МИНИСТЕРСТВО ОБРАЗОВАНИЯ УКРАИНЫ
УЧЕБНО-МЕТОДИЧЕСКИЙ КАБИНЕТ ВЫСШЕГО ОБРАЗОВАНИЯ
СЕВАСТОПОЛЬСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ИНСТИТУТ

В. Г. Новоселов, А. В. Скатков

ПРИКЛАДНАЯ МАТЕМАТИКА ДЛЯ ИНЖЕНЕРОВ-СИСТЕМОТЕХНИКОВ. ДИСКРЕТНАЯ МАТЕМАТИКА В ЗАДАЧАХ И ПРИМЕРАХ

Киев УМК ВО 1992

ъъ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ УКРАИНЫ
УЧЕБНО-МЕТОДИЧЕСКИЙ КАБИНЕТ ВЫСШЕГО ОБРАЗОВАНИЯ
СЕВАСТОПОЛЬСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ИНСТИТУТ

В.Г.Новоселов, А.В.Скатков

ПРИКЛАДНАЯ МАТЕМАТИКА ДЛЯ ИНЖЕНЕРОВ-СИСТЕМОТЕХНИКОВ. ДИСКРЕТНАЯ МАТЕМАТИКА В ЗАДАЧАХ И ПРИМЕРАХ

Утверждено Советом Учебно-методического кабинета высшего образования Министерства образования Украины в качестве учебного пособия для студентов специальности 22.01

Киев УМК ВО 1992

УДК 519.85.681.15

Прикладна математика для інженерів-системотехніків. Дискретна математика в задачах і прикладах: Навч.посібник / В.Г.Новоселов, О.В.Скатков. - К.: НМК ВО, 1992. - 200 с. - Рос. мовою.

В даному посібнику наведено змістовні та формальні постановки задач, основні поняття, форми подання даних і алгоритми розв'язування, які докладно ілюструються прикладами. Пропонується 30 варіантів індивідуальних завдань для кожної задачі. Завдання підібрані так, щоб у них відбивались особливості алгоритмів розв'язувань.

Посібник містить матеріал з основних розділів дискретної математики: оптимізаційні задачі на скінченних множинах і графах, перетворення бульових функцій, комбінаторні обчислення для оцінювання обсягу перебирань.

Іл. 101. Бібліогр.: 23 назви

В данном пособии приведены содержательные и формальные постановки задач, основные понятия, формы представления данных и алгоритмы решения, которые подробно иллюстрируются на примерах. Даются 30 вариантов индивидуальных заданий для каждой задачи. Задания подобраны таким образом, чтобы в них отразились особенности алгоритмов решения.

Пособие содержит материал по основным разделам дискретной математики: оптимизационные задачи на конечных множествах и графах, преобразования булевых функций, комбинаторные вычисления для оценки объема переборов.

ISBN 5-7763-1310-4

ОБЩИЕ СВЕДЕНИЯ

Дискретная математика - это фундаментальная математическая дисциплина для специалистов-системотехников, работающих в различных направлениях разработки и использования вычислительной техники.

Модели и методы преобразования данных дискретной математики существенно отличаются от соответствующих построений непрерывной математики, прежде всего, ярко выраженной алгоритмической направленностью. Используемыми в инженерной практике результатами дискретной математики являются, как правило, не формулы, а алгоритмы решения задач. Обучаемому необходимо получить решение задач алгоритмическими преобразованиями.

Несмотря на разнообразие понятий и моделей, методика решения задач едина и основана на алгоритмах организации и сокращения переборов вариантов решений. Результатами решений являются не только числа, но и оптимальные в некотором смысле подмножества, наборы, формы представления данных. Для студентов заочной формы обучения номер варианта индивидуального задания вычисляют по модулю 30 от числа, определенного двумя последними цифрами зачетной книжки /например, XXXX24 - 24, XXXX76=16/.

В процессе изучения материала данного пособия студенты осваивают решение классических задач дискретной математики, приобретают умения и навыки алгоритмического решения задач, организации переборов, выбора критериев их оптимальности, оценки объема перебора, построения приближенных решений. Эти навыки необходимы инженеру-системотехнику для применения ЭВМ.

Материал пособия основан на опыте более чем двадцатилетнего чтения лекций по дискретной математике. Авторы большое внимание уделили подбору вариантов индивидуальных заданий для самостоятельной работы студентов.

Глава I. ЗАДАЧИ ОПТИМИЗАЦИИ НА МНОЖЕСТВАХ

I.I. Основные понятия теории конечных множеств и отношений

Теория конечных множеств – очень удобное средство формирования и записи основных свойств и объектов дискретной математики. Данный подраздел содержит справочные данные: основные понятия из теории конечных множеств и отношений, которые используются в последующем изложении.

I.I.I. Множества. Способы задания.

Множеством принято называть совокупность объектов, рассматриваемых совместно. Основатель теории Г.Кантор дал такое определение: "Множество есть многое мыслимое как единое". Объекты множества называются элементами. При задании множеств элементы не повторяются, порядок перечисления элементов безразличен. Для конечного множества число элементов называется мощностью. Множества будем обозначать прописными буквами латинского алфавита /A, B, X, M и т.д./; элементы множества – строчными, чаще всего снабженными индексами. Например, можно рассматривать множество $\mathcal{X}$, состоящее из n независимых переменных x_1, x_2, ... x_n. Индекс позволяет различить элементы множества. Для обозначения принадлежности элемента x_i множеству $\mathcal{X}$ используется символ $\in$ /$x_i \in \mathcal{X}$ означает и читается так: "элемент x_i принадлежит множеству $\mathcal{X}$", $x_j \notin \mathcal{X}$ означает, что "элемент x_j не принадлежит множеству $\mathcal{X}$"/.

Первый способ задания множества – перечисление элементов. Перечисляются все элементы множества, перечисление элементов заключается в фигурные скобки.

<u>Примеры</u>. $A = \{a, b, c\}$, $B = \{1, 2, 3, 4\}$, $E = \{0, 1\}$.

Множество может быть задано без перечисления элементов: задается правило $P(x)$, которое определяет для любого объекта x, принадлежит он множеству или нет. В этом втором способе форма записи задания множества $\mathcal{X}$ следующая:

$$\mathcal{X} = \{x \,/\, P(x)\}.$$

Введем векторное обозначение множества $\widetilde{A}$ различных подмножеств. Каждому подмножеству сопоставляем n-разрядный вектор, где n – мощность опорного множества. Сопоставим i-й разряд вектора i-му элементу опорного множества. Присутствие элемента a_i в подмножестве обозначим I, отсутствие – 0, тогда получим следующее представление $\widetilde{A}_{вект}$ множества $\widetilde{A}$

$$\widetilde{A}_{вект} = \{000, 100, 010, 001, 110, 101, 011, 111\}.$$

Последнее обозначение множества $\widetilde{A}_{вект}$ можно интерпретировать еще и так. Возьмем множество $\mathcal{X} = \{x_1, x_2, x_3\}$, где x_1, x_2, x_3 – независимые переменные, каждая из которых может принимать значения из множества $\{0,1\}$ /обозначаем $x_i \in \{0,1\}$ для $1 \le i \le 3$ /. Построим множество M различных комбинаций значений переменных множества $\mathcal{X}$:

$$M = \{000, 100, 010, 001, 110, 101, 011, 111\}.$$

Введем элементарные операции над множествами.

Объединением множеств A и B называется множество C, состоящее из всех элементов, принадлежащих хотя бы одному из множеств A или B. Обозначение операции объединения – $\cup$.

$$C = A \cup B = \{c \,/\, c \in A \ \text{или} \ c \in B\}.$$

Операцию объединения можно проиллюстрировать с помощью кругов Эйлера.

Заштриховано множество $C = A \cup B$.

Пересечением множеств A и B называется множество C, состоящее из всех элементов, принадлежащих одновременно обоим множествам A и B.

$$C = A \cap B = \{c \,/\, c \in A \ и \ c \in B\}.$$

Заштрихована область пересечения.

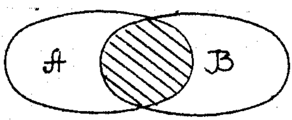
Если $A \cap B = \varnothing$, то множества A и B не пересекаются.

<u>Примеры</u>. $A = \{a \,/\, a$ – целое четное число от I до I00 включительно$\}$,

$$\mathcal{X} = \{x_i \,/\, 1 \le i \le n\}.$$

Для иллюстрации взаимного соотношения нескольких множеств используется геометрическое задание /обозначение/ множеств кругами Эйлера. Множество обозначается внутренней частью геометрической фигуры /чаще всего круга или эллипса/.

Например, высказывание о том, что множества A и B имеют общие элементы, можно проиллюстрировать графически.



Заштрихованы общие элементы множеств A и B.

1.1.2. Подмножества. Операции над множествами.

Два множества равны, если они состоят из одних и тех же элементов /с точностью до порядка/.

Множество B, все элементы которого являются одновременно элементами множества A, называются подмножеством множества A. Используется обозначение $B \subseteq A$ / B поглощается A /. Если не допускается равенство множеств B и A, то используется обозначение $B \subset A$. Следует различать связки: принадлежность $\in$ и поглощение $\subseteq$. Связка $\in$ ставится между элементом a и множеством A, содержащим его, $a \in A$; связка $\subseteq$ ставится между двумя множествами, указывая на то, что одно является подмножеством другого. В частности, можно рассматривать следующие соотношения: $a \in A$, $\{a\} \subseteq A$. Рассмотрим несколько специальных множеств. Отдельное обозначение используется для пустого множества: $A = \phi$ означает, что множество A не содержит элементов. Необходимо отметить множество $\tilde{A}$ всех подмножеств некоторого множества A. Множество A в этом случае называется опорным. Например, $A = \{a, b, c\}$. Образуем различные подмножества: $\phi, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}$.
Тогда множество $\tilde{A}$ всех подмножеств опорного множества следующее:

$$\tilde{A} = \phi, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}.$$

Вычитание $A \setminus B$ множеств A и B определяет множество C, состоящее из всех тех элементов множества A, которые не содержатся в B:

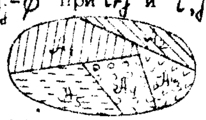
$$C = A \setminus B = \{c / c \in A \text{ и } c \notin B\}.$$

Заштрихована область разности.

Разбиение множества A определяет множество $C = \{A_1, \dots, A_k\}$ такое, что элементы $A_1, \dots, A_k$ – непересекающиеся подмножества множества A, объединение которых есть множество A. Подмножества A_i называют классами.

C = разбиение $A = \{A_1, \dots, A_k\}$ и $\bigcup\limits_{i=1}^{k} A_i = A$,

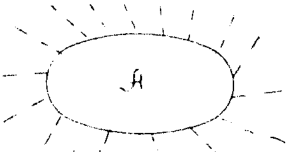
$A_i \cap A_j = \phi$ при $i \neq j$ и $i, j \in \{1, \dots, k\}$.



По-разному заштрихованы классы /непересекающиеся подмножества/, на которые разбивается множество A. Очевидно, разбиение множества A на классы можно выполнить неоднозначно.

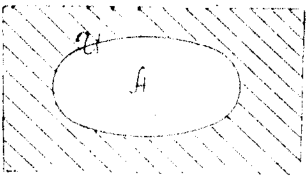
Дополнением множества A называется множество C, состоящее из всех элементов, не принадлежащих A:

$$C = \bar{A} = \{c / c \notin A\}.$$



Заштриховано множество $C = \bar{A}$.

Чаще всего дополнение множества рассматривается в некотором универсальном множестве U , содержащем все рассматриваемые в некоторой задаче элементы:



$$C = \bar{A}_U = \{c / c \notin A \text{ и } c \in U\} .$$

Заштриховано $\bar{A}_U$.

$$A \cup \bar{A}_U = U.$$

I.I.3. Отношения.

Определение отношения. Декартовым произведением $D = X \times Y$ называется множество упорядоченных пар (x_i, y_j) элементов $x_i \in X = \{x_1, \dots, x_n\}$, $y_j \in Y = \{y_1, \dots, y_m\}$:

$$D = \{(x_1, y_1), \dots, (x_1, y_m), (x_2, y_1), \dots, (x_n, y_1), \dots, (x_n, y_m)\}.$$

Мощность множества D равна $n \times m$. В качестве множества Y может быть принято множество X .

Бинарным отношением называется подмножество R множества D $(R \subseteq D)$. Символ R используется для обозначения подмножества и для обозначения того, что элементы x_i и y_j находятся в отношении R :

$$(x_i, y_j) \in R \leftrightarrow x_i R y_j .$$

<u>Примеры:</u>

диагональное отношение $E = \{(x_i, x_i) / x_i \in X\}$;

полное отношение $U = D = X \times Y$;

отношение строгого порядка $R = >$;

отношение нестрогого порядка $R = \geqslant$;

функциональное отношение /определяет функцию $y = R(x)$, при котором для каждого элемента $x_i \in X$ существует один и только один элемент $y_i \in Y$ такой, что $(x_i, y_j) \in R$. Последний пример показывает, что понятие отношения обобщает понятие функция.

Представление бинарных отношений. Множество R однозначно определяет отношение R. Можно задать правило, определяющее отношение R. Часто отношение R задается матрицей отношений $|r_{ij}|$:

$$r_{ij}=\begin{cases}1, & (x_i, y_j)\in R,\\ 0, & (x_i, y_j)\notin R.\end{cases}$$

Примеры.

$R = \geq$, $X = \{1,2,3\}$, $Y = \{1,2,3,4\}$

$\geq$	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

$R = \ni$, $X = \{\{y_1, y_3\}, \{y_1, y_4\}, \{y_2, y_4\}\}$, $Y = \{y_1, y_2, y_3, y_4\}$

$\ni$	y_1	y_2	y_3	y_4
$\{y_1 y_3\}$	1	0	1	0
$\{y_1 y_4\}$	1	0	0	1
$\{y_2 y_4\}$	0	1	0	1

В подразд. I.2 более подробно ознакомимся с этим отношением как отношением покрытия.

R = возведение в квадрат, $X = \{1,2,3\}$, $Y = \{1,4,9\}$.

$y=x^2$	1	4	9
1	1	0	0
2	0	1	0
3	0	0	1

Функциональное отношение $y = x^2$.

I.I.4. Операции над отношениями.

Так как отношения определены как множества, то для них естественно определены операции объединения $\cup$, пересечения $\cap$, дополнения $-$. Кроме того, введены еще две операции:

обратное отношение R^{-1}:

$$(x_i, y_i)\in R \rightarrow (y_j\, x_i)\in R^{-1}.$$

Примеры.

$\geq$	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0

$\geq^{-1}$	1	2	3	4
1	1	1	1	1
2	0	1	1	1
3	0	0	1	1

$\neq$	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1

$\neq^{-1}$	1	2	3	4
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1

Для обратного отношения справедливы два следующих свойства:

$$(R^{-1})^{-1} = R \quad и ,$$

если $R_1 \subseteq R_2$, то $R_1^{-1} \subseteq R_2^{-1}$;

произведение отношений

$$R_1 \subseteq \mathcal{X} \times \mathcal{Y} , \quad R_2 \subseteq \mathcal{Y} \times \mathcal{Z} , \qquad \text{тогда}$$

$$R = R_1 \circ R_2 = \{ (x_i , z_k) / \exists\, y_j \in \mathcal{Y} ((x_i y_j) \in R_1 \text{ и } (y_j , z_k) \in R_2) \} .$$

Пример.

$$R_1 = \subseteq , \quad \mathcal{X} = \{\{1\}, \{1,2\}, \{2,3\}\} ,$$
$$\mathcal{Y} = \{\{1,4\}, \{1,2,3\}\} ,$$
$$R_2 = \subseteq , \quad \mathcal{Z} = \{\{1,3,4\}, \{1,2,4\}\} .$$

R_1	$\{1,4\}$	$\{1,2,3\}$
$\{1\}$	1	1
$\{1,2\}$	0	1
$\{2,3\}$	0	1

R_2	$\{1,3,4\}$	$\{1,2,4\}$
$\{1,4\}$	1	1
$\{1,2,3\}$	0	0

$R_1 \circ R_2$	$\{1,3,4\}$	$\{1,2,4\}$
$\{1\}$	1	1
$\{1,2\}$	0	0
$\{2,3\}$	0	0

I.I.5. Свойства отношений.

Различные отношения имеют довольно общие свойства.

Рефлексивность. Отношение $R \subseteq \mathcal{X} \times \mathcal{X}$ называется рефлексивным, если для всех $x_i \in \mathcal{X}$ пара $(x_i , x_i) \in R$. Например, отношение $R = \geq$ – рефлексивно.

Отношение R антирефлексивно, если для всех $x_i \in \mathcal{X}$ пара $(x_i , x_i) \notin R$. Очевидно, отношение $R = >$ – антирефлексивно. Если же хотя бы для одного $x_i \in \mathcal{X}$ пара $(x_i , x_i) \notin R$, то отношение R нерефлексивно.

Симметричность. Отношение $R \subseteq \mathcal{X} \times \mathcal{X}$ называется симметричным, если из $(x_i\, x_j) \in R$ следует $(x_j\, x_i) \in R$. Если же из $(x_i\, x_j) \in R$ следует $(x_j\, x_i) \notin R$, то отношение R называется асимметричным. Например, отношение $R = \neq$ симметрично, отношение $R = >$ – асимметрично.

Для симметричных отношений

$$R^{-1} = R.$$

Транзитивность. Отношение $R \subseteq \mathcal{X} \times \mathcal{X}$ транзитивно, если из $(x_i, x_j) \in R$ и $(x_j, x_k) \in R$ следует $(x_i, x_k) \in R$. Например, отношение $R = \geqslant$ и отношение $R = \parallel$ /параллельно на множестве прямых линий/ – транзитивны.

I.I.6. Отношения эквивалентности, толлерантности, порядка.

Отношение R называется отношением эквивалентности, если оно обладает свойствами рефлексивности, симметричности и транзитивности. Например, отношение $R = \equiv (mod\, p)$. Отношение эквивалентности $R \subseteq \mathcal{X} \times \mathcal{X}$ разбивает множество $\mathcal{X}$ на классы эквивалентности: элементы x_i, x_j, принадлежащие одному классу, находятся в отношении R $((x_i, x_j) \in R)$, элементы x_i, x_k, принадлежащие разным классам, не находятся в отношении R.

Отношение R называется отношением толлерантности, если оно обладает свойствами рефлексивности и симметричности, но нетранзитивно.

Примеры. На множестве M двоичных наборов может быть задано отношение $R =$ "находиться на расстоянии I по Хэммингу", т.е. "различаться значением одной и только одной переменной". При трех переменных:

$$M = \{000, 00I, 0I0, 0II, I00, I0I, II0, III\}.$$

Наборы 000 и 00I находятся на расстоянии I по Хэммингу, наборы 00I и 0II тоже находятся на расстоянии I, однако наборы 000 и 0II уже не находятся на расстоянии I.

На множестве выбранных портретов может быть задано отношение толлерантности $R =$ "быть похожим".

На множестве подмножеств некоторого множества – отношение толлерантности $R =$ "иметь не пустое пересечение".

Отношением нестрогого порядка называется отношение R, обладающее свойствами рефлексивности, асимметричности и транзитивности.

Примеры: $R = \geqslant$, $R = \supseteq$.

Отношением строгого порядка называется отношение R, обладающее свойствами антирефлексивности, асимметричности и транзитивности.

Примеры: $R = \geqslant$, $R = \supset$.

Если отношение порядка определено для всех пар, т.е. для каждой пары $x_i, x_j \in \mathcal{X}$, либо $(x_i, x_j) \in R$, либо $(x_j, x_i) \in R$, то отношение $R \subseteq \mathcal{X} \times \mathcal{X}$ задает полное линейное упорядочение элементов множества $\mathcal{X}$. Например, отношение $R = \geqslant$ задает линейное упорядочение множества чисел.

Если отношение порядка определено не для всех пар, т.е. существуют не сравнимые элементы $x_i, x_j \in R$, для которых $(x_i, x_j) \notin R$ и $(x_j, x_i) \notin R$, то отношение порядка задает частичное упорядочение. Например, отношение $R = \supseteq$ задает частичное упорядочение на множестве всех подмножеств.

При линейном упорядочении на множестве $\mathcal{X}$ имеется один максимальный и один минимальный элементы, при частичном упорядочении может быть несколько максимальных и минимальных элементов.

Пример. На множестве M наборов двоичных переменных $x_1, \ldots, x_n$ задано отношение $R = \succcurlyeq$, которое определяется следующим образом:

$$R = \{(m_i, m_j) / m_i, m_j \in M \; (\forall x_k \in \mathcal{X} \; (m_i^k \geqslant m_j^k))\}.$$

Отношение $R = \succcurlyeq$ называется векторным отношением больше равно. Оно задает частичное упорядочение наборов.

Пусть задано при $\mathcal{X} = \{x_1, x_2, x_3\}$ подмножество $M_1 \subseteq M$:

$$M_1 = \{001,\ 011,\ 110,\ 101\}.$$

Для множества M_1 выделяются два минимальных элемента, меньше которых в векторном упорядочении других элементов в множестве M_1 нет: 001 и 110. Аналогично можно выделить три максимальных элемента, больше которых в множестве M_1 нет: 110, 011 и 101.

1.2. Задача о покрытии

1.2.1. Постановка задачи о покрытии.

Неформальное описание задачи. Задача о покрытии - это математическая модель большого числа различных оптимизационных задач дискретной математики. Рассмотрим в качестве примера задачу "О полном собрании сочинений". Некоторый писатель выпустил множество $A = \{A_1, \ldots, A_m\}$ сборников, содержащих в совокупности все множество $B = \{B_1, \ldots, B_n\}$ его сочинений. Каждый сборник A_i содержит некоторое подмножество сочинений из $B \; (A_i \subseteq B)$ и имеет некоторую цену a_i. Необходимо найти такое множество $P = \{A_{i_1}, \ldots, A_{i_\ell}\}$, $(i_j \in \{1, \ldots, m\}, \ell \leq m)$ сборников, чтобы в их совокупности содержались все сочинения данного автора:

$$\bigcup_{j=1}^{\ell} A_{i_j} = B, \qquad /1.1/$$

при этом цена $S = \sum_{j=1}^{\ell} a_{i_j}$ такого полного собрания сочинений была наименьшей, либо количество ℓ сборников было минимальным.

2*

Пример I.I.

$$
\begin{array}{llllllllll}
B=\{b_1, & b_2, & b_3, & b_4, & b_5, & b_6, & b_7, & b_8, & b_9\} & \\
A_1=\{b_1, & & b_3, & & & b_6, & b_7 & & \} & a_1 = 1; \\
A_2=\{b_1, & b_2, & & b_4, & & & & b_8 & \} & a_2 = 2,5; \\
A_3=\{ & b_2, & & & b_5, & & b_7 & & \} & a_3 = 2; \\
A_4=\{ & & b_3, & & & & & b_8, & b_9\} & a_4 = 1,5; \\
A_5=\{ & & & b_4, & b_5, & & & b_8, & b_9\} & a_5 = 3; \\
A_6=\{ & & b_3, & & b_5, & b_6, & b_7, & & b_9\} & a_6 = 7; \\
A_7=\{ & b_2, & & b_4, & & b_6 & & & \} & a_7 = 1.
\end{array}
$$

Множество $P_i = \{A_{i_1},\dots,A_{i_\ell}\}$, удовлетворяющее условию /I.I/, является возможным решением задачи. Рассмотрим некоторые из таких решений:

$$P_1 = \{A_1, A_2, A_5\}, \quad A_1 \cup A_2 \cup A_5 = B,$$

цена $\quad S_1 = 1 + 2,5 + 3 = 6,5; \quad \ell_1 = 3;$

$$P_2 = \{A_2, A_6\}, \quad A_2 \cup A_6 = B,$$

цена $\quad S_2 = 2,5 + 7 = 9,5, \quad \ell_2 = 2;$

$$P_3 = \{A_1, A_3, A_4, A_7\}, \quad A_1 \cup A_3 \cup A_4 \cup A_7 = B,$$

цена $\quad S_3 = 1 + 2 + 1,5 + 1 = 5,5, \quad \ell_3 = 4.$

Можно построить еще много других решений. Но не всякое подмножество из A - решение. Например, подмножество $\{A_1, A_2\} \subseteq A$ сборников не является полным собранием сочинений, поскольку не содержит сочинений b_5 и b_9.

Условие /I.I/ - необходимое условие построения решения; условия 1/ $S \to \min$, или 2/ $\ell \to \min$ служат критериями, по которым из возможных решений P_1, P_2, ... выбираются наилучшие.

В данном примере из рассмотренных по критерию I выбирается P_3, по критерию 2 - P_2. Так как мы не рассматривали другие возможные решения, то не можем гарантировать, что найденные решения - наилучшие. Действительно, относительно P_2 для данного примера можно подтвердить оптимальность решения по критерию 2 дополнительными рассуждениями: ни один из сборников A_1, ..., A_7 не содержит все сочинения автора, таким образом $\ell \geqslant 2$; так как для P_2 величина $\ell_2 = 2$, то это решение содержит минимально возможное количество сборников.

Формулировка задачи о покрытии на языке теории множеств.

Абстрагируясь от возможных интерпретаций, сформулируем задачу о покрытии на языке теории множеств. Пусть $B = \{b_1, \dots, b_n\}$ – опорное множество. Имеется множество $A = \{A_1, \dots, A_m\}$ подмножеств множества B ($A_i \subseteq B$, $\bigcup_{i=1}^{m} A_i = B$). Каждому подмножеству A_i сопоставлено число a_i, по-прежнему называемое ценой. Множество $P = \{A_{i_1}, \dots, A_{i_\ell}\}$ ($\ell \le m$) называется решением задачи о покрытии или просто покрытием, если выполнено условие /I.I/. Объяснением термина "покрытие" служит то, что совокупность множеств $A_{i_1}, \dots, A_{i_\ell}$ содержит все элементы множества B, т.е. покрывает множество B.

Теорема I.I.[*] Если P – покрытие, то и $P' \supseteq P$ – тоже покрытие, т.е. множество всех возможных покрытий вогнуто.

Определения. Покрытие P называется безызбыточным, если при удалении из него хотя бы одного элемента оно перестает быть покрытием. Иначе покрытие – избыточно.

Покрытие P называется минимальным, если его цена $S = \sum_{j=1}^{\ell} a_{i_j}$ – наименьшая среди всех покрытий данной задачи.

Покрытие P называется кратчайшим, если ℓ – наименьшее среди всех покрытий данной задачи.

Теорема I.2. Минимальные и кратчайшие покрытия – безызбыточны.

Пример технического приложения задачи о покрытии. К задаче о покрытии сводятся многие технические задачи. Например, задача "О минимальном проверяющем наборе тестов". Пусть некоторое устройство имеет множество $B = \{b_1, \dots, b_n\}$ элементов, каждый из которых может находиться в работоспособном или неработоспособном состоянии. Пусть тест T_i – некоторое воздействие на устройство. Оно позволяет судить о работоспособности подмножества $A_i \subseteq B$ элементов по правильной или неправильной реакции устройства на воздействие T_i.

Задано множество $T = \{T_1, \dots, T_m\}$ допустимых тестов и соответствующее ему множество $A = \{A_1, \dots, A_m\}$ подмножеств проверяемых элементов ($\bigcup_{i=1}^{m} A_i = B$). Требуется найти набор тестов $\{T_{i_1}, \dots, T_{i_\ell}\}$, проверяющих работоспособность всех элементов $b_j \in B$ устройства ($\bigcup_{j=1}^{\ell} A_{i_j} = B$).

---

[*] Доказательства всех теорем опущены.

Можно строить кратчайшие / $\ell \to min$ /, или минимальные, наборы тестов. Цена a_i теста T_i может иметь, например, смысл времени воздействия теста на устройство, тогда минимальный набор тестов $\left(\sum_{j=1}^{\ell} a_{i_j} \to min\right)$ имеет наименьшее время проверки работоспособности всех элементов устройства.

Другие интерпретации задачи о покрытии представлены в задачах теории графов и минимизации булевых функций.

I.2.2. Таблица покрытий.

Таблица покрытий T весьма удобна для представления исходных данных в задаче о покрытиях. Эта таблица - матрица T отношения принадлежности элементов множеств $A_i \in A$ опорному множеству B . Столбцы матрицы T сопоставлены элементам множества B , строки - элементам множества A :

$$T_{ij} = \begin{cases} 1, \text{ если } b_j \in B \text{ и } b_j \in A_i \ , \\ 0, \text{ если } b_j \in B \text{ и } b_j \notin A_i \ . \end{cases}$$

Нули в матрице T не проставляются. Для примера I.I матрица имеет следующий вид:

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	a_i
A_1	1		1			1	1			1
A_2	1	1		1				1		2,5
A_3		1			1		1			2
A_4			1					1	1	1,5
A_5				1	1			1	1	3
A_6			1		1	1	1		1	7
A_7		1		1		1				1

Рис.I.I

На языке таблицы покрытий задача о покрытии формулируется как задача построения таких подмножеств P_i строк таблицы покрытий, чтобы в совокупности строк, входящих в P_i , во всех столбцах были единицы /покрыты единицами все строки матрицы/. Такое подмножество строк называется покрытием.

В данном примере $P_1 = \{A_1, A_2, A_5\}$ - покрытие, в чем легко убедиться по следующей таблице

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9
A_1	1		1			1	1		
A_2	1	1		1				1	
A_5				1	1			1	1

Рис. I.2

По крайней мере хотя бы одна единица есть в каждом столбце. На этом примере видна наглядность представления информации на языке таблицы покрытий. Эта таблица позволяет достаточно просто проверять необходимое условие /I.I/. Вычерчивать соответствующий фрагмент таблицы покрытий для такой проверки не нужно. Таблица покрытий позволяет упростить процесс поиска покрытий, поэтому в дальнейшем используется именно это представление задачи о покрытии.

I.2.3. Решение задачи методом полного перебора.

Очевидным способом построения всех возможных покрытий служит метод перебора всех подмножеств строк таблицы покрытий: пустое подмножество строк, подмножества из одной строки, из двух строк, и т.д., из всех строк. Приходится рассматривать 2^m подмножеств. Их можно представить в виде следующего перечисления подмножеств:

$$\{\Phi, \{A_1\}, \ldots, \{A_m\}, \{A_1, A_2\}, \ldots, \{A_1, A_m\}, \{A_2 A_3\}, \ldots, \{A_2 A_m\}, \{A_3 A_4\}, \ldots$$
$$, \{A_{m-1}, A_m\}, \{A_1, A_2, A_3\}, \ldots, \{A_1, \ldots, A_m\}\} = D. \qquad /I.2/$$

В примере I.I приходится рассматривать $2^7 = 128$ подмножеств. Рассмотрим более простую таблицу покрытий /рис.I.3/.

<u>Пример I.2</u>.

	b_1	b_2	b_3	b_4	b_5
A_1	1		1		1
A_2	1	1		1	
A_3	1			1	1
A_4		1	1		

Рис. I.3

Множество D подмножеств D_i, которые строятся при полном переборе, будет следующим:

$$\{\varnothing, \{A_1\}, \{A_2\}, \{A_3\}, \{A_4\}, \{A_1, A_2\}, \{A_1, A_3\}, \{A_1, A_4\},$$
$$\{A_2, A_3\}, \{A_2, A_4\}, \{A_3, A_4\}, \{A_1, A_2, A_3\}, \{A_1, A_2, A_4\},$$
$$\{A_1, A_3, A_4\}, \{A_2, A_3, A_4\}, \{A_1, A_2, A_3, A_4\}\} = D.$$

Каждое $D_i \in D$ проверяется, является ли оно покрытием: подмножества $\varnothing$, A_1, A_2, A_3, A_4 очевидно не являются покрытиями;

$\{A_1, A_2\}$ – покрытие, совокупность этих двух строк содержит единицы во всех столбцах таблицы покрытия;

$\{A_1, A_3\}$ – нет единицы в столбце b_2;

$\{A_1, A_4\}$ – не содержит единицы в столбце b_4;

$\{A_2, A_3\}$ – не содержит единицы в столбце b_3;

$\{A_2, A_4\}$ – не содержит единицы в столбце b_5;

$\{A_3, A_4\}$ – покрытие;

$\{A_1, A_2, A_3\}$ – покрытие, причем избыточное, так как поглощает $\{A_1, A_2\}$;

$\{A_1, A_2, A_4\}$ – избыточное покрытие;

$\{A_1, A_3, A_4\}$ – избыточное покрытие;

$\{A_2, A_3, A_4\}$ – избыточное покрытие;

$\{A_1, A_2, A_3, A_4\}$ – избыточное покрытие.

Очевидно, что для более сложных случаев представлять множество всех подмножеств будет весьма трудно. Для проверки очередного подмножества на соответствие условию решения /I.I/ другие подмножества не нужны; поэтому достаточно на каждом шаге решения иметь одно подмножество и получать после рассмотрения очередного – другое подмножество, которое подлежит рассмотрению.

Алгоритм последовательного построения /генерации/ подмножеств должен:

гарантировать, что последовательным его применением можно построить, начиная с некоторого начального подмножества, все возможные подмножества;

не требовать построения подмножеств, которые уже построены;

быть простым и наглядным;

иметь критерий окончания перебора.

Таких алгоритмов может быть достаточно много, но все они так или иначе основаны на упорядочении перебора подмножеств. Упорядочение перебора позволяет удовлетворить требованиям I,2 и 4 в алгоритме генерации подмножеств. Можно, например, упорядочить подмножества по следующему признаку: сначала рассматривать все подмножества с i элементами, затем – с i +I элементами и т.д., / i = 0,I,..., m /. Нужно еще упорядочить само рассмотрение подмножеств с i элементами /пример такого перебора приведен в формуле /I.2/ /.

Можно предложить другой алгоритм полного перебора, более удобный для дальнейшего упрощения процесса решения. Будем считать, что подмножества A_i пронумерованы – $A_1, \dots A_m$ и, таким образом, упорядочены. Основная идея алгоритма заключается в следующем: сначала строятся все подмножества D_i, содержащие A_1, затем – содержащие A_2, но не содержащие A_1; если построено подмножество D_i, то за ним строятся подмножества D_{i+j}, целиком содержащие $D_i (D_i \subset D_{i+j})$. Сформулируем алгоритм.

0. Текущее подмножество $D = \{A_1\}$, $i = 0$.

I. $i = i + I$. Запоминаем множество D как очередное построенное подмножество D_i.

2. Находим наибольший номер j элемента $A_j \in D$. Если $j \neq n$, то переходим к п.3 алгоритма. Если $j = n$, то удаляем A_n из D и если $D = \phi$, то заканчиваем построение, если же $D \neq \phi$, то находим наибольший номер j элемента $A_j \in D$ и удаляем A_j из D.

3. $j = j + I$. Вводим в D элемент A_j. Переходим к п.I.

Выполнение алгоритма рассмотрим для примера I.2. Строится такая последовательность подмножества D_i:

$$D_1 = \{A_1\}, D_2 = \{A_1, A_2\}, D_3 = \{A_1, A_2, A_3\}, D_4 = \{A_1, A_2, A_3, A_4\},$$
$$D_5 = \{A_1, A_2, A_4\}, D_6 = \{A_1, A_3\}, D_7 = \{A_1, A_3, A_4\}, D_8 = \{A_1, A_4\},$$
$$D_9 = \{A_2\}, D_{10} = \{A_2, A_3\}, D_{11} = \{A_2, A_3, A_4\}, D_{12} = \{A_2, A_4\},$$
$$D_{13} = \{A_3\}, D_{14} = \{A_3, A_4\}, D_{15} = \{A_4\}.$$

I.2.4. Сокращение перебора при построении безызбыточных покрытий.

Во многих приложениях достаточно находить только безызбыточные покрытия /теорема I.2/. Тогда нужно видоизменить алгоритм так, чтобы не строить избыточные покрытия. Однако не удается найти простой и эффективный алгоритм, не требующий построения всех избыточных покрытий. Хорошо, если уменьшается их количество. Основа изменения алгоритма перебора подмножеств: если текущее подмножество – покрытие, то в это подмножество не нужно вводить новые элементы. Безызбыточные покрытия –

граница вогнутого множества всех покрытий /теорема I.I/, поэтому видоизмененный алгоритм называется <u>алгоритмом граничного перебора по вогнутому множеству</u>.

0. Текущее подмножество $D=\{A_1\}$, i = 0. Переходим к п.4 алгоритма.

I. Находим наибольший номер j элемента в подмножестве D . Рассматриваем следующие ситуации.

А. Если $j \neq n$ и D не покрытие, то переходим к п.3 алгоритма.

В. Если $j \neq n$ и D покрытие, то переходим к п.2 алгоритма.

С. Если $j = n$, то удаляем A_n из D, и если $D = \varnothing$, то переходим к п.5 алгоритма; иначе – снова находим наибольший номер j элемента в D .

2. Удаляем элемент A_j из D .

3. $j := j+1$. Вводим элемент A_j в D .

4. Исследуем, является ли очередное построение D покрытием; если нет, то переходим к п.I алгоритма, иначе – во-первых, удаляем из ранее построенных покрытий те, которые поглощают D /избыточные покрытия/, соответственно сокращая i , и, во-вторых, запоминаем D как покрытие P_i $(i = i+1;\ P_i = D)$. Переходим к п.I алгоритма.

5. Заканчиваем построение всех безызбыточных покрытий.

