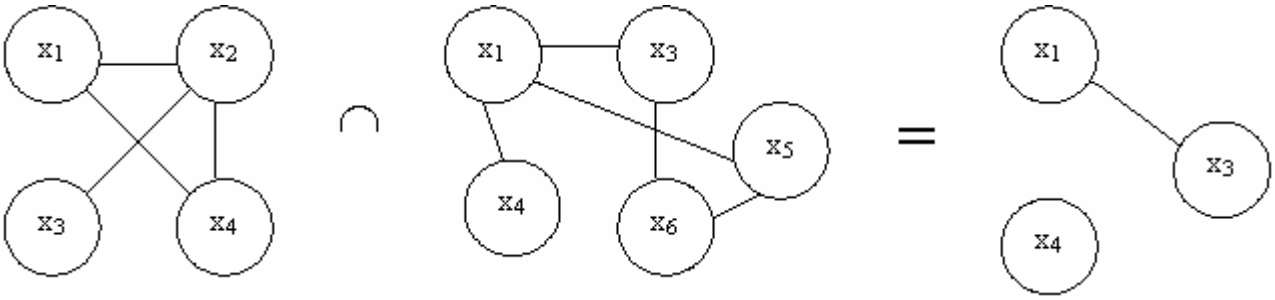


Рисунок 9.12 – Пример пересечения

9.3 Связность графа

Маршрутом в графе G=(X, U) называют некоторую конечную последовательность ребер вида $s=(x_0, x_1)(x_1, x_2),\ldots, (x_{l-1}, x_l)$, где x_0, x_2 – начальная и конечная вершины, соответственно. Число ребер в маршруте называется его ***длиной.***

Маршрут, в котором нет повторяющихся ребер, называют ***цепью***. Если в маршруте различны все вершины, то он называется ***простой цепью***. Замкнутая цепь, у которой начальная и конечная вершины совпадают $x_0=x_l$, называется ***циклом***.

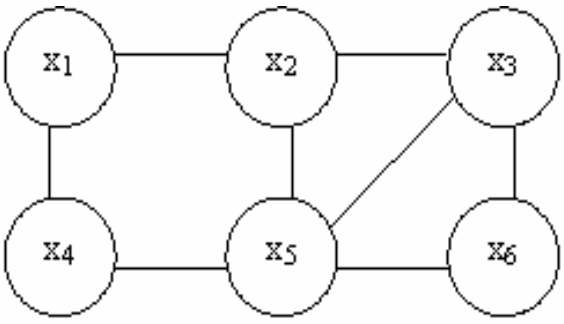


Рисунок 9.13 – Граф G

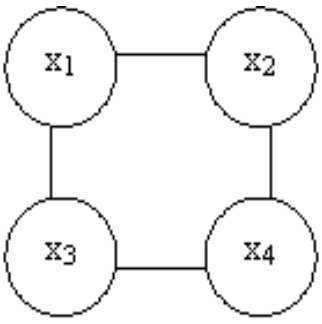


Рисунок 9.14 – Граф

На рисунке 9.13 в графе G построим маршрут, цепь и цикл.

$S_1 = (x_1,x_2)(x_2,x_3)(x_3,x_5)(x_5,x_2)(x_2,x_1)(x_1,x_4)$ – маршрут ;

$S_2 = (x_1,x_2)(x_2,x_3)(x_3,x_5)(x_5,x_2)$ – цепь;

$S_3 = (x_5,x_2)(x_2,x_3)(x_3,x_6)(x_6,x_5)$ – цикл;

$S_4 = (x_4,x_5)(x_5, x_3)(x_3,x_6)$ – простая цепь.

Существует простой способ определения существования маршрутов длины q по матрице R графа G путем возведения ее в q степень.

Пусть задана матрица смежности R графа, изображенного на рисунке 9.14.

$R^2 =$

	1	2	3	4
1	0	1	1	0
2	1	0	0	1
3	1	0	0	1
4	0	1	1	0

R=

	1	2	3	4
1	2	0	0	2
2	0	2	2	0
3	0	2	2	0
4	2	0	0	2

Каждый элемент в матрице R^2 равен числу маршрутов, ведущих из вершины x_i в вершину x_j. Например, $r_{3,2} = 2$ указывает, что существуют два маршрута длины 2 из вершины x_3 в вершину x_2. $s_1 = (x_3,x_1)(x_1, x_2)$; $s_2 = (x_3, x_4)(x_4, x_2)$. (Возведение матрицы в степень производится по правилу умножения матриц). Для определения числа маршрутов длины 3 необходимо определить матрицу R^3.

$R^3 =$

	1	2	3	4
1	0	4	4	0
2	4	0	0	4
3	4	0	0	4
4	0	4	4	0

$r_{1,2}$=4, показывает, что существует 4 маршрута длины 3, ведущие из вершины x_1 в x_2.

$s_1 = (x_1, x_2)(x_2, x_1)(x_1, x_2)$; $s_2 = (x_1, x_3)(x_3, x_1)(x_4, x_2)$;
$s_3 = (x_1, x_2)(x_2, x_4)(x_4, x_2)$; $s_4 = (x_1, x_3)(x_3, x_1)(x_1, x_2)$.

Понятие связности графов относится к одному из наиболее важных понятий теории графов.

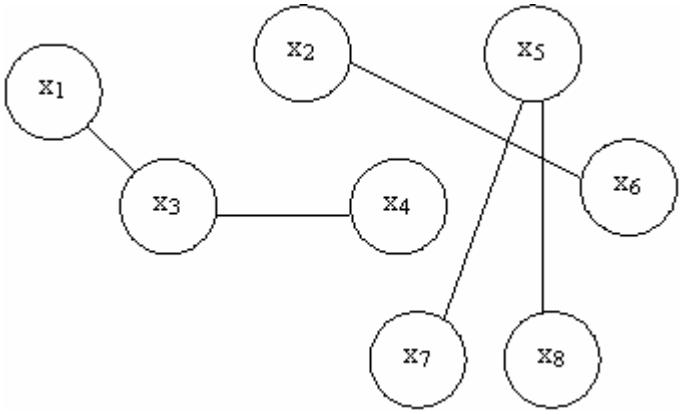
Две произвольные вершины x_i, $x_j \in X$ графа G=(X, U) называются ***связными***, если существует маршрут 5, в котором вершины x_i, x_j будут концевыми. Граф называется ***связным***, если любые две его вершины связны, т.е. 2 вершины объединены простой цепью. В противном случае граф не связан, а каждый из составляющих его связных подграфов G_1, G_2,…, G_e называется ***компонентой связности***.

Из определения связности следует:

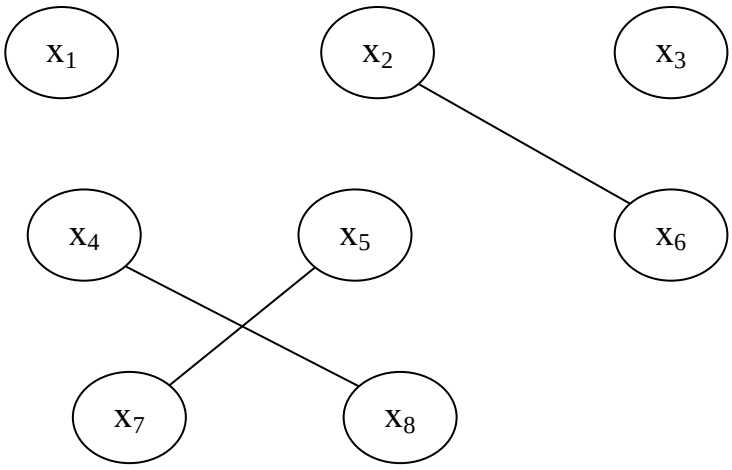
- в связном графе вершина x_i связана сама с собой (рефлексивность);

- если вершина x_i связана с вершиной x_j, то x_j связана с x_i (симметричность);
- если x_i связана с x_j и x_j связана с x_k, то x_i связана с x_k (x_i, x_j, $x_k \in X$) (транзитивность), из чего следует, что отношение связности является отношением эквивалентности.

В этом случае множество вершин графа G=(X, U), который моделирует схему, можно разбить на непересекающиеся классы X_i, причем ребра графа будут соединять только вершины внутри этих классов. Число компонент, из которых состоит граф, называется ***степенью связности***. Связный граф состоит из одной компоненты связности. Примеры графов, состоящих из нескольких компонент связности, приведены на рисунке 9.15.



а)



б)

Рисунок 9.15 – Граф, состоящий из трех компонент связности (а), из пяти компонент связности (б)

Одной из характеристик связных графов является число ребер в графе с n вершинами и заданным числом k компонент связности. Число ребер удовлетворяет неравенству:

$$n - k \le m \le (n - k)(n - k - 1)/2.$$

Граф с n вершинами, содержащий более чем (n–1)(n–2)/2 ребер, связан.

Нахождение простых цепей

Необходимо найти все простые цепи графа G, соединяющие две произвольные вершины. Граф может быть связан, а может быть не связан. Если вершины принадлежат одной компоненте связности, то решение можно найти. Если вершины принадлежат разным компонентам связности, то решения нет.

Алгоритм нахождения простых цепей из вершины i в вершину j состоит из следующих шагов:

1. Выбрать вершину i.
2. Для выбранной вершины составить список смежных вершин таких, каких не было в цепи.
3. Проверить есть ли среди них искомая вершина.
4. Если есть искомая вершина – записать цепь отдельно.
5. Для остальных вершин (для каждой из вершин) перейти к п.2.

Цикл необходимо выполнить n–1 раз для связного графа.

Рассмотрим пример нахождения простых цепей для графа, приведенного на рисунке 9.16. Построим все простые цепи из вершины x_1 в вершину x_7.

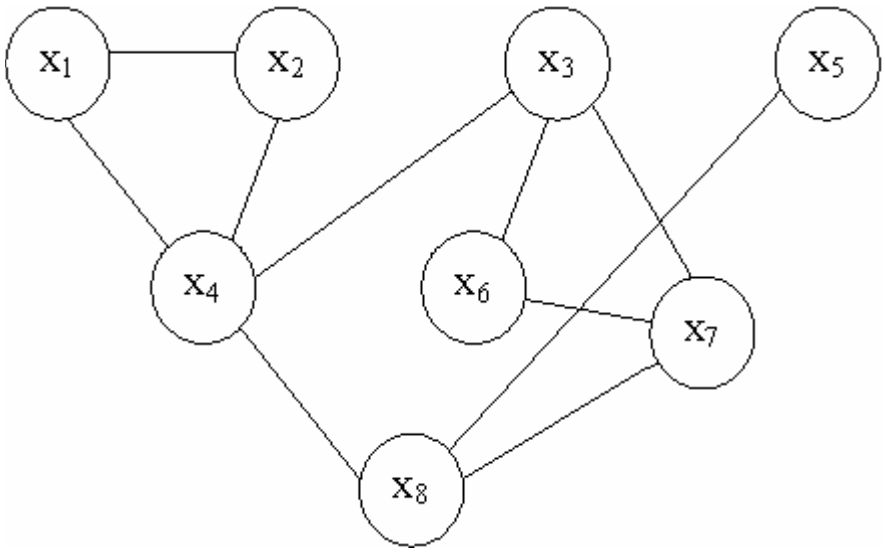


Рисунок 9.16

Для поиска в ширину на первом шаге строим простые цепи 1-2 и 1-4. На втором шаге выбираем вершину 2 в качестве исходной и рассматриваем все смежные с ней вершины. С ней смежны вершины 1 и 4, но вершина 1 уже присутствует в цепи, поэтому полученная цепь будет 1-2-4. Для вершины 4 получим цепи 1-4-2, 1-4-3, 1-4-8. На третьем шаге необходимо обратить внимание на то, что цепь 1-4-2 тупиковая, поскольку все вершины, смежные с вершиной 2, а это вершины 2 и 4, уже присутствуют в цепи.

Шаг 1	Шаг 2	Шаг 3	Шаг 4	Шаг 5
1-2 1-4	1-2-4 1-4-2 1-4-3 1-4-8	1-2-4-3 1-2-4-8 1-4-3-7 1-4-3-5 1-4-8-7 1-4-8-6	1-2-4-3-7 1-2-4-3-5 1-2-4-8-7 1-2-4-8-6 1-4-3-5-7	1-2-4-3-5-7
Либо				
1-2 1-4	1-2-4 1-4-3 1-4-8	1-2-4-3 1-2-4-8 1-4-3-7 1-4-3-7 1-4-8-6 1-4-8-7	1-2-4-3-5 1-2-4-3-7 1-2-4-8-6 1-2-4-8-7 1-4-3-5 тупик	1-2-4-3-5-7 тупик 1-4-3-5-7

Пусть задан связный граф G=(X, U). Подмножество U’⊆U называется ***разделяющим***, если после его удаления граф становится несвязным. Разделяющее подмножество всегда существует. Если разделяющее множество состоит из одного ребра $u_i \in U$, то u_i называется ***перешейком*** или ***мостом***. Ребра (x_3,x_4) (x_4,x_7) графа, приведенного на рисунке 9.17, являются перешейком.

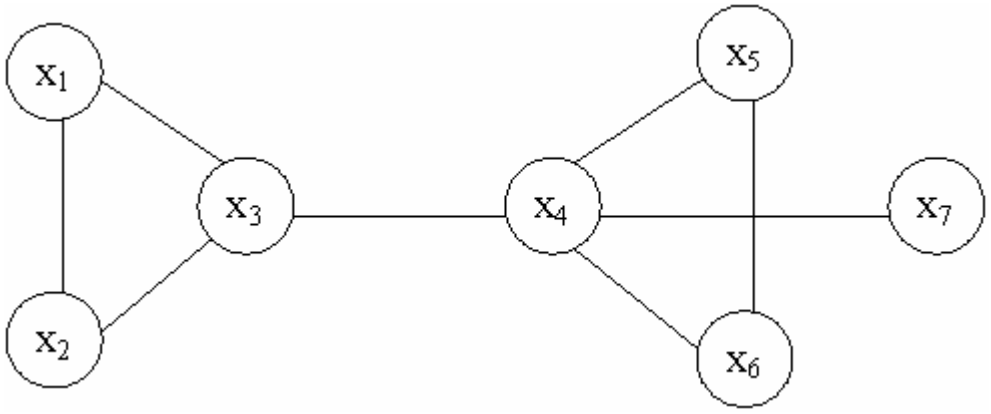


Рисунок 9.17 – Пример графа содержащего мост

9.4 Эйлеровы и гамильтоновы графы

Связный граф G=(X, U) называется ***эйлеровым***, если существует замкнутая цепь (цикл), проходящая через каждое ребро графа один раз.