Для примера I.2 реализуется такая последовательность построения подмножеств D /подмножества D , являющиеся покрытиями, подчеркнуты одной чертой, безызбыточные покрытия – двумя чертами/.

$$D = \{A_1\},\ \underline{\underline{D_1 = \{A_1, A_2\}}},\ D = \{A_1, A_3\},\ \underline{D_2 = \{A_1, A_3, A_4\}};$$
$$D = \{A_1, A_4\},\ D = \{A_2\},\ D = \{A_2, A_3\},\ \underline{D_3 = \{A_2, A_3, A_4\}},$$
$$D = \{A_2, A_4\},\ D = \{A_3\},\ \underline{\underline{D_2 = \{A_3, A_4\}}},\ D = \{A_4\}.$$

Из пяти возможных избыточных покрытий получено только два.

Рассмотренный алгоритм граничного перебора достаточно прост и универсален, он применим для любых задач, в которых возможные решения составляют вогнутое множество. Однако сокращение перебора по этому алгоритму не достаточно, так как алгоритм граничного перебора мало учитывает специфику задачи. Используя представление информации таблицей, можно достигнуть большего сокращения процесса решения.

I.2.5. Вопросы и задачи к пп. I.2.I – I.2.4.

А. Найти другие интерпретации задачи о покрытии.

Б. Как понимается процесс сокращения перебора при решении задач на поиск оптимального решения?

В. Почему таблица покрытий удобна при решении задачи на ЭВМ? Какие операции ЭВМ лучше использовать при решении данной задачи?

Г. Построить минимальные покрытия

①	I	2	3	4	5	6	Q
А	I		I			I	2
Б	I			I	I		I
В	I				I	I	3
Г		I	I			I	4
Д		I	I	I			I

②	I	2	3	4	5	6	Q
А	I			I	I		I
Б	I	I	I				3
В		I		I		I	3
Г		I	I			I	5
Д			I		I	I	I

③	I	2	3	4	5	6	Q
А	I		I			I	I
Б		I		I		I	I
В	I	I			I		2
Г		I		I	I		5
Д			I	I	I		2

④	I	2	3	4	5	6	Q
А	I	I				I	2
Б	I			I			3
В		I		I	I		2
Г		I	I			I	I
Д	I		I		I		2

⑤	I	2	3	4	5	6	Q
А	I	I	I				3
Б			I	I	I		2
В	I		I	I			5
Г	I			I		I	2
Д		I			I	I	2

⑥	I	2	3	4	5	6	Q
А	I		I			I	2
Б			I	I	I		2
В		I			I	I	2
Г	I	I				I	4
Д	I	I		I			I

⑦	I	2	3	4	5	6	Q
А	I	I	I				3
Б		I	I		I		2
В	I			I	I		4
Г	I	I				I	2
Д			I	I		I	I

⑧	I	2	3	4	5	6	Q
А		I	I			I	2
Б	I			I		I	I
В	I		I		I		3
Г		I		I		I	3
Д		I		I	I		2

⑨	I	2	3	4	5	6	Q
А		I	I			I	2
Б	I			I		I	4
В	I		I		I		2
Г		I	I			I	3
Д		I		I	I		2

⑩	I	2	3	4	5	6	Q
А	I				I	I	2
Б		I	I		I		4
В		I		I		I	2
Г	I			I		I	3
Д	I		I	I			2

⑪	I	2	3	4	5	6	Q
А	I		I	I			3
Б	I		I		I		2
В				I	I	I	2
Г	I	I		I			2
Д		I	I			I	I

⑫	I	2	3	4	5	6	Q
А		I	I		I		I
Б	I	I		I			3
В				I	I	I	3
Г	I			I		I	5
Д	I		I			I	I

⑬	I	2	3	4	5	6	Q
А	I		I	I			2
Б	I			I	I		2
В	I		I			I	I
Г		I	I	I			2
Д		I			I	I	4

⑭	I	2	3	4	5	6	Q
А	I	I			I		4
Б	I	I		I			2
В			I	I	I		3
Г	I				I	I	I
Д		I	I			I	3

⑮	I	2	3	4	5	6	Q
А	I	I			I		2
Б		I	I	I			2
В		I	I		I		3
Г			I		I	I	2
Д	I			I		I	4

⑯	I	2	3	4	5	6	Q
А	I		I	I			4
Б	I	I		I			I
В			I	I		I	2
Г	I		I		I		2
Д		I			I	I	2

⑰	I	2	3	4	5	6	Q
А	I				I	I	4
Б	I		I			I	I
В	I			I	I		2
Г		I	I	I			2
Д		I			I	I	2

⑱	I	2	3	4	5	6	Q
А		I	I			I	I
Б	I		I	I			4
В	I	I			I		3
Г	I			I	I		5
Д				I	I	I	I

⑲	I	2	3	4	5	6	Q
А		I	I			I	4
Б			I	I		I	2
В	I	I	I				2
Г		I			I	I	I
Д	I			I	I		2

⑳	I	2	3	4	5	6	Q
А	I	I		I			2
Б			I	I	I		4
В		I			I	I	2
Г	I	I				I	3
Д	I		I			I	2

21	I	2	3	4	5	6	Q
А	I			I		I	2
Б	I		I	I			4
В	I		I		I		2
Г		I	I	I			2
Д		I			I	I	2

22	I	2	3	4	5	6	Q
А		I			I	I	I
Б			I		I	I	2
В		I	I	I			3
Г	I	I			I		2
Д	I			I		I	2

23	I	2	3	4	5	6	Q
А			I		I	I	4
Б	I		I		I		2
В	I			I		I	3
Г		I	I			I	I
Д		I		I	I		3

24	I	2	3	4	5	6	Q
А		I	I	I			3
Б		I			I	I	2
В	I		I			I	2
Г	I				I	I	4
Д	I			I	I		2

25	I	2	3	4	5	6	Q
А	I	I			I		4
Б		I			I	I	2
В	I	I	I				I
Г	I			I	I		2
Д			I	I		I	2

26	I	2	3	4	5	6	Q
А	I	I		I			2
Б	I				I	I	2
В	I	I			I		4
Г		I	I		I		2
Д			I	I		I	3

27	I	2	3	4	5	6	Q
А	I	I				I	4
Б		I	I		I		2
В	I			I	I		2
Г			I	I	I		3
Д			I	I		I	2

28	I	2	3	4	5	6	Q
А	I		I	I			2
Б				I	I	I	2
В	I	I			I		I
Г	I	I	I				3
Д		I	I			I	2

29	I	2	3	4	5	6	Q
А	I	I		I			2
Б	I		I		I		3
В		I	I			I	4
Г				I	I	I	I
Д	I				I	I	2

30	I	2	3	4	5	6	Q
А	I	I				I	2
Б		I		I	I		4
В			I		I	I	3
Г	I		I	I			2
Д	I		I			I	I

I.2.6. Сокращение таблицы покрытий до циклической.

Теоремы о сокращении таблицы покрытий. Объем перебора существенно зависит от размеров таблицы покрытий, поэтому изложенные в данном разделе способы вычеркивания некоторых строк и столбцов таблицы покрытий имеют большое значение для сокращения перебора.

Теорема I.3 /о ядре/. Если в столбце таблицы покрытий содержится единственная единица, то строка, имеющая эту единицу, входит во все покрытия. Эта строка называется ядерной.

Множество ядерных строк заранее выделяется и запоминается для введения во все покрытия. Ядерные строки из таблицы удаляются и вычеркиваются все покрытые ими столбцы, т.е. вычеркиваются все столбцы, в которых есть единицы в ядерных строках.

Теорема I.4 /об антиядре/. Если после удаления ядерных строк и покрытых ими столбцов в таблице покрытий в какой-либо строке не остается единиц, то эта строка не входит ни в одно безызбыточное покрытие. Такие строки называются антиядерными и они вычеркиваются из таблицы без запоминания.

Прежде, чем сформулировать следующие теоремы, напомним понятие поглощающего вектора: вектор E поглощает вектор F, $/ E \succcurlyeq F /$, если для всех компонент e_i , f_i этих векторов можно одновременно записать

$$e_i \geqslant f_i \cdot \qquad /I.3/$$

Примеры. E = IIOI, F = OIOI, S = IOII, T= IIII.
$E \geqslant F$, , $E \leqslant T$, вектора E и S , F и S - несравнимы.
E - поглощающий вектор для F .
E - поглощаемый вектор для T.

Теорема I.5 /о поглощающих столбцах/. В таблице покрытий могут быть вычеркнуты все поглощающие столбцы /рассматриваемые как векторы/ без ущерба для построения всех безызбыточных покрытий.

Теорема I.6 /о поглощаемых строках при поиске одного кратчайшего покрытия/. Если при решении задачи покрытия достаточно гарантировать получение хотя бы одного кратчайшего покрытия, то можно удалять все поглощаемые строки.

Теорема I.7 /о поглощаемых строках при построении минимальных покрытий/. Если при решении задачи покрытия достаточно гарантировать построение всех /хотя бы одного/ минимальных покрытий, то можно вычеркивать поглощаемую строку, если цена ее больше /или равна/ цены поглощающей строки.

Примеры использования теорем. Таблица покрытий для примера I.I /см. рис.I.I/ не упрощается: в ней нет ни ядерных, ни антиядерных строк, ни поглощаемых столбцов, ни поглощаемых строк. Таблица покрытий для примера I.2 /см. рис.I.3/ тоже упрощается очень мало: в ней только столбец b_1 можно вычеркнуть как поглощающий, т.е. $(b_1 \geqslant b_4$, $b_1 \geqslant b_5)$.

Рассмотрим более информативный пример I.3 /рис.I.4/.

В столбцах b_3 и b_4 по одной I в каждом, поэтому строки A_3 и A_5 , содержащие эти I, являются ядерными. Эти строки запоминаем для введения во все безызбыточные покрытия этой таблицы и после удаления этих строк и всех покрытых ими столбцов, таблица покрытий сократится до таблицы /рис. I.5/. Можно таблицу не перечерчивать, а вычеркнуть соответствующие строки и столбцы непосредственно на рис. I.4.

На рис. I.5 строка A_6 не содержит I, эта строка является антиядерной и ее вычеркиваем из таблицы.

Столбец $b_7 \geqslant b_2$, поэтому поглощающий столбец b_7 вычеркиваем. Прежде чем перейти к дальнейшим упрощениям еще раз перечертим оставшуюся таблицу.

На рис. I.6 строка A_2 поглощается строкой A_1 $(A_2 \leqslant A_1)$. Возможно несколько разветвлений процесса решения.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	a
A_1		1				1	1				3
A_2		1			1		1				2
A_3			1		1			1			2
A_4	1									1	3
A_5				1				1			3
A_6					1			1			3
A_7	1				1		1		1		2
A_8						1			1	1	1

Рис. I.4

	b_1	b_2	b_6	b_7	b_9	b_{10}	a	
A_1		1	1	1			3	
A_2		1		1			2	
A_4	1					1	3	$\{A_3, A_5\}$
v A_6							3	
A_7	1			1	1		2	
A_8			1		1	1	1	

Рис. I.5

	b_1	b_2	b_6	b_9	b_{10}	a	
A_1		1	1			3	
A_2		1				2	
A_4	1				1	3	$\{A_3, A_5\}$
A_7	1			1		2	
A_8			1	1	1	1	

Рис. I.6

	b_1	b_9	b_{10}	a	
A_4	1		1	3	
A_7	1	1		2	$\{A_1, A_3, A_5\}$
A_8		1	1	1	

Рис. I.7

I. Если строятся все безызбыточные покрытия, то никакие сокращения строк делать нельзя и таблица /рис. I.6/ далее не упрощается.

2. Если ищутся минимальные покрытия, то согласно теореме I.7 нужно обратить внимание на цены a_i строк: $A_2 \leq A_1$ и $a_2 < a_1$, поэтому строку A_2 нельзя вычеркивать и таблица также не упрощается.

3. Если ищем одно кратчайшее покрытие, то строку A_2 как поглощаемую, можно вычеркнуть. Тогда строка A_1 становится ядерной, так как в столбце b_2 останется единственная единица. Строка A_1 добавляется к множеству ядерных строк – $\{A_1, A_3, A_5\}$, таблица покрытий упрощается: вычеркивается строка A_1 и покрытые ею столбцы b_2 и b_6.

Оставшаяся таблица /рис. I.7/ не упрощается.

Алгоритм сокращения таблицы покрытий. Используя теоремы I.3 – I.7, можно упростить таблицу покрытий, запомнив ядерные строки. Возможны два исхода процесса решения.

I. Таблица покрытий после упрощений становится пустой – вычеркнуты все столбцы. В этом случае множество ядерных строк – требуемое покрытие.

2. Остаток таблицы покрытий более не упрощается приемами, предлагаемыми теоремами I.3-I.7. Получаем циклический остаток таблицы покрытий. Покрытия для циклического остатка таблицы можно строить только методами перебора покрытий: граничным перебором или разложением по столбцу /см. подразд. I.9/. Когда к ядерным строкам добавляются строки покрытия циклического остатка, получается покрытие исходной таблицы покрытий. Сформулируем алгоритм построения циклического остатка таблицы покрытий и множества ядерных строк для случая построения одного кратчайшего покрытия.

0. Считаем исходную таблицу покрытий текущей таблицей покрытий, а множество ядерных строк – пустым.

I. Находим ядерные строки, запоминаем множество ядерных строк. Текущую таблицу покрытий сокращаем: вычеркиваем ядерные строки и все столбцы, покрытые ими.

2. Вычеркиваем антиядерные строки.

3. Вычеркиваем поглощающие столбцы.

4. Вычеркиваем поглощаемые строки.

5. Если в результате выполнения пп. I-4 текущая таблица покрытий изменилась, снова выполняем п.I, иначе преобразования заканчиваем.

Подчеркиваем еще раз, что получаются два результата – множество ядерных строк и циклический остаток таблицы покрытий. Таблица покрытий примера I.I – циклическая, множество ядерных строк – пусто. Для примера I.3 множество ядерных строк содержит три элемента – $\{A_1, A_3, A_5\}$;

циклический остаток при построении одного кратчайшего покрытия показан на рис.I.7.

При сокращении таблицы покрытий до циклической при построении одного минимального покрытия изменяется только п.4 алгоритма, который формулируется следующим образом:

вычеркиваются поглощаемые строки, веса которых не меньше весов соответствующих поглощаемых строк.

Для примера I.3 в этом случае множество ядерных строк $\{A_3, A_5\}$, циклический остаток таблицы покрытий показан на рис. I.6.

При сокращении таблицы покрытий до циклической при условии построения всех безызбыточных покрытий п.4 алгоритма не выполняется: в этом случае нельзя вычеркивать никакие поглощаемые строки. Ответ для примера I.3 тот же, что и при построении одного минимального покрытия.

I.2.7. Вопросы и задачи к п. I.2.6.

А. Сформулируйте п.4 алгоритма при решении задачи построения всех минимальных покрытий.

Б. Таблица покрытий преобразовывалась при условии построения всех безызбыточных покрытий. В результате получились множество ядерных строк и пустая таблица покрытий. Что можно сказать о числе безызбыточных покрытий данной таблицы?

В. Сократить следующие таблицы покрытий с гарантией построения одного кратчайшего покрытия.

I	I	2	3	4	5	6	7	8	9	IO
А			I							I
Б	I	I					I		I	
В	I			I		I				I
Г			I	I	I					
Д						I			I	I
Е							I	I		
Ж		I	I		I					
И						I		I		

2	I	2	3	4	5	6	7	8	9	IO
А		I					I	I		
Б	I			I						
В						I			I	
Г		I	I						I	I
Д				I				I		
Е	I					I				I
Ж			I		I				I	
И	I				I	I	I			

3	I	2	3	4	5	6	7	8	9	IO
А	I	I						I		
Б		I	I		I		I			
В				I			I			
Г				I				I		
Д		I				I			I	I
Е			I				I			I
Ж	I				I	I			I	
И			I			I				

4	I	2	3	4	5	6	7	8	9	IO
А				I		I	I	I		
Б	I		I		I				I	
В				I	I	I				
Г		I								I
Д	I			I						
Е		I	I				I			
Ж	I							I	I	
И						I				I

5	I	2	3	4	5	6	7	8	9	IO
А				I	I		I	I		
Б			I	I		I				
В		I							I	
Г					I			I		I
Д			I					I		
Е						I			I	
Ж	I		I			I	I			
И	I	I								I

6	I	2	3	4	5	6	7	8	9	IO
А		I						I		
Б			I			I				
В	I	I			I					
Г				I				I	I	
Д	I			I				I		I
Е				I		I				
Ж		I			I		I			I
И			I				I		I	

7	I	2	3	4	5	6	7	8	9	IO
А	I						I	I		I
Б			I	I						
В					I			I	I	
Г						I				I
Д		I	I		I	I				
Е	I	I								I
Ж				I					I	
И			I			I	I			

8	I	2	3	4	5	6	7	8	9	IO
А	I			I				I		
Б	I		I			I			I	
В					I			I		
Г		I				I	I			
Д					I				I	
Е		I		I						
Ж			I	I				I		I
И		I					I			I

9	I	2	3	4	5	6	7	8	9	IO
А	I				I					
Б		I		I		I				
В	I	I						I	I	
Г			I		I		I			
Д			I		I				I	
Е	I					I	I	I		
Ж				I						I
И			I							I

IO	I	2	3	4	5	6	7	8	9	IO
А			I	I						
Б		I		I		I	I			
В					I	I				
Г		I	I					I	I	
Д	I				I					
Е				I		I				I
Ж	I		I				I	I		
И	I								I	I

II	I	2	3	4	5	6	7	8	9	IO
А	I							I		I
Б		I			I					
В				I			I			
Г		I		I		I				I
Д			I				I		I	
Е	I		I			I	I			
Ж					I			I		
И		I		I					I	

I2	I	2	3	4	5	6	7	8	9	IO
А				I					I	
Б		I					I			I
В	I		I		I					I
Г		I				I	I			
Д			I						I	
Е					I		I			
Ж	I			I				I		
И			I		I	I		I		

13	1	2	3	4	5	6	7	8	9	10
А			I	I		I	I			
Б		I						I	I	
В	I					I				I
Г	I	I							I	
Д				I			I	I		I
Е				I	I					
Ж							I		I	
И			I		I					

14	1	2	3	4	5	6	7	8	9	10
А	I		I				I			
Б	I			I						
В			I	I		I			I	
Г					I		I			
Д	I						I	I		
Е		I			I					
Ж				I				I	I	I
И		I				I				I

15	1	2	3	4	5	6	7	8	9	10
А	I				I	I				
Б		I		I						
В			I					I	I	I
Г	I						I			
Д				I					I	
Е		I				I	I			I
Ж		I	I				I			
И	I				I			I		

16	1	2	3	4	5	6	7	8	9	10
А			I		I				I	I
Б		I		I				I		
В	I		I					I	I	
Г		I					I			
Д	I				I	I				
Е		I		I						I
Ж						I	I			
И				I					I	

17	1	2	3	4	5	6	7	8	9	10
А			I					I		I
Б		I	I	I			I			
В				I		I	I			I
Г	I					I			I	
Д					I			I		
Е	I			I						
Ж	I	I							I	
И					I		I			

18	1	2	3	4	5	6	7	8	9	10
А	I	I	I			I				
Б				I	I					I
В						I		I		
Г		I			I	I			I	
Д	I						I		I	
Е		I		I						
Ж							I	I		
И			I	I						I

19	1	2	3	4	5	6	7	8	9	10
А	I		I							I
Б		I				I			I	
В			I	I	I					
Г		I								I
Д				I				I		
Е		I			I	I	I			
Ж	I						I		I	I
И						I		I		

20	1	2	3	4	5	6	7	8	9	10
А	I				I	I				I
Б		I	I				I			
В								I	I	I
Г			I		I	I		I		
Д						I	I			
Е				I					I	
Ж				I	I					
И	I	I					I			

21	1	2	3	4	5	6	7	8	9	10
А		I			I				I	
Б	I		I							
В					I		I			
Г	I							I		
Д		I	I	I			I			
Е				I		I		I		I
Ж					I				I	I
И			I			I	I			

22	1	2	3	4	5	6	7	8	9	10
А		I					I	I		I
Б					I				I	
В	I		I	I						
Г				I			I			
Д			I	I				I		
Е	I	I				I	I			
Ж			I		I					
И						I			I	I

23	1	2	3	4	5	6	7	8	9	10
А						I			I	
Б		I								I
В				I		I	I	I		
Г			I				I		I	
Д		I		I						
Е				I	I	I				
Ж	I		I						I	
И	I				I			I		I

24	1	2	3	4	5	6	7	8	9	10
А	I			I	I		I			
Б			I				I			
В	I					I				I
Г					I		I	I		
Д		I		I				I	I	
Е			I							I
Ж		I			I					
И		I				I			I	

25	1	2	3	4	5	6	7	8	9	10
А				I	I			I		I
Б				I			I			
В						I	I	I		
Г	I		I	I						I
Д		I							I	
Е			I			I	I			
Ж	I	I			I					
И						I			I	

26	1	2	3	4	5	6	7	8	9	10
А	I		I					I		
Б		I				I				
В						I				I
Г		I		I	I		I			
Д							I		I	I
Е	I			I						
Ж		I	I	I						
И	I				I			I	I	

27	1	2	3	4	5	6	7	8	9	10
А		I			I				I	
Б				I		I			I	I
В	I		I				I			
Г		I						I		
Д	I	I			I	I				
Е			I	I			I			
Ж			I		I					
И								I		I

28	1	2	3	4	5	6	7	8	9	10
А			I		I				I	
Б					I			I		
В				I			I		I	I
Г		I	I			I	I			
Д		I		I						
Е	I			I						I
Ж						I		I		
И	I	I				I				

4*

(29)	1	2	3	4	5	6	7	8	9	10
А	1			1		1		1		
Б			1						1	
В		1					1			
Г					1	1			1	
Д					1				1	1
Е			1	1			1			
Ж		1						1		
И	1		1				1			1

(30)	1	2	3	4	5	6	7	8	9	10
А			1		1					1
Б								1		1
В		1	1			1	1			
Г	1	1			1	1				
Д				1			1		1	
Е		1		1						
Ж						1		1		
И	1			1					1	

1.2.8. Построение покрытий методом разложения по столбцу.

Алгоритм разложения и дерево процесса решения. Основная идея метода разложения по столбцу заключается в следующем: каждый столбец таблицы покрытий должен быть покрыт и покрывается только за счет строк, содержащих единицу в этом столбце. Для упрощения процесса решения выбирается очередной столбец с меньшим числом единиц. В выбранном столбце находится несколько единиц и необходимо рассмотреть варианты его покрытия, в каждом из которых выбирается в покрытие одна строка, содержащая единицы в этом столбце. Строки выбираются в порядке уменьшения числа единиц в них. Выбранная в покрытие строка содержит единицы не только в покрываемом столбце, но и в ряде других, поэтому таблица покрытий сокращается: вычеркивается выбранная строка и все покрытые ею столбцы. В каждом очередном варианте не включаются строки, выбранные в покрытие для предыдущих вариантов разложения, получается свой остаток таблицы покрытий, к которому применяется алгоритм сокращения таблицы покрытий, выделяются ядерные строки и приписываются к выбранной для данного варианта строке.

Процесс решения фиксируется в виде дерева, в каждой вершине которого записаны множество введенных в решение строк и соответствующий циклический остаток. Начальной вершине /корню/ дерева приписаны циклический остаток и множество ядерных строк исходной таблицы покрытий. Из вершины выходит столько ветвей, сколько единиц в столбце, выбранном для разложения. Дерево достраивается разложением очередного циклического остатка, приписанного вершине, в которой разложение еще не проведено, и циклический остаток не пуст. Лист, в котором остаток пуст, называется конечным, для него разложение, естественно, не проводится. Конечные листы дерева решения нумеруются римскими цифрами.

Примеры построения дерева решения. Рассмотрим процесс разложения для примера I.2 /см. рис. I.3/. Алгоритм сокращения удаляет столбец $b_1 (b_1 \geqslant b_4)$. Циклический остаток показан на рис. I.8.

	b_2	b_3	b_4	b_5
A_1		1		1
A_2	1		1	
A_3			1	1
A_4	1	1		

Рис.I.8

Множество ядерных строк пусто. Все столбцы содержат по две единицы, поэтому выбираем первый – b_2. Ветвление ведется в два варианта по строкам A_2 и A_4. В каждом варианте остаток таблицы сокращается, выделяется соответствующая ядерная строка, присоединяемая к множеству строк при соответствующей вершине. Так как остатки таблиц покрытий после этого оказываются пустыми, процесс разложения заканчивается /рис. I.9/. Конечные вершины пронумерованы римскими цифрами.

Для примера I.I /см.рис. I.I/ дерево решения изображено на рис. I.IO без комментариев.

Построение покрытий по дереву решения. На рис. I.9 и I.IO показаны деревья решения, по ним нужно получить результат – множество покрытий. Для этого достаточно по каждой ветви дерева решения пройти от конечной вершины к корню, объединяя множества строк, введенных в покрытие на каждой пройденной вершине. Каждое такое множество – покрытие исходной таблицы покрытий. Для выделения оптимальных решений нужно привести подобные, удалить избыточные покрытия и выбрать из оставшихся оптимальные покрытия по соответствующему критерию – кратчайшие или минимальные. В примере /рис.I.9/ имеем две конечные вершины. Строим два покрытия:

1/ $\varnothing \cup \{A_1, A_2\} = \{A_1, A_2\}$;

2/ $\varnothing \cup \{A_3, A_4\} = \{A_3, A_4\}$.

Оба покрытия – кратчайшие. Объем перебора существенно сократился по сравнению с граничным перебором: построены только два покрытия, а не I2 подмножеств, проверяемых на покрытие.

В примере I.I /рис. I.IO/ конечных вершин более 5. Соответственно получаем покрытия:

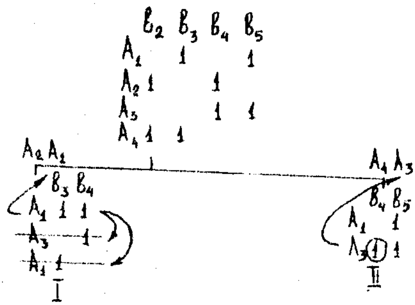


Рис. I.9

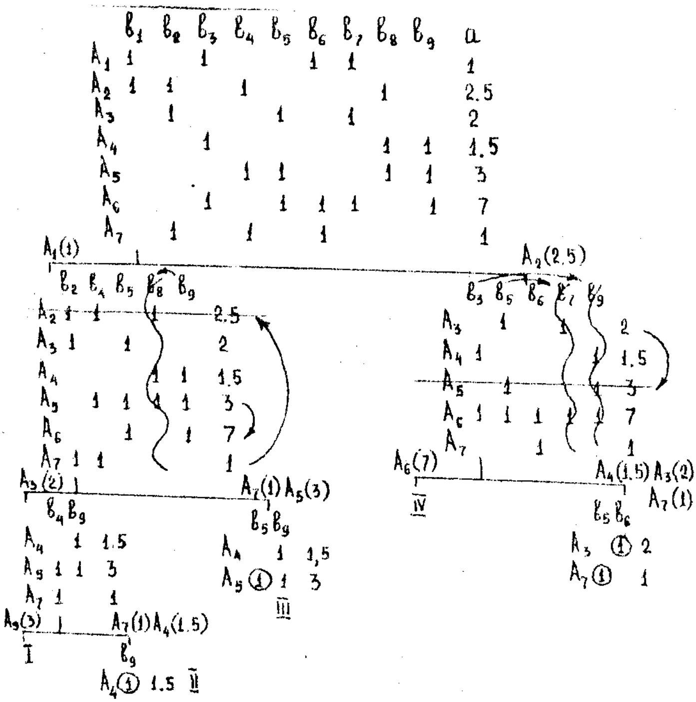


Рис. I.10

1/ $\{A_1\} \cup \{A_3\} \cup \{A_5\} = \{A_1, A_3, A_5\}$; $S_1 = \sum a_i = 1+2+3=6$;
2/ $\{A_1\} \cup \{A_3\} \cup \{A_4, A_7\} = \{A_1, A_3, A_4, A_7\}$; $S_2 = 1+2+1,5+1=5,5$;
3/ $\{A_1\} \cup \{A_5, A_7\} = \{A_1, A_5, A_7\}$, $S_3 = 1+3+1 = 5$;
4/ $\{A_2\} \cup \{A_6\} = \{A_2, A_6\}$, $S_4 = 2,5+7=9,5$;
5/ $\{A_2\} \cup \{A_3, A_4, A_7\} = \{A_2, A_3, A_4, A_7\}$; $S_5 = 2,5+2+1,5+1=7$.

Из построенных покрытий - одно минимальное, а именно 3/ $\{A_1, A_5, A_7\}$ с ценой 5. Так как проведен весь необходимый перебор, имеется гарантия, что это покрытие действительно минимально, а покрытие P_3, построенное в п. I.2.I, не минимально.

Краткая формулировка алгоритма построения покрытий методом разложения по столбцу.

I. Исходная таблица покрытий упрощается и полученные циклический остаток и ядерные строки приписываются начальной вершине /корню/ дерева решения.

2. Находится не конечная очередная вершина /лист/, при которой имеется не пустой остаток таблицы покрытий. Если таких вершин нет, то процесс построения дерева решения заказчивается и выполняется п.3, иначе находится столбец с меньшим числом единиц и проводится разложение по этому столбцу. В каждом варианте разложения остаток таблицы сокращается и, если оказывается пустым, то соответствующая вершина объявляется конечной. Снова выполняется п.2 алгоритма.

3. По дереву решений строятся покрытия. Их оценивают и выбирают наилучшие.

I.2.9. Вопросы и задачи к п.I.2.8.

А. На примере I.I. оцените объем перебора при решении: а/ полным перебором; б/ граничным перебором; в/ разложением по столбцу.

Б. Решите пример I.I без использования приемов сокращения таблицы покрытий, пользуясь только методом разложения по столбцу без сокращения таблицы. Насколько увеличилось дерево решения? Усложнился ли процесс решения?

В. Решите задачи I.2.5.Г при условии построения всех безызбыточных покрытий.

Г. Закончите решение задач о покрытии примеров I.2.7.В при условии построения одного кратчайшего покрытия.

Д. Решите следующие задачи о покрытии при условии построении одного минимального покрытия:

I	I	2	3	4	5	6	7	8	9	a
А			I	I			I			I
Б	I				I		I			I
В		I	I	I					I	2
Г		I		I	I		I	I		4
Д	I		I			I			I	3
Е					I			I	I	I
Ж		I				I		I		2

2	I	2	3	4	5	6	7	8	9	a
А				I	I		I			I
Б	I					I			I	2
В	I			I	I			I	I	3
Г			I					I	I	I
Д		I	I			I	I			3
Е	I	I		I		I	I			2
Ж		I			I			I		2

3	I	2	3	4	5	6	7	8	9	a
А		I	I		I			I		3
Б	I		I	I				I		2
В	I			I		I	I		I	4
Г		I				I			I	I
Д			I	I					I	2
Е	I						I	I		2
Ж					I	I	I			I

4	I	2	3	4	5	6	7	8	9	a
А		I		I				I		2
Б			I			I			I	I
В		I			I		I		I	3
Г	I			I			I		I	2
Д	I		I		I					I
Е	I		I	I		I		I		3
Ж						I	I	I		2

5	I	2	3	4	5	6	7	8	9	a
А		I					I	I		I
Б						I		I	I	2
В		I	I		I	I				3
Г			I	I					I	2
Д	I				I		I			I
Е	I		I	I		I				2
Ж	I			I			I	I	I	3

6	I	2	3	4	5	6	7	8	9	a
А	I			I		I	I		I	3
Б					I		I		I	I
В		I				I	I			2
Г		I	I		I			I		3
Д	I		I	I	I					2
Е	I					I		I		2
Ж			I	I					I	I

7	I	2	3	4	5	6	7	8	9	a
А			I					I		I
Б		I		I		I				2
В	I	I		I	I			I		3
Г	I			I		I			I	2
Д	I				I		I			I
Е		I			I				I	2
Ж			I			I	I		I	3

8	I	2	3	4	5	6	7	8	9	a
А		I				I		I		I
Б		I		I			I		I	3
В				I	I			I	I	2
Г			I			I			I	2
Д	I				I		I			I
Е	I		I	I						2
Ж	I		I		I	I		I		3

9	I	2	3	4	5	6	7	8	9	a
А	I			I					I	2
Б	I						I		I	I
В	I		I			I		I		3
Г					I		I	I		2
Д		I		I	I		I		I	4
Е		I	I	I						I
Ж		I			I	I				I

I0	I	2	3	4	5	6	7	8	9	a
А	I	I						I		I
Б		I		I	I		I	I		4
В		I		I		I				2
Г					I	I	I			I
Д			I				I	I		I
Е	I		I			I			I	3
Ж				I	I				I	2

II	I	2	3	4	5	6	7	8	9	a
А			I		I				I	I
Б		I	I			I	I			3
В		I	I	I					I	2
Г	I			I	I			I	I	3
Д					I	I		I		I
Е	I	I		I		I				2
Ж	I						I	I		2

I2	I	2	3	4	5	6	7	8	9	a
А					I		I	I		2
Б	I			I		I				I
В	I	I	I							I
Г		I				I	I		I	3
Д			I		I		I		I	2
Е	I		I	I	I			I		4
Ж				I				I	I	I

I3	I	2	3	4	5	6	7	8	9	a
А			I		I				I	I
Б		I	I	I				I		3
В			I	I			I		I	2
Г	I				I	I	I		I	4
Д					I	I		I		2
Е	I	I					I			I
Ж	I			I		I				I

I4	I	2	3	4	5	6	7	8	9	a
А	I	I					I			2
Б				I	I				I	2
В					I	I	I	I		2
Г		I				I			I	I
Д	I		I		I	I				3
Е			I	I				I		I
Ж	I			I			I	I	I	4

I5	I	2	3	4	5	6	7	8	9	a
А	I			I	I		I		I	4
Б	I		I			I	I			2
В	I			I				I		I
Г			I				I		I	I
Д				I	I	I				2
Е		I	I			I		I		3
Ж		I			I				I	I

I6	I	2	3	4	5	6	7	8	9	a
А		I			I	I				2
Б	I	I		I						I
В	I	I			I			I	I	4
Г			I					I	I	2
Д	I					I	I	I		2
Е					I		I		I	I
Ж			I	I		I	I			3

5 - 2-IOI8

(I7)	I	2	3	4	5	6	7	8	9	a
А		I	I	I			I	I		3
Б			I	I	I					I
В			I				I		I	2
Г	I	I		I						I
Д		I				I		I		I
Е					I	I	I	I		2
Ж	I				I	I			I	3

(I8)	I	2	3	4	5	6	7	8	9	a
А		I			I	I	I			3
Б		I						I	I	I
В	I		I				I			2
Г			I		I	I			I	2
Д	I		I	I				I	I	3
Е	I			I		I				I
Ж				I	I			I		I

(I9)	I	2	3	4	5	6	7	8	9	a
А	I			I					I	2
Б			I			I		I		I
В			I	I	I	I			I	3
Г	I			I	I		I			2
Д	I	I					I	I		3
Е			I				I		I	I
Ж		I			I	I				I

(20)	I	2	3	4	5	6	7	8	9	a
А			I	I			I			I
Б		I	I		I				I	2
В					I	I		I		2
Г	I			I	I				I	3
Д							I	I	I	I
Е	I	I				I				I
Ж		I	I			I	I	I		3

(21)	I	2	3	4	5	6	7	8	9	a
А	I			I				I		I
Б		I	I			I	I			2
В	I				I				I	I
Г			I		I	I				2
Д	I			I	I	I	I			3
Е		I	I					I	I	3
Ж		I		I			I			I

(22)	I	2	3	4	5	6	7	8	9	a
А				I			I	I	I	3
Б		I	I						I	2
В	I	I	I		I	I				3
Г			I	I	I			I		2
Д	I					I	I			I
Е	I			I	I					I
Ж		I				I		I		I

(23)	I	2	3	4	5	6	7	8	9	a
А					I		I		I	I
Б	I		I			I			I	3
В			I	I		I				2
Г	I	I					I			I
Д		I	I					I		2
Е		I		I	I		I	I		3
Ж					I	I		I		I

(24)	1	2	3	4	5	6	7	8	9	a
А			I			I			I	2
Б	I				I		I	I		2
В		I	I	I						I
Г	I			I	I				I	3
Д					I	I		I		2
Е	I	I					I			I
Ж		I	I			I	I	I		3

(25)	I	2	3	4	5	6	7	8	9	a
А			I		I				I	I
Б	I			I					I	2
В		I					I	I		I
Г	I		I		I			I		2
Д	I		I			I	I			3
Е		I		I	I			I	I	3
Ж		I		I		I				2

(26)	I	2	3	4	5	6	7	8	9	a
А		I	I				I	I		3
Б		I	I	I	I					2
В						I		I	I	I
Г	I			I	I	I			I	3
Д	I		I	I						2
Е	I	I				I				I
Ж					I		I		I	2

(27)	I	2	3	4	5	6	7	8	9	a
А						I	I		I	2
Б		I	I		I	I	I			3
В					I	I		I		I
Г	I			I				I	I	3
Д	I	I	I	I						2
Е			I	I			I			I
Ж	I	I			I					I

(28)	I	2	3	4	5	6	7	8	9	a
А	I	I				I				I
Б		I	I	I	I					2
В						I	I	I		I
Г		I	I				I		I	3
Д	I		I	I						2
Е	I			I	I	I		I		3
Ж					I			I	I	I

(29)	I	2	3	4	5	6	7	8	9	a
А		I			I			I		I
Б		I				I	I		I	3
В	I		I	I	I			I		3
Г			I	I			I			2
Д	I		I			I	I			2
Е	I			I		I				I
Ж					I			I	I	I

(30)	I	2	3	4	5	6	7	8	9	a
А	I				I				I	I
Б						I	I	I	I	2
В			I			I	I			I
Г		I	I					I		2
Д		I	I		I		I		I	3
Е	I			I		I				3
Ж		I		I	I					2

I.2.I0. Приближенное решение задачи о покрытии.

Замечания о приближенности решений задач дискретной математики. Если из постановки задачи видно, что даже сокращенный перебор приводит к очень трудоемкому процессу решения, то для получения ответа приходится отказываться от построения результата с наилучшей стоимостью /максимального или кратчайшего покрытия/, достаточно получить результат, удовлетворяющий необходимым условиям [условие /I.I/]. Оптимальности качества результата гарантий уже нет, но целесообразно получить не самый худший результат /хотя бы безызбыточное, лучше близкое к кратчайшему покрытие/. При этом, в ущерб качеству, необходимо значительно упростить процесс решения.

Метод минимального столбца - максимальной строки. Покрытие, близкое к кратчайшему, дает следующий простой алгоритм преобразования таблицы покрытий.

I. Исходная таблица считается текущей преобразуемой таблицей покрытий, множество строк покрытия - пусто.

2. В текущей таблице выделяется столбец с наименьшим числом единиц. Среди строк, содержащих единицы в этом столбце, выделяется одна с наибольшим числом единиц. Эта строка включается в покрытие, текущая таблица сокращается вычеркиванием всех столбцов, в которых выбранная строка имеет единицы. Если в таблице есть невычеркнутые столбцы, то выполняется п.2, иначе покрытие построено. Заметим, что при подсчете числа единиц в строке учитываются единицы в невычеркнутых столбцах.

Примеры приближенного решения задачи. Решение задачи /см. рис.I.3/ сводится к выбору столбца b_2 и строки A_2 как имеющей три единицы, после чего остаются только столбцы b_3 и b_5, по любому из которых выбирается строка A_1, покрывающая оба оставшихся столбца. Покрытие - $\{A_1, A_2\}$

Для задачи /см. рис. I.I/ процесс решения показан на рис. I.I, на котором цифрами отмечена последовательность выбора столбцов и строк. Цифры в кружочках указывают число единиц в строке, сокращенной к моменту рассмотрения таблицы.

		1)	2)			3)				
		$\cancel{b_1}$	$\cancel{b_2}$	$\cancel{b_3}$	$\cancel{b_4}$	b_5	$\cancel{b_6}$	$\cancel{b_7}$	$\cancel{b_8}$	b_9
1)	A_1	1 (4)!		1			1	1		
2)	A_2	1 (4)	1 (3)!		1				1	
	A_3		1 (2)			1 (1)		1		
	A_4			1					1	1
	A_5				1	1 (2)!			1	1
	A_6			1		1 (2)	1	1		1
	A_7		1 (2)		1		1			

Рис. I.II

Получено решение $\{A_1, A_2, A_5\}$, на одну строку большее кратчайшего покрытия. Кстати, если бы по первому рассматриваемому столбцу b_1 была выбрана строка A_2, имеющая то же, что и A_1 число единиц, то было бы получено кратчайшее покрытие.

Индивидуальное задание. Решить методом минимального столбца максимальной строки задачи пп. I.2.7.Г и I.2.9Д, сравнить с качеством результатов, полученных точными методами.

I.3. Метод ветвей и границ в задаче о покрытии

Как показано на примерах в предыдущем подразделе метод разложения по столбцу с применением сокращений таблицы покрытий значительно уменьшает объем вычислений по сравнению с граничным перебором. Можно еще сократить процесс решения, если ввести в этот метод оценку из метода ветвей и границ. Последний является довольно общим методом дискретной математики, применяемым для решения многих задач. Сначала рассмотрим этот метод без ориентации на конкретные задачи.

I.3.I. Основные идеи метода ветвей и границ.

Основу метода составляют три преобразования - разложение очередной рассматриваемой задачи на подзадачи, вычисление оценок качества решения подзадач и, наконец, организация и сокращение процесса рассмотрения подзадач.

Разложение задачи. Исходная задача или любая полученная из нее подзадача разлагается на несколько более простых подзадач при выполнении двух условий:

каждая подзадача имеет ту же формулировку и представление, что и исходная задача;

объединение результатов всех подзадач содержит все искомые результаты решения исходной задачи.

Разложение изображается в виде дерева процесса решения /см. рис. I.10/. Исходная задача помещена в корне дерева, подзадачи - на его листьях. Каждая подзадача проще исходной, и к ней можно снова применить разложение, пока этот процесс не приводит к получению результата решения на очередном листе дерева, тогда этот лист является конечным.

Функция оценки и требования к ней. Для каждой подзадачи вычисляется значение функции оценки, которая должна удовлетворять следующим двум требованиям:

значение оценки должно быть нижней границей значений функции качества результатов решения рассматриваемой подзадачи, т.е. значение оценки должно быть меньше или равно значению функции качества каждого из этих результатов /формулировка приведена для поиска результатов с минимумом функции качества, при поиске результатов с максимальным значением функции качества оценка должна быть верхней границей/. Таким образом, если получена оценка для рассматриваемой подзадачи, то гарантируется, что не найдется ни одного результата с качеством решения лучшим, чем значение оценки;

вычисление оценки значительно проще, чем построение всех возможных результатов рассматриваемой подзадачи.

Организация и сокращение процесса решения. Процесс решения исходной задачи сводится к ее разложению на подзадачи, оценке подзадач, выбору очередной подзадачи для дальнейшего разложения и сокращению дерева разложения за счет удаления тех подзадач, оценки которых хуже, чем значения функции качества уже полученных результатов. Приведем общую формулировку метода ветвей и границ.

А. Исходная задача вместе с оценкой приписывается корню дерева.

Б. Если есть еще вершины /не конечные листья дерева/, в которых решение еще не доведено до результата, то выбирается одна из них с минимальным значением функции оценки и выполняется п.В, иначе выполняется п.Г.

В. Производится разложение подзадачи, приписанной выбранной вершине. Процесс фиксируется построением новых листов дерева, каждому из которых приписывается новая, более простая, подзадача и ее оценка. Если в каком-нибудь варианте разложения подзадача упростилась до построения результата, то вместо оценки вычисляется значение функции качества этого результата, и тогда в дереве процесса решения удаляются все листья, значения оценок на которых больше /или равны при поиске хотя бы одного лучшего результата/, чем полученное значение функции качества. Выполняется п.Б.

Г. По дереву процесса решения формируются результаты с лучшими значениями функции качества.

Процедуры разложения и вычисления оценки не должны быть сложными, так как в процессе решения они выполняются многократно, но они должны быть достаточно эффективными, так как от этого зависит сложность дерева процесса решения. К сожалению, для каждого класса задач необходимо формулировать свои процедуры разложения и оценки, учитывающие особенности задач этого класса. В данном подразделе рассматриваются формулировки этих процедур при решении задачи о покрытии, в разделе теории графов будут рассмотрены соответствующие процедуры для задачи поиска гамильтонова контура минимальной длины.

1.3.2. Формулировка метода ветвей и границ для задачи о покрытии

Разложение по столбцу таблицы покрытий полностью удовлетворяет всем требованиям процедуры разложения метода ветвей и границ. Действительно, при разложении получаются подтаблицы, имеющие ту же форму и те же требования при решении, что и исходная таблица. В совокупности таблиц разложения содержатся все покрытия исходной таблицы, при использовании алгоритмов сокращения таблицы покрытий сохраняются все безызбыточные /минимальные, кратчайшие/ или хотя бы одно кратчайшее /минимальное/ в зависимости от применяемого алгоритма удаления поглощаемых строк таблицы покрытий. Процедура разложения таблицы покрытий по столбцу достаточно проста.

Для вычисления функций оценки можно предложить несколько процедур, различающихся по сложности.

Оценка 0 при построении кратчайших покрытий

А. $O^A =]\frac{n}{K}[+ \ell$, где $]x[$ – наименьшее целое, большее x; n – число столбцов в таблице покрытий; K – наибольшее число единиц в строке таблицы покрытий; ℓ – число строк, уже введенных в покрытие на пути к данному листу. Оценка достаточно груба, но вычисляется очень просто.

Б. Оценка $O^Б = O_g + \ell$ вычисляется при выполнении следующего неравенства:

$\sum_{i=1}^{O_g} K_i \geq n$, где K_i – число единиц в очередной строке с наибольшим числом единиц, оставшейся после удаления предыдущих выбранных строк.

Для примера, изображенного на рис.I.I: $O^A =]\frac{9}{5}[+ 0 = 2$. $O^Б$ вычисляется по следующей процедуре: сначала выбирается строка A_6 с $K_1 = 5$, затем любая из строк с $K_2 = 4$, и так как 5 + 4 = 9, то $O^Б$ = 2 + 0 = 2.

Для примера, изображенного на рис. I.I2.

$O^A =]\frac{9}{5}[+ 0 = 2$, $O^Б$ вычисляется по следующей процедуре: выбирается строка A_6 с K_1 = 5, затем любые две строки с K_2 = 3 и K_3 = 3, и так как 5 + 3 + 3 > 9, то $O^Б$ = 3. Очевидно, что $O^Б$ более точная оценка, ее вычисление не намного сложнее, чем вычисление O^A.

Оценка при поиске минимальных покрытий. Вычисление оценки для этого случая более громоздко, можно использовать оценки O^A или $O^Б$ и на их основе построить оценки O^{AM} и $O^{БM}$ как суммы O^A или $O^Б$ наименьших весов.

$O^{AM} = \sum_{i=1}^{O^A} a_i^{min} + \varphi_\ell$; $O^{БM} = \sum_{i=1}^{O^Б} a_i^{min} + \varphi_\ell$, где a_i^{min} – очередной наименьший вес оставшихся строк таблицы покрытий, после удаления ранее выбранных; φ_ℓ – сумма весов всех ℓ строк, уже вошедших в покрытие.

Для примера /см.рис.I.I/

$$O^{AM} = 1 + 1 = O^{БM}.$$

Для примера /рис.I.I2/

$$O^{AM} = 1+2=3\ , \quad O^{БM} = 1+2+2=5\ .$$

Реализация метода ветвей и границ для задачи о покрытии. Данный метод конкретизируется сформулированными процедурами разложения и оценки, в которых используются все приемы сокращения остатков таблицы покрытий для каждого варианта разложения. При построении покрытий по дереву процесса решения используется процедура прохода от конечных вершин к корневой, которая была сформулирована в алгоритме разложения таблицы покрытий по столбцу.

На рис.I.I3 изображено дерево процесса решения для примера I.I при использовании оценки $O^{БM}$.

I.3.3. Вопросы и задачи к подразд. I.3.

А. Какие особенности общей формулировки метода ветвей и границ приводят к необходимости требования, чтобы оценка была оценкой снизу?

Б. Решить задачи п. I.2.9.Д с использованием оценок $O^{Б}$ и $O^{БM}$ при поиске, соответственно одного кратчайшего и одного минимального покрытия.

I.4. Задача о совместимых подмножествах

Рассмотрим еще одну задачу, имеющую большое число различных интерпретаций, - построение максимальных подмножеств на выпуклом множестве.

I.4.I. Отношение совместимости и совместимые подмножества

Задано множество $X=\{x_1,\dots,x_n\}$ и бинарное отношение R несовместимости на множестве $X\,(R\subseteq X{\times}X)$: пара $(x_i,x_j)\in R$ означает, что элементы x_i и x_j несовместны, т.е. не могут одновременно быть включены в одно совместимое множество $Y\subseteq X$. Отношение R - антирефлексивно, т.е. для всех $x_i\in X$ пара $(x_i,x_j)\notin R$. Отношение R - симметрично, т.е. если пара $(x_i,x_j)\in R$, то и пара $(x_j,x_i)\in R$. Если исходное отношение R не симметрично, то его нужно дополнить до симметричного. Отношение R при небольшом n удобно задавать матрицей отношений:

$$R_{ij}=\begin{cases} 1, \text{ при } (x_i,x_j)\in R,\\ 0, \text{ при } (x_i,x_j)\notin R.\end{cases}$$

Подмножество $Y\subseteq X$ совместимо, если никакая пара элементов $x_i\in Y$ и $x_j\in Y$ не является несовместимой, т.е. $(x_i,x_j)\notin R$.

Теорема I.8. Если Y_k совместимое подмножество, то и любое $Y_l\subseteq Y_k$ - тоже совместимо, т.е. множество совместимых подмножеств - выпукло.

	1	2	3	4	5	6	7	8	9	a
A_1	1		1	1			1			1
A_2		1			1			1		3
A_3		1	1				1			2
A_4				1				1	1	3
A_5				1		1				2
A_6	1		1		1	1			1	5

Рис. I.12

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	a
A_1	1		1			1	1			1
A_2	1	1		1				1		2,5
A_3		1			1		1			2
A_4			1					1	1	1,5
A_5				1	1			1	1	3
A_6			1		1	1	1		1	7
A_7		1		1		1				1

$O^{Б} = 2+0=2$

$O^{БМ} = 1+1=2$

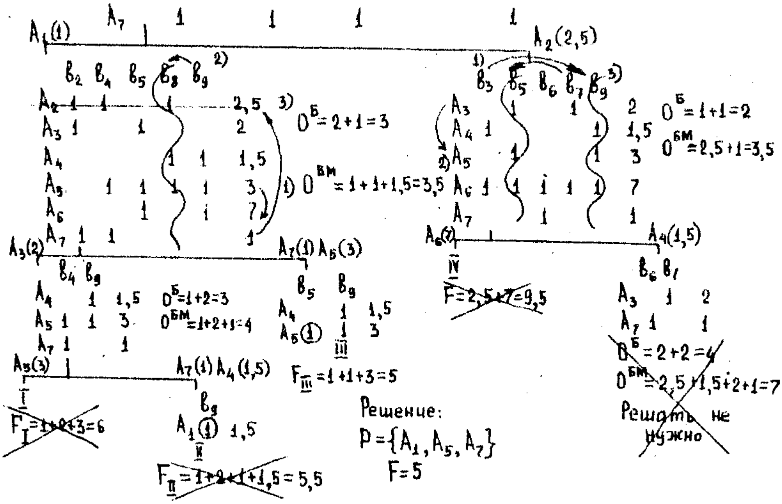


Рис. I.13

Согласно этой теореме достаточно строить <u>максимальные совместимые подмножества</u>.

<u>Пример 1.3.</u> $Y_1 = \{x_1, x_4\}$, $R_{1,4} = 0$, $R_{1,2} = R_{1,3} = R_{4,5} = 1$.

R	x_1	x_2	x_3	x_4	x_5
x_1	–	1	1	0	0
x_2	1	–	0	1	0
x_3	1	0	–	0	0
x_4	0	1	0	–	1
x_5	0	0	0	1	–

Таким образом, Y_1 – максимальное совместимое подмножество;

$Y_2 = \{x_2, x_3, x_5\}$ – то же максимальное совместимое подмножество.

Задача построения совместимых подмножеств возникает, например, при подборе коллектива космического корабля или набора цветов на одной клумбе. Другие интерпретации встретятся нам при построении внутренне устойчивых подмножеств вершин графа в задаче его раскраски или при построении максимальных интервалов в задаче минимизации булевых функций.

1.4.2. Алгоритм граничного перебора построения максимальных совместимых подмножеств

В п. 1.2.4 сформулирован алгоритм граничного перебора построения всех безызбыточных /минимальных/ покрытий на вогнутом множестве покрытий. Его модификация позволяет получить алгоритм граничного перебора на выпуклом множестве совместимых подмножеств.

0. Текущее подмножество $Y = \{x_1\}$, $i = 0$.

1. Находим наибольший номер j такой, что $x_j \in Y$. Рассматриваем ситуации:

А. Если $j \neq n$ и $i = 0$, то выполняем п.3.

Б. Если $j \neq n$ и $i \neq 0$, то выполняем п.2.

В. Если $j = n$, то удаляем элемент x_n из Y $(Y = Y \setminus \{x_n\})$, и если $Y \neq \emptyset$, то выполняем п.1, если $Y = \emptyset$, то выполняем п.6.

2. Удаляем элемент x_j из Y $(Y = Y \setminus \{x_j\})$.

3. $j = j+1$ и если $j > n$, выполняем п.4, иначе проверяем совместимость элемента x_j со всеми элементами множества Y, если хотя бы для одного элемента $x_l \in Y$ $(x_j, x_l) \in R$, то выполняем п.3, иначе, если для всех $x_l \in Y$ $(x_j, x_l) \notin R$, то вводим элемент x_j в Y, выполняем п.3.

4. Если $i = 0$, то выполняем п.5, иначе проверяем на поглощение множество Y с множествами Y_S, $1 \leq S \leq i$.

5. $i = i+1$, $Y_i = Y$, выполняется п.1. Если да, то выполняем п.1, иначе – п.5.

6. Построение всех максимальных совместимых подмножеств Y_S, $1 \leq S \leq i$ закончено.

Наиболее наглядно алгоритм граничного перебора выполняется в векторном представлении подмножеств Y :

$$\vec{Y} = Y^1 \dots Y^n, \quad \text{где} \quad y^k = \begin{cases} 1 & \text{при } x_k \in Y, \\ 0 & \text{при } x_k \in Y. \end{cases}$$

Проиллюстрируем выполнение алгоритма на примере I.4 /рис. I.I4/.

R	x_1	x_2	x_3	x_4	x_5
x_1	–	1	1		
x_2	1	–		1	
x_3	1		–		
x_4		1		–	1
x_5				1	–

x_1	x_2	x_3	x_4	x_5	
1			1		y_1
1				1	y_2
	1	1		1	y_3
	1			1	$-\ y \subseteq y_3$
		1	1		y_4
		1		1	$-\ y \subseteq y_3$
			1		$-\ y \subseteq y_1$
				1	$-\ y \subseteq y_2$

Рис. I.I4

Построены четыре максимальных совместимых подмножества:

$$y_1 = \{x_1, x_4\}, \quad y_2 = \{x_1, x_5\}, \quad y_3 = \{x_2, x_3, x_5\} \quad \text{и} \quad y_4 = \{x_3, x_4\}.$$

Других максимальных подмножеств не может быть, так как при организации решения перебор ведется "плотно", без пропусков подмножеств. Хотя при переборе получаются немаксимальные подмножества, но они сразу же обнаруживаются /$y \subseteq y_s$, $1 \le s \le i$ / и не запоминаются.

Рассмотрим еще один пример /рис. I.I5/.

В векторном представлении при переходе от предыдущего Y к последующему удаляется (если она есть) крайняя справа единица, на n-м месте затем находится крайняя справа единица /место j /, которая тоже удаляется, все оставшиеся единицы переписываются в новый вектор Y и по нему проверяется возможность включения переменных с номерами $j+1, \dots$ /это отмечено в примере стрелкой у крайней справа единицы в каждом предыдущем векторе Y /.

6*

R	1	2	3	4	5	6	7
1	–	1		1		1	
2	1		1				1
3		1	–		1	1	
4	1			–	1		
5			1	1	–		1
6	1		1			–	
7		1			1		–

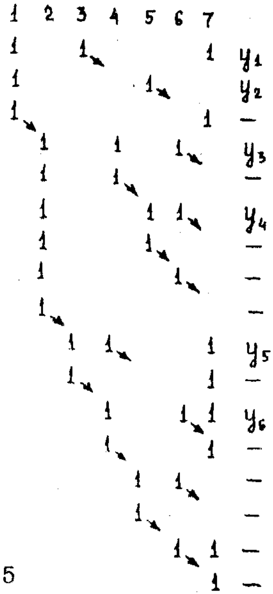


Рис. I.15

I.4.3. Вопросы и задачи к подразд. I.4.

А. Почему подмножество Y_1 максимально /см. рис. I.14/?

Б. Обратите внимание! Векторное представление подмножеств в алгоритме граничного перебора удобно при реализации алгоритма на ЭВМ.

В. Построить максимальные совместимые подмножества с использованием векторной реализации алгоритма граничного перебора

(I)

	I	2	3	4	5	6
I	–	I		I	I	
2	I	–	I			I
3		I	–	I		
4	I		I	–		I
5	I				–	I
6		I		I	I	–

(2)

	I	2	3	4	5	6
I	–		I			I
2		–	I	I		
3	I	I	–			I
4		I		–	I	I
5				I	–	I
6	I		I	I	I	–

(3)

	I	2	3	4	5	6
I	–	I			I	I
2	I	–		I		I
3			–	I	I	
4		I	I	–		I
5	I		I		–	
6	I	I		I	I	–

(4)

	I	2	3	4	5	6
I	-		I	I		
2		-	I		I	
3	I	I	-			I
4	I			-	I	
5		I		I	-	I
6			I		I	-

(5)

	I	2	3	4	5	6
I	-	I			I	
2	I	-		I	I	
3			-	I	I	I
4		I	I	-		
5	I	I	I		-	I
6			I		I	-

(6)

	I	2	3	4	5	6
I	-		I	I		I
2		-	I		I	
3	I	I	-		I	
4	I			-		I
5		I	I		-	I
6	I			I	I	-

(7)

	I	2	3	4	5	6
I	—			I	I	I
2		-	I		I	
3		I	-		I	I
4	I			-	I	I
5	I	I		I	-	
6	I		I	I		-

(8)

	I	2	3	4	5	6
I	-		I		I	I
2		-	I	I		
3	I	I	-			I
4		I		-	I	I
5	I			I	-	
6	I		I	I		-

(9)

	I	2	3	4	5	6
I	-	I			I	I
2	I	-		I		I
3			-	I	I	
4		I	I	-		
5	I		I		-	I
6	I	I			I	-

(I0)

	I	2	3	4	5	6
I	-	I				I
2	I	-		I	I	
3			-	I	I	I
4		I	I	-		I
5		I	I		-	
6	I		I	I		-

(II)

	I	2	3	4	5	6
I	-		I	I		I
2		-	I		I	
3	I	I	-			I
4	I			-	I	
5		I		I	-	I
6	I		I		I	-

(I2)

	I	2	3	4	5	6
I	-	I		I		I
2	I	-			I	I
3			-	I	I	
4	I		I	-		
5		I	I		-	I
6	I	I			I	-

(I3)

	I	2	3	4	5	6
I	-	I		I		I
2	I	-	I		I	I
3		I	-	I	I	
4	I		I	-	I	
5		I	I	I	-	
6	I	I				-

(I4)

	I	2	3	4	5	6
I	-		I		I	I
2		-	I	I	I	
3	I	I	-			I
4		I		-	I	
5	I	I		I	-	
6	I		I			-

(I5)

	I	2	3	4	5	6
I	-			I	I	
2		-	I	I	I	
3		I	-		I	I
4	I	I		-		
5	I	I	I		-	I
6			I		I	-

(16)

	I	2	3	4	5	6
I	-	I			I	
2	I	-		I		I
3			-	I		I
4		I	I	-		
5	I				-	I
6		I	I		I	-

(17)

	I	2	3	4	5	6
I	-	I				I
2	I	-		I	I	
3			-		I	I
4		I		-	I	I
5		I	I	I	-	
6	I		I	I		-

(18)

	I	2	3	4	5	6
I	-		I			I
2		-	I	I		I
3	I	I	-		I	
4		I		-	I	I
5			I	I	-	
6	I	I		I		-

(19)

	I	2	3	4	5	6
I	-		I		I	I
2		-	I	I		I
3	I	I	-	I		
4		I	I	-		I
5	I				-	I
6	I	I		I	I	-

(20)

	I	2	3	4	5	6
I	-	I		I	I	
2	I	-			I	
3			-	I		I
4	I		I	-	I	I
5	I	I		I	-	
6			I	I		-

(21)

	I	2	3	4	5	6
I	-	I		I	I	I
2	I	-		I		
3			-	I	I	
4	I	I	I	-		
5	I		I		-	I
6	I				I	-

(22)

	I	2	3	4	5	6
I	-	I	I			I
2	I	-		I		I
3	I		-	I	I	
4		I	I	-		
5			I		-	I
6	I	I			I	I

(23)

	I	2	3	4	5	6
I	-	I				I
2	I	-		I	I	
3			-	I	I	I
4		I	I	-		I
5		I	I		-	
6	I		I	I		-

(24)

	I	2	3	4	5	6
I	-	I			I	
2	I	-	I			I
3		I	-	I		I
4			I	-	I	
5	I			I	-	I
6		I	I		I	-

(25)

	I	2	3	4	5	6
I	-		I			I
2		-	I	I	I	
3	I	I	-			I
4		I		-		I
5		I			-	I
6	I		I	I	I	-

(26)

	I	2	3	4	5	6
I	-			I		I
2		-	I		I	
3		I	-		I	I
4	I			-		I
5		I	I		-	I
6	I		I	I	I	-

(27)

	I	2	3	4	5	6
I	-		I		I	
2		-	I	I		I
3	I	I	-	I		
4		I	I	-	I	
5	I			I	-	I
6		I			I	-

28	I	2	3	4	5	6
I	-	I			I	I
2	I	-	I	I		
3		I	-			I
4		I		-	I	I
5	I			I	-	
6	I		I	I		-

29	I	2	3	4	5	6
I	-			I	I	I
2		-	I		I	
3		I	-	I	I	
4	I		I	-		
5	I	I	I		-	I
6	I				I	-

30	I	2	3	4	5	6
I	-	I		I		I
2	I	-			I	I
3			-	I	I	
4	I		I	-		I
5		I	I		-	
6	I	I		I		-

I.5. Комбинаторные вычисления на конечных множествах

Комбинаторика - это характерная составляющая, грань дискретной математики, связанная с построением, изучением свойств и расчетом числа наборов элементов конечного множества, используемых при решении задач дискретной математики. Другие ее грани - модели дискретной математики, доказательства оптимизируемых построений и т.д. В рассмотренных задачах о покрытии, совместимых подмножествах к комбинаторике относятся свойства, связанные с организацией перебора вариантов решений, расчетом числа вариантов, оценкой сокращения процесса решений и т.д. Собственно комбинаторные аспекты - существенная составляющая всего материала данного учебного пособия, так как в нем большое внимание уделяется алгоритмическому подходу к решению задач дискретной математики, оптимизации перебора вариантов, сокращению числа рассматриваемых решений. В данном подразделе излагаются основные понятия, связанные с комбинаторными вычислениями - расчетом числа наборов, исходя из их свойств.

I.5.I. Основные свойства наборов.

Конечное множество с точки зрения комбинаторных вычислений определено числом n элементов: $X=\{x_1,\dots,x_n\}$, элементы x_i - различны. В конкретных комбинаторных задачах формулируются свойства, которым должны удовлетворять решения - наборы элементов конечного множества. Для каждой задачи эти свойства специфичны, однако выделяют три основных свойства наборов, которые следует учитывать во всех задачах: число элементов, возможность упорядочения элементов и допущение повторения элементов.

В зависимости от комбинации этих основных свойств определяют следующие шесть основных типов наборов комбинаторики.

Размещение без повторений. Набор, содержащий K различных элементов, выбранных из n-элементного множества, с учетом порядка расположения этих элементов, называется размещением без повторений.

Примеры.

А. K-значные числа /$K < 10$/, состоящие из различных цифр 0, 1, 2,...9:

$K = 2$; 01, 02,... 09, 10, 12,...19, 20, 21, 23, 29,...90, 91, 92,..., 98

Б. Трехбуквенные размещения на множестве букв А, Б, В, Г:
АБВ, АБГ, АВБ, АВГ, АГБ, АГВ, БАВ, БАГ, БВА, БВГ, БГА, БГВ, ВАБ, ВАГ, ВБА, ВБГ, ВГА, ВГБ, ГАБ, ГАВ, ГБА, ГБВ, ГВА, ГВБ.

Размещение с повторениями. Набор, содержащий K элементов n-элементного множества, построенный с учетом порядка расположения элементов и допускающий повторение элементов, называется размещением с повторениями.

А. K-значные числа десятичной системы счисления:

$K = 2$: 00, 01...,09, 10, 11,...19,...,90, 91,...,99.

Б. Трехбуквенные размещения на множестве букв А, Б, В, Г:
ААА,ААБ, ААВ, ААГ, АБА, АБВ, АБГ, АВА, АВБ, АВГ,..., ГГГ.

Перестановки без повторения. Набор, получаемый различным упорядочением /перестановкой/ элементов n-элементного множества, называется перестановкой.

Пример. Все возможные наборы, получаемые перестановкой букв множества А,Б,В,О:

АБВО, АБОВ, АВГО, АВОБ, АОБВ, БАВО, БАОВ, БВАО, БВОА, БОАВ, БОВА, ВАБО, ВАОБ, ВБАО, ВБОА, ВОАБ, ВОБА, ОАБВ, АОВБ, ОБАВ, ОБВА, ОВАБ, ОВБА.

Перестановки с повторениями. Набор, в котором каждый элемент x_i повторяется n_i раз и при построении которого учитывается порядок расположения различных элементов, называется перестановкой с повторениями.

Примеры. А. Перестановки, полученные различным расположением букв из исходного слова МАМА / $n_м = 2$, $n_А = 2$/: ММАА, МАМА, МААМ, АММА, АМАМ, ААММ.

Б. Перестановки из 0 и 1 при $n_0 = 2$, $n_1 = 3$:
00111, 01011, 01101, 01110, 10011, 10101, 10110, 11001, 11010, 11100.

Сочетания. Набор, содержащий K различных элементов n-элементного множества без учета порядка их расположения, называется сочетанием. Иначе, сочетания - это K-элементные подмножества n-элементного множества.

Примеры. А. Трехбуквенные сочетания на множестве АБВГ: $\{АБВ\}$, $\{АБГ\}$, $\{АВГ\}$, $\{БВГ\}$.

Б. Трехэлементные подмножества $\{1,2,3,4,5\}$: $\{1,2,3\}$, $\{1,2,4\}$, $\{1,2,5\}$, $\{1,3,4\}$ $\{1,3,5\}\{1,4,5\}$, $\{2,3,4\}$, $\{2,3,5\}$, $\{2,4,5\}$, $\{3,4,5\}$.

Сочетания с повторениями. Наборы, содержащие k элементов n-элементного множества без учета порядка и с разрешением повторений, называются сочетаниями с повторениями.

Примеры. А. Четырехбуквенные сочетания с повторениями, построенные по множеству $\{А, М\}$:
АААА, АААМ, ААММ, АМММ, ММММ.

Б. Четырехбуквенные сочетания с повторениями на множестве $\{0,1\}$: 0000, 0001, 0011, 0111, 1111.

1.5.2. Мощности множеств основных типов наборов.

При комбинаторных вычислениях используются два "золотых" правила комбинаторики.

Правило произведения. Если некоторый выбор A можно осуществить n вариантами, а для каждого из них некоторый другой выбор B можно реализовать m вариантами, то выбор A и B /в указанном порядке/ можно произвести $n \times m$ вариантами.

Правило числа элементов суммы множеств. Пусть $N(A)$ – мощность множества A , тогда для вычисления $N(A \cup B \cup C)$ используется правило включения и исключения:

$$N(A \cup B \cup C) = N(A) + N(B) + N(C) - (N(A \cap B) + N(A \cap C) + N(B \cap C)) + N(A \cap B \cap C).$$

Если множества A, B, C не пересекаются, то очевидно формула включения и исключения значительно упрощается

$$N(A \cup B \cup C) = N(A) + N(B) + N(C),$$
$$(A \cap B = A \cap C = B \cap C = \varnothing).$$

На рис.1.16 изображена классификация и приведены формулы расчета числа наборов для каждого из основных типов наборов. Приведем выводы этих формул.

Размещения: $$A_n^k = \frac{n!}{(n-k)!}.$$

Первый элемент может быть выбран n вариантами, так как повторения не разрешены, второй элемент может быть выбран $n-1$ вариантами, 3-й элемент – $n-2$ вариантами, k-й элемент – $n-k+1$ вариантами. По правилу произведения

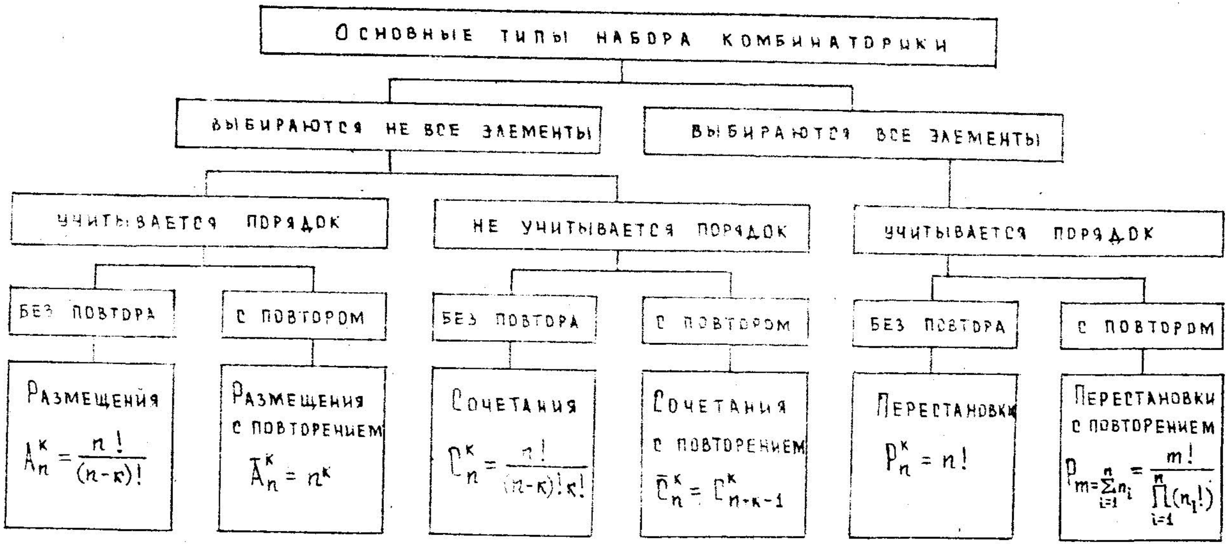

Основные типы набора комбинаторики
выбираются не все элементы
выбираются все элементы
учитывается порядок
не учитывается порядок
учитывается порядок
без повтора
с повтором
без повтора
с повтором
без повтора
с повтором
Размещения $A_n^k = \frac{n!}{(n-k)!}$
Размещения с повторением $\bar{A}_n^k = n^k$
Сочетания $C_n^k = \frac{n!}{(n-k)!k!}$
Сочетания с повторением $\bar{C}_n^k = C_{n+k-1}^k$
Перестановки $P_n^k = n!$
Перестановки с повторением $P_{m=\sum_{i=1}^{n} n_i} = \frac{m!}{\prod_{i=1}^{n}(n_i!)}$

Рис. I.16

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

Размещения с повторениями

Первый элемент может быть выбран n -вариантами, остальные, так как повторения разрешены, тоже n - вариантами:

$$\bar{A}_n^k = n \times n \times \ldots \times n = n^k .$$

Перестановки: $P_n = n!$

Очевидно $P_n = A_n^n = \frac{n!}{(n-n)!} = n!$ $\quad$ / $0! = 1$ по определению/.

Перестановки с повторениями:

$$P_{m=\sum_{i=1}^{q} n_i} = \frac{m!}{n_1! \times n_2! \times \ldots \times n_n!} .$$

Формула получена следующим последовательным рассуждением. Сначала все m элементов считаем различными, тогда $P_m = m!$, затем различие для первого элемента убираем, тогда перестановки, отличающиеся только порядком расположения первого элемента, очевидно, становятся не различимыми. Перестановок первого элемента $P_{n_1} = n_1!$, т.е.

$$P_m = n_1 + (m - n_1) = \frac{P_m}{P_{n_1}} = \frac{m!}{n_1!} .$$

Рассуждая аналогичным образом для 2-, 3-го и т.д. элементов, получаем формулу для перестановок с повторениями.

Сочетания: $\quad C_n^k = \frac{n!}{(n-k)!\,k!} .$

Из каждого сочетания можно построить $k!$ размещений перестановкой элементов, поэтому

$$C_n^k = \frac{A_n^k}{P_k} = \frac{n!}{(n-k)!\,k!} .$$

Сочетания с повторениями:

$$\bar{C}_n^k = C_{n+k-1}^k .$$

Каждое сочетание с повторениями полностью определяется, если указано сколько элементов $x_i\,(1 \le i \le n)$ в него входит. Поставим в соответствие набору последовательность нулей и единиц по следующему правилу: последовательно для каждого элемента запишем столько единиц, сколько элементов x_i входит в сочетание, между этими группами единиц вставляется один нуль.

Таким образом в последовательности будет K единиц и $n-1$ нуль. Различные перестановки $n_1 = K$ единиц и $n_0 = n-1$ нулей определяют различные сочетания с повторениями

$$\bar{C}_n^K = P_{K+n-1} = \frac{(n+K-1)}{K!\,(n-1)!} = C_{n+K-1}^K .$$

Четырехбуквенные сочетания с повторениями на множестве $\{A,M\}$:
$n = 2$, $K = 4$, $\bar{C}_2^4 = C_{2+4-1}^4 = C_5^4 = C_5^1 = 5$,

AAAA, AAAM, AMMM, MMMM, AAMM.