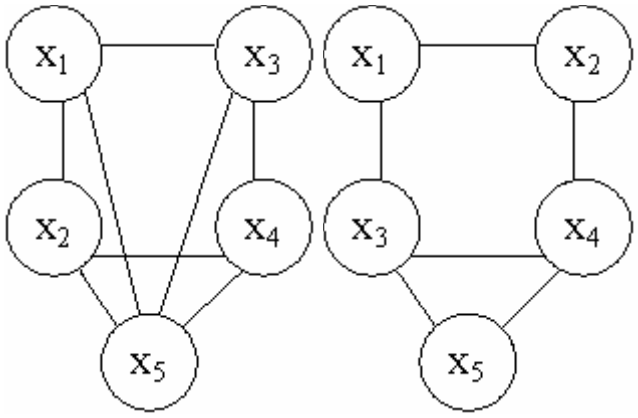


Рис. 9.18 – Неэйлеров граф

Рис. 9.19 – Полуэйлеров граф

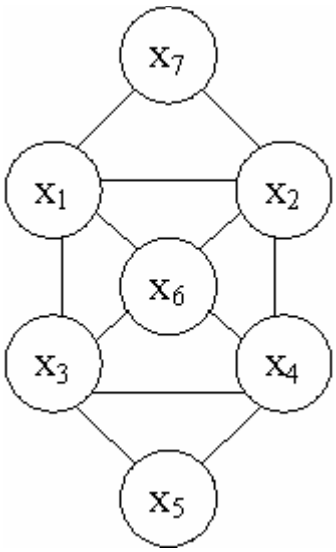


Рис. 9.20 – Эйлеров граф

Граф называется ***полуэйлеровым***, если существует незамкнутая цепь, проходящая через каждое ребро один раз.

Рассмотрим условие существования эйлерова цикла.

Конечный граф G является эйлеровым, если он связан и все его локальные степени четные (рис. 9.20). Заметим, что граф яв-

ляется полуэйлеровым, если в нем не более двух вершин имеют нечетные локальные степени.

Идея алгоритма ***Флери*** построения эйлеровой цепи в эйлеровом графе. Пусть G – эйлеров граф. Тогда необходимо выйти из произвольной вершины и проходить по ребрам G, соблюдая правила:

1. Стирать ребра по мере их прохождения и стирать изолированные вершины.

2. По мосту можно проходить только тогда, когда нет других возможностей.

При программировании использовать 2 стека.

Цикл, проходящий по всем вершинам графа G один раз, называется ***гамильтоновым***, а граф G называется гамильтоновым графом. Для гамильтонова графа не существует критерия существования гамильтонова цикла, есть только теоремы, дающие достаточные условия его существования или отсутствия.

Теорема 1. Если в графе с n ($n \geq 3$) вершин для любой пары несмежных вершин x_i, x_j $\rho(x_i)+ \rho(x_j) \geq n$, то граф имеет гамильтонов цикл.

Теорема 2. В графе без гамильтонова цикла длина его наибольших простых цепей удовлетворяет неравенству $\ell \geq \rho(x_{n1}) + \rho(x_{n2})$, где $\rho(x_{n1})$ и $\rho(x_{n2})$ – две наименьшие локальные степени графа.

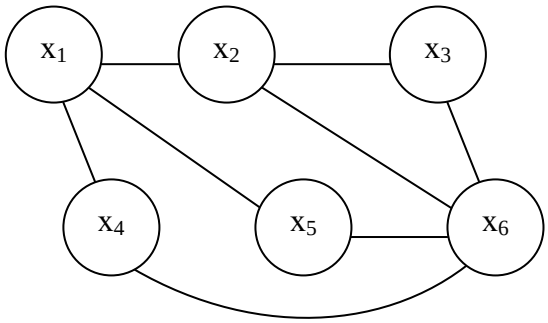


Рисунок 9.21

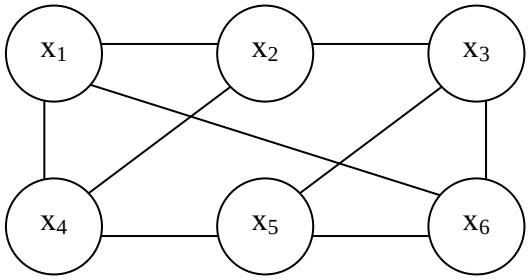


Рисунок 9.22

Если в графе G существует висячая вершина, то гамильтонов цикл отсутствует. Если граф G полный, то он содержит гамильтонов цикл.

9.5 Деревья

Связный граф без циклов называется деревом и обозначается $T = (X, U)$, $|X| = n$, $|T| = n-1$. Начальную вершину называют ***корнем***, из которого выходят ребра, называемые ***ветвями*** дерева. Любые две вершины дерева связаны единственной цепью. В любом связном графе G можно выделить некоторое дерево T.

Дерево, у которого число вершин равно числу вершин графа из которого выделено дерево, а ребро является подмножеством этого графа, называется ***покрывающим*** деревом.

Для одного и того же связного графа можно выделить некоторое множество покрывающих его деревьев.

Теорема 3. Число t покрывающих деревьев в полном графе K_n составляет $t=n^{n-2}$. Множество деревьев называется ***лесом***.

Задачи выделения эйлеровых и гамильтоновых циклов, а также покрывающих деревьев, связаны с задачами о лабиринте, коммивояжере, с построением деревьев минимальной точности. Задача о лабиринте в терминах теории графов формулируется как задача отыскания в связном графе $G=(X, U)$ маршрута с наименьшим числом ребер, который начинается в заданной вершине $x_i \in X$ и приводит в искомую вершину $x_j \in X$.

Метод Тремо

От вершины x_i необходимо перейти ко всем вершинам, находящимся на расстоянии 1 дин. Каждое ребро $u_i = (x_i, x_\ell)$ помечается один раз, когда оно смежно с вершинами x_i, x_ℓ в вершине x_i, это ребро помечается как «открытое». Если окажется, что нет ребер, инцидентных x_ℓ, кроме u_i, то, вернувшись в x_i, ребро u_i помечается как «закрытое». Если некоторое другое ребро $u'_i = (x_i, x_\ell)$ также ведет из x_i в x_ℓ оно также помечается как закрытое.

Для попадания в вершины, находящиеся на расстоянии 2, берется открытое ребро $u = (x_i, x_\ell)$ и снова помечается. В x_ℓ открытые ребра проходятся и помечаются как закрытые, если они ведут к пройденным вершинам, и так со всеми открытыми ребрами. Если искомая вершина расположена на расстоянии n, то при ее достижении все открытые ребра будут помечены n раз.

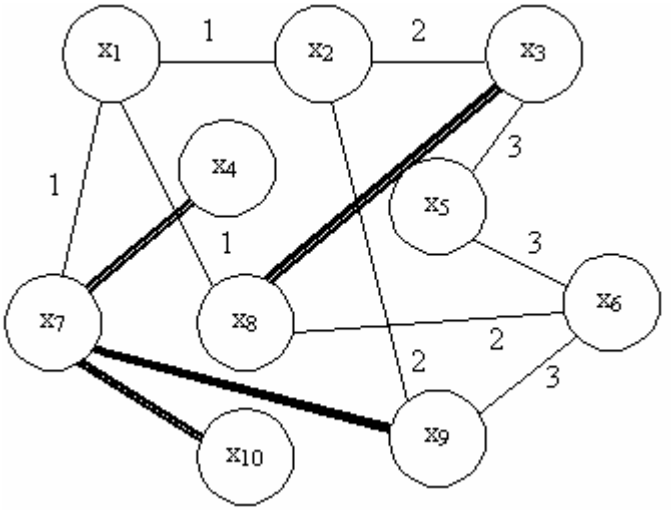


Рисунок 9.23

Задача 2. Пусть необходимо проложить сеть проводов между терминалами при условии, что количество затраченного провода должно быть минимальным, т.е. построить граф типа дерево.

Задача состоит в нахождении одного из n^{n-2} деревьев.

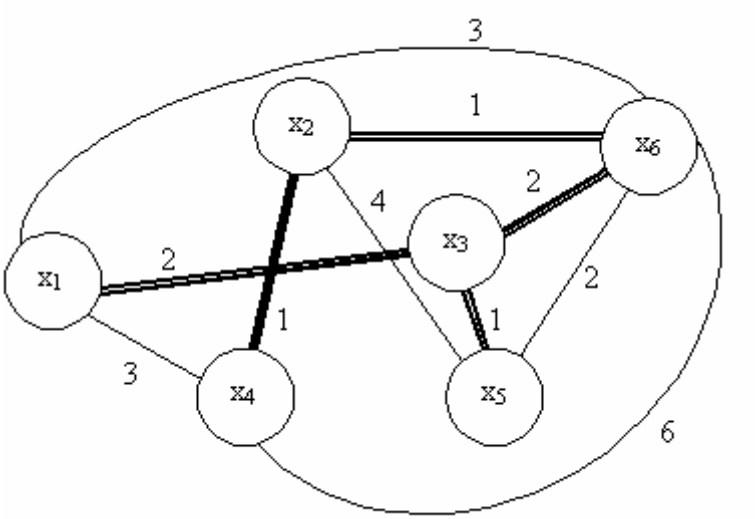
Пусть G=(X, U) связной граф, и каждому ребру $u_i \in U$ ставится в соответствие некоторое неотрицательное число ν_i, называемое мерой. Необходимо найти покрывающее дерево T, для которого сумма мер, взятая по всем ребрам, минимальна.

$$\sum_{i=1}^{m} \nu(u_i) = \min.$$

Метод Краскала

1. В связном графе G=(X, U), | X | = n, определяется ребро с наименьшей мерой.

2. Строится по индукции последовательность ребер u_2, u_3,…, u_{n-1}, причем на каждом шаге выбирается ребро с наименьшей мерой, не совпадающее с выбранными и не образующее циклов с предыдущими ребрами u_i. Полученный подграф T графа G является искомым деревом.



$(x_2x_6)=1$
$(x_6x_3)=2$
$(x_3x_5)=1$
$(x_3x_1)=2$
$(x_2x_4)=1$

9.6 Понятие метрики графа

Метрика графа основана на понятии расстояния. Назовем расстоянием $d(x_i, x_j) = d_{ij}$ между вершинами $x_i, x_j \in X$ графа $G=(X, U)$ длину кратчайшей цепи, соединяющей эти вершины. Под длиной цепи понимается число входящих в нее ребер.

Функция $d(x_i, x_j)$, определенная на множестве ребер графа G, называется ***метрикой*** графа. Метрика удовлетворяет аксиомам Фреше:

$\forall\ x_i\ x_j \in X[d(x_i, x_j) \geq 0]$;

$\forall\ x_i\ x_j \in X[d(x_i, x_j) = 0 \leftrightarrow x_i = x_j]$;

$\forall\ x_i\ x_j \in X[d(x_i, x_j) = d(x_i, x_j)]$;

$\forall\ x_i, x_j, x_k \in X[d(x_i, x_j) + d(xj, x_k) \geq d(x_i, x_k)]$,

т.к. $d(x_i, x_j)$ кратчайшая цепь

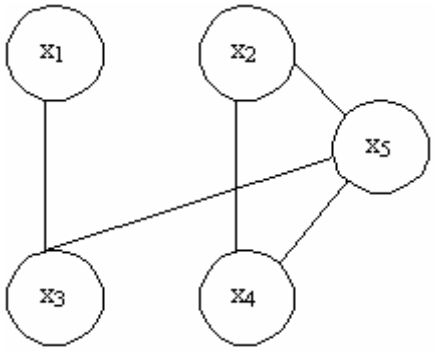
Функция расстояний задается матрицей

$$D = \| d_{ij} \| = \begin{cases} 0, \text{ если } x_i = x_j; \\ \\ d_{ij}, \text{ если } x_i \neq x_j; \end{cases}$$

Список расстояний можно задать в виде множества кортежей длины три $< x_i, x_j, d_{ij} >$.

Для графа, приведенного на рисунке 9.24.

D={<1,2,3>, <1,3,1>, <1,4,3>, <1,5,2>, <2,3,2>, <2,4,1>, <2,5,1>, <3,4,2>, <3,5,1>, <4,5,1>}



	1	2	3	4	5
1	0	3	1	3	2
2	3	0	2	1	1
3	1	2	0	2	1
4	3	1	2	0	1
5	2	1	1	1	0

Рисунок 9.24

Диаметр графа d(G) определяется как максимальное расстояние между его вершинами.

$D(G) = \max dij,\ x_i\ x_j \in X$

Интерес представляет нахождение расстояний в графах частного вида, называемых координатной решеткой.

$G_r = (X_r, U_r)$

В графе $G_r = (X_r, U_r)$ множество X_r соответствует узлам решетки, а U_r – вертикальным и горизонтальным отрезкам, соединяющим узлы решетки.

Введем декартову систему координат с осями s и t.

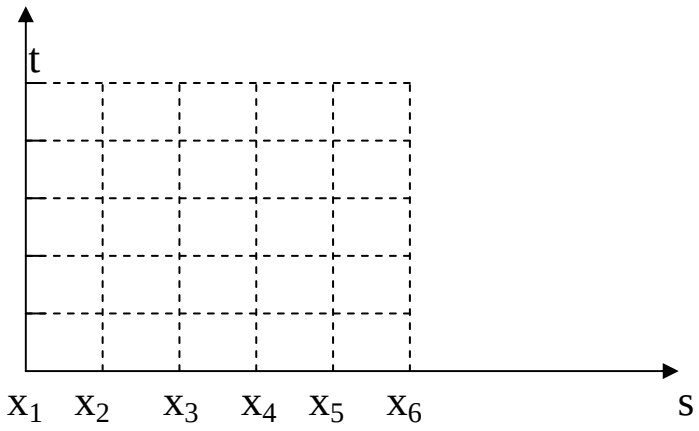


Рисунок 9.25

Расстояние между соседними узлами решетки называют ***шагом решетки*** и принимают равным единице. Расстояние между двумя произвольными вершинами в решетке G_r рассчитывается:

$$d_{ij} = |s_i - s_j| + |t_i - t_j|,$$

где s_i, s_j и t_i, t_j – координаты x_i и x_j.

Обычно задаются размеры решетки p x q, где p – число узлов решетки по оси s, q – not. Например, d(7, 13) = |1 – 2| + |5 – 0| = 6.

Граф G_r удовлетворяет аксиомам Фреше. Если произвольный граф G отображается в G_r так, что любые вершины G размещаются в узлах решетки, то расстояние между вершинами G определяется как расстояние между соответствующими узлами решетки G_r.

Любой граф G может быть отображен в решетку G_r.

Для подсчета суммарной длины L(G) ребер графа G, отображенного в решетку G_r, введем понятие матрицы геометрии D_v.

D_v представляет собой часть матрицы расстояний D, в которой исключены элементы d_{ij}, если вершины x_i $x_j \in X$ не смежны в графе G.

$$d^v_{ij} = \begin{cases} 0, \text{ если } r_{ij} = 0; \\ \\ d_{ij}, \text{ если } r_{ij} \neq 0. \end{cases}$$

Сумма элементов матрицы D_v определяет удвоенную суммарную длину L(G) ребер графа G при данном его отображении в решетку G_r.

9.7 Цикломатическое число, раскраска

Наименьшее число ребер, которое необходимо удалить из графа G, чтобы он стал ациклическим, называется ***цикломатическим числом*** графа. Для графа G=(X, U), |X| = n, |U| = m цикломатическое число j(G) = m – n + k, где m – число ребер графа, n – число вершин, k – число компонент связности графа. Для графа, состоящего из одной компоненты связности j(G) = m – n + 1, например, на рис. 9.26 j(G) = 7 – 5 = 2, т.е. после удаления двух ребер граф становится ациклическим, в примере это ребра (например, (x_4, x_2), (x_4, x_3)). Чтобы узнать, какие ребра удалять, необходимо каждый раз удалять ребро, которое разрушает хотя бы один цикл.