Наборы, которые строятся в соответствие сочетаниям с повторениями из K единиц и $n-1$ нулей при выборе формулы:
IIII0, III0I, II0II, I0III, 0IIII.

Для вычисления формул комбинаторики можно использовать таблицу факториалов:

n	$n!$	n	$n!$	n	$n!$
0	I	6	720	II	39916800
I	I	7	5040	I2	479001600
2	2	8	40320	I3	6227020800
3	6	9	372880	I4	87I78291200
4	24				
5	I20	I0	3628800	I5	I307674367000

При больших n для вычисления факториалов полезно использовать формулу Стирлинга:

$$n! \simeq \left(\frac{n}{e}\right)^n \sqrt{2\pi n}, \quad e \simeq 2,718\ldots .$$

I.5.3. Приемы решения комбинаторных вычислительных задач.

Решение комбинаторных задач - процесс творческий, тем не менее существуют несколько общих приемов, которые необходимо изучить для творческого использования при решении конкретных задач. Эти приемы в дальнейшем будут использоваться при выводе различных формул, расчете числа графов, числа булевых функций и т.д. Рассмотрим основные приемы на классических примерах.

Сведение исходной задачи к известной. Рассмотрим этот прием на примере задачи о ящиках. Имеется n ящиков, вмещающих каждый n_i предметов $(1 \le i \le n)$. Все $m=\sum n_i$ предметов различны, но имеют одинаковый объем. Найти число распределений предметов по ящикам. Для решения задачи распределим номера ящиков 1,2,..., n по предметам: очевидно, каждый номер i будет повторяться n_i раз. Получаем перестановки $m=\sum n_i$ номеров i с повторениями n_i, тогда их число

$$P_{m=\sum_{i=1}^{n} n_i} = \frac{m!}{\prod_{i=1}^{n}(n_i!)}.$$

Итак, при решении задачи о ящиках ответ получается сразу же, как только исходная постановка сведена к известной. Тот же прием фактически был применен при выводе формул числа перестановок без повторений, числа сочетаний с повторениями. Однако переформирование каждой исходной задачи - процесс творческий.

Разбиение исходной задачи на несколько известных взаимодействующих задач. Рассмотрим этот прием на примере расчета числа X возможных вариантов выбора президиума на собрании. Присутствует n человек, из них нужно выбрать K человек в президиум, включая председателя и секретаря собрания. Задачу разбиваем на две - выбор президиума /сочетание C_n^K / и выбор среди президиума председателя и секретаря/размещение A_K^2 /. Применяя правило произведения, получаем $X = C_n^K \times A_K^2$.

Тот же прием использовался при выводе формул числа сочетаний $\left(C_n^K = \frac{A_n^K}{P_K}\right)$, числа перестановок с повторениями.

Использование формулы включения и исключения. Формула включений и исключений в общем случае имеет следующий вид:

$$N\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} N(A_i) - \sum_{1 \le i_1 < i_2 \le n} N(A_{i_1} \cap A_{i_2}) + \dots$$
$$\dots + (-1)^K \sum_{1 \le i_1 < \dots < i_K \le n} N(A_{i_1} \cap A_{i_2} \cap \dots \cap A_{i_K}) + \dots$$
$$\dots + (-1)^{n-1} N(A_1 \cap A_2 \cap \dots \cap A_n).$$

Проиллюстрируем этот прием на примере расчета числа Y натуральных чисел на отрезке $(1,N)$, не делящихся ни на одно из заданных чисел $a_1, \dots, a_K$, где a_i - взаимно простые числа. Пусть A_i - множество натуральных чисел, не превышающих N и делящихся на a_i. Тогда $N(A_i) = \left[\frac{N}{a_i}\right]$, где $[x]$ - наибольшее целое, не превосходящее x.

Очевидно, что $A_{i_1} \cap A_{i_2} \cap \ldots \cap A_{i_s}$ - множество чисел, делящихся на $a_{i_1}, \ldots, a_{i_s}$ и $N(A_{i_1} \cap \ldots \cap A_{i_s}) = \left[\frac{N}{a_{i_1} \ldots a_{i_s}}\right]$. По формуле включений и исключений число X чисел, делящихся по крайней мере на одно из чисел $a_1, \ldots, a_k$ и не больших N,

$$X = \sum_{i=1}^{k}\left[\frac{N}{a_i}\right] + \sum_{1 \le i_1 < i_2 \le k}\left[\frac{N}{a_{i_1} a_{i_2}}\right] + \sum_{1 \le i_1 < i_2 < i_3 \le k}\left[\frac{N}{a_{i_1} a_{i_2} a_{i_3}}\right] + \ldots + (-1)^{k-1}\left[\frac{N}{a_1 a_2 \ldots a_k}\right].$$

Число $Y = N - X$.

Применение производящих функций для вывода формул комбинаторики. Производящая функция - аналитическая функция, в которой используются сочетания, размещения и т.д. Характерным примером служит известный бином Ньютона:

$$(1+x)^n = \sum_{i=0}^{n} C_n^i x^i.$$

Проиллюстрируем использование бинома Ньютона для вывода нескольких интересных формул и одновременно покажем прием использования производящих функций.

А. $x = 1, \quad (1+1)^n = \sum_{i=0}^{n} C_n^i, \qquad 2^n = \sum_{i=0}^{n} C_n^i.$

Б. $x = -1, \quad (1-1)^n = \sum_{i=0}^{n} C_n^i (-1)^i,$

$$0 = \sum_{i \in \text{чет}} C_n^i - \sum_{i \in \text{нечет}} C_n^i, \qquad \sum_{i \in \text{чет}} C_n^i = \sum_{i \in \text{нечет}} C_n^i = 2^{n-1}.$$

В. $(1+x)^n = (1+x)^m (1+x)^{n-m},$

$$\sum_{i=0}^{n} C_n^i x^i = \left(\sum_{i=0}^{m} C_m^i x^i\right)\left(\sum_{i=0}^{n-m} C_{n-m}^i x^i\right).$$

При $\quad x^k C_n^k = \sum_{i=0}^{k}\left(C_m^i \times C_{n-m}^{k-i}\right).$

В частности, при $m = 1$

$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$ - это формула - известный треугольник Паскаля.

Еще одна частность при $n=2k$, $m=k$:

$$C_{2k}^{k}=\sum_{i=0}^{k}C_k^i\,C_k^{k-i}\text{ , так как }\ C_k^{k-i}=C_k^i\ \text{ , то }\ C_{2k}^{k}=\sum_{i=0}^{k}(C_k^i)^2 .$$

Для сочетаний с повторениями используется функция $(1+x+x^2+\ldots)^n=$ $=\sum_{i=0}^{n}C_{n+i-1}^{i}x^i$, применяя которую можно получить формулы, где будет появляться C_{n+i-1}^{i} – формула для сочетаний с повторениями. Для вывода формул над производящими функциями выполняються различные операции – интегрирование, дифференцирование, приравнивание переменной к константам и т.д.

I.5.4. Задачи к подразд. I.5.

А. Каждый студент группы либо брюнет, либо любит математику, либо занимается спортом. В группе 20 брюнетов, из них 10 любят математику; брюнет, любящий математику и занимающийся спортом, всего один. Из 16 спортсменов группы 10 брюнетов. Всего любит математику 15 студентов, из них только 6 – спортсмены. Сколько студентов в группе?

Б. В азбуке Морзе – два знака: точка и тире. Символы представляются последовательностями точек и тире. Не используются последовательности длиннее, чем 5. Рассчитать число возможных символов, которые могут быть представлены азбукой Морзе.

В. Каждого из 24 студентов группы можно направить на практику в одно из 5 мест. Сколько имеется различных вариантов распределения?

Г. Сколько вариантов выбора 4 пирожных, если в магазине 7 типов пирожных?

ГЛАВА 2. ЗАДАЧИ ОПТИМИЗАЦИИ НА ГРАФАХ

2.1. Краткие сведения об основных определениях

Графом (G,X), $X=\{x_1,\ldots,x_n\}$ называется антирефлексивное отношение $G\subseteq X\times X$, которое рассматривается в следующей интерпретации: элементы $x_i\in X$ – вершины графа, элементы $(x_i,x_j)\in G$ – дуги графа. Название "граф" происходит от его наглядного графического представления: вершины изображаются точками, дуги – линиями со стрелками /от вершины x_i к вершине x_j/.

G	x_1	x_2	x_3	x_4
x_1	–	0	I	I
x_2	I	–	0	I
x_3	0	I	–	I
x_4	I	0	I	–

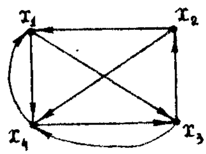


Рис.2.I

На рис.2.I заданы матрица отношения G и графическое представление (G,X). Если допускаются дуги $(x_i,x_i)\in G$, то граф называется графом с петлями. Вершины, соединенные дугой, называются смежными. Дуга $/x_i,x_j/$ и вершины x_i и x_j называются инцидентными. В графе, показанном на рис.2.I, все вершины смежны, вершине x_1 инцидентны дуги (x_2,x_1), (x_1,x_3), (x_1,x_4) и (x_4,x_1). Две встречные дуги (x_i,x_j) и (x_j,x_i) изображаются одной линией без стрелки, которая называется ребром.

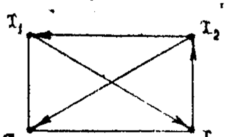


Рис.2.2

На рис.2.2 встречные дуги (x_4,x_3), (x_3,x_4) и (x_1,x_4), (x_4,x_1) заменены ребрами. Если отношение G симметрическое, то в графе есть только ребра и такой граф называется неориентированным. Если G – несимметрическое отношение, то в графе есть дуги /и ребра/, такой граф называется ориентированным /см. рис. 2.2/. В ориентированном графе для каждой вершины x_i определяются А/ $G(x_i)$ - исход вершины, т.е. множество вершин x_j таких, что $(x_i, x_j) \in G$, Б/ $G^{-1}(x_i)$ - вход вершины, т.е. множество вершин x_j, таких, что $(x_j, x_i) \in G$. Для графа /рис. 2.2/ $G(x_1) = \{x_3, x_4\}$, $G^{-1}(x_1) = \{x_2, x_3\}$; $G(x_4) = \{x_1, x_3\}$, $G^{-1}(x_4) = \{x_1, x_2, x_3\}$.

Мощности $S(x_i) = |G(x_i)|$ и $S^{-1}(x_i) = |G^{-1}(x_i)|$ множеств исхода и входа вершины x_i называются степенью $S(x_i)$ исхода и степенью $S^{-1}(x_i)$ входа. Для графа /рис. 2.2/ $S(x_1)=2$, $S^{-1}(x_1)=2$; $S(x_4)=2$, $S^{-1}(x_4)=3$. Для неориентированного графа очевидно $G(x_i) = G^{-1}(x_i)$ и тогда определяется степень $S(x_i) = |G(x_i)|$ вершины x_i. Очевидно, степень исхода /входа/ вершины равна числу дуг, исходящих /входящих/ из вершины. Для неориентированного графа степень вершины равна числу ребер, инцидентных ей.

Граф (G_1, X) называется частичным для графа (G, X), если $G_1 \subset G$, т.е. частичный граф получается из графа (G, X) удалением некоторых дуг. Граф (G_1, X_1) называется подграфом (G, X), если $X_1 \subset X$ и G_1 получается из G удалением дуг, инцидентных удаленным вершинам. Например, на рис. 2.3 показаны частичный граф /a/ и подграф /$б$/ графа (G, X) на рис.2.2.

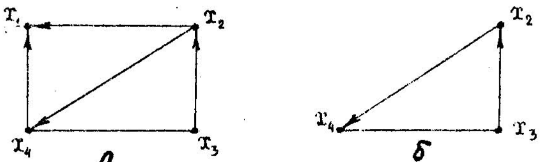


Рис. 2.3

Кроме перечисленных способов задания графа /матрица отношения G, графическое представление/ часто используются еще два: матрица инциденций и множества X и G. Матрица инциденций, имеющая n строк и m столбцов, где $m = |G|$ - число дуг, определяется следующим образом. Множество дуг $(x_i, x_j) \in G$ нумеруются $(1 \le \ell \le m)$:

$$M_{i\ell} = \begin{cases} 1, & \text{если } x_i \text{ есть начало } \ell\text{-й дуги } (x_i, x_j)_\ell; \\ 0, & \text{если } x_i \text{ не инцидента } \ell\text{-й дуге } (x_k, x_j)_\ell; \\ -1, & \text{если } x_i \text{ есть конец } \ell\text{-й дуги } (x_j, x_i)_\ell. \end{cases}$$

Например, если дуги графа /см. рис. 2.1/ пронумерованы

$$\begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ (x_1 x_3) & (x_1 x_4) & (x_2 x_1) & (x_2 x_4) & (x_3 x_2) & (x_3 x_4) & (x_4 x_1) & (x_4 x_3), \end{array}$$

то матрица инциденций имеет вид

	I	2	3	4	5	6	7	8
x_1	I	I	-I	0	0	0	-I	0
x_2	0	0	I	I	-I	0	0	0
x_3	-I	0	0	0	I	I	0	-I
x_4	0	-I	0	-I	0	-I	I	I

В представлении множествами X и G граф на рис. 2.I имеет следующий вид: $X=\{x_1, x_2, x_3, x_4\}$,

$$G=\{(x_1 x_3),(x_1 x_4),(x_2 x_1),(x_2 x_4),(x_3 x_2),(x_3 x_4),(x_4 x_1),(x_4 x_3)\}.$$

Это представление удобно, если в графе относительно мало дуг /ребер/.

Граф называется <u>взвешенным</u>, если задана функция ℓ_{ij} на дугах графа; например, ℓ_{ij} – длина дуги (x_i, x_j). Чаще всего $\ell_{ij} \geqslant 0$, во многих приложениях ℓ_{ij} – целочисленна. Функцию ℓ_{ij} можно задать в матричном виде:

$$\ell_{ij}=\begin{cases} \ell_{ij} & \text{при } (x_i, x_j) \in G; \\ \infty & \text{при } (x_i, x_j) \notin G. \end{cases}$$

В графическом представлении вес дуги изображается числом на дуге /рис. 2.4/.

ℓ	x_1	x_2	x_3	x_4
x_1	∞	∞	2	5
x_2	3	∞	∞	1
x_3	∞	4	∞	5
x_4	3	∞	4	∞

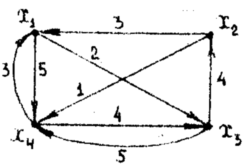


Рис. 2.4

Отметим, что графы – удобная модель представления данных почти во всех областях знаний /блок-схемы и сложные структуры данных в программировании, графы автомата в теории дискретных устройств, графы состояний в теории управления, схемы соединений в электронике, марковские цепи в теории случайных процессов, структуры молекул в химии и т.д./.

<u>Задание к подразделу</u>. Построить графы по матрицам отношений задания I.4.3.В, матрицы инциденций, исходы и входы вершин x_1 и x_4, подграфы, полученные удалением этих вершин.

8 – 2-IOI8

2.2. Кратчайший путь

2.2.1. Пути в графах

Последовательность дуг, начинающаяся в вершине $x_н \in X$, заканчивающаяся в вершине $x_к \in X$ и такая, что конец очередной дуги является началом следующей, называется путем из вершины $x_н$ к вершине $x_к$: $(x_н x_{i_1})(x_{i_1} x_{i_2})(x_{i_2} \dots x_{i_\ell})(x_{i_\ell} x_к) =$ путь $(x_н x_к)$. Если в пути $(x_н, x_к)$ вершины не повторяются, то путь называется элементарным, если не повторяются дуги - простым. В неориентированном графе пути соответствует маршрут - последовательность ребер, составляющих как и путь цепочку. Во взвешенном графе длиной пути называется сумма ℓ весов его дуг. Если граф не взвешен, то можно считать веса дуг равными I. Кратчайшим путем между выделенной парой вершин $x_н$, $x_к$ называется путь, имеющий наименьшую длину среди всех возможных путей между этими вершинами /рис. 2.5/.

Пример.

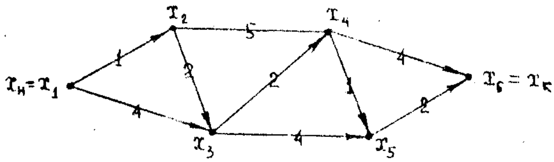


Рис. 2.5

Возможные пути между вершинами $x_н = x_1$ и $x_к = x_6$:

I $(x_1 x_2)(x_2 x_4)(x_4 x_6)$;
$\ell = 1 + 5 + 4 = 10$

II $(x_1 x_3)(x_3 x_5)(x_5 x_6)$;
$\ell = 4 + 4 + 2 = 10$

III $(x_1 x_2)(x_2 x_3)(x_3 x_5)(x_5 x_6)$;
$\ell = 1 + 2 + 4 + 2 = 9$ и т.д.

Очевидно, что из просмотренных путей меньшую длину $\ell = 9$ имеет третий, но является ли он кратчайшим? Конечно, для такого небольшого графа можно еще перебирать пути и выбрать самый короткий, но для больших и сложных графов нужен метод, сокращающий перебор. Задача о кратчайшем пути в графе имеет множество интерпретаций:

А/ расстояние между геометрическими точками;

Б/ время реализации программы между выделенными метками;

В/ стоимость передачи информации по каналам связи между заданными пунктами в сети связи, и т.д.

2.2.2. Волновой алгоритм построения кратчайшего пути в невзвешенном графе

При невзвешенном графе длина пути – это число дуг, входящих в него.

<u>Алгоритм построения кратчайшего пути</u> основан на пошаговом распространении волны от вершины начала пути по графу до тех пор, пока волна не коснется вершины конца пути, и затем выделяются дуги, по которым волна дошла до конечной вершины.

0. Вершина $x_н$ помечается нулем, остальные считаются непомеченными; $i=0$.

1. $i=i+1$. Помечаются индексом i все не помеченные ранее вершины, в которых есть дуги от помеченных вершин. Если помечена вершина $x_к$, то выполняется п.2, иначе, если текущим значением индекса i оказались помеченными какие-либо вершины, то выполняется п.1, иначе делается вывод, что пути из вершины $x_н$ в вершину $x_к$ нет.

2. Обратным проходом по дугам, начиная от вершины $x_к$, выделяются те дуги и вершины, которые инцидентны выделенным вершинам и разность между весами которых равна 1. При движении от вершины $x_н$ по выделенным дугам оказываются построены все кратчайшие пути к вершине $x_к$ /рис. 2.6/.

<u>Пример</u>.

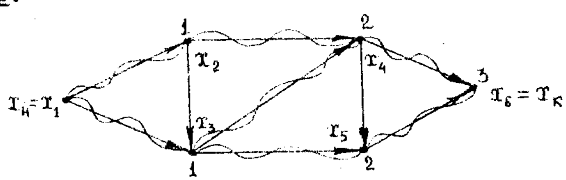


Рис. 2.6

Вершина x_1 помечена 0, вершины x_2 и x_3 помечаются 1, затем вершины x_4 и x_5 – помечаются 2 и, наконец, вершина $x_6 = x_к$ помечается i = 3. Обратный проход: вершина x_6 выделена, разности весов пар вершин x_4, x_6 и x_5, x_6 равны 1, выделяются дуги / x_4, x_6 / и / x_5, x_6 / и вершины x_4 и x_5 . От этих выделенных вершин согласно их входам вычисляются разности весов пар вершин $x_4 x_2$, $x_4 x_3$ и $x_5 x_3$, которые равны единице, и выделяются дуги / x_2, x_4 /, / x_3, x_4 / и / x_3, x_5 / и вершины x_2 и x_3 .Окончательно разность весов пар вершин $x_2 x_1$ и $x_3 x_1$ равна 1, выделяются дуги / $x_1 x_2$ / и / $x_1 x_3$ /. Двигаясь от вершины x_1 по выделенным дугам, получаем три кратчайших пути:

I/ / x_1, x_2 /, / x_2, x_4 /, / x_4, x_6 /; II/ / x_1, x_2 /, / x_3, x_4/, / x_4, x_6 /;
III/ / x_1, x_3 /, / x_3, x_5/, / x_5, x_7 /.

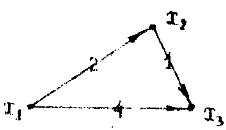
Длины ℓ всех путей одинаковы и равны числу i шагов распространения волны / $\ell = i = 3$/.

2.2.3. Волновой алгоритм для взвешенного графа.

Отметим особенности построения кратчайшего пути для взвешенного графа волновым алгоритмом.

I. Вместо единичного изменения индекса i /номер шага распространения волны/ при расчете пройденного пути для каждой вершины должна использоваться сумма весов ℓ_{ij} дуг / x_i, x_j /, ведущих к этой вершине от вершины $x_н$. К каждой вершине могут идти от вершины $x_н$ несколько путей, суммы длин дуг по разным путям различны. При поиске кратчайшего пути необходимо выбирать меньшую сумму. Волны распространения веса по разным путям доходят до каждой вершины последовательно, при очередной волне необходимо либо оставить старый вес вершины, либо заменить его на новый /меньший/. Поэтому при расчете веса вершины x_i за счет волны, подошедшей к ней по дуге / x_j, x_i /, вычисление веса V_i определяется по формуле $V_i = min\ (V_i, V_j + \ell_{ij})$.

<u>Примеры</u>. Для графа на рис. 2.7



Шаг 0: $V_1 = 0$.

Шаг I: $V_2 = 0+2=2$,

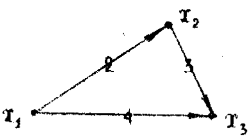
$V_3 = 0+4=4$.

Шаг 2:

$V_3 = min/4, 2+1/=3$.

Рис. 2.7

Для графа на рис. 2.8



Шаг 0: $V_1 = 0$.

Шаг I: $V_2 = 0+2=2$,

$V_3 = 0+4=4$.

Шаг 2: $V_3 = min/4, 2+3/=4$.

Рис. 2.8

2. Веса вершин в процессе распространения волны могут изменяться неоднократно. При каждом изменении веса V_i вершины это уменьшение веса необходимо передать вершинам исхода x_i, т.е. необходимы специальные средства, отражающие факты получения вершиной нового веса и передачу его другим вершинам. В качестве такого средства используется мас-

сив номеров вершин, получивших новый вес /при каждом изменении веса номер вершины включается в этот массив, если его там не было, при передаче веса - исключается/.

Формулировка алгоритма.

I. Вершина $x_н$ получает вес $V_н=0$, ее номер вводится в массив M номеров вершин, изменивших вес. Остальные вершины x_i получают вес $V_i=\infty$ и их номера не попадают в массив M.

2. Если массив M пуст, то выполняется п.3, иначе выбирается с исключением из него очередная вершина x_i и пересчитываются веса вершин, принадлежащих исходу $G(x_i)$ вершины x_i:

$$\forall x_j \in G(x_i)(V_j = \min(V_j, V_i + \ell_{ij})).$$

Если вес V_j уменьшается, то номер j включается с приведением подобных в M. Снова выполняется п.2.

3. Если вес $V_к=\infty$, то делается вывод, что пути из вершины $x_н$ к вершине $x_к$ нет, иначе выполняется процедура выделения дуг, такая же, как в волновом алгоритме для невзвешенного графа, за исключением того, что разность весов вершин x_j и x_i должна быть равна ℓ_{ij}. После выделения дуг строятся кратчайшие пути, длины которых равны $V_к$.

Рассмотрим рис. 2.5.

I: $V_н = V_1 = 0$, $V_2 = V_3 = V_4 = V_5 = V_6 = \infty$, $M = \{1\}$.

2: $M \neq \phi$, $i=1$, $M=\phi$, $G(x_1) = \{x_2, x_3\}$;
$V_2 = \min(\infty, 0+1) = 1$, $M=\{2\}$;
$V_3 = \min(\infty, 0+4) = 4$, $M=\{2,3\}$.

2: $M \neq \phi$, $i=2$, $M=\{3\}$, $G(x_2) = \{x_3, x_4\}$;
$V_3 = \min/4, I+2/=3$, $M=\{3\}$;
$V_4 = \min/\infty, I+5/=6$, $M=\{3,4\}$.

2: $M=\phi$, $i=3$, $M=\{4\}$, $G(3) = \{x_4, x_5\}$;
$V_4 = \min/6, 3+2/=5$, $M=\{4\}$;
$V_5 = \min/\infty, 3+4/=7$, $M=\{4,5\}$.

2: $M=\phi$ $i=4$, $M=\{5\}$, $G(x_4)=\{x_5, x_6\}$;

$V_5=\min/7, 5+1/=6$, $M=\{5\}$;

$V_6=\min/\infty, 5+4/=9$, $M=\{5,6\}$.

2: $M\neq\phi$, $i=5$, $M=\{6\}$, $G(x_5)=\{x_6\}$;

$V_6=\min/9, 6+2/=8$, $M=\{6\}$;

2: $M\neq\phi$, $i=6$, $M=\phi$, $G(x_6)=\phi$;

2: $M=\phi$ и выполняется п.3.

Для иллюстрации выполнения п.3 изобразим граф /см. рис. 2.5/ с весами вершин, полученными в результате решения /рис. 2.9/.

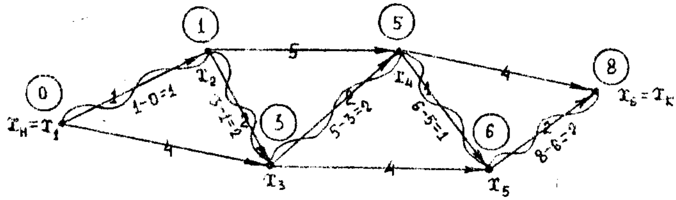


Рис. 2.9

Начинаем с вершины x_6 . $G^{-1}(x_6)=\{x_4, x_5\}$; $(\ell_{4,6}=4)\neq(V_6-V_4=3)$, поэтому дуга (x_4, x_6) не выделяется;

$(\ell_{5,6}=2)=(V_6-V_5=2)$ и дуга (x_5, x_6) выделяется, одновременно выделяется вершина x_5 .

Для выделенной вершины x_5 : $G^{-1}(x_5)=\{x_3, x_4\}$;

$(\ell_{3,5}=4)\neq(V_5-V_3=3)$, дуга $(x_3\ x_5)$ не выделяется;

$(\ell_{4,5}=1)=(V_5-V_4=1)$, выделены дуга $(x_4\ x_5)$ и вершина x_4 .

Для вершины x_4 : $G^{-1}(x_4)=\{x_2, x_3\}$; $\{\ell_{2,4}=5\}\neq(V_4-V_2=4)$ дуга (x_2, x_4) не выделяется; $(\ell_{3,4}=2)=(V_4-V_3=2)$, выделяется дуга (x_3, x_4) и вершина x_3. Для вершины x_3: $G^{-1}(x_3)=\{x_1, x_2\}$; $(\ell_{1,3}=4)\neq(V_3-V_1=3)$, дуга (x_1, x_3) не выделяется; $(\ell_{2,3}=2)=(V_3-V_2=2)$, следовательно, выделяются дуга (x_2, x_3) и вершина x_2 . $G^{-1}(x_2)=\{x_1\}$; $(\ell_{1,2}=1)=(V_2-V_1=1)$, выделяется дуга (x_1, x_2). Построен кратчайший путь $(x_н, x_к)$ длиной 8

$$(x_н=x_1, x_2)\,(x_2, x_3)\,(x_3, x_4)\,(x_4, x_5)\,(x_5, x_6=x_к).$$

Процесс решения можно оформить в виде следующей таблицы:

x_1	x_2	x_3	x_4	x_5	x_6	M
0	∞	∞	∞	∞	∞	1
0	1	4	∞	∞	∞	2, 3
0	1	3	6	∞	∞	3, 4
0	1	3	5	7	∞	4, 5
0	1	3	5	6	9	5, 6
0	1	3	5	6	8	6
0	1	3	5	6	8	∅

Далее процесс решения выполняется по рис. 2.9.

Приведем еще один пример /рис. 2.10/ с краткой записью процесса решения.

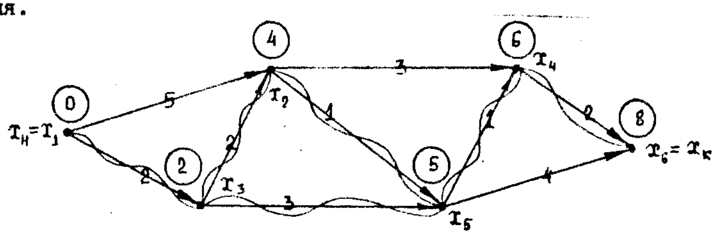


Рис. 2.10

x_1	x_2	x_3	x_4	x_5	x_6	M
0	∞	∞	∞	∞	∞	1
0	5	2	∞	∞	∞	2, 3
0	5	2	8	6	∞	3, 4, 5
0	4	2	8	5	∞	4, 5, 2
0	4	2	8	5	10	5, 2, 5
0	4	2	6	5	9	2, 6, 4
0	4	2	6	5	9	6, 4
0	4	2	6	5	9	4
0	4	2	6	5	8	4
0	4	2	6	5	8	∅

На графе /рис. 2.10/ отметим вершины полученными числовыми значениями весов и выделим кратчайшие пути. Получены два кратчайших пути длиной 8:

1. $(x_1,x_3)(x_3,x_2)(x_2,x_5)(x_5,x_4)(x_4,x_6)$;

2. $(x_1,x_3)(x_3,x_5)(x_5,x_4)(x_4,x_6)$.

2.2.4. Алгоритм Дейкстры построения кратчайшего пути во взвешенном графе.

Как видно из последнего примера веса вершин V_i пересчитывались неоднократно и появлялись несколько раз в массиве M, являясь источником новых волн распространения весов. Этот недостаток волнового алгоритма заставил разрабатывать другие алгоритмы построения кратчайшего пути, один из лучших – алгоритм Дейкстры, отличия которого состоят в том, что на каждом шаге выбирается для распространения веса вершина, имеющая наименьший вес, не распространившая его на другие вершины. Таким образом, в алгоритме Дейкстры нужно отметить вершины уже распространившие свой вес, а из остальных выбирать очередную с наименьшим весом.

Формулировка алгоритма:

1. $V_н=0, \forall x_i\,(i \neq H)(V_i=\infty)$. Все вершины не отмечены.

2. Среди неотмеченных вершин выбираем x_i с наименьшим весом V_i, вершину отмечаем, если $V_i=\infty$, то делаем вывод, что пути из вершины $x_н$ к вершине $x_к$ нет, иначе, если выбранная вершина $x_i=x_к$, то выполняем п.3, иначе пересчитываем веса вершин, принадлежащих $G(x_i)$: $V_j=\min(V_j, V_i+\ell_{ij})$. Выполняем п.2.

3. Аналогичен п.3 волнового алгоритма. Приведем сокращенную запись решения для примеров /см. рис. 2.5; 2.9/, рассмотренные вершины изображены кружками. Кратчайшие пути не выделяем, так как этот процесс выполняется также, как и в волновом алгоритме /рис. 2.11 и 2.12/.

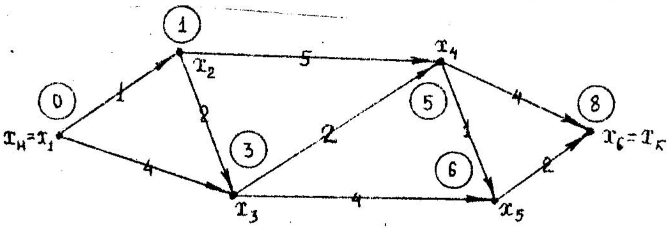


Рис. 2.11

x_1	x_2	x_3	x_4	x_5	x_6
0	∞	∞	∞	∞	∞
	1	4	∞	∞	∞
		3	6	∞	∞
			5	7	∞
				6	9
					8

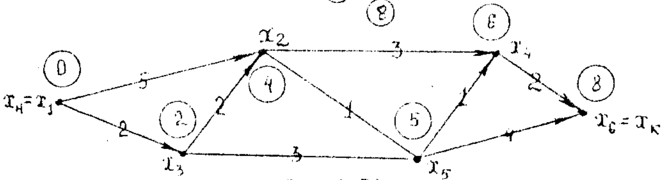


Рис.2.I2

x_1	x_2	x_3	x_4	x_5	x_6
0	∞	∞	∞	∞	∞
	5	2	∞	∞	∞
	4		∞	5	∞
			7	5	∞
			6		9
					8

Как видим, процесс решения для обоих примеров немного сократился, хотя на каждом шаге необходимо выбирать вершину меньшего веса.

2.2.5. Замечание о длиннейших путях.

Во многих приложениях наряду с кратчайшими путями встает задача построения длиннейших путей. Например, в программе необходимо знать не только наименьшее, но и наибольшее время реализации. При составлении графика работ наибольшее время выполнения работ имеет существенное значение и т.д. Если в алгоритмах поиска кратчайшего пути заменить процедуру расчета весов вершин

$$V_j = max\,(V_j, V_i + \ell_{ij})$$

и в первом пункте волнового алгоритма присвоить всем вершинам вес 0, то автоматически будут строиться длиннейшие пути.

При построении длиннейших путей рассматриваются элементарные или простые длиннейшие пути, длиннейшие пути с заданным числом выполнения циклов.

Пример

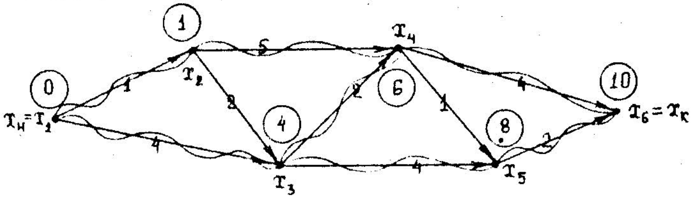


Рис. 2.13

x_1	x_2	x_3	x_4	x_5	x_6	M
0	0	0	0	0	0	1
0	1	4	0	0	0	2,3
0	1	4	6	0	0	3,4
0	1	4	6	8	0	4,5
0	1	4	6	8	10	5,6
0	1	4	6	8	10	6
0	1	4	6	8	10	$\varnothing$

Построены три длиннейших пути:

1. $(x_1, x_2)(x_2, x_4)(x_4, x_6)$;
2. $(x_1, x_3)(x_3, x_4)(x_4, x_6)$;
3. $(x_1, x_3)(x_3, x_5)(x_5, x_6)$.

Все они имеют длину 10.

2.2.6. Вопросы и задачи к подразд. 2.2.

А. Что гарантирует в алгоритме Дейкстры выбор на очередном шаге нерассмотренной вершины с наименьшим весом?

Б. Поясните, в каких случаях алгоритм Дейкстры лучше, а в каких хуже волнового алгоритма.

В. Найти кратчайшие пути согласно волновому алгоритму и алгоритму Дейкстры, используя графы п. 2.7.4. Считается, что графы ориентированы: дуги имеют направления слева направо. Веса дуг - цифры, стоящие на дугах вне скобок.

Например, $x_i \xrightarrow{4(3)} x_j$ означает $x_i \xrightarrow{4} x_j$.

Г. Для тех же графов найти длиннейшие элементарные пути.

2.3. Компоненты связности

2.3.1. Определение связности.

В ряде приложений /системах связи, программных комплексах, электрических цепях/ важное значение имеет понятие связности. Вершины x_i

и x_j слабо связны, если существует путь (x_i, x_j) в графе (G, X). Вершины x_i и x_j сильно связны, если существуют пути (x_i, x_j) и (x_j, x_i) в графе $/G, X/$. Если в графе нет путей из вершины x_i в x_j и нет обратного пути из x_j в x_i, то вершины x_i, x_j не связны. Очевидно, что для неориентированного графа имеет смысл только понятие сильной связности. Отношение сильной связности рефлексивно, симметрично и транзитивно, т.е. оно является отношением эквивалентности и однозначно разбивает множество вершин графа на компоненты связности: максимальные подмножества сильно связных между собой вершин.

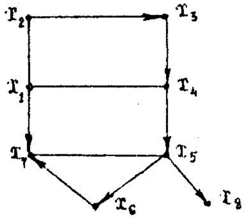


Рис. 2.14

Пример.

В графе /рис. 2.14/ три компоненты связности:

I. $\{x_1, x_2, x_3, x_4\}$;

П. $\{x_5, x_6, x_7\}$; Ш. $\{x_8\}$.

Легко убедиться, что внутри каждой компоненты между всеми парами вершин есть пути в обоих направлениях. Между компонентами – только слабая связность: есть пути из вершин компоненты I в вершины компонент П и Ш, и из вершин компоненты П в вершину x_8.

2.3.2. Построение компонент связности в неориентированном графе.

Как следует из однозначности разбиения неориентированного графа на компоненты связности, их выделение не требует перебора вариантов. Поэтому алгоритм простой.

I. $i = 0$. Все вершины графа не отмечены.

2. $i = i + 1$. Выбираем очередную неотмеченную вершину, отмечаем ее и все связанные с нею вершины значением индекса i с помощью распространения волны отметок по ребрам, идущим от уже отмеченных индексом i вершин. Таким образом выделяется i-я компонента связности. Если есть еще неотмеченные вершины, то выполняем п.2, иначе выделение компонент связности закончено /рис. 2.15/.

Пример.

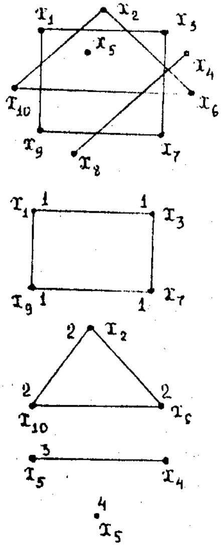


Рис. 2.15

I: $i = 0$.

2: $i = 1$. Отмечаем индексом $i = 1$ вершину x_1 и связанные с нею вершины x_3, x_7 и x_9. Получена I-я компонента связности

$$1\{x_1, x_3, x_7, x_9\}.$$

2: $i = 2$. Отмечаем индексом $i=2$ вершину x_2 и вершины x_6, x_{10}. Построена 2-я компонента связности. 2: $i = 3$. Отмечаются индексом $i = 3$ вершины x_4 и x_8. Третья компонента связности $-\{x_4, x_8\}$.

2: $i = 4$. Отмечается индексом $i = 4$ вершина x_5, которая формирует четвертую компоненту связности $-\{x_5\}$.

2.3.3. Построение компонент связности в ориентированном графе

Алгоритм заключается в сведении исходного ориентированного графа к неориентированному путем преобразований, не нарушающих сильную связность вершин графа. Для полученного неориентированного графа алгоритм выделения компонент связности уже сформулирован в предыдущем подразделе.

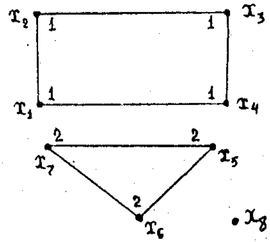
Алгоритм сведения исходного графа к неориентированному.

1. Все встречные дуги в исходном неориентированном графе заменяем ребрами.

2. Если в преобразуемом графе уже нет дуг, то процедура завершена, иначе выбирается какая-либо дуга $/x_i, x_j/$ и, если в графе есть путь $/x_j, x_i/$, то дуга $/x_i, x_j/$ и все дуги пути $/x_j, x_i/$ заменяются ребрами, если же пути $/x_j, x_i/$ в графе нет, то дуга $/x_i, x_j/$ удаляется из графа /должна сохраняться только сильная связность/. Выполняется п.2.

Проведем процедуру выделения компонент связности для графа /см. рис. 2.14/. Для дуги $/x_2, x_3/$ есть путь $/x_3, x_4/ /x_4, x_1/ /x_1, x_2/$, равный пути $/x_3, x_2/$, поэтому дуги $/x_2, x_3/$ и $/x_3, x_4/$ заменяем ребрами. Для дуги $/x_4, x_5/$ пути $/x_5, x_4/$ в графе нет, поэтому дуга

/ x_4, x_5 / удаляется. Аналогично удаляются дуги / x_1, x_7 / и / x_5, x_8 /. Для дуги / x_5, x_6 / есть путь / x_6, x_5 /, равный пути / x_6, x_7 / / x_7, x_5 /, поэтому дуги / x_5, x_6 / и / x_6, x_7 / заменяются ребрами. Получен неориентированный граф /рис. 2.16/, выделение компонент связности для которого не представляет трудностей.



I. $\{x_1, x_2, x_3, x_4\}$.
II. $\{x_5, x_6, x_7\}$.
III. $\{x_8\}$.

Рис. 2.16

2.3.4. Вопросы и задания к подразд. 2.3.

А. Показать, что если любая вершина сильно связна со всеми остальными вершинами графа, то граф – одна компонента связности.

Б. Убедиться, что графы задания подразд. 2.1 имеют одну компоненту связности.

В. В матрицах отношений подразд. 1.3.3 В заменить из половины матрицы ниже диагонали пять единиц на нули. Построить компоненты связности для полученных графов.

2.4. Система независимых циклов

2.4.1. Определения.

Определение цикла. Замкнутый маршрут в неориентированном графе называется <u>циклом</u> /например, маршрут по выделенным на рис. 2.17 ребрам/.

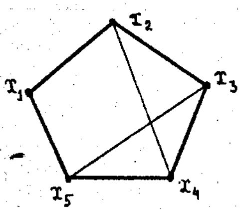


Рис. 2.17

Для ориентированного графа определяется аналогично понятие <u>контур</u>-замкнутый путь. В графе /рис. 2.17/ можно построить несколько циклов:

I. $(x_1, x_2)(x_2, x_3)(x_3, x_4)(x_4, x_5)(x_5, x_1)$;

II. $(x_1, x_2)(x_2, x_4)(x_4, x_5)(x_5, x_1)$;

III. $(x_1, x_2)(x_2, x_3)(x_3, x_5)(x_5, x_1)$;

IV. $(x_1, x_2)(x_2, x_4)(x_4, x_3)(x_3, x_5)(x_5, x_1)$ и т.д.

Определение системы независимых циклов. Цикл называется линейно-зависимым от некоторой совокупности других циклов, если его можно построить линейной комбинацией циклов этой совокупности. Чтобы пояснить понятие линейной комбинации, рассмотрим совокупность двух циклов графа /рис. 2.17/:

I. $(x_1, x_2)(x_2, x_3)(x_3, x_5)(x_5, x_1)$, П. $(x_3, x_5)(x_5, x_4)(x_4, x_3)$;

и организуем их обход, как показано на рис. 2.18. Исключая из обхода ребро / x_3, x_5 /, пройденное туда и обратно, получаем новый цикл

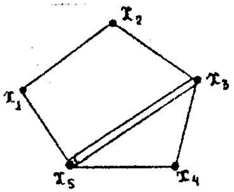


Рис. 2.18

$$(x_1, x_2)(x_2, x_3)(x_3, x_4)(x_4, x_5)(x_5, x_1).$$

Этот цикл построен в результате линейной комбинации исходных циклов и является линейно-зависимым от них. Таким образом, непосредственно на графе неформально пояснено понятие линейной зависимости. Приведем формальное определение. Сопоставим каждому циклу двоичный m -разрядный вектор, где m -число ребер графа. Пронумеруем ребра. Для i -го цикла компоненты $Ц_{ij}$ вектора $Ц_i$ определятся следующим образом:

$$Ц_{ij} = \begin{cases} 0, & \text{если ребро } j \text{ не входит в цикл;} \\ 1, & \text{если ребро } j \text{ входит в цикл.} \end{cases}$$

Тогда линейной комбинацией векторов $Ц_i$ называется результат векторной операции сложения по модулю два. Для данного примера /нумерация ребер графа показана на рис. 2.19/.

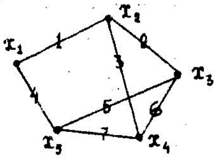


Рис. 2.19

$$\begin{array}{c|ccccccc|c}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \\
\oplus & 1 & 1 & 0 & 1 & 1 & 0 & 0 & \oplus\ Ц_{I} \\
 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & Ц_{II} \\
\hline
 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & Ц_{I} \oplus Ц_{II}
\end{array}$$

Системой независимых циклов графа называется максимальная линейно независимая совокупность циклов графа.

Цикломатическое число ν. Граф в общем случае может иметь не одну систему независимых циклов, однако число ν циклов в системе для каждого графа определяется его структурой однозначно. Число ν называется цикломатическим числом графа.

Теорема 2.1 /Эйлера/. Мощность ν системы независимых циклов в графе определяется формулой $\nu = m - n + p$, где m – число ребер; n – число вершин; p – число компонент связности.

В нашем примере ν = 7–5+1=3, т.е. в систему включаются три взаимно независимых цикла, каждый следующий цикл уже можно построить их линейной комбинацией.

Система независимых циклов плоского графа. Граф, который можно представить на плоскости, без пересечения его ребер, называется плоским.

Теорема 2.2. Края конечных граней плоского изображения графа составляют систему независимых циклов.

Конечной гранью называется внутренняя область плоскости, ограниченная ребрами в плоском изображении графа.

Граф, взятый нами для примера, – плоский, его изображение показано на рис. 2.20.

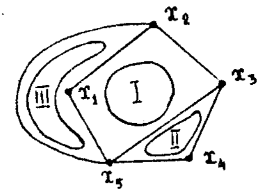


Рис. 2.20

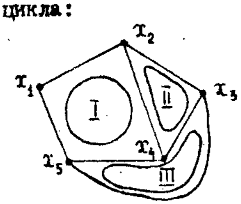
Система независимых циклов этого графа также показана на рис. 2.20 и содержит ν = 3 цикла:

I. $(x_1,x_2)(x_2,x_3)(x_3,x_5)(x_5,x_1)$.

II. $(x_3,x_5)(x_5,x_4)(x_4,x_3)$.

III. $(x_1,x_2)(x_2,x_5)(x_5,x_4)$.

Можно представить другое плоское изображение графа /рис. 2.21/ и получить другую систему независимых циклов, содержащую тоже ν = 3 цикла:



I. $(x_1,x_2)(x_2,x_4)(x_4,x_5)(x_5,x_1)$.

II. $(x_2,x_3)(x_3,x_4)(x_4,x_2)$.

III. $(x_3,x_5)(x_5,x_4)(x_4,x_3)$.

Рис. 2.21

В электротехнике, электронике при составлении законов Кирхгофа для электрической сети используют эту закономерность /рис. 2.22/.

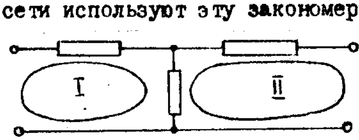


Рис. 2.22

Большинство графов не являются плоскими. Общий алгоритм построения системы независимых циклов сформулирован в п. 2.4.3.

2.4.2. Дерево, остов графа.

Определение дерева. Конечный связный граф без циклов называется деревом. Из свойств отсутствия циклов и связности следует, что p = I и $\nu = 0 = m - n + 1$, т.е. $m = n - 1$; число ребер в дереве на единицу меньше числа вершин. Мы уже встречались с понятием дерева, когда фиксировали процесс решения в задаче о покрытии в виде дерева. Примеры графов-деревьев показаны на рис. 2.23.

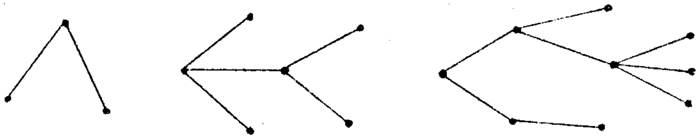


Рис. 2.23

Напомним, что вершины дерева, которым инцидентно только одно ребро, называются листьями /висячими вершинами/. Одна из вершин может быть объявлена корнем дерева.

Теорема 2.3. Существует $D_n = n^{n-2}$ различных деревьев на n вершинах.

Доказательство сводится к двум конструктивным построениям, сведению дерева к набору номеров вершины и, наоборот, восстановлению по этому набору дерева. Однозначность процесса позволяет заключить, что число деревьев равно числу наборов и таким образом рассчитать число деревьев, уточнив свойства наборов. Процесс построения набора по дереву.

I. Рассматриваем дерево с $n \geq 2$ вершинами. Исходный набор пуст.

2. Если преобразуемое дерево сохранило только две вершины, то процесс закончен, иначе находим среди листьев вершину с наименьшим номером. Эту вершину исключаем из дерева вместе с инцидентным ей ребром, в набор включаем номер вершины, с которой соединена удаляемая вершина. Снова выполняем п.2.

На рис. 2.24 изображена вся последовательность построения набора A и преобразования дерева. Восклицательным знаком отмечена выбираемая вершина с наименьшим номером.

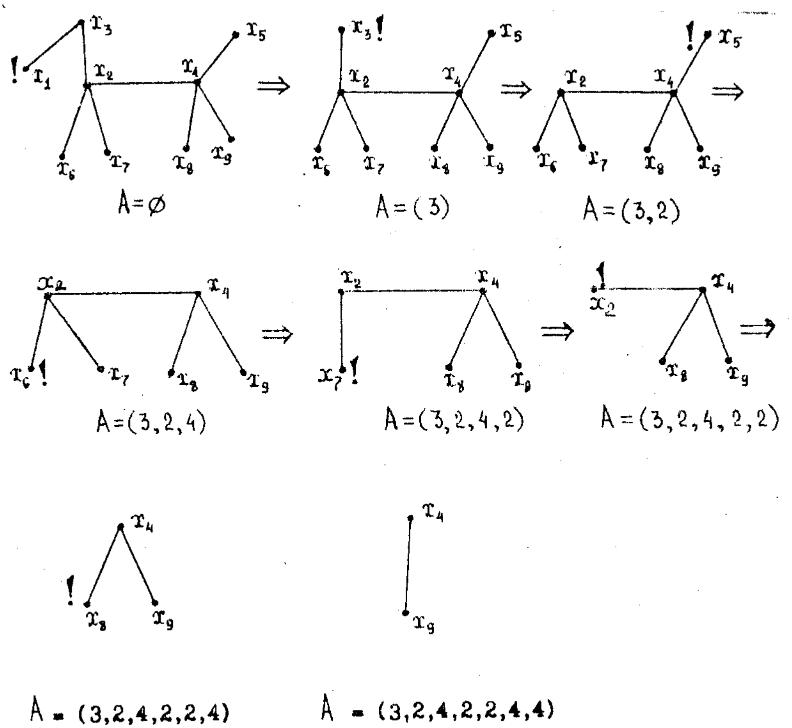


Рис. 2.24

Сформулируем процесс восстановления дерева по набору A :

I. Строим набор B , равный последовательности номеров вершин в порядке возрастания: $B = (1,2,\dots,n)$.

2. Если в наборе B остались всего два номера вершин, то соединим их и искомое дерево построено, иначе находим в B минимальный номер, которого нет в наборе A , и соединяем эту вершину с вершиной, номер который указан первым в наборе A . Вычеркиваем эти номера из A и B . Снова выполняем п.2. Для данного примера процесс показан на рис. 2.25.

Итак, из приведенного примера видно, что процесс построения набора A и восстановления дерева однозначен. Набор A - это размещение /важен порядок/ с повторениями. Длина набора равна $n-2$.

Отсюда $D_n = n^{n-2}$. Формула доказана.

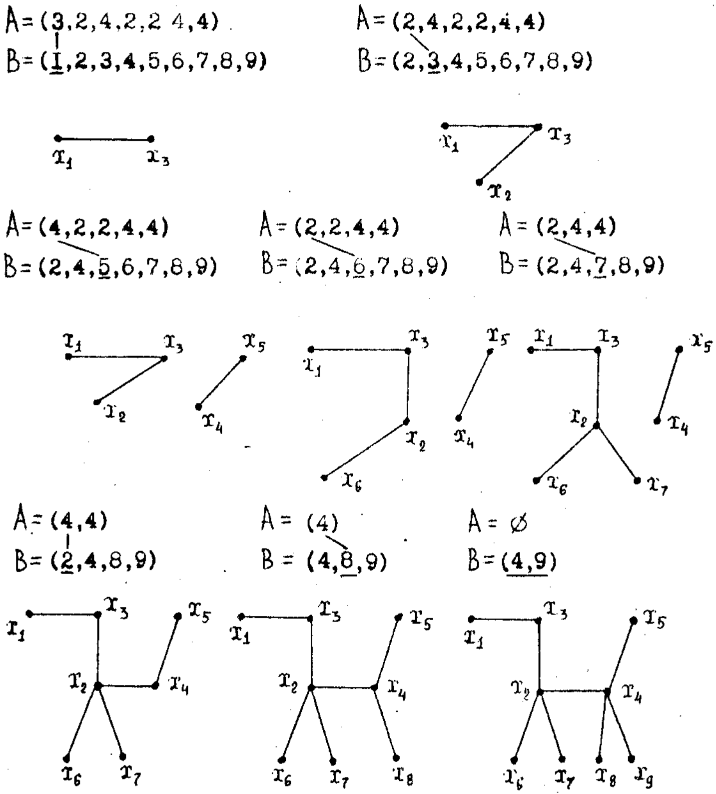
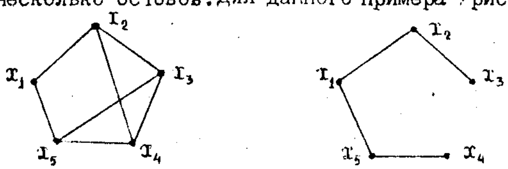


Рис. 2.25

Определение остова графа. Если граф не связан, то рассматриваем его отдельные компоненты. Остов такого графа определен как совокупность остовов его компонент. Для каждой компоненты частичный подграф, который может быть построен из нее удалением некоторых ребер и который является деревом, называется остовом. В общем случае для графа можно построить несколько остовов. Для данного примера /рис. 2.26/.



Исходный граф

Один из возможных остовов

Рис. 2.26

Один из возможных остовов для несвязного графа показан на рис. 2.27.

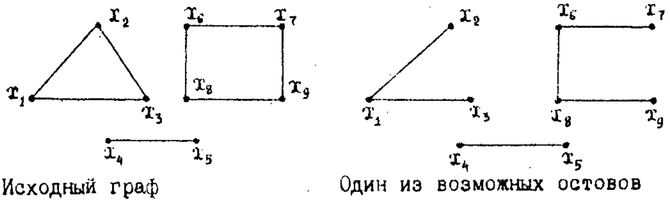


Рис. 2.27

Алгоритм построения произвольного остова.

I. Для каждой компоненты i графа выполняем п.2 и 3.

2. Строим частичный граф, содержащий все n_i вершин компоненты и не содержащий ребер /ϕ граф/.

3. Если в текущий частичный граф включены уже n_i-1 ребер, то остов для компоненты i построен, иначе выбираем очередное не рассмотренное ребро компоненты и пытаемся его включить в текущий граф. Если в текущем графе это не приводит к образованию цикла, то включаем ребра, иначе - не включаем. Ребро считаем рассмотренным. Выполняем п.3.

Пример для графа показан на рис. 2.I7. В нем только одна компонента. На рис. 2.28 изображена последовательность включения ребер в порядке их нумерации на рис. 2.I9.

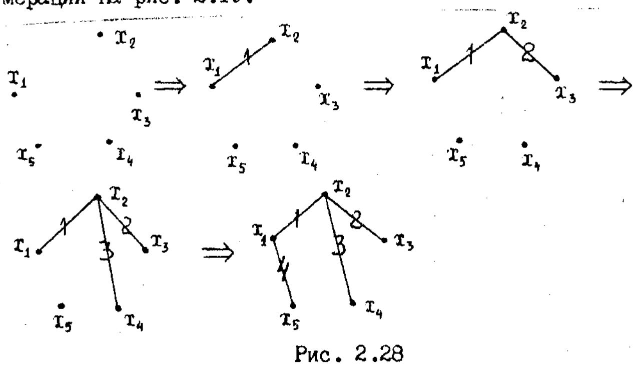


Рис. 2.28

Поскольку цикла не образовалось, то все ребра с номерами I, 2, 3 и 4 включены в остов и так как $m=n-1$, то остов построен.

Если выбрать другую последовательность рассмотрения ребер для включения в остов /например, 5, 6, 7, 2, 3, I, 4/, то не все рассмотренные ребра будут включены в остов и поэтому строится другой остов /рис. 2.29/.

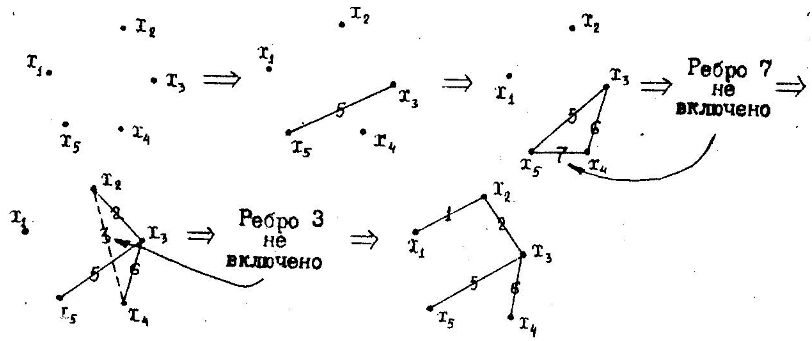


Рис. 2.29

Определение и алгоритм построения минимального остова. Для взвешенного графа остов с наименьшей суммой весов вошедших в него ребер называется минимальным /кратчайшее связывающее дерево/. В 2.3 обосновано, что при рассмотрении ребер в сформулированном ранее алгоритме построения обычного остова в порядке возрастания их веса будет построен минимальный остов. Например, если веса ребер соответствуют их нумерации на рис. 2.19, то на рис. 2.28 построен минимальный остов. Для графа /см. рис. 2.17/ с другими весами минимальный остов построен на рис. 2.30.

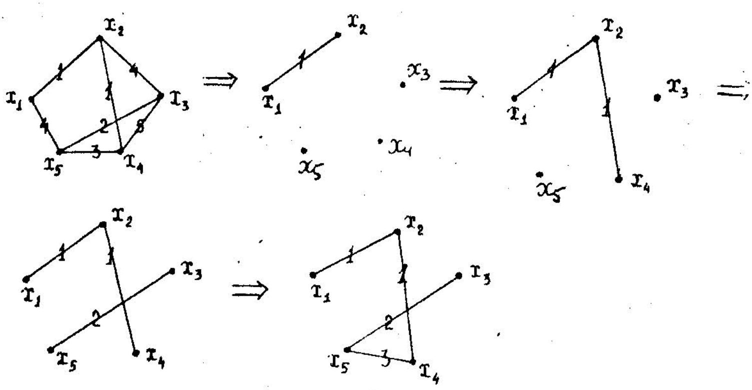


Рис. 2.30

Задача о минимальном остове имеет очевидные интерпретации: l_{ij} – длина, тогда имеется кратчайшая связь всех вершин; l_{ij} – стоимость, самая дешевая связь всех вершин и т.д.

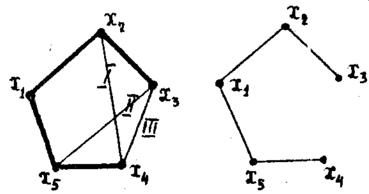
2.4.3. Алгоритм построения системы независимых циклов графа.

I. Строится произвольный остов графа. В исходном графе отмечаются ребра, включенные в остов.

2. Выбирается очередное неотмеченное ребро и строится цикл, содержащий это ребро и ребра остова. Такой цикл обязательно существует, так как вершины остова связны. Рассмотренное ребро отмечается и если есть еще неотмеченные ребра, то выполняется п.2, иначе – п.3.

3. По формуле Эйлера $\nu = m - n + p$ производится проверка числа построенных циклов /для контроля/.

Пример для графа на рис.2.3I.



Система циклов

I. $(x_2,x_4)(x_4,x_5)(x_5,x_1)(x_1,x_2)$;

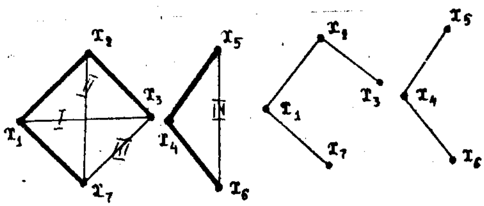
II. $(x_3,x_5)(x_5,x_1)(x_1,x_2)(x_2,x_3)$,

III. $(x_3,x_4)(x_4,x_5)(x_5,x_1)(x_1,x_2)(x_2,x_3)$;

$\nu = 9 - 7 + 1 = 3.$

Рис. 2.3I

Рассмотрим более сложный пример несвязного графа /рис. 2.32/



Система циклов

I. $(x_1,x_3)(x_3,x_2)(x_2,x_1)$;

II. $(x_2,x_7)(x_7,x_1)(x_1,x_2)$.

III. $(x_3,x_7)(x_7,x_1)(x_1,x_2)(x_2,x_3)$.

IV. $(x_5,x_6)(x_6,x_4)(x_4,x_5)$;

$\nu = 9 - 7 + 2 = 4.$

Рис. 2.32

2.4.4. Задание к подразд. 2.4.

А. Построить остов и систему независимых циклов для графов задания подразд. 2.I, заменив все дуги на ребра.

Б. Построить минимальный остов для графов задания подразд. 2.I, если веса ребер берутся как веса ребер следующего полного графа /рис. 2.33/.

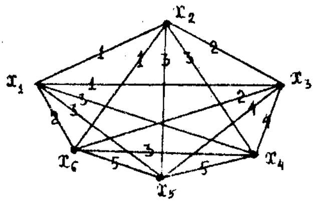


Рис. 2.33

В. Рассчитать число ν независимых циклов для графов задания п. . 2.7.4, если считать их неориентированными.

Г. Построить систему циклов графов задания п. 2.8.4.

2.5. Кратчайшая раскраска графа

2.5.I. Постановка задачи.

Вершины графа необходимо <u>раскрасить</u> так, чтобы никакие две смежные вершины не были окрашены одной краской. Вместо красок можно использовать числа 0, I, 2,... Условием оптимальности раскрашивания является использование минимального числа красок $\varkappa$, называемого <u>хроматическим числом графа</u>.

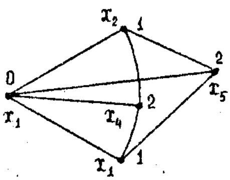


Рис. 2.34

Хроматическое число $\varkappa$ для графа /рис.2.33/ равно трем. Действительно, если вершину x_1 покрасить краской 0, то все смежные с ней вершины x_2, x_3, x_4, x_5 нельзя покрасить краской 0. Однако вершины x_2 и x_4 - смежные, значит для их раскраски необходимы две различные краски /I и 2/. Таким образом, изображенная на рис. 2.34 раскраска - минимальна.

<u>Теорема 2.4</u>. Для плоских графов $\varkappa \leq 4$.

2.5.2. Внутренне и внешне устойчивые множества вершин графа.

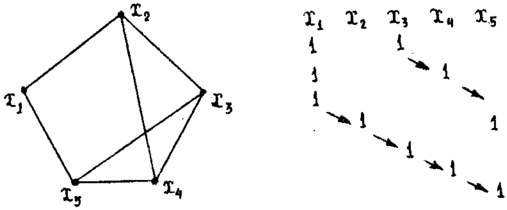
<u>Внутренне устойчивое множество.</u>

Подмножество $X_{внутр} \subseteq X$ вершин графа (G,X) называется внутренне устойчивым, если вершины этого подмножества не смежны, т.е.

$$X_{внутр} \cap \Big(\bigcup_{x_i \in X_{внутр}} G(x_i) \Big) = \varnothing .$$

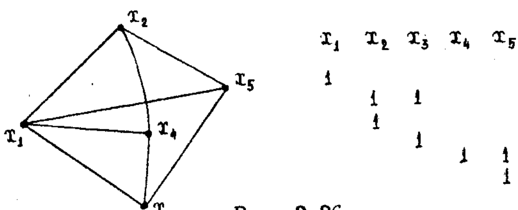
Очевидно следует искать максимальные внутренне устойчивые подмножества вершин графа. Решить задачу поиска этих подмножеств можно применением алгоритма граничного перебора с поиском максимальных совместимых подмножеств по матрице отношений графа или непосредственно по самому графу. Отношение G графа считается отношением противоречия.

Приведем решение двух примеров /рис. 2.35-2.36/.



$\{x_1, x_3\}$
$\{x_1, x_4\}$
$\{x_2, x_5\}$

Рис. 2.35



$\{x_1\}$
$\{x_2, x_3\}$
$\{x_4, x_5\}$

Рис. 2.36

Внешне устойчивое множество. Подмножество $X_{внеш} \subseteq X$ вершин графа (G,X) называется внешне устойчивым, если для каждой вершины x_i графа удовлетворяется хотя бы одно из условий:

а/ либо $x_i \in X_{внеш}$;

б/ либо $x_i \in G^{-1}(x_j)(x_j \in X_{внеш})$, т.е.

$$X_{внеш} \cup \Big(\bigcup_{x_i \in X_{внеш}} G^{-1}(x_i) \Big) = X .$$

Эта задача имеет интерпретацию расстановки часовых на вершинах графа с тем, чтобы защитить все вершины графа: вершина x_i защищена либо, если на ней стоит часовой $/x_i \in X_{внеш}/$, либо она видна из вершины $x_j \in X_{внеш}(x_i \in G^{-1}(x_j))$. Очевидно, необходимо строить минимальные внешне устойчивые множества.

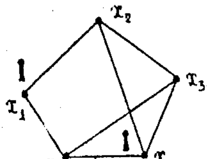


Рис. 2.37

Часовые на вершинах x_1 и x_4 защищают все вершины графа /рис. 2.37/. Другие варианты: $\{x_1, x_2\}$ или $\{x_1, x_3\}$, или $\{x_2, x_5\}$ и т.д.

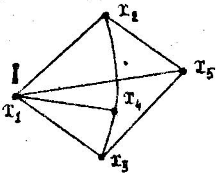


Рис. 2.38

Один часовой на вершине x_4 /рис. 2.38/ защищает все вершины. Другие варианты: $\{x_2,x_3\}$, $\{x_4,x_5\}$ и т.д.

Алгоритм построения минимальных внешне устойчивых множеств вершин графа.

I. Матрица отношения G графа транспонируется /получается отношение G^{-1}/. Диагональные элементы приравниваются к единице /сама вершина себя защищает/.

2. Решается задача построения покрытий строками матрицы преобразованного отношения всех ее столбцов.

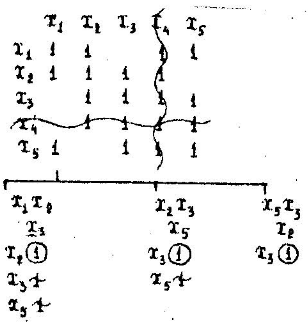


Рис. 2.39

Решаем задачу для графа /рис. 2.38/ при построения хотя бы одного кратчайшего покрытия. На рис. 2.38 построены три кратчайших покрытия $\{x_1,x_2\}$, $\{x_2,x_3\}$, $\{x_5,x_4\}$, которые служат внешне устойчивыми множествами вершины графа /рис. 2.39/. Если необходимо строить все минимальные внешне устойчивые множества, то нужно решать задачу построения всех безызбыточных покрытий.

2.5.3. Алгоритм кратчайшей раскраски графа.

Очевидно, что каждое максимальное внутренне устойчивое множество вершин графа может быть покрашено одной краской. Для решения задачи необходимо выбрать минимальное число максимальных внутренне устойчивых множеств, содержащих в совокупности все вершины графа, т.е. решить задачу кратчайшего покрытия максимальными внутренне устойчивыми множествами всех вершин графа.

Формулировка алгоритма:

I. Методом граничного перебора строятся все максимальные внутренне устойчивые множества вершин графа.

2. Строится таблица покрытий, строками которой являются найденные внутренне устойчивые подмножества, столбцами - вершины графа. Таблица

отражает отношение принадлежности вершин максимальных внутренне устойчивых множеств множеству вершин графа.

3. Решается задача построения одного кратчайшего покрытия.

4. Вершины, принадлежащие каждому подмножеству, вошедшему в найденное кратчайшее покрытие, окрашиваются одной краской. Если некоторая вершина принадлежит нескольким вошедшим в покрытие подмножествам, то она в одном остается, из остальных исключается.

Рассмотрим граф

x_1	x_2	x_3	x_4	x_5	
1		1			A
1			1		B
1					—
	1			1	C
		1			—
			1		—
				1	—

Таблица покрытий

	x_1	x_2	x_3	x_4	x_5
A	1		1		
B	1			1	
C		1			1

Покрытие – A,B,C.

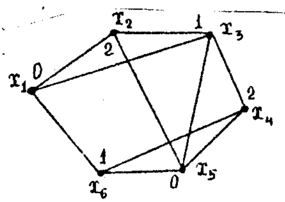
Раскраска

$A = \{x_1, x_3\}$ – краска 0,

$B = \{x_4\}$ – краска I,

$C = \{x_2, x_5\}$ – краска 2.

Проведем раскраску вершин графа, изображенного на рис. 2.40.



x_1	x_2	x_3	x_4	x_5	x_6	
1			1			
1				1		0
	1		1			2
	1				1	
		1			1	1

Рис. 2.40

2.5.4. Задания к подразд. 2.5.

А. Найти внешне устойчивые множества вершин графов задания подразд. 2.I.

Б. Провести кратчайшую раскраску вершин графов задания подразд. 2.I.

В. Провести кратчайшую раскраску вершин графов задания п.2.8.4.

2.6. Гамильтонов контур

2.6.I. Формулировка задач о гамильтоновом контуре минимальной стоимости.

Задан направленный граф (G,X), дуги которого взвешены функцией $\ell(x_i,x_j)$. Необходимо составить гамильтонов контур /ГК/, т.е. замкнутый путь по вершинам графа, который удовлетворяет следующим условиям: а/ контур, если таковой существует, включает в себя все вершины графа; б/ для каждой вершины только одна дуга входа этой вершины принадлежит контуру и только одна дуга исхода вершины принадлежит контуру /каждая вершина включена в контур только 1 раз/; в/ сумма f весов дуг построенного контура должна быть минимальной /эта функция определяет качество ГК/:

$$f=\sum_{(x_i,x_j)\in ГК}(x_i,x_j)\rightarrow min.$$

Такой ГК называется минимальным.

Пример.

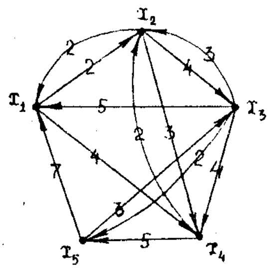


Рис. 2.4I

Возможные гамильтоновы контуры

I. $ГК=(x_1,x_2)(x_2,x_3)(x_3,x_4)(x_4,x_5)(x_5,x_1)$

$f=\ 2+4+4+5+7=22.$

II. $ГК=(x_1,x_2)(x_2,x_4)(x_4,x_5)(x_5,x_3)(x_3,x_1)$

$f=\ 2+3+5+3+5=18.$

III. $ГК=(x_1,x_4)(x_4,x_2)(x_2,x_3)(x_3,x_5)(x_5,x_1)$

$f=\ 4+2+4+2+7=19$

и т.д.

Наилучший из найденных контуров, удовлетворяющих условиям а/ и б/, имеет значение функции $f=18$. Но является ли этот результат минимальным ГК?

2.6.2. Возможные интерпретации задачи о ГК.

Выбор расписания проведения работ: x_i – работы; (x_i,x_j) – допустимые последовательности выполнения работ; $\ell(x_i,x_j)$– стоимость выполнения работы x_j после x_i. ГК минимальной стоимости – расписание /последовательность/ выполнения всех работ с минимальной суммарной стоимостью их выполнения.

Быстрейший путь сбора подписей. Нужно собрать подписи всех визирующих документ и как можно быстрее: x_i – визирующие; (x_i, x_j) – возможная последовательность подписания / x_j после x_i /; $\ell(x_i, x_j)$ – время поиска x_j после x_i.

Гамильтонов контур – путь сбора всех подписей на документе за минимальное время.

Классическая итерпретация /задача коммивояжера/: x_i – города; (x_i, x_j) – односторонняя дорога их x_i в x_j; $\ell(x_i, x_j)$ – ее длина. Необходимо выбрать путь коммивояжера так, чтобы последний посетил все города, причем каждый город – I раз.

2.6.3. Применение метода ветвей и границ к задаче построения ГК минимальной длины.

Исходным служит представление графа в виде матрицы отношения $\ell(x_i, x_j)$ /если дуги (x_i, x_j) в графе нет, то $\ell(x_i, x_j) = \infty$/. Например /см. рис. 2.4I/:

	x_1	x_2	x_3	x_4	x_5
x_1	∞	2	∞	4	∞
x_2	2	∞	4	3	∞
x_3	5	3	∞	4	2
x_4	∞	2	∞	∞	5
x_5	7	∞	3	∞	∞

Идея оценки. ГК содержит только по одной дуге, принадлежащей $G^{+1}(x_i)$ и $G^{-1}(x_i)$ – входу и исходу этой вершины. При вычислении функции качества ГК только одно число из каждой строки матрицы $|\ell(x_i, x_j)|$ будет входить в сумму весов ребер ГК. Пусть $\ell(x_i) = \min\limits_{(x_j)} \ell(x_i, x_j)$ – минимальное число в строке. Преобразуем матрицу $|\ell(x_i, x_j)|$ в $|\ell'(x_i, x_j)| = |\ell(x_i, x_j) - \ell(x_i)|$. Хотя мы вычли в каждой строке величину $\ell(x_i)$ из всех элементов этой строки, но поскольку в функцию качества гамильтонова контура входит только одно из них, то справедливо утверждение, что функция качества гамильтонова контура, полученного по исходной матрице $f = f' + \sum\limits_{i=1} \ell(x_i) = f' + \xi_{вых}$.

Аналогично можно провести вычисление для столбцов:

$$\ell'(x_j) = \min \ell'(x_i, x_j); \qquad |\ell''(x_i, x_j)| = |\ell'(x_i, x_j) - \ell'(x_j)|$$

$$\xi_{вх} = \sum_{j=1}^{n} \ell'(x_j), \quad \xi = \xi_{вых} + \xi_{вх}, \quad f = f'' + \xi.$$

	x_1	x_2	x_3	x_4	x_5	$\ell(x_i)$
x_1	∞	2	∞	4	∞	2
x_2	2	∞	4	3	∞	2
x_3	5	3	∞	4	2	2
x_4	∞	2	∞	∞	5	2
x_5	7	∞	3	∞	∞	3
						/11

	x_1	x_2	x_3	x_4	x_5	
x_1	∞	0	∞	2	∞	
x_2	0	∞	2	1	∞	
x_3	3	1	∞	2	0	
x_4	∞	0	∞	∞	3	
x_5	4	∞	0	∞	∞	
	0	0	0	1	0	/1

	x_1	x_2	x_3	x_4	x_5
x_1	∞	0	∞	1	∞
x_2	0	∞	2	0	∞
x_3	3	1	∞	1	0
x_4	∞	0	∞	∞	3
x_5	4	∞	0	∞	∞

$$|\ell(x_i,x_j)|,\ \xi_{вых}=11,\ |\ell'(x_i,x_j)|,\ \xi_{вх}=1,\ |\ell''(x_i,x_j)|,$$
$$\xi=\xi_{вх}+\xi_{вых}=11+1=12,$$
$$f=f''+12.$$

По крайней мере функция f качества любого ГК матрицы $|\ell(x_i,x_j)|$ больше или равна ξ, таким образом величина ξ может служить оценкой качества возможных ГК по матрице $|\ell(x_i,x_j)|$.