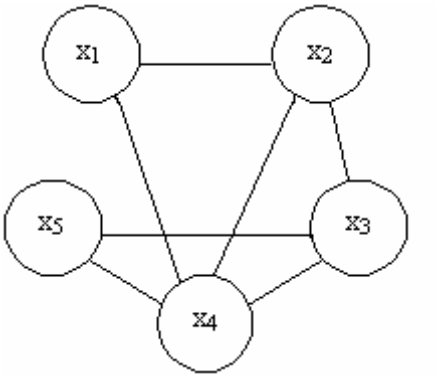


Рисунок 9.26

Раскраской вершин графа называется разбиение множества вершин графа на l непересекающихся классов (подмножеств)

$$X_1, X_2, ..., X_l; \quad X = \bigcup_{i=1}^{l} X_i; X_i \bigcap X_j = \emptyset; \quad i, j \in \overline{1,l}; \quad i \neq j,$$

таких, что внутри каждого подмножества X_i не должно быть смежных вершин.

Если каждому подмножеству X_i поставить в соответствие определенный цвет, то вершины внутри подмножества можно окрасить в один цвет, вершины другого – в другой и т.д. до полной раскраски. В этом случае граф называется l-раскрашиваемым. Наименьшее число подмножеств, на которое разбивается граф при раскраске, называется ***хроматическим числом*** K(G).

Очевидно, полный граф K_n можно раскрасить только в n цветов K(Kn)=n.

Для связного графа G = (X, U) с $n-1 \le m \le n(n-1)/2$ верхняя оценка хроматического числа:

$$K(G) = \left[\frac{3 + \sqrt{9 + 8(m - n)}}{2} \right].$$

Нижней оценкой K(G) является число вершин в наибольшем полном подграфе графа G.

Хроматическое число обычно определяется с помощью методов линейного программирования.

Важное практическое применение имеют 2-раскрашиваемые или двудольные (граф Кенига) (бихроматические) графы. Обозначается двудольный граф $G_{n1,n2} = (X_1, X_2, U)$, где $X_1 \cup X_2 = X$,

$X_1 \cap X_2 = \varnothing$, а ребра соединяют только подмножества X_1 и X_2 между собой.

Пример двудольного графа.

$G_{n1,n2} = (X_1, X_2, U)$; $|X_1| = n_1 = 4$;

$|X_2| = n_2 = 3$; $X_1 = \{x_1, x_2, x_3, x_4\}$;

$X_2 = \{x_5, x_6, x_7\}$.

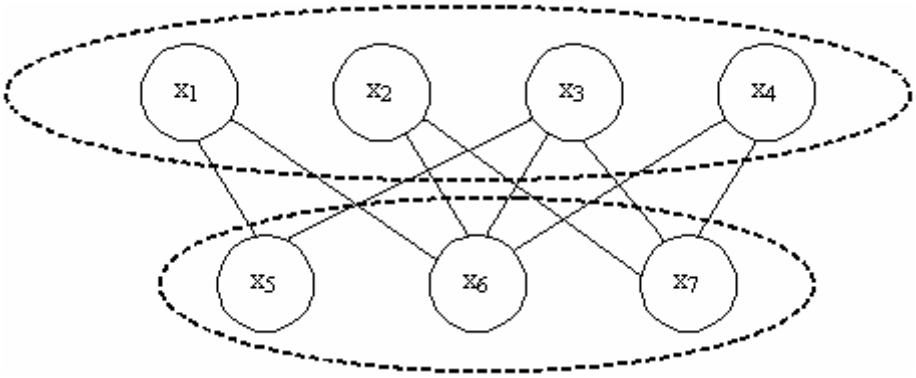


Рисунок 9.27 – Двудольный граф

Граф $K_{n1,\ n2}$ называется полным двудольным графом, если любая вершина $x_i \in X_1$ смежна каждой вершине $x_j \in X_2$ ($i \neq j$).

В двудольном графе множество X_1 можно раскрасить одним цветом, X_2 – другим цветом.

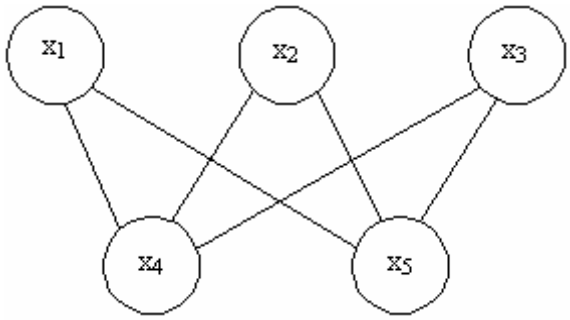


Рисунок 9.28 – Полный двудольный граф K_{32}

При разработке алгоритмов компоновки, размещения и трассировки возникает необходимость определения двудольности некоторого графа или выделения в этом графе максимальных непересекающихся двудольных частей.

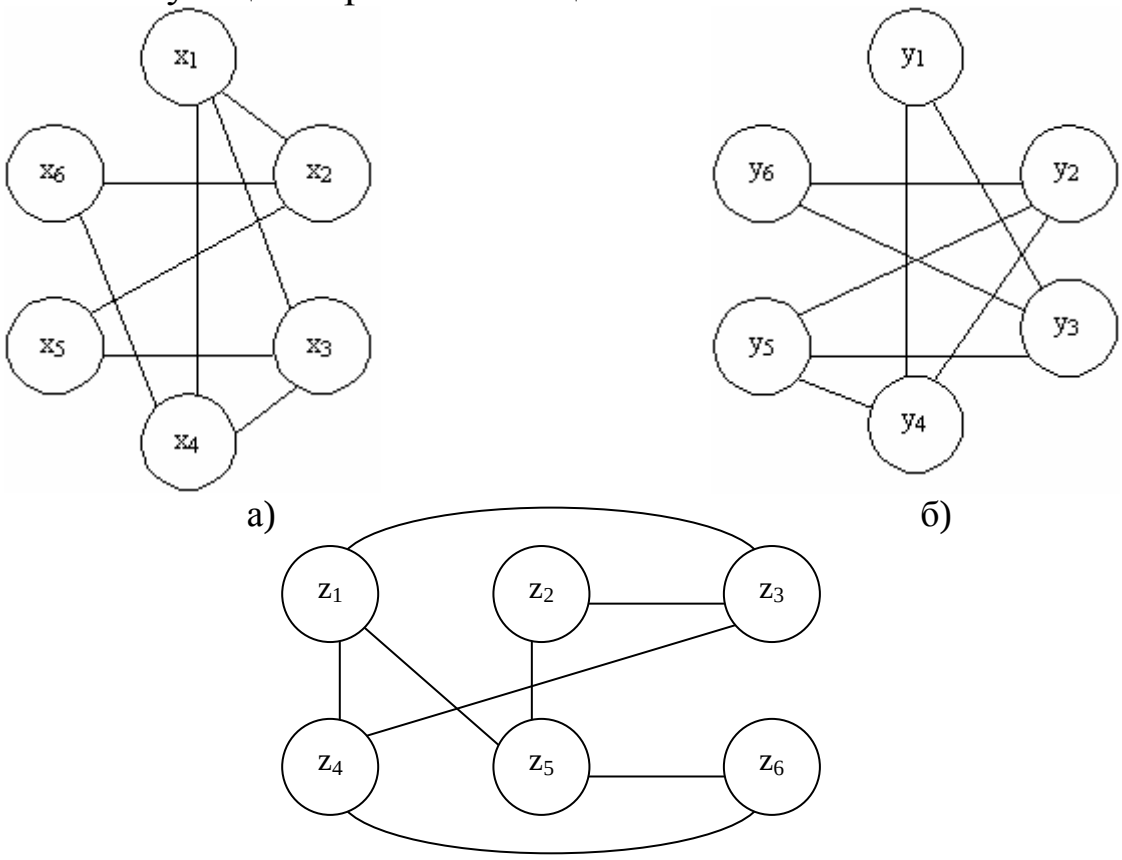
Теорема 4. Граф $G_{n1,\ n2}$ является двудольным тогда и только тогда, когда он не имеет простых циклов нечетной длины.

Рассмотрим число внутренней устойчивости. Если любые вершины в графе G = (X, U) X’⊆X, не смежны, то это подмножество называется ***внутренне устойчивым.***

Тождественные преобразования графов, сводимые только к переобозначению вершин и ребер, приводят к получению изоморфных графов.

9.8 Изоморфизм графов

Два графа G=(X, U) и G’=(X’, U’) называют ***изоморфными***, если можно установить взаимно-однозначное соответствие X↔X’, U↔U’ такое, что если $(x_i, x_j) \in X \leftrightarrow (x_i, x_j) \in X'$, то ребро $u = (x_i, x_j) \in U \leftrightarrow u' = (x'_i, x'_j) \in U'$. Изоморфизм есть отношение эквивалентности на графах. Изоморфные графы могут быть получены один из другого при помощи перенумерации их вершин. Если изоморфные преобразования проводятся с графом, заданным матрицей смежности, то они сводятся к перестановке местами соответствующих строк и столбцов.



а) б)

в)

Рисунок 9.29 – Граф G и изоморфные ему

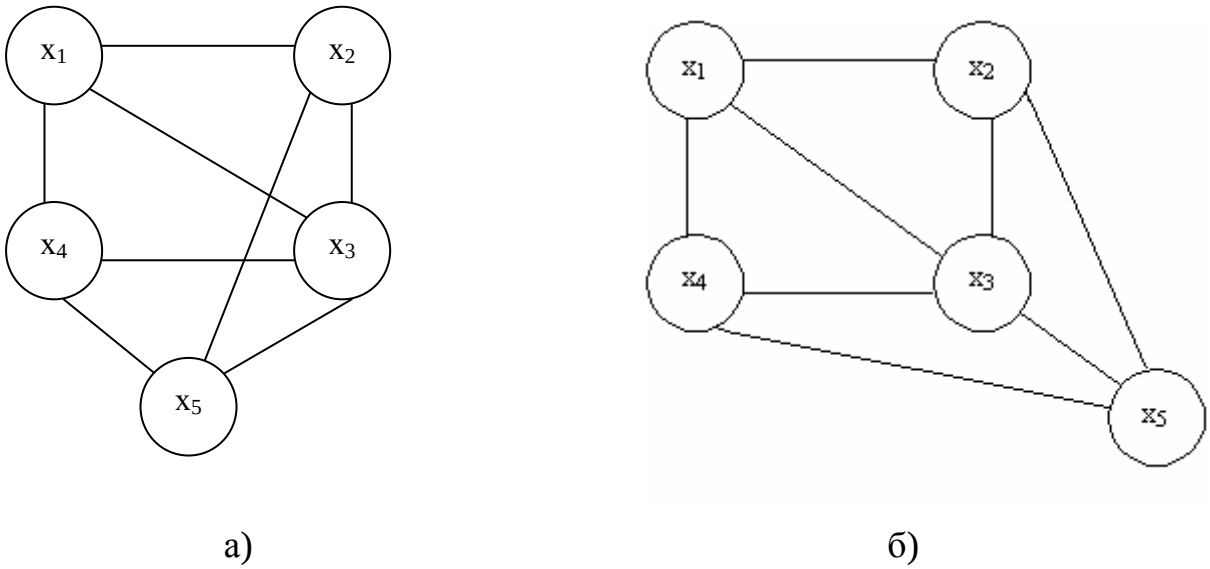
В общем случае для определения изоморфизма необходимо сделать n! сравнений.

При покрытии функциональной схемы набором стандартных модулей или при решении задачи типизации необходимо устанавливать изоморфизм между графом G и какой-либо частью другого графа G’.

При конструировании схем к их топологическому чертежу предъявляются требования получения плоского изображения схем.

Граф G=(X, U) называется ***плоским***, если он расположен на плоскости таким образом, что ребра имеют общие точки лишь в вершинах. Граф, изоморфный плоскому, расположенный на плоскости и имеющий пересечения ребер, называется ***планарным***.

Область плоскости, ограниченная ребрами плоского графа, внутри которой нет ни вершин, ни ребер, называется ***гранью***.



а) б)

Рисунок 9.30 – Планарный граф (а) и изоморфный ему плоский граф (б)

Ребра грани образуют простой цикл. Плоский граф имеет всегда одну бесконечную грань, не ограниченную ребрами. Существует формула Эйлера, позволяющая установить связь между числом вершин и числом ребер плоского графа:

n – m + f = 2, где f – число граней плоского графа.

Определить планарность можно с помощью различных критериев.

Пусть задан граф G=(X, U). Подразбиением ребра $u_k = (x_i, x_j)$ называют замену его двумя ребрами $u_{p1} = (x_i, x_p)$ и $u_{p2} = (x_p, x_j)$ с введением новой вершины x_p. Два графа называют гомеоморфными, если они обладают изоморфными подразбиениями.

Теорема (Понтрягина-Куратовского). Граф планарен тогда и только тогда, когда он не содержит подграфов, гомеоморфных полному графу K_5 и полному двудольному графу $K_{3,3}$.

Граф планарен тогда и только тогда, когда планарны все его связные компоненты.

Распространенная методика определения планарности заключается в нахождении в графе G максимального цикла C, лучше Гамильтонова, и размещении его на плоскости в виде замкнутой самопересекающейся кривой. Далее в оставшейся части определяют пересекающиеся по ребрам пути и предпринимают попытки разместить каждый из путей либо внутри C, либо полностью вне C. Если таким образом размещается весь граф, следовательно, он планарен, в противном случае – не планарен.

Заметим, что если граф связный и плоский, то и двойственный ему граф G_s также будет плоским и связным.

9.9 Орграфы

Орграф $G=(X, \vec{U})$ будем обозначать D=(X, U) и называть графом.

Матрицей смежности графа D называется матрица R(D) = $=\|r_{ij}\|_{nxn}$, причем

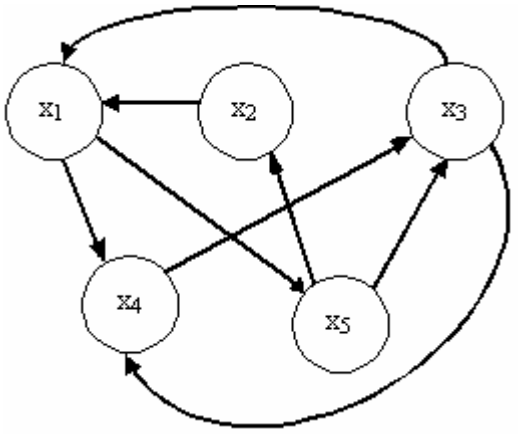
$$r_{ij}\begin{cases} 1, \text{ если } <xi, xj> \text{ – дуга D;} \\ = \\ 0, \text{ в противном случае.} \end{cases}$$

Так как $r_{ij} \neq r_{ji}$, то матрица не симметрична относительно главной диагонали.

Дуга $u_i = <x_i, x_j>$. Считается положительно инцидентной ее конечной вершине x_j. Число дуг, положительно инцидентных вершине x_j, называется ***полустепенью захода*** и обозначается $\varsigma^+(x_j)$. Число дуг, отрицательно инцидентных x_j, т.е. выходящих из x_j, называется полустепенью исхода и обозначается через $\varsigma^-(x_j)$.

$$\sum_{x_j \in X} \rho^{+}(x_j) = \sum_{x_j \in X} \rho^{-}(x_{j)} = |U|.$$

Из матрицы R(D) (рис. 9.31) видно, что суммы элементов по строкам равны полустепеням захода вершин D, а сумма элементов по столбцам – полустепеням исхода.



R(D) =

	1	2	3	4	5
1	0	0	0	1	0
2	1	0	0	0	0
3	1	0	0	1	1
4	0	0	1	0	0
5	0	1	1	0	0

Рисунок 9.31 – Орграф и матрица смежности

Элементы матрицы инциденций принимают значения 0, +1, –1. Элемент равен нулю, если вершина не инцидентна дуге, +1, если дуга ориентирована от вершины, +1, если дуга ориентирована к вершине.

Для графа D на рис. 31 матрица инциденций имеет вид:

	U_1	U_2	U_3	U_4	U_5	U_6	U_7	U_8
x_1	–1	–1	1	1	0	0	0	0
x_2	0	1	0	0	0	0	0	–1
x_3	1	0	0	0	–1	–1	1	0
x_4	0	0	0	–1	1	0	–1	0
x_5	0	0	–1	0	0	1	0	1

Маршрутом графа D считается чередующаяся последовательность вершин и дуг (x_0, u_1, x_1, u_2,…,u_n, x_n) в котором каждая дуга u_i есть кортеж u = <x_i, x_j>. Маршрут, в котором все вершины различны, называется ***путем***. Замкнутый маршрут, у которого все вершины различны, за исключением первой и последней, называется ***контуром***.

Если существует путь из вершины x_i в вершину x_j, то говорят, что x_j достижима из x_i. Граф D называют ***сильносвязным***, если любые две его вершины взаимнодостижимы.

Граф G, полученный из графа D заменой каждой дуги $u_i = <x_i, x_j>$ на соответствующее ребро $u_i = (x_k, x_l)$, т.е. устранением стрелок, называется ***основанием D***.

Два орграфа называются ***изоморфными***, если можно установить изоморфизм между их основаниями при сохранении порядка стрелок на каждой дуге.

Неорграф G называется ***ориентируемым***, если каждое его ребро можно ориентировать так, что полученный граф будет сильно связным. Такой процесс называется заданием ориентации графа G. Очевидно, что произвольный эйлеров граф может быть ориентируемым, так как достаточно пройти по любой эйлеровой цепи, ориентируя ребра в направлении движения.

Граф D называется эйлеровым, если в нем существует замкнутая цепь, содержащая каждую его дугу. Необходимым условием существования эйлерова орграфа является его сильная связность. Связный граф D = (X, U) является эйлеровым, когда $\forall x_i \in X(\varsigma^+(x_i) = \varsigma^-(x_i))$.

Орграф называется гамильтоновым, если в нем существует контур, содержащий каждую вершину орграфа.

Теорема. Пусть D – сильно связный граф. Если для $\forall x_i \in X(\varsigma^+(x_i) \geq n/2 \; n \; \varsigma^-(x_i) \geq n/2)$, то D – гамильтонов граф.

Метод Мальгранжа разбиения графа D на максимально связные подграфы.

Определим прямое и обратное транзитивные замыкания. Прямым транзитивным замыканием Γ^+x_i называют подмножество вершин $X' \subseteq X$, в которые можно попасть из вершины x_i по некоторому пути. Здесь $\Gamma^{+2}x_i = \Gamma^+\{\Gamma^+x_i\}$, $\Gamma^{+3}x_i = \Gamma^+\{\Gamma^+\{\Gamma^+x_i\}\}$,… Обратным транзитивным замыканием называют подмножество вершин, из которых можно попасть в x_i по некоторому пути. Обозначается Γ^-x_i.

Определим обратное и прямое транзитивное замыкание для вершины x_7.

$\Gamma^+x_7 = \{x_7,x_4,x_6\}$, $\Gamma^{+2}x_7 = \Gamma^+\{\Gamma^+x_7\} = \Gamma^+\{x_7,x_4,x_6\} =$

$=\{x_7,x_4,x_6,x_2,x_5\}$, $\Gamma^{+3}x_7 = \Gamma^+\{\Gamma^{+2}x_7\} = \{x_7,x_6,x_4,x_5,x_1,x_2,x_3\}$;

$\Gamma^{-}x_7 = \{x_7, x_2\}$; $\Gamma^{-2}x_7 = \{x_7, x_2, x_4\}$.

$\Gamma^{+}x_7 = \{x_7\} \cup \Gamma^{+}x_7 \cup \Gamma^{+2}x_7 \cup \Gamma^{+3}x_7 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$.

$\Gamma^{-}x_7 = \{x_7\} \cup \Gamma^{-}x_7 \cup \Gamma^{-2}x_7 = \{x_2, x_4, x_7\}$.

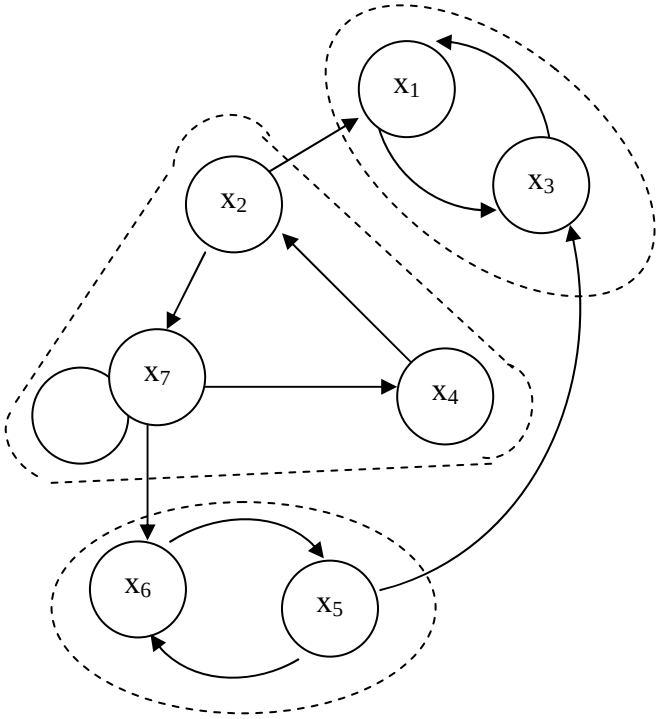


Рисунок 9.32 – Разложение графа на максимально связные графы

Основная идея алгоритма заключается в следующем. Выбирается произвольная вершина $x_i \in X$ графа D и для нее определяется $\Gamma^{+}x_i$, $\Gamma^{-}x_i$ и $C(x_i) = \Gamma^{+}x_i \cap \Gamma^{-}$; далее выбирается вершина $x_j \notin C(x_i)$, и процесс продолжается аналогично, пока возможно.

В результате работы алгоритма получим разбиение графа (рис. 32) на три части (указаны пунктиром).

9.10 Сети Петри

Сеть Петри в графическом представлении является двудольным ориентированным мультиграфом.

Математически сеть Петри основывается на понятии комплекта.

Комплект, как и множество, это набор элементов из некоторой области X и допускающий присутствие одного и того же

элемента несколько раз. Например, к числу комплектов можно отнести {a, b, c}; {a, b, c, c}; {a, a, a}, X= {a, b, c, d}.

Вместо отношения включения, являющегося основным понятием теории множеств, в теории комплектов вводится функция числа повторений каждого элементов и обозначается (x, B) (читается «число x в B»). Если ввести ограничения $0 \le \#(x, B) \le 1$, для $\forall x \in B$, то получим обычное понятие множества элементов.

Приведем теоретико-множественные операции, определенные на комплектах.

1. Операция включения $\#(x, B) > 0$.
2. Операция не включено $\#(x, B) = 0$.
3. Пустое множество $\#(x, B) = 0$, $\forall x \in X$.
4. $A \subseteq B$; $\#(x, A) \le \#(x, B)$, $\forall x \in X$.
5. $A=B$; $\#(x, A) = \#(x, B)$, $\forall x \in X$.
6. $A \cup B$; $\#(x, A \cup B) = \max\{\#(x, A), \#(x, B)\}$.
7. $A \cap B$; $\#(x, A \cap B) = \min\{\#(x, A), \#(x, B)\}$.
8. $A+B$; $\#(x, A+B) = \#(x, A) + \#(x, B)$.
9. $A-B$; $\#(x, A-B) = \#(x, A) - \#(x, A \cap B)$.

Под пространством комплектов X^{-n} понимают множество всех таких комплектов, элементы которых принадлежат X и ни один из них не входит в комплект более n раз:

$$\{B \in X^{-n}\} \Leftrightarrow \{x \in B \Rightarrow x \notin X;\ \#(x,B) \le n,\ \forall x \in X\}.$$

Мощностью комплекта B называется общее число повторений элементов в комплекте.

$$|B| = \sum_x \#(x, B), \quad x \in X.$$

Сетью Петри (C) называется дискретная детерминированная модель, состоящая из четырех элементов

$$C = (P, T, I, 0),$$

где $P = \{p_1, p_2, \ldots, p_n\}$ – конечное множество позиций $n \ge 0$.

$T = \{t_1, t_2, \ldots, t_m\}$ – конечное множество переходов $m \ge 0$, $P \cap T = \varnothing$.

$I : T \to P^{\infty}$ – входная функция, отображающая множество переходов в комплекты событий.

$0 : T \to P^{\infty}$ – выходная функция из T в P^{∞}. Позиция p_i является входной позицией перехода t_j в том случае, если $p_i \in I(t_j)$, и выходной, если $p_i \in 0(t_j)$.

Расширенными входными и выходными функциями сетей Петри соответственно называются отображения:

$$I : P \rightarrow T^{\infty}; \quad 0 : P \rightarrow T^{\infty},$$

для которых $\#(t_j, I(p_i)) = \#(p_i, 0(t_j))$; $\#(t_j, 0(p_i)) = \#(p_i, I(t_j))$.

Пример. Пусть структура сети Петри имеет вид:

$C = (P, T, I, 0)$; $P = \{p_1, p_2, p_3, p_4, p_5\}$; $T = \{t_1, t_2, t_3, t_4\}$;

$I(t_1) = \{p_1\}$ $\quad 0(t_1) = \{p_2, p_3, p_5\}$

$I(t_2) = \{p_2, p_3, p_5\}$ $\quad 0(t_2) = \{p_5\}$

$I(t_3) = \{p_3\}$ $\quad 0(t_3) = \{p_4\}$

$I(t_4) = \{p_4\}$ $\quad 0(t4) = \{p_2, p_3\}$

Расширенными входной и выходной функциями данной сети являются:

$I(p_1) = \{\varnothing\}$ $\quad 0(p_1) = \{t_1\}$

$I(p_2) = \{t_1, t_4\}$ $\quad 0(p_2) = \{t_2\}$

$I(p_3) = \{t_1, t_4\}$ $\quad 0(p_3) = \{t_2, t_3\}$

$I(p_4) = \{t_3\}$ $\quad 0(p_4) = \{t_4\}$

$I(p_5) = \{t_1, t_2\}$ $\quad 0(p_5) = \{t_2\}$

Проиллюстрируем пример графически.

Элементами графа служат кружки, обозначающие позиции сети, и планки, обозначающие ее переходы. Ориентированные дуги соединяют позиции и переходы в обоих направлениях.

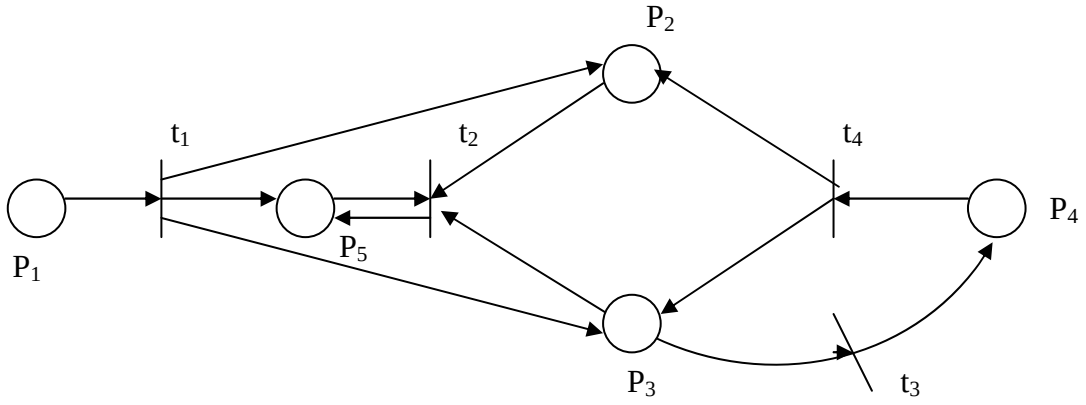


Рисунок 9.33

Граф G ***сети Петри*** – двудольный ориентированный мультиграф $G = (V, A)$, где $V = \{v_1, v_2, \ldots, v_s\}$ – множество вершин $V = P \cup T$;

$A = \{a_1, a_2, \ldots, a_r\}$ – комплект направленных дуг.

$a_i = <v_j, v_k>$, $v_j, v_k \in V$.

Комплект направляющих дуг A вводится следующим образом:

$\#((p_i t), A) = \#(p_i, I(t_j)), \forall p_i \in P, t_j \in T;$

$\#((t_j, p_i), A = \#(p_i, 0(t_j))$.

Сеть можно выполнить. Для выполнения сети применяют маркировку, представляющую присвоение позициям сети фишек, количество и положение которых при выполнении сети могут изменяться.

Маркировкой μ сети Петри C = (P, T, I, 0) называется функция, отображающая множество позиций p в множество натуральных чисел N:

$\mu : P \rightarrow N$

Под маркировкой можно подразумевать n-вектор

$$\mu = (\mu_1, \mu_2, \ldots, \mu_n), \text{ где } n = |P|, \mu_i \in \overline{N, i} = 1,n,$$

который определяет для каждой позиции сети P_i количество фишек (μ_i) в этой позиции. На графе сети Петри фишки изображаются точками в кружке соответствующей позиции, если число точек невелико, и в противном случае указывается натуральное число μ_i для данной позиции. Приведем пример маркировки для сети, заданной рисунком 9.34.

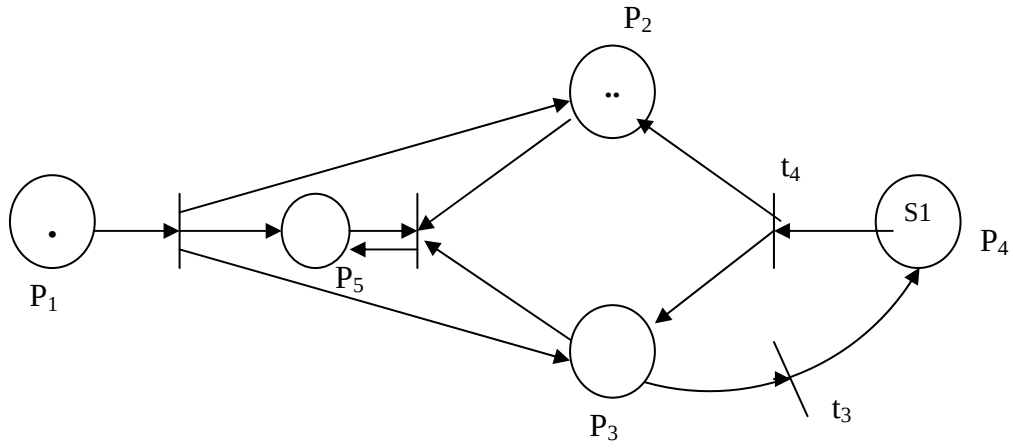


Рисунок 9.34

Фишки во входной позиции, допускающие переход, носят название разрешающих фишек.