Идея ветвления. По любой дуге можно разложить множество $\{ГК\}$ на два непересекающихся подмножества: $\{ГК_1\}$, содержащие данную дугу, и $\{ГК_2\}$, не содержащие ее.

$$\{ГК\}$$

$(x_i,x_j)\in ГК\in\{ГК_1\}$ $\qquad\qquad$ $(x_i,x_j)\notin ГК\in\{ГК_2\}$

Одно или оба подмножества могут быть пустыми.

Используется усиление алгоритма, приводящее к сокращению процесса решения: нужно стараться выбрать дугу так, чтобы в множестве $\{ГК_I\}$ были ГК меньшей длины, а в множестве $\{ГК_2\}$ – большей длины, тогда множество $\{ГК_2\}$ может быть удастся не рассматривать. Для матрицы $|\ell''(x_i,x_j)|$ выбор дуги $/x_i, x_j/$, для которой $\ell(x_i,x_j)=0$ не приводит непосредственно к увеличению длины ГК. Значит, можно рассматривать такие дуги. Их в матрице $|\ell''(x_i,x_j)|$ много, по крайней мере по одной в строке и столбце. Вспомним второе условие, чтобы запрещение дуги приводило к увеличению оценки длины ГК на возможно большую величину $\theta(x_i,x_j)$. Запрещение дуги $/x_i,x_j/$ равносильно тому, что $\ell(x_i,x_j)=\infty$, но тогда в строке x_i и столбце x_j один из нулей исчезает, значит можно пересмотреть оценку

$$\theta(x_i,x_j)=\min_{x_k\neq x_j}\ell(x_i,x_k)+\min_{x_i\neq x_i}\ell(x_k,x_j).$$

Выбирается та дуга $/x_k,x_\ell/$, для которой $\theta(x_i,x_j)\to max$.

Рассмотрим пример:

	x_1	x_2	x_3	x_4	x_5
x_1	∞	0	∞	1	∞
x_2	0	∞	2	0	∞
x_3	3	1	∞	1	0
x_4	∞	0	∞	∞	3
x_5	4	∞	0	∞	∞

$$\theta(x_1,x_2)=\ell(x_1,x_4)+\ell(x_4,x_2)=1+0=1,$$
$$\theta(x_2,x_1)=\ell(x_2,x_4)+\ell(x_3,x_1)=0+3=3,$$
$$\theta(x_2,x_4)=\ell(x_2,x_1)+\ell(x_1,x_4)=0+1=1,$$
$$\theta(x_3,x_5)=\ell(x_3,x_2)+\ell(x_4,x_5)=1+3=4,$$
$$\theta(x_4,x_2)=\ell(x_4,x_5)+\ell(x_1,x_2)=3+0=3,$$
$$\theta(x_5,x_3)=\ell(x_5,x_1)+\ell(x_2,x_3)=4+2=6.$$

Вычислим $\theta(x_k,x_\ell)=max(\theta(x_i,x_j))=\theta(x_5,x_3)=6$. Оценки $\theta(x_i,x_j)$ можно непосредственно изображать на матрице $\ell''(x_i,x_j)$:

$$\begin{array}{c|c|} & x_j \\ \hline x_i & \begin{array}{l} \ell(x_i,x_j) \\ \quad \theta(x_i,x_j) \end{array} \\ \hline \end{array}$$

	x_1	x_2	x_3	x_4	x_5
x_1	∞	0_1	∞	1	∞
x_2	0_3	∞	2	0	∞
x_3	3	1	∞	1	0_4
x_4	∞	0_3	∞	∞	3
x_5	4	∞	$\boxed{0_6}$	∞	∞

Ветвление можно вести по дуге (x_5,x_3)

$$ГК=\phi,\ |\ell''(x_i,x_j)|;\ \xi=12$$

$(x_5,x_3)\in ГК$ — $|\ell^*(x_i,x_j)|$ $\qquad$ $ГК=\phi$ — $|\ell^{**}(x_i,x_j)|$

$$\xi^*=\xi+\ell(x_5,x_3)=12+0=12. \qquad \xi^{**}=\xi+\theta(x_i,x_j)=12+6=18.$$

<u>Проведение ветвления</u> по дуге $/x_k, x_\ell/$. Матрица $|\ell^*(x_i,x_j)|$ строится по матрице $|\ell''(x_i,x_j)|$: а/ удаляются строка x_k и столбец x_ℓ; б/ чтобы не допускать замыкания раньше, чем будут пройдены все вершины, запрещается замыкание всех путей, вошедших в выбранную часть ГК /удаляется дуга, соединяющая конечную вершину пути с начальной/; в/ по полученной матрице $|\ell^*(x_i,x_j)|$ строится оценка $\xi^*_{доп}=\xi^*_{вых}+\xi^*_{вх}$ и матрица преобразуется до $|\ell^{*''}(x_i,x_j)|$ точно так же, как преобразовывалась матрица $|\ell(x_i,x_j)|$ до $|\ell''(x_i,x_j)|$.

Матрица $|\ell^{**}(x_i,x_j)|$ получается заменой в матрице $|\ell''(x_i,x_j)|$ величины $\ell''(x_k,x_\ell)=0$ на $\ell^{**}(x_k,x_\ell)=\infty$.

Для обеих матриц $|\ell^{*}(x_i,x_j)|$ и $|\ell^{**}(x_i,x_j)|$ проводится оценка и дополнительные значения $\xi^{*}_{доп}$ и $\theta(x_k,x_\ell)$ прибавляются к ξ, матрицы $|\ell^{*}|$ и $|\ell^{**}|$ преобразуются к виду $|\ell^{*''}|$ и $|\ell^{**''}|$.

Продолжим разложение для каждого примера. Величины $\theta(x_i,x_j)$ для дуг с $\ell(x_i,x_j)=0$ записываем в соответствующие элементы в качестве нижнего индекса:

ГК$=\varnothing$

$\xi=12$

$$\begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline x_1 & \infty & 0_1 & \infty & 1 & \infty \\ x_2 & 0_3 & \infty & 2 & 0_1 & \infty \\ x_3 & 3 & 1 & \infty & 1 & 0_4 \\ x_4 & \infty & 0_3 & \infty & \infty & 3 \\ x_5 & 4 & \infty & \boxed{0}_6 & \infty & \infty \end{array}$$

$(x_5,x_3)\in$ ГК $\qquad\qquad$ $(x_5,x_3)\notin$ ГК

ГК$\in(x_5,x_3)$

$\xi=12$

$$\begin{array}{c|cccc|c} & x_1 & x_2 & x_4 & x_5 & \\ \hline x_1 & \infty & 0 & 1 & \infty & 0 \\ x_2 & 0 & \infty & 0 & \infty & 0 \\ x_3 & 3 & 1 & 1 & \cancel{\infty} & 1 \\ x_4 & \infty & 0 & \infty & 3 & 0/1 \end{array}$$

$\Downarrow$

$\xi_{вых}=1$

$$\begin{array}{c|cccc} & x_1 & x_2 & x_4 & x_5 \\ \hline x_1 & \infty & 0 & 1 & \infty \\ x_2 & 0 & \infty & 0 & \infty \\ x_3 & 2 & 0 & 0 & \infty \\ x_4 & \infty & 0 & \infty & 3 \\ & 0 & 0 & 0 & 3/3 \end{array}$$

$\Downarrow$

$\xi_{вх}=3$

$$\begin{array}{c|cccc} & x_1 & x_2 & x_4 & x_5 \\ \hline x_1 & \infty & 0 & 1 & \infty \\ x_2 & 0 & \infty & 0 & \infty \\ x_3 & 2 & 0 & 0 & \infty \\ x_4 & \infty & 0 & \infty & 0 \end{array}$$

$\xi=12+\xi_{доп}=12+1+3=16$

ГК$=\varnothing$

$\ell(x_5,x_3)=\infty$

$$\begin{array}{c|ccccc|c} & x_1 & x_2 & x_3 & x_4 & x_5 & \\ \hline x_1 & \infty & 0 & \infty & 1 & \infty & 0 \\ x_2 & 0 & \infty & 2 & 0 & \infty & 0 \\ x_3 & 3 & 1 & \infty & 1 & 0 & 0 \\ x_4 & \infty & 0 & \infty & \infty & 3 & 0 \\ x_5 & 4 & \infty & \cancel{0} & \infty & \infty & 4/4 \end{array}$$

$\Downarrow$

$\xi_{вых}=4$

$$\begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline x_1 & \infty & 0 & \infty & 1 & \infty \\ x_2 & 0 & \infty & 2 & 0 & \infty \\ x_3 & 3 & 1 & \infty & 1 & 0 \\ x_4 & \infty & 0 & \infty & \infty & 3 \\ x_5 & 0 & \infty & \infty & \infty & \infty \\ & 0 & 0 & 2 & 0 & 0/2 \end{array}$$

$\Downarrow$

$$\begin{array}{c|ccccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline x_1 & \infty & 0 & \infty & 1 & \infty \\ x_2 & 0 & \infty & 0 & 0 & \infty \\ x_3 & 3 & 1 & \infty & 1 & 0 \\ x_4 & \infty & 0 & \infty & \infty & 3 \\ x_5 & 0 & \infty & \infty & \infty & \infty \end{array}$$

$\xi=12+\theta(x_5,x_3)=12+4+2=18$

Рис. 2.42

Получение решения. Так как каждой вершине дерева приписывается множество дуг, включенных в ГК, то на висячей вершине дерева решения, для которой матрица $\|\ell^*(x_i, x_5)\|$ оказывается пустой, множество ГК с точностью до упорядочения дуг составит ГК и оценка ξ будет уже функцией f – длиной ГК.

Полное рассмотрение решения примера I.

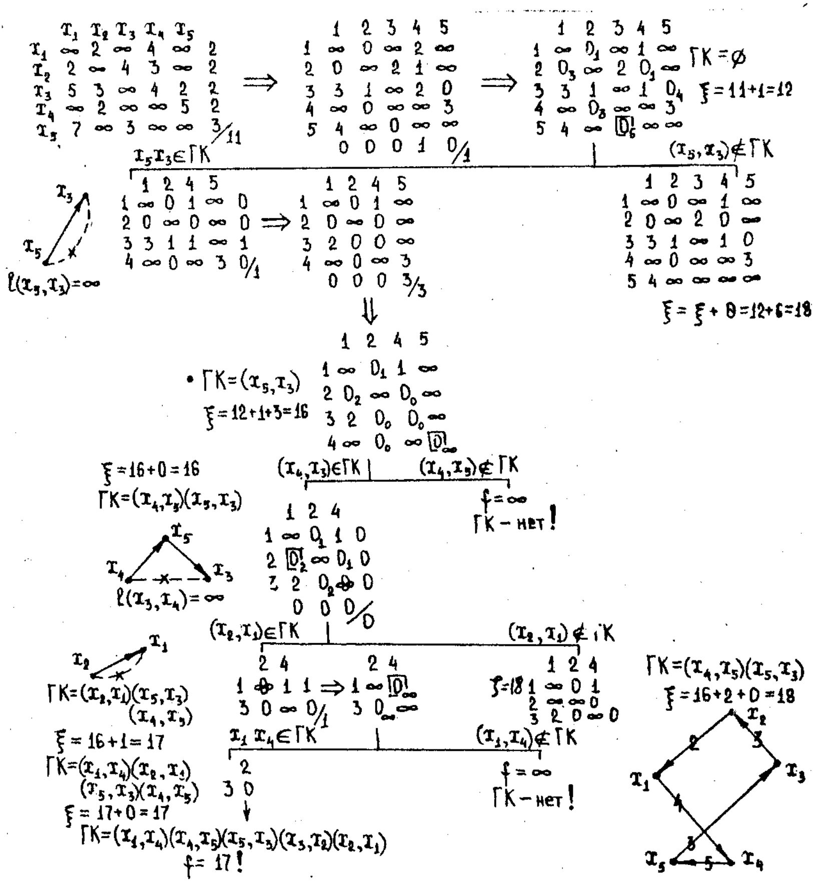


Рис. 2.43

Так как на всех других висячих вершинах $\xi > f = 17$, то найденный ГК - минимален. ГК, найденные для этого примера для рис. 2.41, не минимальны.

2.6.4. Краткая формулировка алгоритма.

1. От исходной матрицы $|\ell|$ переходим к матрице $|\ell''|$ и оценке ξ, $ГК = \phi$. Тройку $|\ell''|$, ξ и $ГК$ помещаем в корень дерева. Рассматриваем вершину - корень дерева.

2. В рассматриваемой вершине дерева строим оценки $\theta(x_i, x_j)$ для дуг, для которых $\ell''(x_i, x_j) = 0$, и выбираем дугу (x_k, x_ℓ) с наибольшей оценкой $\theta(x_k, x_\ell)$.

3. Проводим разложение матрицы $|\ell''|$ на дуге (x_k, x_ℓ) на $|\ell''|$ и $|\ell^{**}|$, уточняем оценки ξ^* и ξ^{**} и получаем матрицы $|\ell^{*''}|$ и $|\ell^{**''}|$.

Если ξ^* или ξ^{**} равны ∞, то обрываем соответствующую ветвь. Если ξ^* или ξ^{**} больше стоимости f_Λ лучшего из найденных ГК, то так-так же поступаем аналогично. Если $|\ell^*| = \phi$, то на этой вершине получим ГК и f. Сравним f и f_Λ. Если $f < f_\Lambda$, то запоминаем f как стоимость f_Λ лучшего результата, иначе обрываем соответствующую ветвь. Обрываем все концевые вершины, где $\xi \geq f_\Lambda$.

4. Если есть висячие вершины с непустой матрицей $|\ell''|$, то выбираем из них одну с меньшим значением ξ и выполняем п.2, иначе процесс заканчивается; решение, соответствующее значению f_Λ, является гамильтоновым контуром минимальной длины.

<u>Замечание</u>. Весь процесс решения изображаем в виде дерева во всех вариантах звездочки $*$, $**$ и штрихи $'$, $''$ не ставим.

Рассмотрим еще один пример решения задачи построения гамильтонова контура минимальной длины /рис.2.44/.

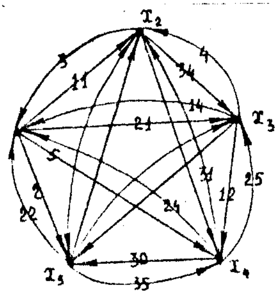


Рис. 2.44

Первый очевидный ГК:

$$ГК = (x_1, x_2)(x_2, x_3)(x_3, x_4)(x_4, x_5)(x_5, x_1)$$
$$f = 11 + 34 + 12 + 30 + 22 = 109.$$

Можно попробовать найти еще один очевидный ГК:

$$ГК = (x_1, x_5)(x_5, x_4)(x_4, x_3)(x_3, x_2)(x_2, x_1)$$
$$f = 2 + 35 + 25 + 4 + 3 = 69.$$

Точное решение задачи найдем, применяя сформулированный алгоритм /рис.2.45/.

$$\begin{array}{c|ccccc|c} & 1 & 2 & 3 & 4 & 5 & \\ \hline 1 & \infty & 11 & 21 & 5 & 2 & 2 \\ 2 & 3 & \infty & 34 & 32 & 15 & 3 \\ 3 & 14 & 4 & \infty & 12 & 23 & 4 \\ 4 & 24 & 31 & 25 & \infty & 30 & 24 \\ 5 & 22 & 1 & 13 & 35 & \infty & 1/34 \end{array} \Rightarrow \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & 9 & 19 & 3 & 0 \\ 2 & 0 & \infty & 31 & 29 & 12 \\ 3 & 10 & 0 & \infty & 8 & 19 \\ 4 & 0 & 7 & 1 & \infty & 6 \\ 5 & 21 & 0 & 12 & 34 & \infty \\ & 0 & 0 & 1 & 3 & 0/4 \end{array} \Rightarrow \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & 9 & 18 & 0_5 & 0_6 \\ 2 & \boxed{0}_{12} & \infty & 30 & 26 & 12 \\ 3 & 10 & 0_5 & \infty & 5 & 19 \\ 4 & 0_0 & 7 & 0_{11} & \infty & 6 \\ 5 & 21 & 0_{11} & 11 & 31 & \infty \end{array}$$

$\Gamma K=\varnothing$

$\xi=34+4=38$

$(x_2,x_1)\in\Gamma K$

$$\begin{array}{c|cccc|c} & 2 & 3 & 4 & 5 & \\ \hline 1 & \infty & 18 & 0 & 0 & 0 \\ 3 & 0_5 & \infty & 5 & 19 & 0 \\ 4 & 7 & \boxed{0}_{17} & \infty & 6 & 0 \\ 5 & 0_{11} & 11 & 31 & \infty & 0 \\ & 0 & 0 & 0 & 0/0 & \end{array}$$

$\Gamma K=(x_2,x_1)$

$\xi=38+0=38$

$(x_2,x_1)\notin\Gamma K$

$$\begin{array}{c|ccccc|c} & 1 & 2 & 3 & 4 & 5 & \\ \hline 1 & \infty & 9 & 18 & 0 & 0 & \\ 2 & \infty & \infty & 30 & 26 & 12 & 12 \\ 3 & 10 & 0 & \infty & 5 & 19 & \\ 4 & 0 & 7 & 0 & \infty & 6 & \\ 5 & 21 & 0 & 11 & 31 & \infty & \\ & 0 & & & & & \end{array} \Downarrow \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & 9 & 18 & 0_5 & 0_6 \\ 2 & \infty & \infty & 18 & 14 & \boxed{0}_{14} \\ 3 & 10 & 0_5 & \infty & 5 & 19 \\ 4 & 0_{10} & 7 & 0_{11} & \infty & 6 \\ 5 & 21 & 0_{11} & 11 & 31 & \infty \end{array}$$

$\xi=38+12=50$

$\Gamma K=\varnothing$

$(x_2,x_5)\notin\Gamma K$

$\xi=50+14=64$

$(x_4,x_3)\in\Gamma K$

$$\begin{array}{c|ccc|c} & 2 & 4 & 5 & \\ \hline 1 & \infty & \boxed{0}_{31} & 0_{19} & 0 \\ 3 & 0_{19} & \infty & 19 & 0 \\ 5 & 0_{31} & 31 & \infty & 0 \\ & 0 & 0 & 0/0 & \end{array}$$

$\Gamma K=(x_4,x_5)(x_2,x_1)$

$\xi=38$

$(x_4,x_3)\notin\Gamma K$

$$\begin{array}{c|cccc|c} & 2 & 3 & 4 & 5 & \\ \hline 1 & \infty & 18 & 0 & 0 & \\ 3 & 0 & \infty & 5 & 19 & \\ 4 & 7 & \cancel{0} & \infty & 6 & 6 \\ 5 & 0 & 11 & 31 & \infty & \\ & & 11 & & & \end{array} \Downarrow \begin{array}{c|cccc} & 2 & 3 & 4 & 5 \\ \hline 1 & \infty & 7 & 0_5 & 0_0 \\ 3 & 0_5 & \infty & 5 & 19 \\ 4 & 1 & \infty & \infty & 0_1 \\ 5 & 0_0 & \boxed{0}_7 & 31 & \infty \end{array}$$

$\Gamma K=(x_2,x_1)$

$\xi=38+6+11=55$

$(x_1,x_4)\in\Gamma K$

$$\begin{array}{c|cc|c} & 2 & 5 & \\ \hline 3 & \infty & 19 & 19 \\ 5 & 0 & \infty & 0/19 \end{array} \Downarrow \begin{array}{c|cc} & 2 & 5 \\ \hline 3 & \infty & 0 \\ 5 & 0 & \infty \end{array}$$

$\Gamma K=(x_4,x_3)$ $(x_2,x_1)(x_1,x_4)$

$\xi=38+19=57$

$(x_1,x_4)\in\Gamma K$

$\xi=38+31=69$

$(x_5,x_3)\in\Gamma K$

$$\begin{array}{c|ccc|c} & 2 & 4 & 5 & \\ \hline 1 & \infty & 0_5 & 0_0 & 0 \\ 3 & \boxed{0}_6 & 5 & \infty & 0 \\ 4 & 1 & \infty & 0_1 & 0 \\ & 0 & 0 & 0/0 & \end{array}$$

$\Gamma K=(x_2,x_1)(x_5,x_3)$

$\xi=55$

$(x_5,x_3)\notin\Gamma K$

$\xi=55+7=62$

$(x_3,x_2)\in\Gamma K$

$$\begin{array}{c|cc} & 4 & 5 \\ \hline 1 & \boxed{0}_{\infty} & \infty \\ 4 & \infty & 0_{\infty} \end{array}$$

$\Gamma K=(x_2,x_1)(x_5,x_3)(x_3,x_2)$

$\xi=55$

$(x_3,x_2)\notin\Gamma K$

$\xi=55+6=61$

$(x_1,x_4)\in\Gamma K$

$$\begin{array}{c|c} & 5 \\ \hline 4 & 0 \end{array}$$

$|(x_4,x_5)\in\Gamma K$

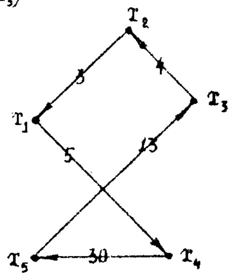
$\Gamma K=(x_4,x_5)(x_1,x_4)(x_2,x_1)(x_5,x_3)(x_3,x_2)$

$f=55!$

$(x_1,x_4)\notin\Gamma K$

$f=\infty$

$\Gamma K=$ нет!



Проверка:

$f=3+4+5+13+30=55$

Рис. 2.45

На рис. 2.45 найден лучший ГК, на всех остальных висячих вершинах дерева решения оценки больше 55, поэтому процесс решения заканчивается.

2.6.5. Индивидуальное задание к подразд. 2.6.

(1)	1	2	3	4	5
1	–	2	17	13	–
2	3	–	14	5	14
3	–	2	–	10	5
4	8	4	15	–	10
5	10	3	–	12	–

(2)	1	2	3	4	5
1	–	19	4	10	1
2	5	–	4	4	19
3	10	11	–	13	2
4	4	16	9	–	17
5	11	17	–	1	–

(3)	1	2	3	4	5
1	–	10	3	4	8
2	17	–	0	18	6
3	11	12	–	3	7
4	5	11	2	–	8
5	1	3	23	15	–

(4)	1	2	3	4	5
1	–	3	12	20	4
2	31	–	7	4	–
3	10	25	–	13	20
4	8	–	20	–	6
5	16	5	23	8	–

(5)	1	2	3	4	5
1	–	12	17	10	13
2	31	–	29	3	1
3	12	9	–	4	32
4	7	2	6	–	6
5	1	24	0	17	–

(6)	1	2	3	4	5
1	–	11	11	0	2
2	3	–	24	27	11
3	14	4	–	7	23
4	14	20	–	–	23
5	22	1	3	26	–

(7)	1	2	3	4	5
1	–	26	–	7	14
2	15	–	3	21	6
3	4	14	–	8	29
4	28	8	2	–	–
5	0	4	27	1	–

(8)	1	2	3	4	5
1	–	4	10	9	18
2	2	–	23	10	15
3	5	16	–	–	17
4	28	14	11	–	19
5	2	29	5	18	–

(9)	1	2	3	4	5
1	–	3	25	4	14
2	2	–	5	15	12
3	4	8	–	7	19
4	15	4	3	–	9
5	3	14	7	23	–

(10)	1	2	3	4	5
1	–	10	2	13	3
2	5	–	19	3	17
3	16	8	–	19	5
4	4	13	20	–	7
5	12	18	6	1	–

(11)	1	2	3	4	5
1	–	5	1	30	10
2	4	–	32	12	27
3	25	–	–	4	2
4	10	38	3	–	6
5	14	17	16	1	–

(12)	1	2	3	4	5
1	–	3	10	29	2
2	1	–	24	–	19
3	20	8	–	5	16
4	10	1	15	–	7
5	2	14	28	15	–

(13)	1	2	3	4	5
1	–	24	4	8	12
2	19	–	14	5	9
3	8	7	–	4	5
4	15	8	15	–	2
5	9	5	0	16	–

(16)	1	2	3	4	5
1	–	16	9	–	2
2	11	–	14	9	6
3	21	4	–	4	18
4	7	–	1	–	22
5	8	9	11	15	–

(17)	1	2	3	4	5
1	–	–	2	4	18
2	17	–	7	12	2
3	3	16	–	21	1
4	22	1	12	–	15
5	–	1	23	16	–

(18)	1	2	3	4	5
1	–	18	25	13	10
2	21	–	3	6	14
3	7	–	–	4	31
4	20	4	–	–	8
5	11	2	3	20	–

(19)	1	2	3	4	5
1	–	1	16	20	3
2	23	–	1	15	–
3	6	1	–	10	16
4	12	15	1	–	22
5	2	18	–	3	–

(20)	1	2	3	4	5
1	–	5	2	10	–
2	–	–	3	12	10
3	14	14	–	5	3
4	15	10	4	–	8
5	17	–	2	13	–

(21)	1	2	3	4	5
1	–	7	12	3	11
2	23	–	3	15	1
3	6	6	–	18	17
4	2	8	11	–	5
5	3	8	10	4	–

(22)	1	2	3	4	5
1	–	2	11	13	10
2	–	–	17	1	11
3	4	19	–	4	5
4	9	17	16	–	4
5	4	1	19	10	–

(23)	1	2	3	4	5
1	–	32	9	4	12
2	0	–	24	17	1
3	29	1	–	3	31
4	6	6	2	–	7
5	17	13	12	10	–

(24)	1	2	3	4	5
1	–	23	4	7	14
2	3	–	1	26	22
3	24	11	–	27	3
4	–	23	20	–	14
5	11	2	11	0	–

(25)	1	2	3	4	5
1	–	29	14	8	4
2	27	–	4	1	0
3	3	6	–	21	15
4	2	–	8	–	28
5	–	14	26	7	–

(26)	1	2	3	4	5
1	–	17	16	–	5
2	5	–	29	18	2
3	23	15	–	10	2
4	11	19	14	–	28
5	10	18	4	9	–

(27)	1	2	3	4	5
1	–	19	8	7	4
2	7	–	14	23	3
3	5	12	–	15	2
4	3	9	4	–	15
5	25	14	3	4	–

(28)	1	2	3	4	5
1	–	5	8	19	16
2	6	–	18	1	12
3	19	17	–	3	–
4	20	7	13	–	4
5	2	6	10	13	–

(29)	1	2	3	4	5
1	–	2	–	4	25
2	16	–	17	1	14
3	32	27	–	12	4
4	3	6	36	–	10
5	1	10	5	30	–

(30)	1	2	3	4	5
1	–	16	2	5	20
2	28	–	14	15	2
3	24	19	–	–	1
4	15	7	1	–	10
5	10	2	3	29	–

(31)	1	2	3	4	5
1	–	5	7	4	8
2	0	–	5	16	9
3	14	9	–	5	19
4	15	2	8	–	15
5	4	12	24	8	–

(32)	1	2	3	4	5
1	–	18	4	4	21
2	11	–	9	15	8
3	14	6	–	9	11
4	1	22	–	–	7
5	9	2	16	–	–

2.7. Потоки в сети

2.7.1. Определение потока в сети.

Суть – это ориентированный граф $/G, X/$, в котором выделены две вершины*: вершина истока $x_u \in X$ и вершина стока $x_c \in X$. Поток вытекает из вершины x_u, проходит по вершинам сети и собирается в вершине x_c.

Несколько интерпретаций сети:

каналы связи ЭВМ и потоки данных в этих каналах;

транспортные артерии и грузопотоки по ним;

токи в электрической схеме;

потоки деталей или рабочей силы для обеспечения работы сборочного конвейера;

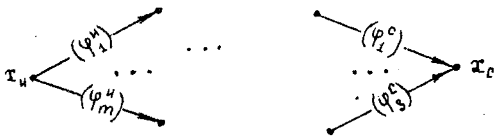
сеть снабжения электричеством /газом и т.д./потребителей.

Поток в сети формируется из потоков по каждой дуге сети. Допускаются только положительные значения потока $\varphi(g)$ по каждой дуге:

$$g \in G \quad (\varphi(g) \geqslant 0). \qquad /2.1/$$

Поток по дуге обозначается числом в скобках: $x_k \xrightarrow{(\varphi)} x_\ell$, например, при $\varphi = 3$ $x_k \xrightarrow{(3)} x_\ell$

Сумма потоков, вытекающих из вершины x_u, равна сумме потоков, приходящих к вершине x_c:



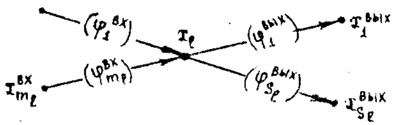
$$V = \sum_{i=1}^{m} \varphi_i^u = \sum_{i=1}^{s} \varphi_i^c. \qquad /2.2/$$

Величина V служит оценкой потока в сети. Подобное /2.2/ условие должно соблюдаться для всех внутренних вершин сети: сумма потоков, приходящих к вершине x_ℓ / $\ell \neq$ и, c /, равна сумме потоков, вытекающих из вершины x_ℓ:

$$\sum_{x_i \in G^{-1}(x_\ell)} \varphi(x_i, x_\ell) = \sum_{x_i \in G^{+1}(x_\ell)} \varphi(x_\ell, x_i). \qquad /2.3/$$

---

* Сети с несколькими вершинами истока и стока сводятся к рассматриваемому случаю.



$$G^{-1}(x_\ell) = \{x_i^{вх}, \ldots, x_{m_\ell}^{вх}\};$$
$$G^{+1}(x_\ell) = \{x_1^{вых}, \ldots, x_{s_\ell}^{вых}\}.$$

Абстрагируясь от интерпретаций, поток в сети – это функция $\varphi(g)$, определенная на дугах $g \in G$ сети, которая удовлетворяет условиям /2.1/-/2.3/. Легко проверить соблюдение условий в сети на рис. 2.46.

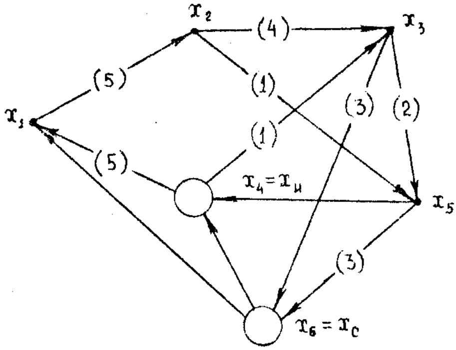


Рис. 2.46

В дальнейшем для наглядности будем упорядочивать изображение сети: вершину $x_и$ помещаем на рисунках слева, вершину x_c – справа, промежуточные вершины – между ними, упорядочивая по ярусам. Дуги входа вершины $x_и$ и дуги исхода вершины x_c будем исключать. В таком упорядоченном изображении сети не будем указывать направления дуг – всегда принимаем направление слева направо. Упорядоченное расположение /рис. 2.46/ показано на рис. 2.47.

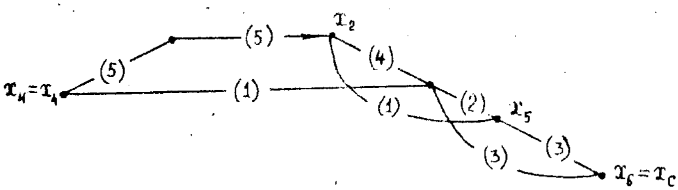


Рис. 2.47

По таким упорядоченным изображениям легче проверить условия /2.1/ – /2.3/, проще решать задачи о минимальном и максимальном потоках в сетях.

2.7.2. Задача о максимальном потоке.

Постановка задачи. На дугах $g \in G$ сети определяется функция $C(g) \geq 0$ - пропускная способность дуги g. Ставится задача построения потока в сети, для которого удовлетворяется условие

$$\varphi(g) \leq C(g) \qquad /2.4/$$

и наибольшее значение принимает оценка V, т.е.

$$V = \sum_{g \in G^{+}(x_u)} \varphi(g) \longrightarrow max. \qquad /2.5/$$

Пропускная способность $C(g)$ ограничивает допустимый наибольший поток по каждой дуге. Величины $C(g)$ изображаются на дугах сети как числа без скобок. Поскольку поток в сети должен удовлетворять условиям /2.1/ -/2.3/, то не на всех дугах он одновременно может достигать своего наибольшего значения. Например, поток в сети на рис. 2.48, очевидно, максимален: по каждой из дуг исхода вершины $x_u = x_1$ поток достигает наибольшего возможного значения, так как равен пропускной способности этих дуг.

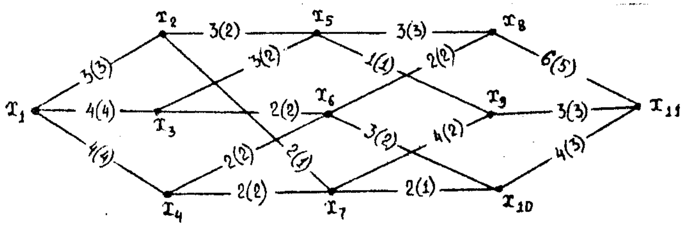


Рис. 2.48

В этом примере поток по дуге / x_2 , x_5 / не достигает своего наибольшего возможного значения: наибольшее суммарное значение потока, приходящего к вершине x_2 , $C(x_1, x_2) = 3$, суммарное значение пропускных способностей дуг, исходящих из вершины x_2 , равно 5, т.е. $C(x_2, x_5) + C(x_2, x_7) = 3+2$, но суммарное значение потоков, исходящих из вершины x_2 , не может быть больше 3 [условие/2.3/].

Построение полного потока. При построении максимального потока вручную удобно разбить процесс решения на два этапа: построение полного потока и коррекция потока до максимального.

Назовем дугу, для которой $\varphi(g) = C(g)$, насыщенной. Путь от вершины x_u к вершине x_c , содержащий хотя бы одну насыщенную дугу, назовем насыщенным. Поток называется полным, если все пути в сети от вершины x_u к вершине x_c насыщены. При упорядоченном изображении сети процесс нахождения путей от вершины x_u к вершине x_c достаточно нагляден, поэтому и процесс построения полного потока в сети прост: находится очередной путь $/x_u \to x_c/$, состоящий только из ненасыщенных дуг и по всем дугам g этого пути поток увеличивается на величину $\Delta\varphi$, равную наименьшей разности $C(g) - \varphi(g)$:

$$\Delta\varphi = \min_{g \in (x_u \to x_c)} \left(C(g) - \varphi(g) \right) . \qquad /2.6/$$

Для организации перебора путей $/x_u \to x_c/$ каждый раз выбираем возможное верхнее ребро очередной вершины, к которой подходит путь. Так как мы увеличиваем на всем пройденном пути поток по дугам на одну и ту же величину $\Delta\varphi$, то условия /2.1/ - /2.3/ в данном алгоритме выполняются автоматически. Рассмотрим сеть /рис.2.49/ с исходным нулевым потоком.

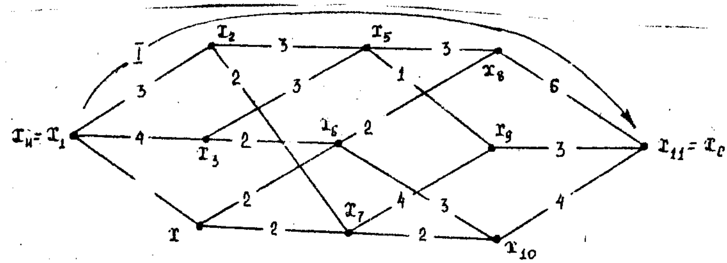


Рис. 2.49

Для выделенного пути $(x_1 \to x_{11}) = (x_1, x_2)(x_2, x_5)(x_5, x_8)(x_8, x_{11})$ величина $\Delta\varphi_I = \min(3-0, 3-0, 3-0, 6-0) = 3$, поэтому увеличиваем поток по всем дугам этого пути на 3 /рис. 2.50/.