Переход $t_i \in T$ в маркированной сети Петри C = (P, T, I, 0) с маркировкой μ ***разрешен***, если

$$\mu_i = \mu(p_i) \geq \#(p_i, I(t_i)), \forall\ p_i \in P.$$

Пример маркировки μ = (1, 2, 0, 51, 0) (рис. 9.34). Переход допускается удалением всех разрешающих фишек из его входных позиций и последующим помещением в каждую из его выходных позиций по одной фишке для каждой дуги.

Запуск перехода в целом заменяет маркировку μ сети Петри на новую маркировку μ'.

Переход t_i в маркированной сети Петри с маркировкой μ может быть запущен, если он разрешен.

Маркировка μ' для разрешенного перехода t_i образуется по правилу

$$\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_i)) + \#(p_i, 0(t_i)).$$

Рассмотрим следующий пример. Задана сеть Петри (рис. 5.3). Маркировка сети μ = (1, 0, 0, 2, 1). При такой маркировке разрешены три перехода t_1, t_3, t_4. Переход t_2 не разрешен, поскольку две из трех входных позиций не содержит ни одной

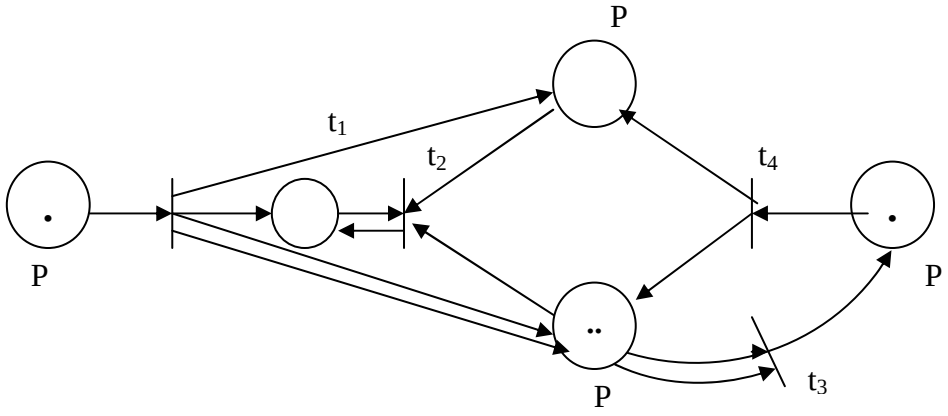


Рисунок 9.35

фишки. Любой из разрешенных переходов может быть запущен. Запустим переход t_4. Фишка удалится из p_5, одна фишка переместится в позицию p_3 и одна – в p_4. В результате сеть Петри будет выглядеть:

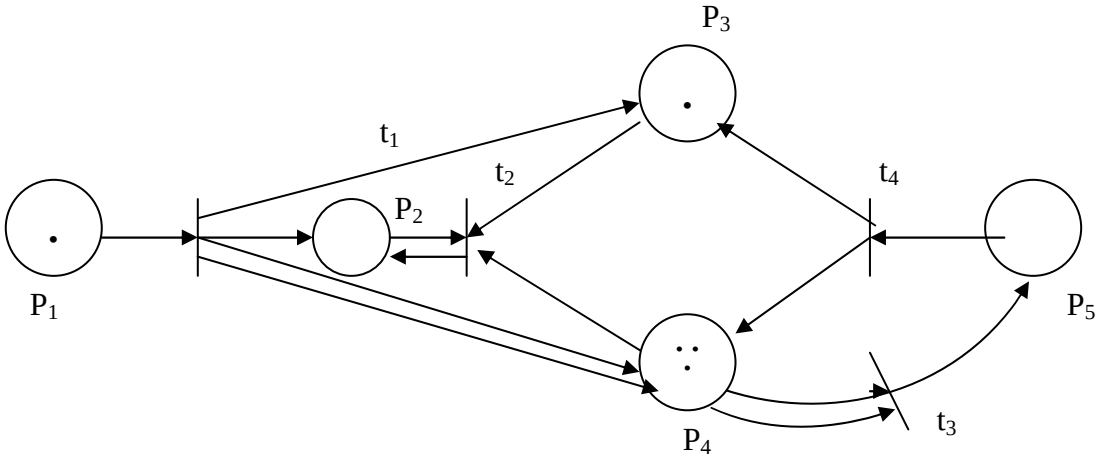


Рисунок 9.36

Пространство состояний сети Петри, обладающей конечным числом позиций (n), есть множество всех маркировок. Изменение в состоянии, вызванное запуском очередного перехода, осуществляется с помощью функции следующего состояния.

Функция следующего состояния $\delta : N^n \times T \rightarrow N^n$ для сети Петри C с маркировкой μ и переходом $t_j \in T$ определена тогда и только тогда, когда

$\mu (p_i) \geq \#(p_i, I(t_j)), \forall p_i \in P.$

В случае определения функции δ осуществляется отображение

$\delta(\mu, t_j) = \mu'$,

где

$\mu (p_i) = \mu (p_i) - (p_i, I(t_j)) + (p_i, 0(t_j)), \forall p_i \in P.$

При выполнении сети Петри получаются две последовательности: последовательность маркировок $\{\mu^0, \mu^1, \ldots\}$ и последовательность переходов, которые были запущены $\{t_{jo}, t_{j1}, \ldots\}$, связанные отношением:

$\delta(\mu^k, t_{jk}) = \mu^{k+1}$, $k = 0, 1, 2, \ldots$

Для сети Петри C = (P, T, I, 0) с маркировкой μ маркировка μ' называется непосредственно ***достижимой*** из μ, если $\exists$ переход $t_j \in T$ такой, что $\delta(\mu, t_j) = \mu'$.

Множеством достижимости R(C, μ) для сети Петри с маркировкой μ называется наименьшая последовательность маркировок, таких что:

1) $\mu \in R(C, \mu)$;

2) если $\mu' \in R(C, \mu)$ и $\mu'' = \delta(\mu, t_j)$, для $t_j \in T$, $\mu'' \in R(C, \mu)$.

Приведем в качестве примера следующую сеть Петри.

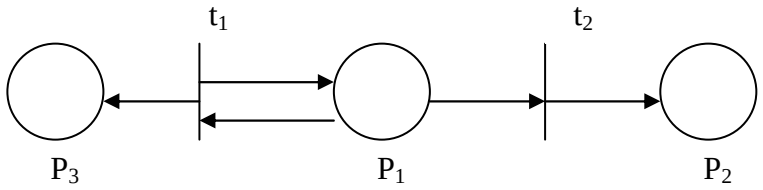


Рисунок 9.37

Для этой сети с начальной маркировкой $\mu = (1, 0, 0)$.

Непосредственно достижимым являются две маркировки:

$\mu_1 = (0, 1, 0)$ и $\mu^2 = (1, 0, 1)$;

из μ^1 нельзя достичь ни одной маркировки. Из μ^2 можно получить (0, 1, 1) и (1, 0, 2). Продолжая процесс, можно получить (1, 0, n) и (0, 1, n), $n \leq 0$.

КОНТРОЛЬНАЯ РАБОТА №1 (ВАРИАНТЫ ЗАДАНИЙ)

В контрольную работу №1 входят соответствующие номера заданий из разделов: «Множества», «Графы».

Множества

Вариант №1

1. Укажите все элементы множества X={x ∈ A | x <10 и A – множество простых чисел}.

2. Сколько элементов в следующих множествах: {d, f, u, df, ff}, {1, 3, 4, 11,34}, {e, 4, ju, 7, 6}, {1, 11, 111,1}, {a, d, b, a, f, r,}?

3. Дано множество A={a,b,c,f,h}. Укажите верные записи: 1) a∈ A, 2) c ⊂ A, 3) ∅∈ A, 4){a,b,h}∈ A, 5){f,h}⊆A.

4. Дайте полное определение операции объединения.

5. Укажите верные утверждения A⊕B⊕C=(A⊕B)⊕C, A⊕B⊕T=A⊕B, A⊕T⊕T=A⊕T.

Вариант №2

1. Сколько элементов в следующих множествах: {x| x ≥ 1 и x ≤ 4}, {rt, y, rt, tr,tt }, {2,4,6, 1+1,1+3}, { ←, ↑, ↓, →, →→, ←, ↓↑}?

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, E= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Проиллюстрировать графически: ($\bar{A}$ ∪$\bar{E}$)∩C.

3. Дано множество A={a,b,c,f,h}. Укажите верные записи: 1) ∅⊂ A, 2) {a,b,c,f,h}⊆A, 3) {a}⊂{a,b}, 4) A⊆ {a,b,c,f,h}.

4.Дайте определение множества.

5. Упростить A∩¬B∪A∩¬B∩C∪A, A∩B∩¬C∪A∩B∩C∪A∩C.

Вариант №3

1. Укажите все элементы множества X={x | x – название месяца, начинающееся с буквы м}.

2. Сколько элементов в следующих множествах: {a, b, c, ac}, {1,r,6,6r,11}, {ℵ, ℜ, ℘, ℑ, ℜ, ℜℵ}, {A, B, P, O, E, HE}?

3. Дано множество A={a,b,c,f,h}. Укажите верные записи: 1) {c}∈{a,c,f}, 2) ∅⊂ A, 3) {a,c}⊂ {b,c,f}, 4) ∅∈ {a,c,h}.

4. Дайте определение операции объединения.

5. Чему равны выражения, если A=B=C=Д=T

A∩B∩Ē∪Ē∩B; A∩¬B∪C∪E.

Вариант №4

1. Укажите все элементы множества X={x | x=2n, n – натуральное число и n<5}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и два подмножества R={2} и Q = {2, 3, 8, 6}. Изобразите эти множества при помощи диаграмм Эйлера-Венна.

3. Известно, что A⊂B и a ∈ A. Какие из записей верны: 1) a⊂A, 2){a}⊂B, 3)a∉ B, 4){a}⊂ A.

4. Дайте полное определение операции дополнения.

5. Упростить, если A⊂B, B=C ¬(A∪B∩¬C)

¬(A∪¬(¬B∩C)).

Вариант №5

1. Укажите все элементы множества X={x |x=2n, n – неотрицательное целое число и n<5}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и два подмножества R={2} и Q = {2, 3, 8, 6}. Укажите элементы, не входящие в множество Q.

3. Даны три множества A,B,C. И a∈ A. Укажите верные утверждения. 1) a ⊂ B, 2) a∈ A∪B, 3) a∈ A∩C, 4) {a}∈ A∪B∪C.

4. Введите понятие универсального множества.

5. Упростить, если C=T и D = 0, (A∪B)∩(C∪D); ¬A ∩¬B ∩C∪B∩C∩D.

Вариант №6

1. Укажите все элементы множества X={x |x=2n+2, n – натуральное число и n<5}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и два подмножества R={2, 3} и Q = {2, 7, 4, 8, 6}. Изобразите эти множества при помощи диаграмм Эйлера-Венна.

3. Даны три множества A,B,C. И $a \in A$. Укажите верные утверждения:

1) $\{a\} \subseteq A$,

2) $\{a\} \subset A \cup B$. 3) $\{a\} \subset A \cap B$. 4) $a \notin B$.

4. Чему равно число всевозможных подмножеств любого конечного множества, содержащего N элементов?

5. Упростить, если C =T; D=0, $(\neg A \cup B \cup) \cap (C \cup D)$; $A \cap C \cup \neg B \cap C \cap A \cap D$.

Вариант №7

1. Укажите все элементы множества $X=\{x \mid x=2(n+1)$, n – натуральное число и $n \leq 3\}$.

2. Дано универсальное множество T = {0, 2, 3, 4, 5, 6, 7, 8, 9} и два подмножества R={2, 3} и Q = {2, 7, 4, 8, 6}. Укажите элементы, не входящие в множество Q.

3. Чему равно:1) $A \cap A=$ 2) $A \cap T=$ 3) $A \cap \varnothing=$

4. Определите отношение включения.

5. Упростить выражение $\neg A \cap (B \cup C \cup D) \cap B \cap C$, $(A \cup B \cup C) \cap (\neg B \cup D)$, если C = T, D=0

Вариант №8

1. Укажите все элементы множества $X=\{x \mid x$ – целое число и $|x| \leq 2\}$.

2. Универсальное множество T представляет собой все гласные буквы русского алфавита. Множество N = {а, е, ы, и, э}. M = ={у, е, а, ы,}. Перечислите все элементы, которые останутся в множестве T, если из него удалить все элементы, не входящие в множества M и N.

3. Чему равно:1) $A \cup A=$, 2) $A \cup T=$, 3) $A \cup \varnothing=$

4. Дайте определение конечного множества.

5. Упростить $A \cap B \cap C \cap D$, $\neg A \cap \neg B \cup \neg C \cap \neg D$, если $C \subset D$, $A \subset B$

Вариант №9

1. Укажите все элементы множества $X=\{x \mid x=3n$, n – целое число и $|n|<3\}$.

2. Универсальное множество T представляет собой все гласные буквы русского алфавита. Множество N = {а, е, ы, и, э}. M = ={у, е, а, ы,}. Укажите буквы, не входящие ни в множество M, ни в множество N.

3. Чему равно 1) $A \cup \bar{A} =$ 2) $A \cap \bar{A} =$

4. Дайте определение подмножества.

5. Упростить $(A \cup B) \cap (C \cup D)$, $A \cap \neg B \cup C \cap \neg D$, если $C \subset D$, $A \subset B$

Вариант №10

1. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} и множества A, B, C.

A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Перечислите элементы множества $M = \{x | x \notin A \text{ и } x \in T\}$.

2. Приведите законы де Моргана.

3. Укажите верные выражения: $(A \cup B) \cap (A \cup C) = A \cup (B \cap C)$, $(B \cup C) \cap A = A \cap B \cup A \cap C$, $(A \cap B) = (B \cap A)$.

4.Дайте определение операции объединения.

5. Упростить $\neg(A \cup B) \cap \neg(C \cup D)$, $(A \cup B \cup C) \cap (B \cup C \cup D)$, если $C \subset D$, $A \subset B$.

Вариант №11

1. Укажите все элементы множества X = {x |x – отличник группы 577–1}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C.

A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Перечислите элементы множества $M = \{x \mid x \in A \cup B, x > 4\}$.

3. Приведите законы ассоциативности.

4. Укажите верные выражения: $(A \cup B) \cap (A \cup C) = A \cap (B \cup C)$, $A \cap B \cup A \cap C = A \cup B \cap C$, $A \cap (B \cup C) = A \cup B \cap C$.

5. Упростить $(\neg A \cup B) \cap (\neg C \cup D)$, $A \cap B \cap \bar{E} \cup \neg A \cap E$, если C= =D =B=T .

Вариант №12

1. Укажите все элементы множества X ={x | x – виды хвойных деревьев, растущих в Томском районе}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} и множества A, B, C.

A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Перечислите элементы множества M = {x| x ∈ A∪B∪C и x – четное число}.

3. Приведите закон инволюции.

4. Чему равны выражения 1) A∪∅= ; 2) ∅∪∅∩A= ; 3) A∩B∩∅= ; 4) T∪∅∩A= .

5. Упростить (¬A∪¬B)∩(B∪E), (¬A∪E)∩(¬B∪Ē), если C=A=B=T.

Вариант №13

1. Укажите все элементы множества X = {x| x – базовые типы языка программирования ПАСКАЛЬ}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C.

A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Перечислите элементы множества T = {x| x∉ A∪B}.

3. Приведите законы идемпотентности.

4. Укажите верные утверждения: A⊕B⊕C=(A⊕B)⊕C, A⊕B⊕T =A⊕B, A⊕T⊕T=A⊕T.

5. Упростить A∩B∩Ē∪Ē∩B, A∩¬B∪C∪E, если C= D= =A=B=T.

Вариант №14

1. Укажите все элементы множества X = { x | x – операторы языка ПАСКАЛЬ}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Перечислите элементы множества H = { x| x∉ A∩B}.

3. Приведите коммутативные законы.

4. Укажите верные утверждения: A⊕B∩C=A⊕B∩C ⊕∅, A⊕∅⊕∅= A⊕∅, A⊕Ā=A∪Ā.

5. Упростить A∪B∪C∪D, ¬A∩¬B∩¬C∩¬C, если C=B=0

Вариант №15

1. Укажите все элементы множества X = { x | x – множество арифметических операций, реализованных в языке Паскаль}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Проиллюстрировать графически: (A∩B)∪C.

3. Приведите дистрибутивные законы.

4. Укажите верные утверждения: 1) A∩(B⊕C)=A∩B⊕A∩C; 2) A⊕B⊕A∩B=A∪B; 3) A⊕B̄⊕A∩B̄=A∪B̄.

5. Упростить ¬A∪B∪D, B∩C∪A∩¬D, если C=B=0

Вариант №16

1. Укажите все элементы множества X = { x | x = n^2, n – целое число и 6≤ n≤ 10}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Проиллюстрировать графически: A \ (B ∪ C).

3. Дайте определение множества.

4. Даны множества: A={0,1,2,5}; B={1,2}; E={2,5,7}; T={0,1,2,3,4,5,6,7}. Найдите элементы множеств: (A∪B∪E)\B; (A∪B)\(AB); (A∪B∪Ē)\(B∪E).

5. Даны множества A={1,2,3}; B={1,2}; C={3,4,5}. Найдите элементы множеств ¬A∩¬B∩¬C, A∩B∩¬C, ¬A∩¬B∩C.

Вариант №17

1. Укажите все элементы множества X = { x | x > 4 и x ∈ {3, 4, 5, 9, 5, 11}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Проиллюстрировать графически: A∩(B⊕C).

3. Перечислите способы задания множества.

4. Даны множества: A={0,1,2,5}; B={1,2}; C={2,5,7}; T={0,1,2,3,4,5,6,7}. Найдите элементы множеств: A⊕C⊕A∩B; A∩B∪A∩C; A∩(B\C).

5. Даны множества A={1,2,3}; B={1,2}; C={3,4,5}. Найдите элементы множеств (A∪B)∩¬C, ¬(¬A∪¬B∪¬C∩A).

Вариант №18

1. Укажите все элементы множества X = { x | x = четное неотрицательное число и x < 1}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Проиллюстрировать графически: A⊕B∩C.

3. Определите отношение равенства.

4. Укажите верные выражения: (A∪B)∩(A∪C)=A∪(B∩C), (B∪C)∩A=A∩B∪A∩C, (A∩B)=(B∩A).

5. Даны множества A={1,2,3}; B={1,2}; C={3,4,5}. Найдите элементы множеств (A∪B)∩(¬A∪¬B), (A∪B)∩A∩C.

Вариант №19

1. Укажите все элементы множества, составленного из десятичного числа 27809673219.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Проиллюстрировать графически: A∪(B\C).

3. Дайте определение операции объединения.

4. Укажите верные выражения: (A∪C)∩(B∪C)=A∪(B∩C), (B∪C)∩A=A∩B∪A∩C, (A∩B)=(C∩A) ∪(B∩C).

5. Расставьте знаки = или ≠

(A∪B)∩(¬A∪B)∩(A∪¬B)……A∪B

(A∩B∪¬A∩B)∩¬B……0

Вариант №20

1. Укажите все элементы множества X={x |x=2(n–1), n – натуральное число и $n \leq 3$}.

2. Дано универсальное множество T = {0, 2, 3, 4, 5, 6, 7, 8, 9}и два подмножества R={2, 3} и Q = {2, 7, 4, 8, 6}. Укажите элементы, не входящие в множество Q∩R.

3. Чему равно:1) A∩A= 2) A∩T= 3) A∩∅=

4. Определите отношение включения.

5. Расставьте знаки = или ≠

(A∩B∪C)∩(¬(A∩B)∪C)……..C

(¬A∪B)∩(A∪¬B) ¬((A∪B)∩(¬A∪¬B))

Вариант №22

1. Укажите все элементы множества X = { x | x = четное неотрицательное число и x < 1}.

2. Дано универсальное множество T = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}и множества A, B, C. A={1, 2, 3, 4, 7}, B= {3, 5, 4, 6}, C= {7, 4, 6, 8}. Проиллюстрировать графически: A∪(B\C).

3. Приведите дистрибутивные законы.

4. Дано множество A={a,b,c,f,h}. Укажите верные записи: 1) a∈ A, 2) c ⊂ A, 3) ∅∈ A, 4){a,b,h}∈ A, 5){f,h}⊆A.

5. Упростить A∩B∩C∪A∩¬B∩C∪B∩C∪¬B∩C =

Графы

Вариант №1

1 Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(1,4), (1,8), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (3,4), (6,8), (5,6), (4,8), (1,9), (9,3), (2,7), (7,6), (4,3), (2,5), (7,7), (3,7)}.

Нарисуйте его, дайте полную характеристику: связность, циклы, цепи, маршруты, ориентированность, постройте матрицу расстояний, задайте матрицей смежности.

2. Постройте минимальное покрывающее дерево для графа, заданного таблицей:

	X1	X2	X3	X4	X5	X6	X7
X1	0	0	2	2	1	0	4
X2	0	0	3	0	1	2	0
X3	2	3	0	5	4	0	5
X4	2	0	5	0	2	3	0
X5	1	1	4	2	0	0	0
X6	0	2	0	3	0	0	5
X7	4	0	5	0	0	5	0

Здесь нулем кодируется отсутствие смежности вершин, цифрой – вес соответствующих ребер.

3. Постройте схему алгоритма выделения из графа суграфа и подграфа с заданным числом ребер.

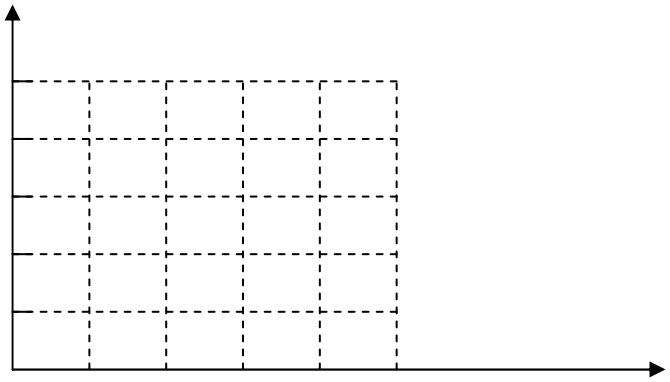
Вариант №2

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9,10}, U={(2,4), (1,8), (2,7), (3,6), (5,3), (1,3), (2,5), (4,6), (3,4), (6,8), (5,6), (1,9), (9,3), (5,7), (7,1), (3,7)}.

Нарисуйте его, задайте матрицей смежности, постройте подграф, сураф, плоский и планарный.

2. Рассчитайте длины всех проводников, заданных решетчатым графом G=(X,U), X={x3,x15,x12,x23) U={(x3,x15),(x15,x23), (x23,X3),(x23,x12).



3. Для полного графа с 4 вершинами постройте все покрывающие неизоморфные деревья.

Вариант №3

1. Задан граф G=(X,U), X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(2,4), (2,8), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (5,6), (4,8), (1,9), (9,3), (1,7), (7,4), (3,7)}.

Нарисуйте его, двойственный ему граф, дайте полную характеристику (связность, циклы, ориентированность, матрица расстояний и т.д.), задайте матрицей смежности.

2. Задан граф G=(X,U), X={x1,x2,x3,x4}, U={(x1,x2), (x1,x3), (x2,x3), (x2,x4), (x3,x4), (x1,x4),(x4,x4)}. Построить простую цепь из X2 в X3.

3. Постройте произвольный мультиграф G=(X,U), |X| =n, |U|=m. N=8,m=14, определите его мультичисло.

Вариант №4

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7}, U={(1,4), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (3,4), (5,6), (2,7), (3,7),(6,7),(1,5)}.

Нарисуйте его, двойственный ему граф, постройте граф, изоморфный исходному, матрицу расстояний, задайте списком смежности.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x2), (x1,x3), (x2,x3), (x2,x4), (x3,x4), (x3,x5),(x4,x4),(x4,x5)}. Построить его графическое представление и матрицы инциденций и смежности, все простые цепи и циклы.

3. Постройте граф G=(X,U), |X| =n, |U|=m. n=7, m=13. Задайте его с помощью матриц смежности и инциденций. Постройте схему алгоритма перехода от одной матрицы к другой.

Вариант №5

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(1,4), (1,8), (2,7), (3,6), (2,3), (1,3), (2,5), (6,8), (5,6), (1,9), (9,3), (2,7), (7,7), (3,7)}.

2. Нарисуйте его, двойственный ему граф, дайте полную характеристику (связность, циклы, ориентированность, матрица расстояний и т.д.), задайте матрицей смежности.

Задан граф G=(X,U), X={x1,x2,x3,x4,x5},U={(x1,x2), (x1,x3), (x2,x3), (x2,x4), (x3,x4), (x3,x5),(x4,x4),(x4,x5)}. Построить его графическое представление и матрицы инциденций и смежности, все простые цепи и циклы.

3. Подсчитайте число суграфов, включая изоморфные, в графе G=(X,U), |X| =n.

Вариант №6

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(1,4), (1,8), (2,7), (3,6), (2,3), (1,3), (2,5), (6,8), (5,6), (1,9), (9,3), (2,7), (7,7), (3,7)}.

2. Нарисуйте его, двойственный ему граф, найдите подграф, суграф, дополненние до полного графа, постройте матрицу расстояний, определите диаметр графа.

Задан граф G=(X,U), X={x1,x2,x3,x4,x6,x6}, U={(x4,x1), (x1,x2), (x2,x6), (x6,x5),(x2,x3),(x3,x5),(x5,x2)}. Построить раскраску графа и найти его хроматическое число.

3. Подсчитайте число суграфов, включая изоморфные, в графе G=(X,U), |X| =n.

Вариант №7

1.Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7}, U={(1,4), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (3,4), (5,6), (2,7), (3,7),(6,7),(1,5)}.

Нарисуйте его, двойственный ему граф, задайте матрицей инциденций, постройте подграф, суграф, дополнение до полного графа, определите диаметр графа и его хроматическое число.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5,x6}, U={(x1,x2), (x2,x4), (x4,x3), (x1,x1), (x3,x6), (x3,x2), (x1,x5), (x3,x1), (x6,x5), (x6,x1), (x3,x5), (x5,x4)}.Постройте граф. Найдите минимальное покрывающее его дерево. Вес вершин 2,6,4,2,5,4,7,9,2,4,6,4 соответственно.

3. Предложите методы построения графов с эйлеровыми и гамильтоновыми циклами на заданных наборах вершин и ребер.

Вариант №8

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(2,4), (2,8), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (5,6), (4,8), (1,9), (9,3), (1,7), (7,4), (3,7)}.

2. Нарисуйте его, двойственный ему граф, задайте матрицей инциденций, смежности, найдите эйлеров цикл, гамильтонов цикл, постройте плоский и планарный граф.

Задан граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x2), (x2,x4), (x4,x3), (x1,x1), (x3,x4), (x3,x2), (x1,x5), (x3,x1), (x5,x5), (x2,x1), (x3,x5), (x5,x4)}. Постройте его, найдите все простые цепи.

3. Предложите алгоритм построения двудольных графов с заданным числом вершин.

Вариант №9

1. Задан граф G=(X,U),

X={1,2,3,4,5}, U={(1,3),(2,4),(5,1)(3,5),(5,4)(1,4)}.

Нарисуйте его, задайте матрицу смежности и инциденций, нарисуйте двойственный ему граф и для него задайте матрицы

смежности и инциденций, найдите диаметр графа и его хроматическое число.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x2), (x2,x4), (x4,x3), (x1,x1), (x3,x4), (x3,x2), (x1,x5), (x3,x1), (x5,x5), (x2,x1), (x3,x5), (x5,x4)}. Построить граф. Найдите минимальное покрывающее его дерево. Вес вершин 2,1,4,2,7,4,7,3,2,4,6,4 соответственно.

3. Определите хроматическое число неполного связного графа G=(X,U), |X| =n, |U|=m. n=8, m=18.

Вариант №10

1. Задан граф G=(X,U),

X={1,2,3,4,5}, U={(1,3),(2,4),(2,1)(3,5),(5,4)(1,4),(4,4)}.

Нарисуйте его, задайте матрицу смежности и инциденций, нарисуйте двойственный ему граф и для него задайте матрицы смежности и инциденций, постройте плоский и планарный граф.

2. Дайте определение Эйлерова графа. Приведите пример. Будет ли граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x2), (x2,x3), (x2,x5), (x3,x4), (x1,x4), (x3,x5)} Эйлеровым? Нарисуйте его. Постройте минимальный цикл.

3. Постройте произвольный граф G=(X,U), |X| =n, |U|=m. n=10, m=18. Определите его цикломатическое число. Укажите, какие ребра должны быть удалены из графа, чтобы он стал ациклическим.

Вариант №11

1. Задан граф G=(X,U),

X={1,2,3,4,5,6}, U={(1,3),(2,4),(5,1)(3,5),(5,4)(1,4),(6,3)}.

Нарисуйте его, задайте матрицу смежности и инциденций, нарисуйте двойственный ему граф и для него задайте матрицы смежности и инциденций, нарисуйте изоморфный ему граф, определите его диаметр.

2. Задан граф. Найдите минимальное покрывающее его дерево. G=(X,U), X={x1,x2,x3,x4,x5,x6,x7},
U={(x1,x3)(x1,x2)(x1,x4)(x2,x3)(x2,x6)(x2,x5)(x3,x4)(x3,x7)(x4,x5)(x4,x6)(x1,x6)(x7,x6)}. «Вес» ребер равен: 4,3,1,6,1,4,5,2,7,8,1,9 соответственно.

3. Определите множество полных графов, содержащих одновременно эйлеровы и гамильтоновы циклы.

Вариант №12

1. Задан граф G=(X,U),

X={1,2,3,4,5}, U={(2,4),(5,1)(3,5),(5,5)(1,4)}.

Нарисуйте его, задайте матрицу смежности и инциденций, нарисуйте двойственный ему граф и для него задайте матрицы смежности и инциденций.

Для исходного графа нарисуйте планарный и плоский граф, определите его хроматическое число.

2. Задан граф G=(X,U), X={x1,x2,x3,x4}, U={(x1,x2), (x1,x3), (x2,x3), (x2,x4), (x3,x4), (x1,x4),(x4,x4)}. Построить все простые цепи из X1 в X3.

3. Доказать, что среди любых 6 человек есть либо 3 попарно знакомых, либо 3 попарно незнакомых.