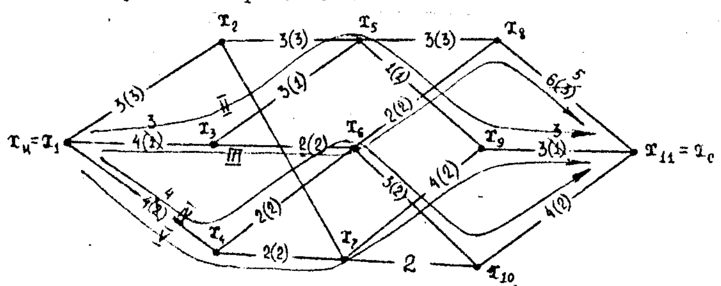


Рис. 2.50

Очевидно, ненасыщенных путей, содержащих дугу $/x_1, x_2/$, нет, поскольку уже сама эта дуга насыщена, поэтому последовательно строим пути II, III, с дугой $/x_1, x_3/$, IУ, У с дугой $/x_1, x_4/$:

$$\Delta\varphi_{\overline{II}} = min(4-0, 3-0, 1-0, 3-0) = 1;$$
$$\Delta\varphi_{\overline{III}} = min(4-1, 2-0, 2-0, 6-3) = 2;$$
$$\Delta\varphi_{\overline{IV}} = min(4-0, 2-0, 3-0, 4-0) = 2;$$
$$\Delta\varphi_{\overline{V}} = min(4-2, 2-0, 4-0, 3-1) = 2.$$

Можно убедиться, что поток, изображенный на рис. 2.50, полон. Однако из сравнения с потоком на рис. 2.48 следует, что он не максимален, так как $/V_{рис.50} = 10 < V_{рис.48} = 11/$.

Итак, не каждый полный поток масимален. Дело в том, что мы совершенно произвольно находили ненасыщенные пути и увеличивали поток по этим путям, не заботясь о других возможных вариантах изменения потока, и таким образом "портили" возможное увеличение потока по другим путям. Это обычная ситуация, когда последовательно решая задачи с оптимизацией решения на каждом отдельном шаге /а не решения в целом/, мы может зайти "в тупик" и не получить общего наилучшего решения.

Коррекция произвольного потока до максимального. Форд и Фалкерсон предложили достаточно простой алгоритм, в котором с помощью конечного числа коррекций произвольного потока его можно перестроить до максимального.

Алгоритм разметки вершин и выбора цепочки дуг для коррекции потока.

1. Все вершины сети считаются непомеченными. Вершина x_u помечается индексом +0.

2. Выбирается очередная помеченная, но еще не рассмотренная вершина x_ℓ:

индексом "$+\ell$" помечаются непомеченные вершины x_s, достигаемые из вершины x_ℓ по дуге $/x_\ell, x_s/$, для которой $\varphi(x_\ell, x_s) < c(x_\ell, x_s)$ /т.е. дуге, по которой можно увеличить поток из вершины x_ℓ /;

индексом "$-\ell$" помечаются непомеченные вершины x_k, из которых можно достичь вершину x_ℓ по дуге $/x_k, x_\ell/$ с $\varphi(x_k, x_\ell) > 0$ /т.е. по дуге, по которой можно уменьшить поток в вершину x_ℓ с тем, чтобы направить его по другому пути к вершине x_c и таким образом скорректировать поток в сети/.

3. Если помечается вершина x_c, то процесс расстановки пометок прекращается, иначе, если есть помеченные, но не рассмотренные вершины, то выполняется п.2; иначе, если все помеченные вершины рассмотрены, но не достигнута вершина x_c, делается вывод, что поток максимален.

В формулировке алгоритма мы попытались изложить идею коррекции; увеличить поток из вершины x_5 к вершине x_c, а часть прежнего потока к вершине x_c вернуть и направить по новому пути.

Рассмотрим пример /рис.2.5I/.

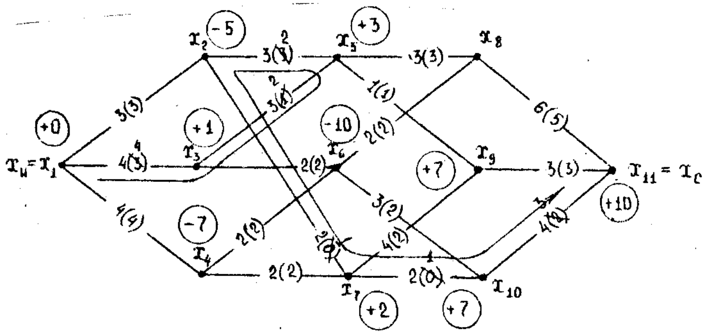


Рис. 2.5I

На рис. 2.5I расставлены индексы $\pm \ell$, и помеченной оказалась вершина x_c, значит поток в графе не максимален и следует найти цепочку дуг, по которым нужно скорректировать поток. Выделить эту цепочку дуг очень просто, так как индексы $/\pm \ell/$ вершин, просматриваемые, начиная с вершины x_c, помогают в этом:

вершина x_c помечена индексом +10, значит, выделяется дуга $/x_{10}, x_c/$; вершина x_{10} помечена индексом +7, выделяется дуга $/x_7, x_{10}/$ и так далее выделяются дуги $/x_2, x_7/$, $/x_2, x_5/$; $/x_3, x_5/$, $/x_1, x_3/$. Величина $\Delta\varphi$ изменения потока

$$\Delta\varphi = \min_{g \in \text{выделенной цепочке дуг}} \begin{cases} C(g) - \varphi(g), & \text{если дуга проходится по цепочке в прямом направлении;} \\ \varphi(g), & \text{если дуга проходится по цепочке в обратном направлении} \end{cases}$$

Для данного примера $\Delta\varphi = \min(4-3, 3-1, 3, 2-0, 2-0, 2-0, 4-2) = 1.$

Поток по дугам выделенной цепочки изменяется на $\Delta\varphi$; увеличивается на $\Delta\varphi$ для дуг, которые проходятся по цепочке в прямом направлении и уменьшаются на $\Delta\varphi$ для дуг, которые проходятся в обратном направлении / см. изменение потока на рис. 2.5I/.

В этом примере мы увеличили поток на единицу от вершины x_u к вершине x_5 и направили дальше это увеличение $\Delta\varphi = 1$ потока по дугам, исходящим из вершины x_5 к вершине x_c, но затем вернули единицу потока, приходящую к x_5 из x_2, и направили из x_2 к вершине x_c по выделенной цепочке дуг.

Итак, приведем общую формулировку алгоритма коррекции потока:

а/ производится разметка вершин сети, и если вершина x_c достигается, то выполняется п.б/, иначе делается вывод, что поток максимален;

б/ выделяется цепочка дуг, вычисляется величина $\Delta\varphi$ и производится коррекция потока на $\Delta\varphi$ в зависимости от направления прохода дуги от вершины x_u к вершине x_c по выделенной цепочке дуг;

в/ индексы всех вершин стираются и снова выполняется п.А алгоритма.

В данном примере при повторном выполнении разметки вершин оказывается помеченной только вершина x_u , а дальше разметка не проходит, так как все дуги, исходящие из этой вершины, насыщены, а дуг, входящих в вершину x_u , нет, что и означает максимальность скорректированного потока.

Рассмотрим пример. Решение проведем без пояснений. На рис. 2.52 показан полный поток.

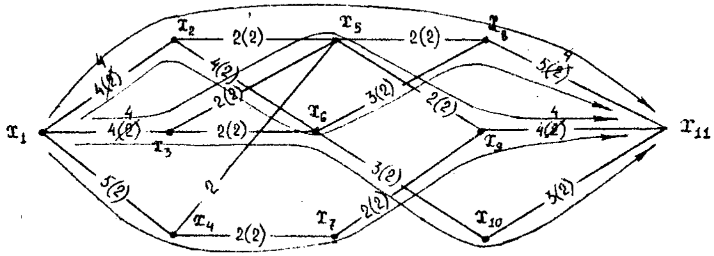


Рис. 2.52

Проведем первую коррекцию /рис. 2.53/.

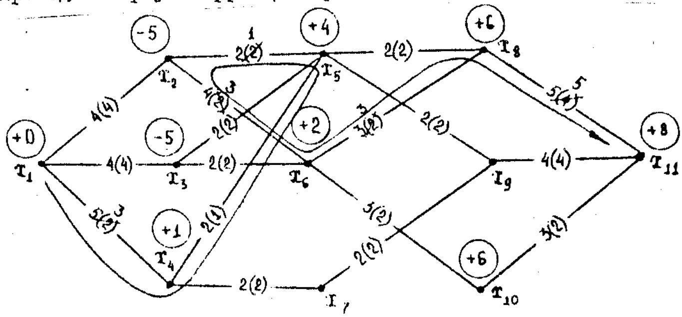


Рис. 2.53

Возможна следующая коррекция /рис. 2.54/.

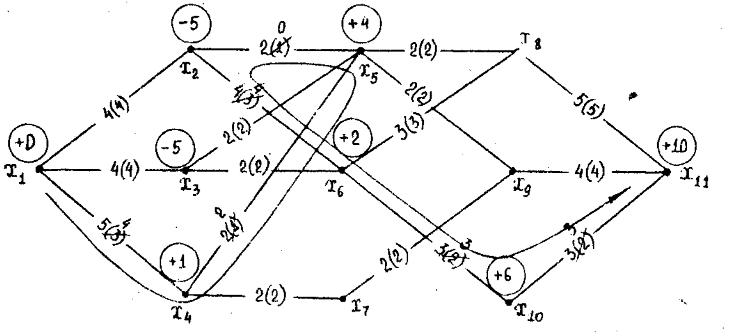


Рис. 2.54

Третья коррекция уже не получается, разметка вершин не достигает вершины x_c /рис. 2.55/.

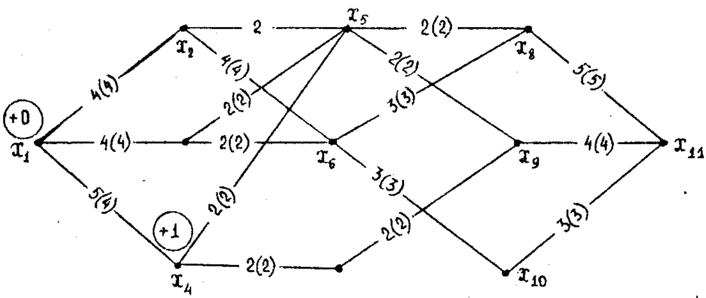
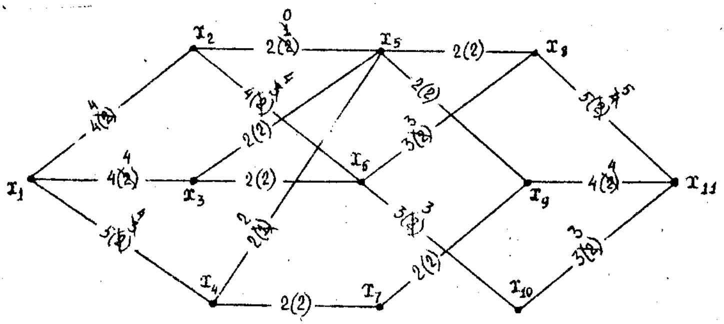


Рис. 2.55

Чтобы не перечерчивать многократно сеть, можно предложить запись преобразований и процесса решения в следующей форме /рис. 2.56/.

В этой записи все преобразования потока ведутся как бы с использованием одного алгоритма коррекции, причем, пока есть возможность расстановки индексов "+ℓ", не используется расстановка индексов "-ℓ".



Номер преобразований	Номер вершины											
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	$\Delta\varphi$
I	$+0$	$+1_4$			$+2_2$			$+5_2$			$+8_5$	2
П	$+0$	$+1_2$				$+2_4$		$+5_3$			$+8_3$	2
Ш	$+0$		$+1_4$		$+3_2$				$+5_2$		$+9_4$	2
IУ	$+0$		$+1_2$			$+3_2$				$+6_3$	$+10_3$	2
У	$+0$			$+1_5$			$+4_2$		$+7_2$		$+9_2$	2
УI	$+0$	-5_2	-5_2	$+1_3$	$+4_2$	$+2_2$		$+6_1$		$+6_1$	$+8_1$	1
УП	$+0$	-5_1	-5_2	$+1_2$	$+4_1$	$+2_1$				$+6_1$	$+10_1$	1
УШ	$+0$			$+1_1$								

Рис. 2.56

2.7.3. Задача о минимальном потоке.

Задача построения минимального потока требует удовлетворения противоположного соотношения между потоком и функцией $C(g)$:

$$\forall g \in G\,(\varphi(g) \geqslant C(g)). \qquad /2.7/$$

Для этой задачи функцию $C(g)$ можно назвать <u>потребностью</u> дуги, и поток должен ее удовлетворить. Естественно, что в такой постановке нужно найти <u>минимальный</u> возможный поток, т.е.

$$V = \sum_{x_i \in G^{-1}(x_c)} \varphi(x_i, x_c) \to min .$$

Построение произвольного потока, удовлетворяющего условие /2.7/.

I. Находится в исходной сети дуга g, для которой $\varphi(g) < C(g)$, и выделяется произвольный путь из вершины x_u к вершине x_c, содержащий эту дугу. Добавляется по дугам этого пути такой поток $\Delta\varphi$, чтобы для всех дуг пути выполнить условие /2.7/:

$$\Delta\varphi = \max_{g \in \text{ выделенному пути.}} \left(C(g) - \varphi(g) \right)$$

Если в сети есть еще дуги, для которых не выполнено условие /2.7/, то снова выполняется п.I, иначе построение потока завершается /рис. 2.57/.

<u>Пример</u>.

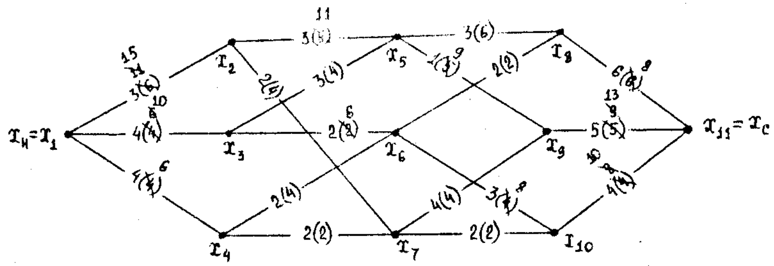


Рис. 2.57

Краткая запись выделяемых путей и необходимых увеличений потоков по каждой дуге приводится в той же нотации, что и для максимального потока.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	$\Delta\varphi$
I путь	0	$+1_3$			$+2_3$			$+5_3$			$+8_6$	6
П путь	0	$+1_0$			$+2_0$				$+5_1$		$+9_5$	5
Ш путь	0	$+1_0$					$+2_2$		$+7_4$		$+9_0$	4
IУ путь	0		$+1_4$		$+3_3$				$+5_0$		$+9_0$	4
У путь	0		$+1_0$			$+3_2$		$+6_2$			$+8_0$	2
УI путь	0		$+1_0$			$+3_0$				$+6_3$	$+10_4$	4
УП путь	0			$+1_4$		$+4_2$				$+6_0$	$+10_0$	4
УШ путь	0			$+1_0$			$+4_2$			$+7_2$	$+10_0$	2

Произвольный поток φ^n, удовлетворяющий условию /2.7/, построен, по многим дугам протекает лишний поток. Для его сокращения строится исходная сеть с пропускными способностями $C^{\wedge}(g) = \varphi^n(g) - C(g)$. В рамках этих пропускных способностей $C^{\wedge}(g)$ строится максимальный лишний поток $\varphi^{\wedge}(g) \leq C^{\wedge}(g)$ /рис. 2.58/.

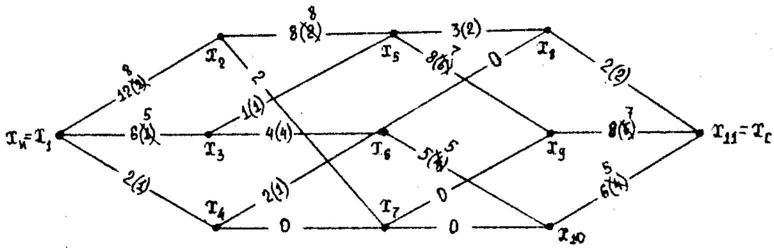


Рис. 2.58

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	$\Delta\varphi$
0	$+1_{12}$			$+2_8$			$+5_3$			$+8_2$	2
0	$+1_{10}$			$+2_6$				$+5_6$		$+9_6$	6
0		$+1_6$		$+3_1$				$+5_2$		$+9_2$	1
0		$+1_5$			$+3_4$				$+6_5$	$+10_6$	4
0			$+1_2$		$+4_2$				$+6_1$	$+10_2$	1
0	$+1_4$	$+1_1$	$+1_1$		$+4_1$	$+2_2$					

Если из произвольного потока $\varphi^n(g)$ вычесть максимальный лишний поток $\varphi^л(g)$, то получится поток $\varphi(g) = \varphi^n(g) - \varphi^л(g)$, который будет минимальным по построению /рис. 2.59/.

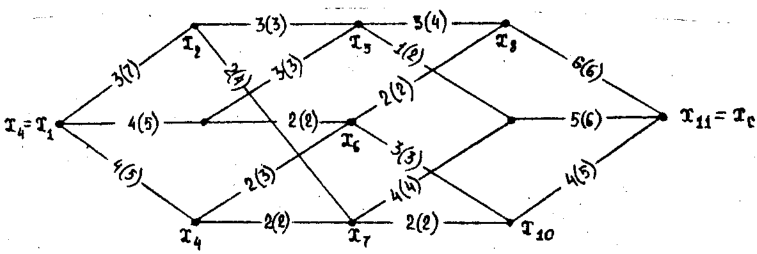


Рис. 2.59

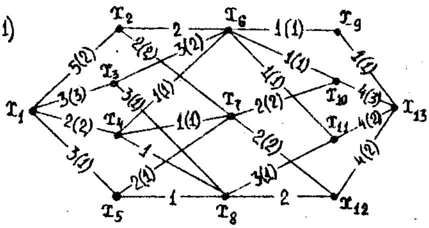
Задача решена.

Формулировка алгоритма построения минимального потока.

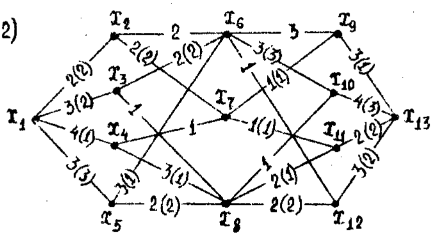
I. Строится произвольный поток $\varphi^n(g)$, удовлетворяющий условию /2.7/.

2.7.4. Варианты индивидуальных заданий к подразд. 2.7

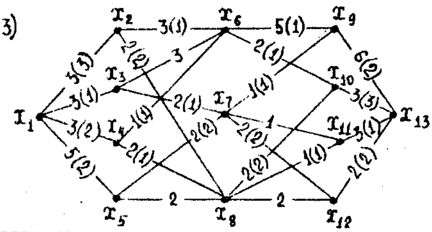
1)



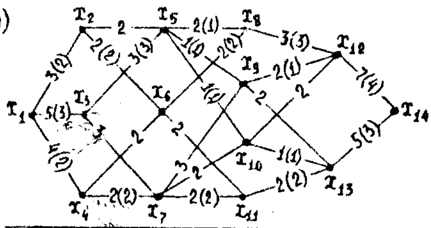
2)



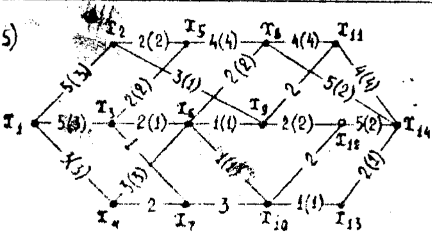
3)



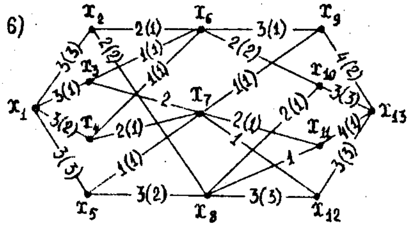
4)



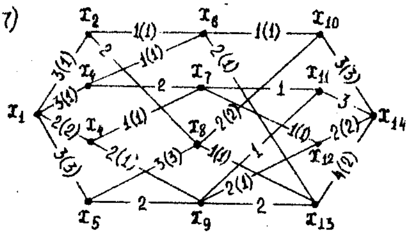
5)



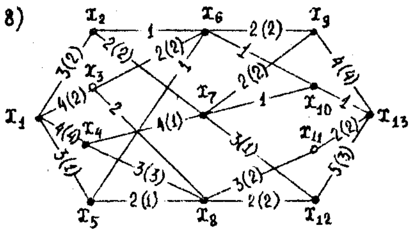
6)



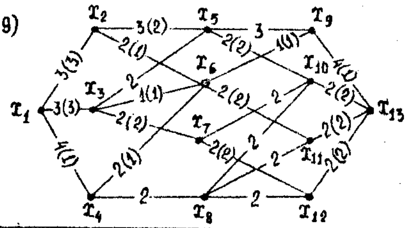
7)



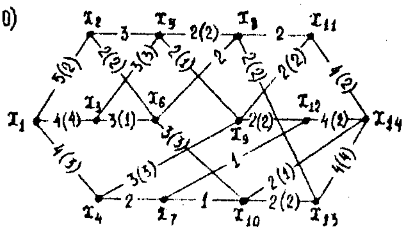
8)



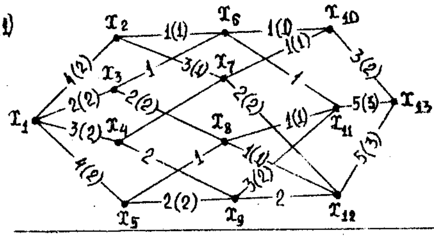
9)



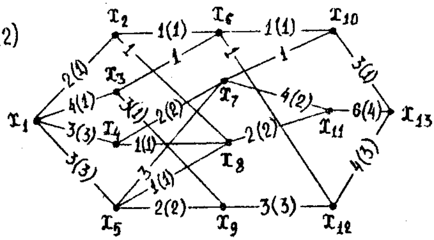
10)



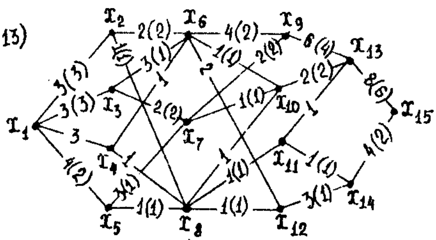
11)



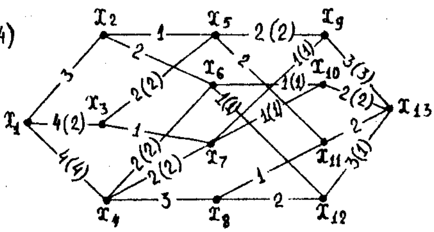
12)



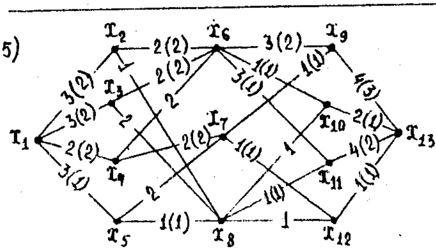
13)



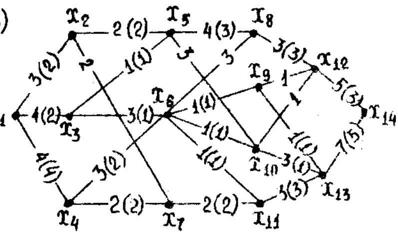
14)



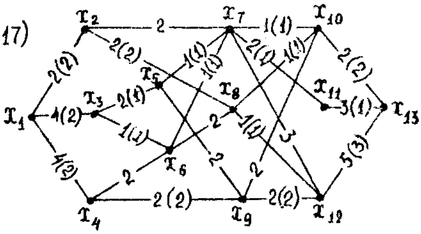
15)



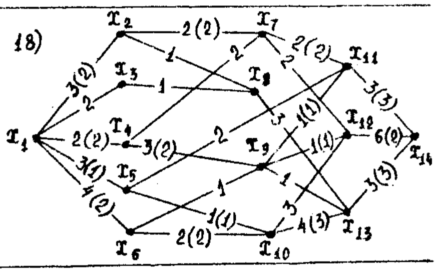
16)



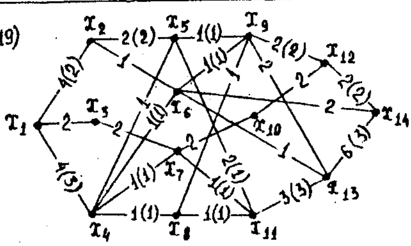
17)



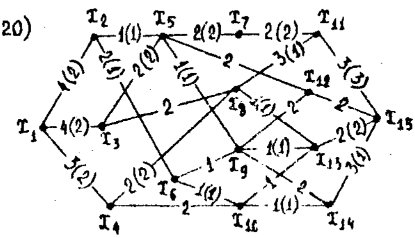
18)



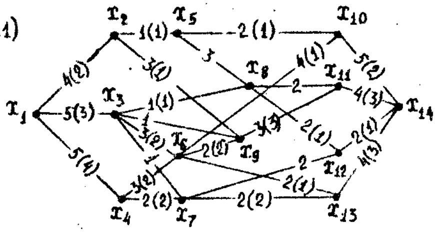
19)



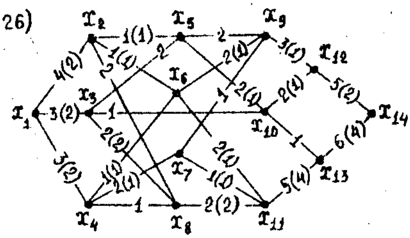
20)



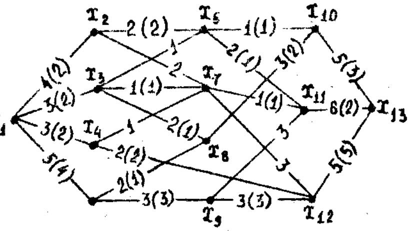
21)



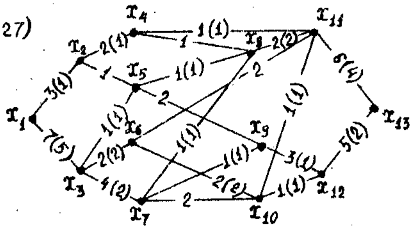
26)



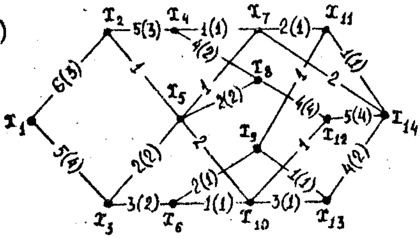
22)



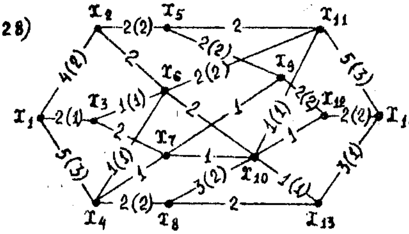
27)



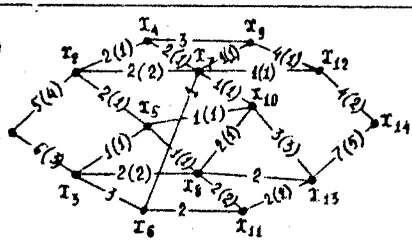
23)



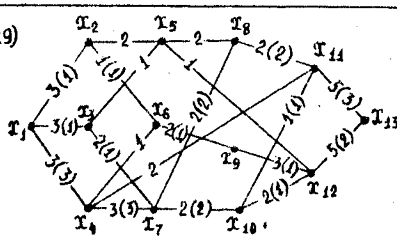
28)



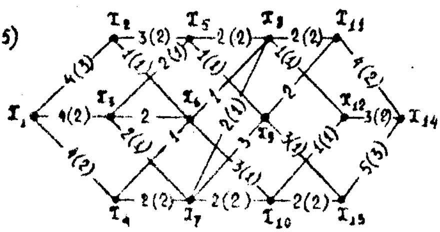
24)



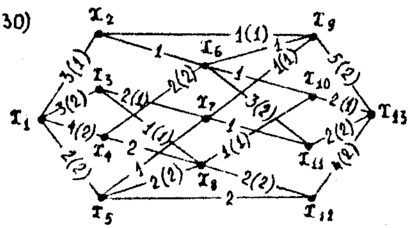
29)



25)



30)



2. В исходной сети формируется пропускная способность $C^1(g) = \varphi^n(g) - C(g)$ и в рамках этой пропускной способности строится максимальный лишний поток $\varphi^1(g)$.

3. Из произвольного потока вычитается максимальный лишний поток, получается искомый минимальный поток

$$\varphi(g) = \varphi^n(g) - \varphi^1(g).$$

2.8. Планарные графы

2.8.1. Определение и свойства планарного графа.

В ряде положений возникает задача изображения графа на плоскости без пересечения ребер, например при изготовлении однослойного печатного монтажа электронных схем /пересечение связей означает появление не предусмотренного контакта/. Граф, для которого можно построить изображение на плоскости без пересечения ребер, называется планарным, а само изображение - плоским.

Пример планарного графа показан на рис. 2.60.

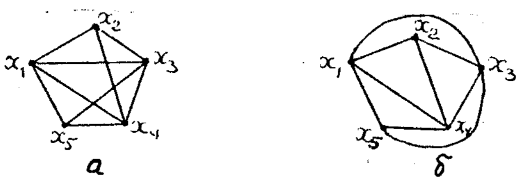


Рис. 2.60

Граф /рис.2.60,а/ планарен, так как для него построено плоское изображение /рис.2.60,б/. Для большинства графов нельзя построить плоское изображение. Такие графы не являются планарными. Классическими примерами непланарных графов являются полный пятивершинник /рис.2.61,а/ и двудольный граф, называемый "три дома и три колодца" /рис.2.61,б/.

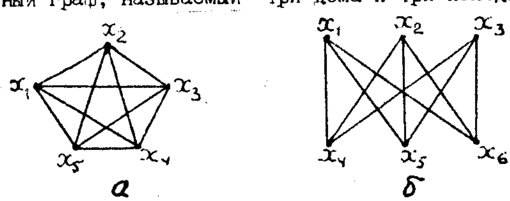


Рис. 2.61

Название последнего непланарного графа связано со следующей интерпретацией. Имеются три дома /x_1, x_2, x_3/ и три колодца /x_4, x_5, x_6/. Необходимо от каждого дома к каждому колодцу проложить дорожки так, чтобы они не пересекались /жители домов не желают встречаться друг с другом/. Непланарность графа "три дома и три колодца" определяет невозможность решения этой задачи.

Введем операцию стягивания вершин: соединение связанных ребром вершин в одну с сохранением всех остальных связей /рис. 2.62/.

Пример.

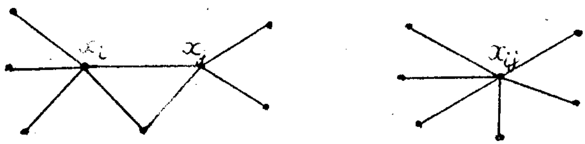


Рис. 2.62

Теорема 2.5 /Понтрягин, Куратовский/. Граф не является планарным, если он содержит подграфы /или подграфы, полученные стягиванием вершин/ вида полного пятивершинника или "три дома и три колодца".

К сожалению, эта теорема не дает алгоритма проверки планарности графа, тем более не представляет алгоритма построения плоского изображения планарного графа. Такой алгоритм изложен в п. 2.8.2.

Плоское изображение графа производит разбиение плоскости на несколько отдельных кусочков - одно бесконечного /внешнего/ и нескольких конечных /внутренних/. Эти кусочки называются гранями.

Пример.

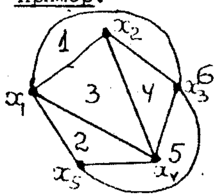


Рис. 2.63

На рис. 2.63 в плоском изображении получены пять конечных граней /I-5/ и одна бесконечная /6/. Цикломатическое число ν равно числу конечных граней -

$$\nu = m - n + p = 9 - 5 + 1 = 5.$$

Теорема 2.6. Края конечных граней плоского изображения образуют систему независимых циклов графа.

Для графа /рис. 2.63/ получается следующая система независимых циклов

I/ $(x_1 x_2)(x_2 x_6)(x_6 x_1)$; 2/ $(x_1 x_5)(x_5 x_4)(x_4 x_1)$; 3/$(x_1 x_4)(x_4 x_2)(x_2 x_1)$;
4/ $(x_2 x_4)(x_4 x_3)(x_3 x_2)$; 5/ $(x_3 x_4)(x_4 x_5)(x_5 x_3)$.

Этой теоремой обычно пользуются в электротехнике и электронике, составляя уравнения Кирхгоффа для напряжений.

Теорема 2.7 о четырех красках. Планарные графы имеют $æ \leq 4$ /рис. 2.64/.

Пример.

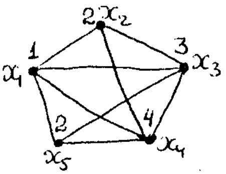


Рис. 2.64

Методом граничного перебора строим максимальные совместимые подмножества:

x_1	x_2	x_3	x_4	x_5	
1					1
	1			1	2
		1			3
			1		4

$æ = 4$

Четыре максимальных совместимых подмножества входят в кратчайшее покрытие /все являются ядерными/.

Географические карты соответствуют плоскому изображению графа соседства стран /вершины - страны, ребра соединяют вершины, соответствующие странам с общей границей/. Для окраски географических карт достаточно четырех красок.

2.8.2. Построение плоского изображения или выявление непланарности произвольного графа.

Решается следующая задача: задан произвольный граф, необходимо либо определить, что он не планарный, либо построить его плоское изображение.

Если граф не содержит циклов, то он планарен /рис. 2.65/. Не формулируя какой-либо алгоритм для этого случая, полагаем, что построение плоского изображения дерева не представляет сложности.

Пример.

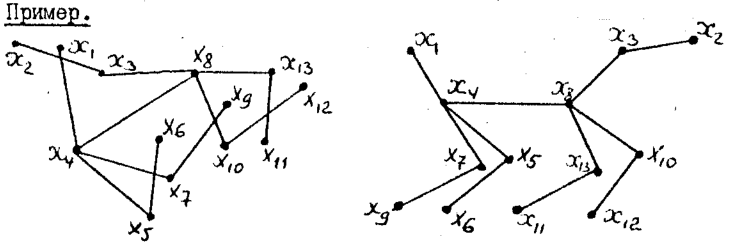


Рис. 2.65

Конечно, этот пример простой. Прежде чем формулировать общий алгоритм, введем некоторые понятия.

Если исходный граф не является деревом, то в нем можно выделить простой цикл, построение плоского изображения которого тоже не представляется сложным /рис. 2.66,б/.

Пример.

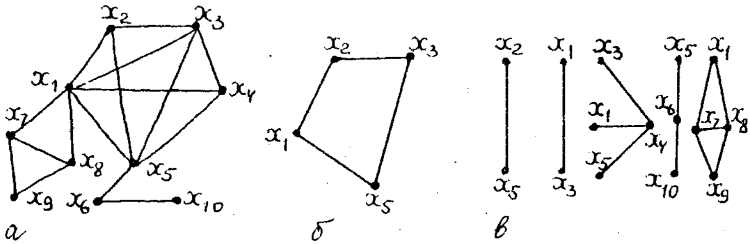


Рис. 2.66

В исходном графе в общем случае остаются вершины и ребра, не вошедшие в полученное плоское изображение выделенного цикла.

Сегментом назовем частичный связный подграф исходного графа, обладающий одним из следующих свойств:

а/ ребро исходного графа, не вошедшее в плоское изображение, вершины которого принадлежат этому изображению /ребра / $x_2 x_5$ / и / $x_1 x_3$ / /;

б/ связная компонента исходного графа, которая содержит одну или несколько вершин, не принадлежащих плоскому изображению, и все ребра, инцидентные этим вершинам /вершина x_4 и ребра /$x_3 x_4$ //$x_1 x_4$ / и /$x_5 x_4$ /; вершины x_6 , x_{10} и ребра /$x_5 x_6$ //$x_6 x_{10}$ /; вершины x_7 , x_8 , x_{10} и ребра /$x_1 x_7$ //$x_1 x_8$ // $x_7 x_8$ //$x_7 x_9$ // $x_8 x_9$ / /. Все сегменты графа /рис. 2.66,а/изображены на рис. 2.66,в. Вершины сегмента, принадлежащие выделенному плоскому изображению, назовем контактными.

Допустимой гранью сегмента назовем грань выделенного плоского подграфа, содержащую все контактные вершины сегмента. Все сегменты для этого примеры /рис. 2.66/ имеют две допустимые грани /конечную и бесконечную/.

В общем случае различаются три ситуации.

I. Для сегмента нет ни одной допустимой грани.

2. Для сегмента существует одна допустимая грань.

3. Для сегмента число допустимых граней больше одной.

В любой допустимой грани можно пытаться строить плоское изображение сегмента, если оно существует. Однако не будем сразу добиваться построения плоского изображения для всего сегмента, достаточно изобразить хотя бы его часть, чтобы получить новое выделенное плоское изображение и новые сегменты. Рассмотрим три ситуации, когда сегмент имеет:

а/ одну контактную вершину и представляет собой дерево, тогда можно построить полное его плоское изображение. При этом число граней не изменится /сегмент / $x_5 x_6$ // $x_6 x_{10}$ / на рис. 2.66/.

б/ одну контактную вершину, но не является деревом, тогда в нем можно выделить простой цикл, содержащий контакную вершину, и присоединить этот цикл к плоскому изображению. Число граней увеличится, и формируются новые сегменты /рис. 2.67/.