Вариант №13

1. Задан граф G=(X,U),

X={1,2,3,4,5}, U={(1,3),(3,4),(5,2)(3,5),(5,4)(1,4)}.

Нарисуйте его, задайте матрицу смежности и инциденций, нарисуйте двойственный ему граф и для него задайте матрицы смежности и инциденций.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x2), (x1,x3), (x2,x3), (x2,x4), (x3,x4), (x3,x5),(x4,x4),(x4,x5)}. Построить его графическое представление, все простые цепи и циклы.

3. Описать в терминах теории графов отношение эквивалентности на конечном множестве.

Вариант №14

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(2,4), <2,8>, <2,7>, <3,6>, (2,3), (1,3), (2,5), (4,6), (5,6), <4,8>, <1,9>, <9,3>, (1,7), (2,5), (7,4), (3,7), <7,2>, <8,3>, <3,6>}.

Нарисуйте его, дайте полную характеристику.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x6,x5}, U={(x1,x2), (x2,x4), (x4,x3), (x1,x1), (x3,x5), (x3,x2), (x1,x5), (x3,x1), (x2,x5), (x6,x1), (x3,x5), (x5,x4)}. Дайте определения: маршрута, цепи,

цикла, связности. Постройте на заданном графе маршрут, цепь, цикл, покрывающее его дерево.

3. Доказать, что если граф связен и конечен, то поиск в ширину и поиск в глубину обойдут все его вершины по одному разу.

Вариант №15

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6}, U={(2,4), (2,5), (4,6), (5,6), (2,5), (6,1),(3,6).(2,1),(1,1)} Нарисуйте его, дайте полную характеристику, (связность, ориентированность, матрица расстояний, хроматическое число, планарность и т.д.).

2. Дайте определение Эйлерова графа. Приведите пример. Будет ли граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x2), (x1,x3), (x1,x2), (x2,x4), (x3,x4), (x3,x5) (x2,x5)} Эйлеровым? Нарисуйте его. Постройте минимальный цикл.

3. Доказать, что граф связен тогда и только тогда, когда его нельзя представить в виде объединения двух графов.

Вариант №16

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(2,4), (2,8), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (5,6), (4,8), (1,9), (9,3), (1,7), (2,5), (7,4), (3,7)}.

Нарисуйте его, дайте полную характеристику, (связность, циклы, ориентированность, матрица расстояний и т.д.) задайте матрицей смежности.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x2), (x1,x3),(x2,x3),(x2,x4),(x3,x4),(x1,x4),(x4,x4),(x3,x5),(x5,x4),(x1,x5)}. Построить все простые цепи из X2 в X5.

3. Нарисовать диаграммы всех деревьев с 7 вершинами.

Вариант №17

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(1,4), (1,8), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (3,4), (6,8), (5,6), (4,8), (1,9), (9,3), (2,7), (7,6), (4,3), (2,5), (7,7), (3,7)}.

Нарисуйте его, дайте полную характеристику (связность, циклы, ориентированность, матрица расстояний и т.д.), задайте матрицей смежности.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5}, U={(x1,x4), (x1,x3), (x2,x1), (x2,x4), (x3,x4), (x3,x5),(x4,x4),(x4,x5)}. Построить его графическое представление и матрицы инциденций и смежности, все простые цепи.

3. Доказать, что полный граф имеет n^{n-2} деревьев.

Вариант №18

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9,10}, U={(2,4), (1,8), (2,7), (3,6), (5,3), (1,3), (2,5), (4,6), (3,4), (6,8), (5,6), (1,9), (9,3), (5,7), (7,1), (3,7)}.

Нарисуйте его, дайте полную характеристику (связность, циклы, ориентированность, матрица расстояний и т.д.), задайте матрицей смежности.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5,x6}, U={(x1,x2), (x2,x4), (x4,x6), (x1,x1), (x3,x6), (x3,x2), (x1,x5), (x3,x1), (x6,x5), (x6,x1), (x2,x5), (x5,x4)}.Постройте граф. Найдите раскраску графа. Начиная с 1 вершины и с 5.

3. Построить сеть Петри для решения задачи о 5 мудрецах. (5 мудрецов. Могут есть или думать. Для еды им даны 5 палочек. Чтобы поесть, мудрецу нужны 2 палочки.)

Вариант №19

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(2,4), (2,8), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (5,6), (4,8), (1,9), (9,3), (1,7), (7,4), (3,7)}.

Нарисуйте его, дайте полную характеристику, (связность, циклы, ориентированность, матрица расстояний и т.д.) задайте матрицей смежности.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x5,x6}, U={(x1,x2), (x2,x4), (x4,x3), (x1,x1), (x3,x4), (x3,x2), (x1,x5), (x3,x1), (x5,x5), (x2,x1), (x3,x5), (x5,x4),(x1,x6), (x2,x6)}. Постройте его, найдите все простые цепи из 3 в 6 вершину.

3. Показать, что граф, имеющий мост, не является эйлеровым.

Вариант №20

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7}, U={(1,4), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (3,4), (5,6), (2,7), (3,7),(6,7),(1,5)}.

Нарисуйте его, двойственный ему граф, плоский и планарный, дайте полную характеристику, задайте матрицей смежности.

2. Задан граф G=(X,U), X={x1,x2,x3,x4,x6,x5}, U={(x1,x2), (x2,x3), (x4,x3), (x5,x1), (x3,x6), (x3,x2), (x1,x5), (x3,x1), (x6,x5), (x6,x1), (x3,x5), (x5,x4)}. Дайте определения: маршрута, цепи, цикла, связности. Постройте на заданном графе маршрут, цепь, цикл, покрывающее его дерево.

3. Определить число неизоморфных деревьев двудольного графа $К_{2,3}$.

Вариант №21

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(1,4), (1,8), (2,7), (3,6), (2,3), (1,3), (2,5), (6,8), (5,6), (1,9), (9,3), (2,7), (7,7), (3,7)}.

Нарисуйте его, Нарисуйте двойствегнный ему граф. Дайте полную характеристику двойственного графа, задайте матрицей смежности, постройте плоский и планарный графы.

2. Дайте определение цикла, маршрута. Приведите пример, используя граф заданный таблицей:

	X1	X2	X3	X4	X5
X1	0	1	1	0	1
X2	1	0	1	0	0
X3	1	1	0	1	1
X4	0	0	1	0	1
X5	1	0	1	1	0

Найдите хроматическое число и диаметр заданного графа.

3. Доказать, что удаление одного ребра, которое принадлежит какому-то циклу связного графа, не делает этот граф несвязным.

Вариант №22

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7, 8, 9}, U={(1,4), (1,8), (2,7), (3,6), (2,3), (1,3), (2,5), (6,8), (5,6), (1,9), (9,3), (2,7), (7,7), (3,7)}.

Нарисуйте его, найдите подграф, сураф, дополненние до полного графа.

2. Задан граф G=(X,U),|X|=6 (6 вершин) и степень каждой вершины $\rho \geq 3$. Нарисовать такой граф и построить все его простые циклы.

3. Доказать, что эйлеров граф не имеет мостов.

Вариант №23

1. Задан граф G=(X,U),

X={1, 2, 3, 4, 5, 6, 7}, U={(1,4), (2,7), (3,6), (2,3), (1,3), (2,5), (4,6), (3,4), (5,6), (2,7), (3,7),(6,7),(1,5)}.

Нарисуйте его, задайте матрицей инциденций, постройте подграф, сураф, дополнение до полного графа.

2. Дайте определение и назначение решетчатого графа. Приведите пример основных соотношений. Найдите расстояние между вершинами x5 и x27 (решетку взять из лекций).

3. Постройте примеры графов, для которых алгоритм последовательного раскрашивания строит не минимальную раскраску.

КОНТРОЛЬНАЯ РАБОТА №2

Вариант №1

1. Построить таблицу истинности функции, реализуемую следующей формулой:

$(x \to y) \oplus ((y \to z) \sim (\neg z \vee x))$.

Привести к виду ДНФ, используя алгебраические преобразования.

2. Задана булева функция. Получить СДНФ, используя разложение Шеннона.

$X1X2 \vee X1X3 \vee \neg X2 \neg X3 \vee \neg X1X2X3 \vee X2 \neg X3$.

3. Задана булева функция.

$F = X1 \neg X2X5 \vee X1X2X4 \neg X5 \vee X1 \neg X2 \neg X3X5 \vee X2X3 \neg X4 \neg X5 \vee X1X2 \neg X3X4 \neg X5$.

Построить карту Карно.

4. Минимизировать, используя метод Квайна и метод Петрика.

$F = X1X2 \neg X3 \neg X4 \vee \neg X2 \neg X3 \neg X4 \vee X1X2X3 \vee \neg X2X3X4 \vee X1X3 \neg X4 \vee X1 \neg X2 \neg X3X4$.

5. Минимизировать, используя карты Карно. Задана КНФ булевой функции.

$F=(X1 \vee X2 \vee \neg X3)(X2 \vee X3 \vee X4)(X3 \vee \neg X4)(\neg X2 \vee \neg X5)$.

Вариант №2

1. Построить таблицу истинности функции, реализуемую следующей формулой:

$((x \wedge y) \vee z) \oplus (z \to x)$.

Привести к виду ДНФ, используя алгебраические преобразования.

2. Задана булева функция. Получить СДНФ, используя разложение Шеннона.

$X1X2X3 \vee X1X3 \vee \neg X2 \neg X3 \vee \neg X1X2 \neg X3 \vee X2 \neg X3$.

3. Задана булева функция от 5 переменных. Построить карту Карно.

$F=\neg X1 \neg X2X4 \vee X1X2X4 \neg X5 \vee X1 \neg X2 \neg X4X5 \vee X2X3 \neg X4 \neg X5 \vee X1 \neg X3X4 \neg X5$.

4. Минимизировать, используя метод Квайна и метод Петрика.

F = X1X2¬X3¬X4 ∨ ¬X2¬X3∨X1X2X3 ∨ ¬X2X3X4 ∨ X1X3¬X4 ∨ ¬X2¬X3X4∨X1X2¬X3X4.

5. Минимизировать, используя карты Карно.

¬X1X2¬X3¬X4¬X5 ∨ X1¬X2X3¬X4¬X5∨X1X4¬X5 ∨ ¬X2¬X3X4X5 ∨ X1¬X2¬X3X4¬X5 ∨ X1X2¬X3¬X5 ∨ X1¬X3X4X5 ∨ X1X2¬X4X5 ∨ X1¬X3¬X4X5.

Вариант №3

1. Построить таблицу функции, реализуемую следующей формулой:

(x⊕¬y)∧(x∨z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Получить СДНФ функции f=(x∧y∧¬z∨¬x∧y∧z)→(x∨y).

3. Используя карты Карно, сравнить две функции:

F_1 = X1X2¬X3¬X4 ∨ ¬X2¬X3¬X4 ∨ X1X2X3 ∨ ¬X2X3X4 ∨ X1X3¬X4 ∨ X1¬X2¬X3X4.

F_2 = X1 ∨ ¬X1¬X2¬X3¬X4 ∨ ¬X2X3X4.

4. Минимизировать, используя метод Квайна и метод Петрика.

X1X2¬X4 ∨ ¬X2¬X3¬X4 ∨ X1X2X3X4 ∨ X1¬X2X3 ∨ X1X3¬X4 ∨ X1¬X2¬X3X4∨¬X1¬X2¬X3X4.

5. Минимизировать, используя карты Карно.

X1X2¬X3X4 ∨ X1¬X2X3¬X5 ∨ X2X3X5 ∨ X2X3¬X5 ∨ X2¬X3X5.

Вариант №4

1. Построить таблицу функции, реализуемую следующей формулой:

¬((¬x⊕y)∨(x~z)).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

F = X1X3¬X4 ∨ ¬X2¬X3¬X4 ∨ X1X2X3 ∨ ¬X1X3X4 ∨ X1X3X4 ∨ X1¬X2¬X3X4.

3. Представить функцию в виде вершин n – мерного куба.

F= x1x2 ∨ ¬x1x2¬x4 ∨ x1¬x2x3x4 ∨ ¬x1x2x3x4 ∨ x1¬x3.

4. Минимизировать, используя карты Карно.

X1X2¬X3X4 ∨ X1¬X2X3¬X5 ∨ X2X3X5 ∨ X2X3¬X5 ∨ X2¬X3X5.

5. Получить СДНФ (ac∨¬a¬b)→(b~ac).

Вариант №5

1. Построить таблицу функции, реализуемую следующей формулой:

(¬y→(z∧x))⊕(x∨y).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Представить функцию в вектором виде.

¬X1X2¬X3¬X4¬X5 ∨ X1¬X2X3¬X4¬X5∨X1X4¬X5 ∨ ¬X2¬X3X4X5 ∨ X1¬X2¬X3X4¬X5 ∨ X1X2¬X3¬X5 ∨ X1¬X3X4X5 ∨ X1X2¬X4X5 ∨ X1¬X3¬X4X5.

3. Минимизировать, используя метод Квайна и метод Петрика.

F = X1X2¬X3¬X4 ∨ ¬X1¬X2¬X3¬X4 ∨ X1X2X3 ∨ ¬X1X3X4 ∨ X1X3¬X4 ∨ X1¬X2¬X3X4.

4. Найдите СДНФ: f=(x1→x2)⊕x2¬x3.

5. Минимизировать функцию, используя карты Карно

(X1X2∨X4¬X5) ∧ (X3¬X4 → X5).

Вариант №6

1. Построить таблицу функции, реализуемую следующей формулой:

((x~z)⊕y)∨(x∧z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

x1x2 ∨ ¬x1¬x4 ∨ x1¬x2x3x4 ∨ ¬x1x2x3x4 ∨ x1¬x3¬x4 ∨ x1¬x2x3¬x4.

3. Построить карту Карно для функции.

¬X1X2X3X4¬X5 ∨ ¬X1X2X3¬X4¬X5∨X1¬X4¬X5 ∨ X2¬X3X4X5 ∨ X1X2¬X3X4¬X5 ∨ ¬X1X2¬X3¬X5 ∨ ¬X1X2¬X4X5 ∨ X1X4X5 ∨ X1¬X3¬X4X5.

4. Функция задана в виде КНФ. Приведите к виду СДНФ.

F=(X1∨¬X2)(X3∨X1)(¬X2∨X4)(X3∨X4).

5. Минимизировать функцию, заданную в форме КНФ, используя карты Карно.

(X1∨¬X2∨¬X3∨X5) ∧ (X1∨X2∨¬X3) ∧(X2∨X4∨¬X5) ∧ (X1∨¬X2∨X3) ∧ (X2∨X3∨X5).

Вариант №7

1. Построить таблицу истинности функции, реализуемую следующей формулой:

(X∧Y)∧(¬Y→(X∧¬Z)).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

X1X2X4X3 ∨ ¬X1X2¬X4 ∨ X1¬X2X4 ∨ X1¬X2X3X4 ∨ X1¬X2X3¬X4 ∨ X1X2¬X3¬X4.

3. Построить карту Карно для функции.

X1X2X4X5∨ ¬X1¬X5 ∨ X1¬X2X4X6 ∨ ¬X2X2X3X4 ∨ X1¬X3¬X4.

4. Найдите СДНФ для функции

F=(A¬B ⊕ AC) ∨ ¬(ACD→A¬C).

5. Минимизировать функцию, используя карты Карно:

¬X1X2¬X3X4¬X5 ∨ X1X2¬X3X4¬X5 ∨ X1¬X4¬X5 ∨ ¬X2X3X4X5 ∨ X1X2¬X3X4¬X5 ∨ ¬X1X2¬X3¬X5 ∨ ¬X1X2¬X4X5 ∨ X1X4X5 ∨ X1X3¬X4X5 ∨ X1¬X2¬X3¬X4X5 ∨ ¬X1X2¬X3X4X5

Вариант №8

1. Построить таблицу функции, реализуемую следующей формулой:

$(X \rightarrow \neg Y) \vee (\neg Y \sim X)$.

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

$X1X2X4X3 \vee \neg X1X2\neg X4 \vee X1\neg X2X4 \vee X1\neg X2X3X4 \vee X1\neg X2X3\neg X4 \vee X1\neg X2\neg X4$.

3. Построить карту Карно.

$X1X2\neg X3X4\neg X5 \vee \neg X1X2\neg X3X4X5 \vee X1\neg X4X2\neg X5 \vee X1\neg X2X3X4X5 \vee \neg X1X2\neg X3X4\neg X5 \vee \neg X1X2\neg X3\neg X5 \vee \neg X1X2\neg X4X5 \vee X1X4X5 \vee X1\neg X4X5$.

4. Получить СДНФ по заданной КНФ

$(x1 \vee x2 \vee x3)(\neg x1 \vee x2 \vee x4)(\neg x2 \vee x3 \vee x5)$.

5. Минимизировать функцию, используя карты Карно.

$(X \rightarrow (Y \wedge \neg (\neg X \sim A))) \vee Z \wedge \neg (A \oplus Z)$.

Вариант №9

1. Построить таблицу функции, реализуемую следующей формулой:

$\neg X \rightarrow (\neg Z \sim Y) \sim (Z \rightarrow (X \oplus Z))$.

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

$\neg A\neg BD \vee (ABC\neg D \vee AB\neg CD) \rightarrow \neg ACD$.

3. Построить карту Карно.

$X1X2X4\neg X5 \vee \neg X1X2\neg X3X4\neg X5 \vee X1\neg X4\neg X5 \vee \neg X2X3X4X5 \vee X1X2 \vee \neg X3X4\neg X5 \vee \neg X1X2\neg X3X5 \vee \neg X1X2\neg X4\neg X5 \vee X1X2\neg X4X5 \vee X1X3\neg X4X5$.

4. Преобразовать к виду СДНФ.

$(A \vee B \vee C)(\neg A \vee B \vee C)(A \vee \neg B \vee C) \vee \neg (AC \vee BC)$.

5. Преобразовать к виду СДНФ и минимизировать функцию, используя карты Карно.

$(X1X2 \vee X3\ X4X5) \rightarrow (X3\neg X4X5 \vee X1)$.

Вариант №10

1. Построить таблицу функции, реализуемую следующей формулой:

(¬Y⊕X)→(X∨¬Z)~(¬XY→¬Z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

((x→y)~(z⊕x)∧y).

3. Построить карту Карно ¬A¬B ∨ B¬D ∨ B¬CD ∨ ABCD.

4. Получить СДНФ (A∨C)(¬A∨¬B∨C) ~ (A∨¬B∨C).

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(X1∨X3∨¬X4∨X5) ∧ (X1∨¬X2∨¬X3) ∧ (X2∨X4∨¬X5) ∧ (X1∨¬X3∨X5) ∧ (X2∨X3∨X5).

Вариант №11

1. Построить таблицу функции, реализуемую следующей формулой:

(Y⊕X∧Y)(X∨Z).

2. Выяснить, является ли формула тождественно истинной:

(X→Y)→((X∨Z)→(Y∨Z)).

3. Минимизировать, используя метод Квайна и метод Петрика.

¬A¬BD∨ABC¬D∨AB¬CD∨AD¬AB∨AB¬C¬D.

4. Получить СДНФ (a∨¬b) ∧ (a∨c∧b)→((¬a∨d) ∧ (c∧d)).

5. f(x1,x2,x3,x4,x5)=¬x1¬x2x5∨x1x2x4¬x5∨x1¬x2¬x3x5∨ x2x3¬x4¬x5∨ x1x2¬x3x4¬x5. Минимизировать заданную выше функцию при помощи карт Карно.

Вариант №12

1. Построить таблицу функции, реализуемую следующей формулой:

(x∨¬y)~(¬x∨z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

¬X2X3X4¬X5 ∨ ¬X2X3¬X4¬X5 ∨ ¬X4¬X5 ∨ X2¬X3X4X5 ∨ X2¬X3X4¬X5 ∨ ¬X2¬X3¬X5 ∨ ¬X2¬X4X5 ∨ X4X5 ∨ ¬X3¬X4X5.

3. Построить Карту Карно.

X1X2¬X3X5¬X6 ∨ ¬X1X2¬X3X5∨X1¬X4X2¬X6 ∨ X1¬X2X3X4X5 ∨ ¬X1X2¬X3X4¬X6 ∨ ¬X1X2¬X3¬X5 ∨ ¬X1X2¬X4X5 ∨ X1X4X5 ∨ X1¬X4X5X6.

4. Преобразовать к виду СДНФ следующую формулу

¬(x∨y)∧¬(x→z)∧(x∨¬y).

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

x1x2¬x3x4x5 ∨ ¬x1¬x2x3 ∨ x2¬x3¬x5 ∨ x1x2x4 ∨ x1¬x4¬x5 ∨ ¬x1¬x3 ∨ ¬x1x2¬x4x5.

Вариант №13

1. Построить таблицу функции, реализуемую следующей формулой:

(¬Y∧X∨Z)∨(Z⊕Y).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

X2¬X3¬X4 ∨ X1¬X2¬X3¬X4 ∨ ¬X1¬X2¬X3 ∨ ¬X1¬X3X4.

3. Построить Карту Карно

¬X1X2X3X4¬X5 ∨ ¬X1X2X3¬X4¬X5∨X1¬X4¬X5 ∨ X2¬X3X4X5 ∨ X1X2¬X3X4¬X5 ∨ ¬X1X2¬X3¬X5 ∨ ¬X1X2¬X4X5 ∨ X1X4X5 ∨ X1¬X3¬X4X5.

4. Получить СДНФ F=A¬C∨B∨¬A¬C.

5. Задана КНФ булевой функции. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(X1∨X2∨¬X4) (¬X3∨¬X4∨X5)(X1∨X2)(X4∨¬X5).

Вариант №14

1. Построить таблицу функции, реализуемую следующей формулой:

(X→Y) →Z ∨¬ (X⊕Z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

¬X1X3X4¬X5 ∨ ¬X1X3¬X4¬X5 ∨ X1¬X4¬X5 ∨ ¬X3X4X5 ∨ X1¬X3X4¬X5 ∨ ¬X1¬X3¬X5 ∨ ¬X1¬X4X5 ∨ X1X4X5 ∨ X1¬X3¬X4X5 ∨ ¬X1X4¬X5.

3. Построить Карту Карно.

X1X2¬X3X4X5 ∨ ¬X1¬X2X6 ∨ X2¬X3X4¬X5 ∨ X1X2X4 ∨ X1¬X4¬X5 ∨ ¬X3X4X6 ∨ ¬X1¬X2X4X3 ∨ X1X2¬X3¬X5

4. Получить СДНФ:

F=X2¬X3∨X1X2X3∨X1X2∨X1X3∨X1¬X3X4∨¬X2X4∨X1X2X4.

5. Задана КНФ булевой функции. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(X1∨¬X3)(¬X2∨¬X3∨X4)(X1∨X3)(X1∨¬X5).

Вариант №15

1. Построить таблицу функции, реализуемую следующей формулой:

((X⊕Y)→(X∨Y)∨Z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

(X1X2 ∨ X4X3X4) →(X3¬X4 → X1).

3. Построить Карту Карно.

¬X1X2X3X4¬X5 ∨ ¬X1X2X3¬X4¬X5∨X1¬X4¬X5 ∨ X2¬X3X4X5 ∨ X1X2¬X3X4¬X5 ∨ ¬X1X2¬X3¬X5 ∨ ¬X1X2¬X4X5 ∨ X1X4X5 ∨ X1¬X3¬X4X5.

4. Получите СДНФ для функции ¬A¬BD∨ABC¬D∨AB¬CD.

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(¬X1X2 ∨ X4¬X3¬X5) → (X3¬X4¬X5 ∨ X1) ∧ (X3∨ X4).

Вариант №16

1. Построить таблицу истинности функции, реализуемую следующей формулой:

(x→y)⊕ (y→z).

Приведите к ДНФ, используя математические преобразования.

2. Задана булева функция от 5 переменных

f(x1,x2,x3,x4,x5)=¬x1¬x2x5∨x1x2x4¬x5∨x1¬x2¬x3x5∨x2x3¬x4¬x5∨ x1x2¬x3x4¬x5.

Найти СДНФ.

3. Построить карту Карно.

¬x1¬x2x4∨x1x2x4¬x5∨x1¬x2¬x4x5∨x2x3¬x4¬x5∨ x1¬x3x4¬x5.

4. Задана КНФ. Минимизировать, используя карты Карно и метод Квайна.

(X1∨X2∨¬X3)(X2∨X3∨X4)(X3∨¬X4)(¬X2∨¬X4).

5. Найти простые импликанты функции

¬x1x2 ∨ x1¬x2 ∨ x2x3 ∨ x1¬x3.

Вариант №17

1. Построить таблицу функции, реализуемую следующей формулой:

((X⊕Y)→(X∨Y) →Z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

(X1X2 ∨ X4X3) ∧ (X3¬X4 ∨ X1) ∨ $\bar{X}$1X2X3 ∨ ¬X2X3¬X4.

3. Построить Карту Карно.

¬X1X2¬X3¬X4¬X5 ∨ X1¬X2X3¬X4¬X5∨X1X4¬X5 ∨ ¬X2¬X3X4X5 ∨ X1¬X2¬X3X4¬X5 ∨ X1X2¬X3¬X5 ∨ X1¬X3X4X5 ∨ X1X2¬X4X5 ∨ X1¬X3¬X4X5.

4. Получите СДНФ для функции ¬A¬BD∨ABC¬D∨AB¬CD.

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(¬X1X2 ∨ ¬X3¬X4) → (X3¬X4 → X1) ∨ (X3X4).

Вариант №18

1. Построить таблицу функции, реализуемую следующей формулой:

((¬x⊕y)∧(x→z)).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика. Функция задана в КНФ.

(x1∨x4)(x2∨¬x3∨¬x4)(¬x1∨¬x2∨x3).

3. Получить СДНФ x1→(x2¬x3)⊕x2.

4. Постройте карту Карно для функции.

¬X1X2X3X4¬X5 ∨ ¬X1X2X3¬X4¬X5∨X1¬X4¬X6 ∨ X2¬X3X4X6 ∨ X1X2¬X3X4¬X5 ∨ ¬X1X2¬X3¬X6 ∨ ¬X1X2¬X4X5 ∨ X1X4X6 ∨ X1¬X3¬X4X5.

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(x2∨x3∨¬x4)(¬x1∨¬x3∨x5)(x3∨x2)(x4∨¬x2).

Вариант №19

1. Построить таблицу функций, реализуемых следующими формулами:

z→y¬x→(y⊕z)∧x.

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

(x1∨x4)(x2∨¬x3∨¬x4)(¬x1∨¬x2∨x3).

3. Получить СДНФ по заданной КНФ

(x1∨x2∨x3)(¬x1∨x2∨x4)(¬x2∨x3∨x5).

4. Получить СДНФ (x1→x2)⊕¬x2∨x1x3.

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(x1∨x3∨¬x4)(x1∨¬x2∨x4)(x2∨x3)(x3∨¬x4).

Вариант №20

1. Построить таблицу функции, реализуемую следующей формулой:

(y⊕x∨¬z∧y) ∧ (z⊕y).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Минимизировать, используя метод Квайна и метод Петрика.

f(x1,x2,x3,x4)= ¬x1¬x3 ∨ x1x2x4 ∨ x1¬x2¬x3 ∨ x2x3¬x4 ∨ x1x2¬x3x4.

3. Преобразовать к виду СДНФ (a∨b∨c)(¬a∨b∨c)(a∨¬b∨c).

4. Получить ДНФ для функции : f=(a∨b∨c)(¬a∨¬b)¬bc.

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно.

(X1X2∨ X3 X4X5) →(X3¬X4X5 → X1).

Вариант №21

1. Построить таблицу функции, реализуемую следующей формулой:

¬(y∨¬x)⊕(x∨¬y∨¬z).

Привести к виду ДНФ, используя алгебраические преобразования.

2. Получить СДНФ, используя разложение Шеннона.

¬a¬b∨b¬d∨b¬cd∨abcd.

3. Найти КНФ функции:

X1X2¬X3X4¬X5 ∨ ¬X1X2¬X3X4X5∨X1¬X4X2¬X5 ∨ X1¬X2X3X4X5 ∨ ¬X1X2¬X3X4¬X5 ∨ ¬X1X2¬X3¬X5 ∨ ¬X1X2¬X4X5 ∨ X1X4X5 ∨ X1¬X4X5.

4. Построить СДНФ (x1⊕x2)→x2x3.

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно и метод Квайна.

(x1∨x3∨x4)(¬x2∨x3)(x1∨¬x3)(¬x3∨x4).

Вариант №22

1. Построить таблицу истинности для следующей формулы:

(x→¬z∧y) ∨ (x⊕y)→x.

Привести к виду ДНФ, используя алгебраические преобразования.

2. Построить карту Карно.

X1X2X4¬X5 ∨ ¬X1X2¬X3X4¬X5∨X1¬X4¬X5 ∨ ¬X2X3X4X5 ∨ X1X2∨¬X3X4¬X5 ∨ ¬X1X2¬X3X5 ∨ ¬X1X2¬X4¬X5 ∨ X1X2¬X4X5 ∨ X1X3¬X4X5.

3. Определить, является ли формула F тождественно истинной.

(x→y)→((x∨z)→(y∨z)).

4. Построить СДНФ (x1∨x4)(x2∨¬x3∨¬x4)(¬x1∨¬x2∨x3).

5. Преобразовать к виду СДНФ, минимизировать функцию, используя карты Карно и метод Квайна.

(a∨c)⊕(b∨¬d)→(a∨b¬c).

СПИСОК ЛИТЕРАТУРЫ

1. Нефедов В.Н., Осипова В.А. Курс дискретной математики. – М.: Наука, 1992.

2. Яблонский С.В. Введение в дискретную математику. – М.: Наука, 1978.

3. Мендельсон Э. Введение в математическую логику. – М., 1976.

4. Вольвачев Р.Т. Элементы математической логики и теории множеств.

5. Харари Ф., Палмер Э. Перечисление графов. – М.: Мир, 1997.

6. Баранов С.И. Синтез микропрограммных автоматов. – Л.: Энергия, 1979.

7. Корячко В.П., Курейчик В.М., Норенков И.П. Теоретические основы САПР. – М.: Радио и связь, 1987.

8. Кориков А.М., Сафьянова Е.Н. Основы системного анализа и теории систем. – Томск: Изд-во Томского университета, 1989.

9. Шевелев Ю.П. Дискретная математика. Теория множеств. Булева алгебра. Часть 1.