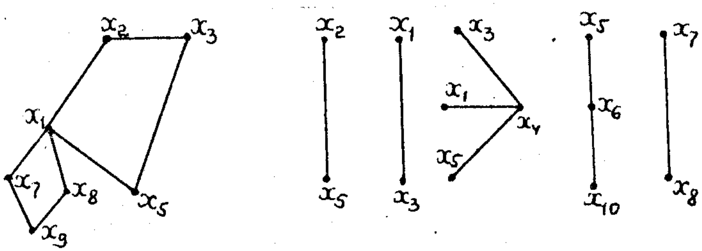


Рис. 2.67

в/ более одной контактной вершины, тогда в нем можно выделить простой маршрут между этими вершинами и включить этот маршрут в плоское изображение. Число граней увеличится, и изменится множество сегментов. /рис. 2.67/. Рассмотрим сегмент с вершиной x_4 и в нем выделим маршрут / $x_3 x_4$ / / $x_4 x_5$ /. Получим результат /рис. 2.68/.

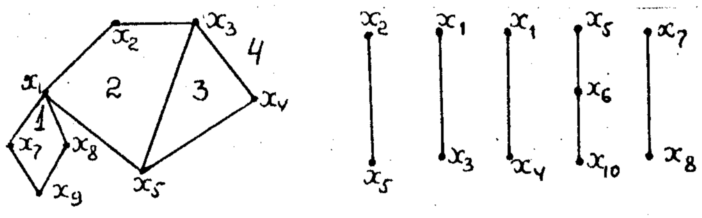


Рис. 2.68

Таким образом, построение плоского изображения планарного графа сводится к последовательному построению текущего плоского изображения частичного подграфа, выбору очередного сегмента и достройке плоского изображения в допустимой грани. Алгоритм фактически сводится к выбору очередного рассматриваемого сегмента и допустимой для него грани. Возможны следующие три ситуации:

а/ имеется сегмент, для которого нет ни одной допустимой грани, тогда построение плоского изображения невозможно, и исходный граф не планарен;

б/ имеется сегмент, для которого есть только одна допустимая грань, тогда существует единственный вариант помещения этого сегмента в этой грани;

в/ для всех сегментов число допустимых граней больше одной, тогда необходимо организовать перебор вариантов, но доказано, что для данной задачи можно обойтись без перебора. В ситуации в/ можно выбрать произвольный сегмент и поместить его в произвольную допустимую для него грань. При этом при любом выборе для планарного графа будет построено плоское изображение, а если на некотором шаге алгоритма реализуется ситуация а/, то вывод о непланарности графа остается справедливым.

Формулировка алгоритма.

I. Если исходный граф - дерево, то строим его плоское изображение, иначе выбираем простой цикл, строим текущее плоское изображение и множество сегментов.

2. Сегменты, имеющие одну контактную точку и являющиеся деревьями, присоединяем к плоскому изображению. Если множество сегментов пусто, то процесс построения плоского изображения завершен, граф планарен. Иначе, если есть сегмент, для которого множество допустимых граней пусто, то граф не планарен. Иначе, если найдется сегмент с одной допустимой гранью, то выбираем его и его допустимую грань, выполняем п.3, иначе выбираем произвольный сегмент и произвольную его допустимую грань, выполняем п.3.

3. Если выбранный сегмент имеет одну контактную точку, то выделяем в нем произвольный простой цикл и выполняем п.4. Если выбранный сегмент имеет две контактные точки, то строим маршрут между этими точками и тоже выполняем п.4.

4. Дополняем выделенные цикл или маршрут в выбранную грань текущего плоского изображения, строим новое множество сегментов. Выполняем п.2.

Продолжим решение примера /рис. 2.68/. Сегменты / $x_2 x_5$ /, / $x_1 x_3$ / и / $x_7 x_8$ / имеют две допустимые грани /2 и 4, 2 и 4 и I и 4/. Сегмент / $x_5 x_6$ / / $x_6 x_{10}$ / - дерево и может быть изображен произвольно в любой допустимой грани 4,2 или 3 /выберем грань 4/. Сегмент / $x_1 x_4$ / имеет одну допустимую грань 4 и является маршрутом между контактными вершинами x_1 , x_4 . Проведем построение /рис. 2.69/.

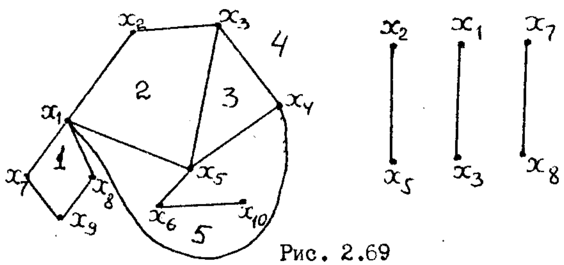


Рис. 2.69

Сегмент / $x_2\ x_5$ / имеет одну допустимую грань. Выбираем сегмент / $x_2\ x_5$ / и грань 2. Получаем новое плоское изображение и сегменты.

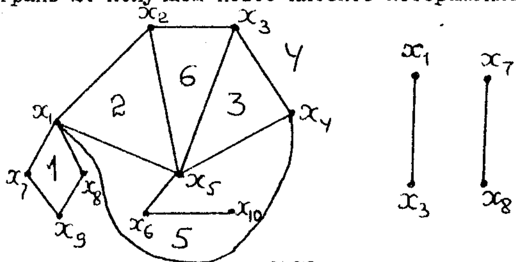


Рис. 2.70

Сегмент / x_1, x_3 / имеет одну допустимую грань 4 и помещается в ней. Для сегмента / $x_7\ x_8$ / выберем грань I /рис.2.70/. Получили пустое множество сегментов и плоское изображение /рис. 2.7I/ исходного графа /см. рис. 2.66/.

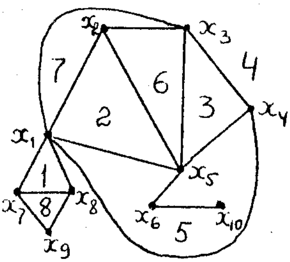


Рис. 2.7I

Рассмотрим пример, проведя решение без пояснений, изобразив процесс решения последовательностью рис. 2.72–2.80, указывая выбранные сегмент и грань восклицательными знаками.

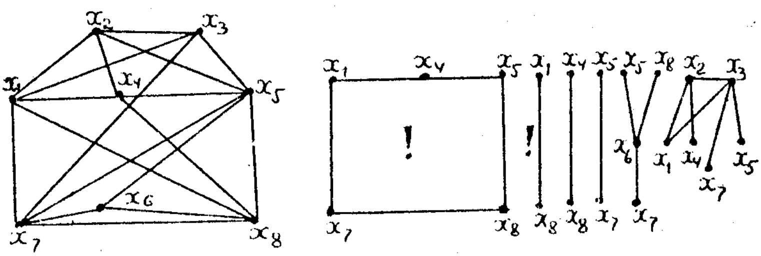


Рис. 2.72

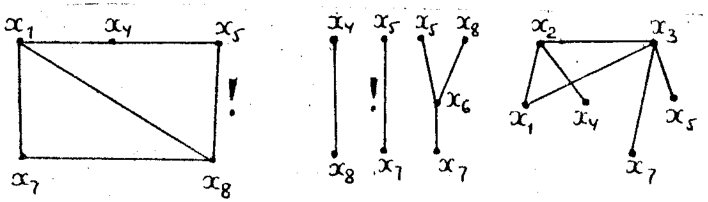


Рис. 2.73

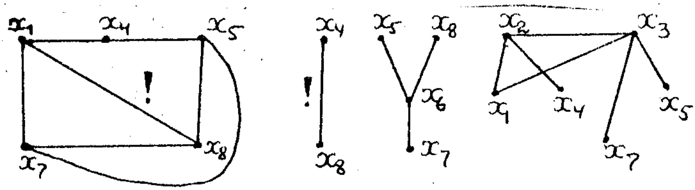


Рис. 2.74

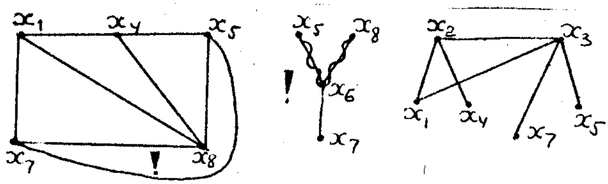


Рис. 2.75

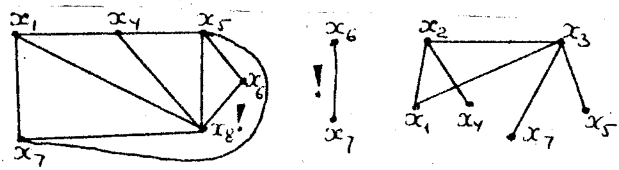


Рис. 2.76

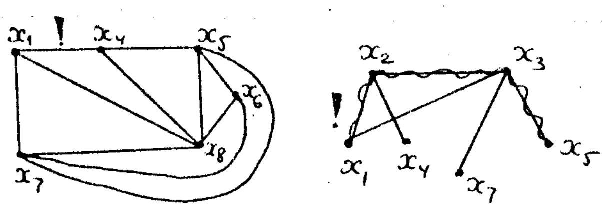


Рис. 2.77

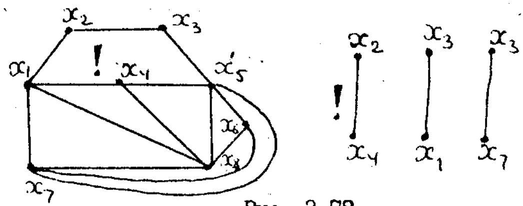


Рис. 2.78

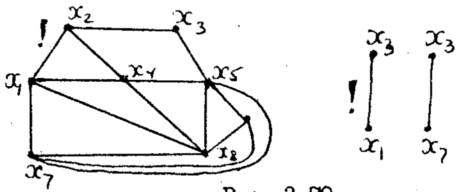


Рис. 2.79

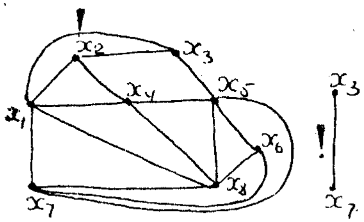


Рис. 2.80

Окончательно на рис. 2.8I построено плоское изображение исходного /см. рис. 2.72/ графа.

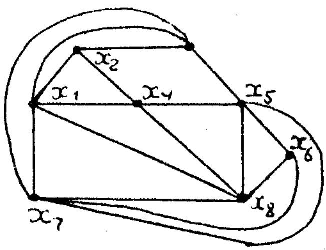
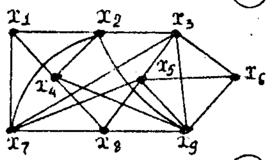


Рис. 2.8I

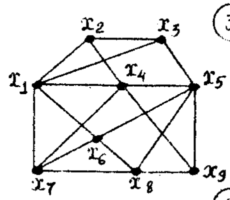
2.8.3. Индивидуальные задания к подразд. 2.8.

Построить плоское изображение следующих графов, записать систему независимых циклов.

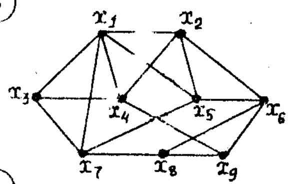
1



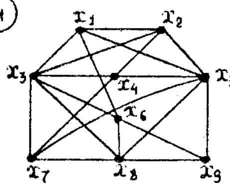
2



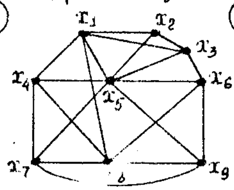
3



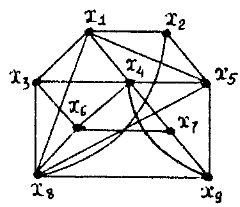
4



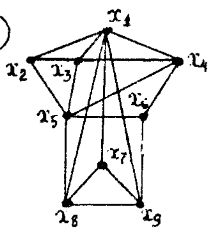
5



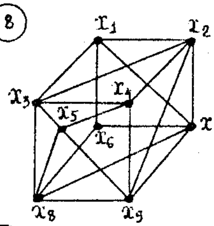
6



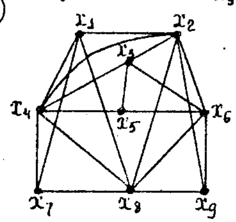
7



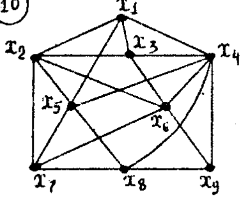
8



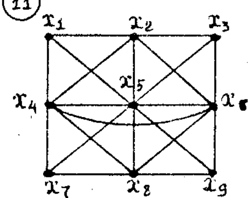
9



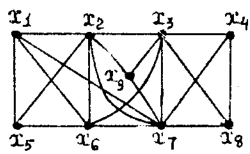
10



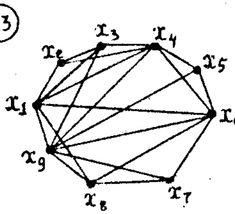
11



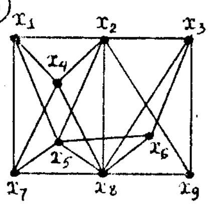
12



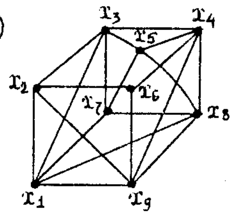
13



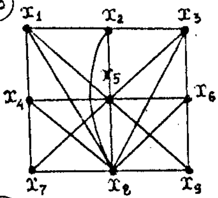
14



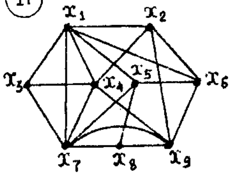
15



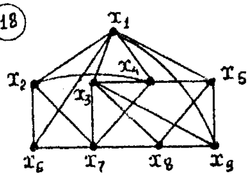
16



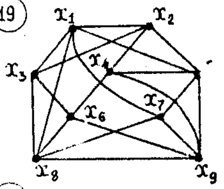
17



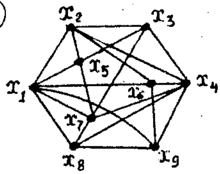
18



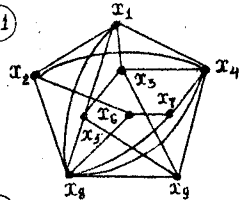
19



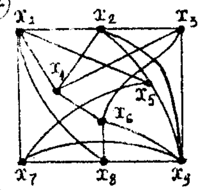
20



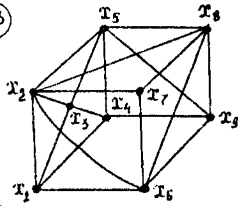
21



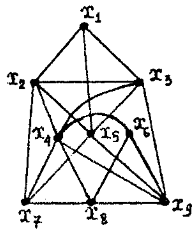
22



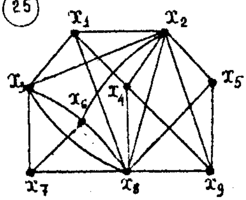
23



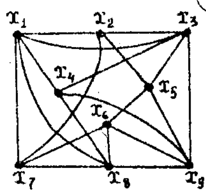
24



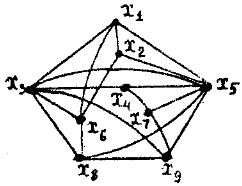
25



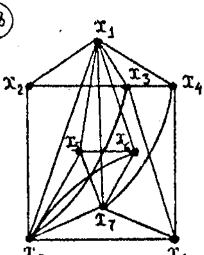
26



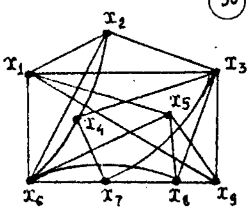
27



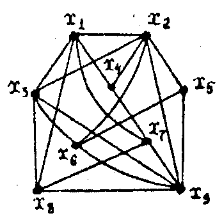
28



29



30



ГЛАВА 3. БУЛЕВЫ ФУНКЦИИ И ИХ ПРЕОБРАЗОВАНИЯ

3.1. Алгебраические формы представления булевых функций

3.1.1. Определения и примеры.

Булевой функцией $f(x_1,\dots,x_n)$ называется двоичная функция двоичных переменных, т.е. f, $x_i \in \{0,1\}$, $1 \le i \le n$. Набором называется комбинация значений n переменных.

Теорема 3.1. Существует 2^n различных наборов n двоичных переменных.

Табличное задание булевой функции. Задается таблица: слева перечисляются все 2^n различных наборов, справа – значение булевой функции на каждом наборе.

Очевидно, что табличное задание однозначно определяет булеву функцию.

Пример двоичного суммирования. При вычислении суммы C двоичных чисел А и В в каждом разряде a_i, b_i участвует еще P_{i-1} – перенос из предыдущего младшего разряда. Очевидно, a_i, b_i и $P_{i-1} \in \{0,1\}$. Результаты c_i и P_i тоже двоичные.

a_i	b_i	p_{i-1}	c_i	p_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Получено табличное задание двух булевых функций $C_i = f_1(a_i, b_i, P_{i-1})$,

$$P_i = f_2(a_i, b_i, P_{i-1}).$$

В таблице наборы удобно перечислять в порядке возрастания значений соответствующих им двоичных чисел.

Пример функции большинства /мажоритарная функция/. Значение булевой функции равно на очередном наборе 1, если в наборе единиц больше, чем нулей, и 0, если нулей больше, чем единиц. Очевидно, функция P_i – пример такой мажоритарной функции.

Пример использования булевой функции для описания поведения физического объекта. Комната с тремя дверями и одной лампочкой, у каждой двери выключатель K_i, изменением состояния x_i которого можно изменить состояние f_λ лампочки.

$f_\lambda \in \{0$ /не горит/, 1 /горит/$\}$, $x_i \in \{0$ /выключен/, 1/включен/$\}$.

При перечислении наборов использован другой, более подходящий для изложения алгоритма работы системы освещения принцип – наборы упорядочены по возрастанию в них единиц. Пусть на наборе 000 лампочка не горит /$f_\lambda(0,0,0)=0$/. Тогда используем следующий алгоритм работы системы

x_1	x_2	x_3	$f_\wedge$
0	0	0	0
0	0	1	1
0	1	0	1
1	0	0	1
0	1	1	0
1	0	1	0
1	1	0	0
1	1	1	1

освещения - при нечетном числе включенных K_i лампочка горит, при четном - выключена. Очевидно, поворотом каждого выключателя /переходом от чет к нечет/ можно изменить состояние лампочки.

Булевы функции - математическое описание алгоритма работы дискретных устройств /комбинационных схем/. Цифровые вычислительные машины и другие устройства дискретного действия используют физические сигналы, квантованные по значению на два уровня /рис. 3.I/
$U_1^{min} \le U_1 \le U_1^{max}$, $U_0^{min} \le U_0 \le U_0^{max}$.

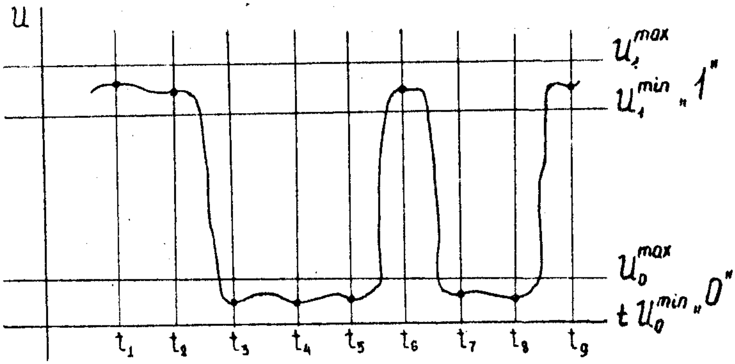


Рис. 3.I

В моменты использования / t_1 , t_2 , .../ значение физического сигнала U должно находиться либо в области U_1 , либо в области U_0 . Одно из этих значений можно сопоставить с I, другое - с 0. Таким образом, комбинационные схемы могут быть представлены следующей моделью /рис. 3.2/, где все $f_i(x_1,...,x_n)$ - булевы функции. Каждому физическому сигналу U_i сопоставляется двоичная переменная.

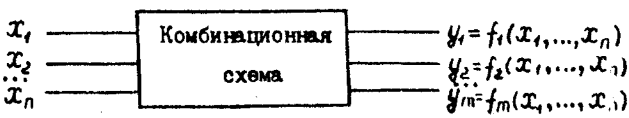


Рис. 3.2

Одноразрядный двоичный сумматор изображен на рис. 3.3.

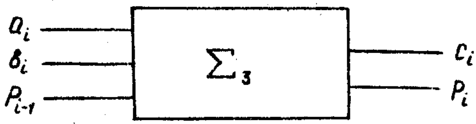


Рис. 3.3

Перечисляющее задание булевых функций. Табличное описание булевой функции - избыточно, достаточно задавать одно из множеств M_1 или M_0 наборов, на которых функция равна соответственно I или 0.

Для функции $f_\wedge$

$$M_1: \begin{array}{ccc} x_1 & x_2 & x_3 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array} \qquad M_0: \begin{array}{ccc} x_1 & x_2 & x_3 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

Задание одного из этих множеств однозначно определяет второе.

Теорема 3.2. Существуют 2^{2^n} различные булевы функции n переменных;

n	0	I	2	3	4	5	6
2^n	I	2	4	8	I6	32	64
2^{2^n}	2	4	I6	256	65536	$4{,}3\cdot 10^9$	$16\cdot 10^{18}$

Основные булевы функции одной и двух переменных. При $n = 0$ две булевы функции - константы 0 и I. При $n = $ I четыре булевы функции, константы 0 и I, переменная x и инверсия $\bar{x}$ переменной /функция НЕ/.

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

При $n = 2$ шестнадцать булевых функций. Приведем только те из них, которые получили специальные названия и применение.

Название функции	Обозначение функции
Конъюнкция, И, логическое умножение	$x_1 \wedge x_2 = x_1 x_2$ /знак конъюнкции можно опустить/
Дизъюнкция, ИЛИ, логическое сложение	$x_1 \vee x_2$
Сумма по модулю 2, ИЛИ с исключением, сравнение	$x_1 \oplus x_2$
Эквивалентность	$x_1 \equiv x_2 = x_1 \sim x_2$
Стрелка Пирса, ИЛИ-НЕ	$\overline{x_1 \vee x_2}$
Штрих Шеффера, И-НЕ	$\overline{x_1 x_2}$
Импликация	$x_1 \rightarrow x_2$

Табличное определение функций

x_1	x_2	$x_1\ x_2$	$x_1 \vee x_2$	$x_1 \oplus x_2$	$x_1 \equiv x_2$	$\overline{x_1 \vee x_2}$	$\overline{x_1\ x_2}$	$x_1 \rightarrow x_2$
0	0	0	0	0	1	1	1	1
0	1	0	1	1	0	0	1	1
1	0	0	1	1	0	0	1	0
1	1	1	1	0	1	0	0	1

В теории дискретных автоматов и программировании наиболее распространены функции конъюнкции, дизъюнкции, сложения по модулю 2; при преобразованиях схем дискретных устройств применяются операции И-НЕ, ИЛИ-НЕ. Операция импликации используется в теории доказательств: x_1 - посылка; x_2 - следствие, 0 - ложь, I - истина. Из ложной посылки можно вывести что угодно, из правильной посылки - только истинное следствие. Таким образом, выражение вида $x_1 \rightarrow x_2$ - это утверждение /теорема/, которое верно, если из правильной посылки выведено истинное следствие. Выражение $x_1 \rightarrow x_2$ можно прочитать так: если x_1, то x_2.

3.I.2. Булева алгебра.

Обобщение операций конъюнкции и дизъюнкции n переменных.

Необходимо отметить два момента: в операциях могут участвовать более, чем две переменных; переменные могут входить как в прямом виде, так и с инверсией.

Для конъюнкции: если хотя бы один из членов конъюнкции равен нулю, то вся конъюнкция равна нулю; конъюнкция равна единице, только если все ее члены равны единице.

Для дизъюнкции: если хотя бы один из членов дизъюнкции равен единице, то вся дизъюнкция равна единице; дизъюнкция равна нулю, только если все ее члены равны нулю.

$$x_i^{\sigma_i} = \begin{cases} x_i\text{, при } \sigma_i = 1\text{.} & \text{Конъюнкция} - x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n}. \\ \bar{x}_i\text{, при } \sigma_i = 0\text{.} & \text{Дизъюнкция} - x_1^{\sigma_1} \vee x_2^{\sigma_2} \vee \ldots \vee x_n^{\sigma_n}. \end{cases}$$

Примеры.

x_1	x_2	x_3	$x_1 \bar{x}_2 \bar{x}_3$	$\bar{x}_1 x_2 x_3$	$x_1 \vee \bar{x}_2 \vee \bar{x}_3$	$\bar{x}_1 \vee x_2 \vee x_3$
0	0	0	0	0	1	1
0	0	1	0	0	1	1
0	1	0	0	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	1
1	1	0	0	0	1	1
1	1	1	0	0	1	1

Аксиомы и теоремы булевой алгебры. При использовании трех операций /конъюнкции, дизъюнкции и инверсии/, констант 0 и I и двоичных переменных $x_1, \ldots, x_n$ строится булева алгебра, основу которой составляют следующие аксиомы:

I/ двойное отрицание $\bar{\bar{x}} = x$;

2/ идемпотентность $x \wedge x = xx = x$, $x \vee x = x$;

3/ коммутативность $x_1 \vee x_2 = x_2 \vee x_1$, $x_1 x_2 = x_2 x_1$;

4/ ассоциативность $x_1 \vee x_2 \vee x_3 = x_1 \vee (x_2 \vee x_3)$, $x_1 x_2 x_3 = x_1 (x_2 x_3)$;

5/ дистрибутивность $x_1(x_2 \vee x_3) = x_1 x_2 \vee x_1 x_3$, $(x_1 \vee x_2)(x_1 \vee x_3) = x_1 \vee x_2 x_3$;

6/ аксиома де'Моргана $\overline{x_1 \vee x_2} = \bar{x}_1 \bar{x}_2$, $\overline{x_1 x_2} = \bar{x}_1 \vee \bar{x}_2$;

7/ исключенного третьего $x_1 \vee \bar{x}_1 = 1$, $x_1 \bar{x}_1 = 0$;

8/ операции с константами $x \vee 1 = 1$, $x \vee 0 = x$;

$x \wedge 1 = x$, $x \wedge 0 = 0$.

Не все аксиомы независимы, часть из них можно вывести из других. Например, аксиому дистрибутивности $(x_1 \vee x_2)(x_1 \vee x_3) = x_1 \vee x_2 x_3$ можно доказать $(x_1 \vee x_2)(x_1 \vee x_3)$:

по первой аксиоме дистрибутивности

$$(x_1 \vee x_2)(x_1 \vee x_3) = (x_1 \vee x_2) x_1 \vee (x_1 \vee x_2) x_3 = x_1 x_1 \vee x_2 x_1 \vee x_1 x_3 \vee x_2 x_3;$$

по аксиоме идемпотентности $x_1 x_1 = x_1$;

согласно операциям с константами $x_1 = x_1 \wedge 1$;

по аксиоме дистрибутивности $x_1 \wedge 1 \vee x_1 x_2 \vee x_1 x_3 = x_1(1 \vee x_2 \vee x_3)$;

согласно операциям с константами $1 \vee x_2 \vee x_3 = 1$;

окончательно $(x_1 \vee x_2)(x_1 \vee x_3) = x_1 \vee x_2 x_3$.

Это и есть вторая аксиома дистрибутивности. Одновременно получена иллюстрация использования аксиом булевой алгебры для преобразования выражений.

На основе аксиом доказаны следующие основные теоремы:

3.3 /поглощения/ $x_1 \vee x_1 x_2 = x_1$.

3.4 /склеивания/ $x_1 x_2 \vee x_1 \bar{x}_2 = x_1$.

3.5 /неполного склеивания/ $x_1 \vee \bar{x}_1 x_2 = x_1 \vee x_2$.

3.6 /обобщенного склеивания/ $x_1 x_2 \vee \bar{x}_2 x_3 \vee x_1 x_3 = x_1 x_2 \vee \bar{x}_2 x_3$.

3.7 /разложения функции $f(x_1, \ldots, x_n)$ /:

дизъюнктивное

$$f(x_1, \ldots, x_n) = x_1 f(1, x_2, \ldots, x_n) \vee \bar{x}_1 f(0, x_2, \ldots, x_n);$$

конъюнктивное

$$f(x_1, \ldots, x_n) = (x_1 \vee f(0, x_2, \ldots, x_n))(\bar{x}_1 \vee f(1, x_2, \ldots, x_n)).$$

3.1.3. Основные алгебраические функции.

Обобщенное представление конъюнкции. Конъюнкция многих переменных может содержать как сами переменные, так и их инверсии. Например: $x_1 \bar{x}_2 \bar{x}_3$. Эта конъюнкция равна единице на единственном наборе 100, на остальных равна 0. Таким образом, любая конъюнкция n переменных /или их инверсий/ равна единице на единственном наборе. Набор значений переменных можно в обобщенном виде представить как последовательность констант σ_i, $1 \le i \le n$: $\sigma_1 \sigma_2 \ldots \sigma_n$. Введем условное обозначение:

$$x_i^{\sigma_i} = \begin{cases} x_i & \text{при } \sigma_i = 1, \\ \bar{x}_i & \text{при } \sigma_i = 0. \end{cases}$$

Очевидно, что для набора 100 можно построить конъюнкцию $x_1^1 x_2^0 x_3^0 = x_1 \bar{x}_2 \bar{x}_3$. Тогда для произвольного набора $\sigma_1 \sigma_2 \ldots \sigma_n$ строится конъюнкция $x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_n^{\sigma_n}$, которая равна единице на этом единственном наборе.

Совершенная дизъюнктивная нормальная форма /ДНФ/ представления булевой функции. Это одна из наиболее часто используемых алгебраических форм.

Теорема 3.8. Любую булеву функцию можно представить в совершенной ДНФ, т.е. в следующем виде:

$$f(x_1,\ldots,x_n)=\bigvee_{\sigma_1\ldots\sigma_n\in M_1} x_1^{\sigma_1}\ldots x_n^{\sigma_n}. \qquad /3.1/$$

Эта формула имеет тот же смысл, что и формула вида $\sum_{i\in I} Q_i$. Формула /3.1/ определяет алгоритм построения совершенной ДНФ по множеству M_1 наборов, на которых функция равна единице.

Пример:

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

$$f=\bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3$$

Доказательство формулы /3.1/ можно провести конструктивно: при подстановке каждого из 2^n наборов дизъюнктивная форма превращается в ту константу /0 или 1/, которой равна функция на этом наборе: при подстановке набора $\sigma_1\ldots\sigma_n\in M_1$ в совершенной ДНФ найдется конъюнкция $x_1^{\sigma_1}\ldots x_n^{\sigma_n}$, которая станет равна единице, в ДНФ достаточно одной единицы, чтобы она была равна 1; при подстановке набора $\sigma_1\ldots\sigma_n\in M_0$ все конъюнкции совершенной ДНФ станут равны нулю, и вся ДНФ равна нулю.

Доказательство формулы /3.1/, основанное на алгебраических преобразованиях. По теореме дизъюнктивного разложения

$$f(x_1,\ldots,x_n)=x_1 f(1,x_2,\ldots,x_n)\vee \bar{x}_1 f(0,x_2,\ldots,x_n).$$

Повторное применение теоремы дизъюнктивного разложения дает

$$f(x_1,\ldots,x_n)=x_1(x_2 f(1,1,\ldots)\vee\bar{x}_2 f(1,0,\ldots))\vee\bar{x}_1(x_2 f(0,1,\ldots)\vee\bar{x}_2 f(0,0,\ldots)).$$

По аксиоме дистрибутивности

$$f=x_1x_2 f(1,1,\ldots,x_n)\vee x_1\bar{x}_2 f(1,0,\ldots,x_n)\vee\bar{x}_1 x_2 f(0,1,\ldots,x_n)\vee\bar{x}_1\bar{x}_2 f(0,0,\ldots,x_n).$$

Последнюю формулу можно условно записать в следующем виде:

$$f(x_1,\ldots,x_n)=\bigvee_{\sigma_1\sigma_2=00}^{\sigma_1\sigma_2=11} x_1^{\sigma_1}x_2^{\sigma_2} f(\sigma_1,\sigma_2,x_3,\ldots,x_n).$$

Проведя дизъюнктивное разложение по остальным переменным $x_3,\ldots,x_n$, получим

$$f(x_1,\dots,x_n)=\bigvee_{\sigma_1\dots\sigma_n=0\dots0}^{\sigma_1\dots\sigma_n=1\dots1} x_1^{\sigma_1}\dots x_n^{\sigma_n} f(\sigma_1,\dots,\sigma_n).$$

В этой формуле 2^n конъюнкций, причем $f(\sigma_1,\dots,\sigma_n)$- константа, равная единице на наборе $\sigma_1\dots\sigma_n\in M_1$ и 0 на наборе $\sigma_1\dots\sigma_n\in M_0$, но логическое умножение на 0 превращает всю конъюнкцию в 0 и она пропадает из ДНФ, а умножение на I можно не указывать, т.е. получаем /3.I/.

Совершенная конъюнктивная нормальная форма (КНФ) булевой функции.

Теорема 3.9. Любую булеву функцию можно представить в совершенной КНФ, т.е. в следующем виде:

$$f(x_1,\dots,x_n)=\bigwedge_{\sigma_1\dots\sigma_n\in M_0}(x_1^{\bar\sigma_1}\vee\dots\vee x_n^{\bar\sigma_n}). \qquad /3.2/$$

Доказательство проведем, использовав закон де Моргана.

$$\bar f=(x_1,\dots,x_n)=\bigvee_{\sigma_1\dots\sigma_n\in M_0} x_1^{\sigma_1}\dots x_n^{\sigma_n}.$$

Функция $\bar f$ принимает единичное значение на наборах множества M_0.

$$f=\bar{\bar f}=\overline{\bigvee_{\sigma_1\dots\sigma_n\in M_0} x_1^{\sigma_1}\dots x_n^{\sigma_n}}=$$

$$\bigwedge_{\sigma_1\dots\sigma_n\in M_0}\overline{x_1^{\sigma_1}\dots x_n^{\sigma_n}}=\bigwedge_{\sigma_1\dots\sigma_n\in M_0}(x_1^{\bar\sigma_1}\vee\dots\vee x_n^{\bar\sigma_n}).$$

Получена формула /3.2/.

Пример.

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

$$p_i=(x_1\vee x_2\vee x_3)(x_1\vee x_2\vee\bar x_3)(x_1\vee\bar x_2\vee x_3)(\bar x_1\vee x_2\vee x_3)$$

16*

Совершенная шефферовская нормальная форма /НФ/

$$f(x_1,\dots,x_n)=\bigwedge_{\sigma_1\dots\sigma_n\in M_1}\overline{x_1^{\sigma_1}\dots x_n^{\sigma_n}}.$$

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

$$p_i=\overline{\overline{\bar x_1\,x_2\,x_3}\;\;\overline{x_1\,\bar x_2\,x_3}\;\;\overline{x_1\,x_2\,\bar x_3}\;\;\overline{x_1\,x_2\,x_3}}$$

Совершенная пирсовская НФ

$$f(x_1,\dots,x_n)=\bigvee_{\sigma_1\dots\sigma_n M_0}\overline{x_1^{\bar\sigma_1}\vee\dots\vee x_n^{\bar\sigma_n}}\ .$$

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

$$p_i=\overline{\overline{x_1\vee x_2\vee x_3}\vee\overline{x_1\vee x_2\vee\bar x_3}\vee\overline{x_1\vee\bar x_2\vee x_3}\vee\overline{\bar x_1\vee x_2\vee x_3}}$$

Доказательства обеих форм осуществляются применением аксиом двойного отрицания и де'Моргана соответственно к совершенной ДНФ и КНФ.

Совершенная полиномиальная НФ

$$f(x_1,\dots,x_n)=a_0\oplus a_1x_1\oplus\dots\oplus a_nx_n\oplus a_{n+1}x_1x_2\oplus\dots\oplus a_{\frac{n(n+1)}{2}}x_{n-1}x_n\oplus$$
$$\oplus\, a_{\frac{n(n+1)}{2}+1}x_1x_2x_3\oplus\dots\oplus a_{2^n-1}x_1x_2\dots x_n.$$
$$a_i\in\{0,1\},\quad 0\le i\le 2^n-1.$$

Совершенная полиномиальная НФ для каждой булевой функции определяется набором значений коэффициентов $a_0,\dots,a_{2^n-1}$, и если $a_i=0$, то соответствующий член отсутствует, а если $a_i=1$, то присутствует в форме.

Приведем конструктивное доказательство и одновременно алгоритм построения совершенной полиномиальной НФ. В совершенной ДНФ операцию $\vee$

можно заменить на $\oplus$, так как в этой форме при подстановке любого набора возможна одна из двух следующих ситуаций:

$$0 \vee 0 \vee \ldots \vee 0 = 0 \oplus 0 \oplus \ldots \oplus 0 \quad \text{при} \quad \sigma_1 \ldots \sigma_n \in M_0,$$

$$0 \vee 1 \vee 0 \vee \ldots \vee 0 = 0 \oplus 1 \oplus 0 \oplus \ldots \oplus 0 \quad \text{при} \quad \sigma_1 \ldots \sigma_n \in M_1.$$

Во всех конъюнкциях можно провести замену $\bar{x}_i = (1 \oplus x_i)$, затем раскрыть скобки и привести подобные по правилам

$$K_i \oplus K_i = 0, \qquad K_i \oplus K_i \oplus K_i = K_i.$$

В результате будет построена совершенная полиномиальная НФ.

Примеры:

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

$$p_i = \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 \vee x_1 x_2 x_3 =$$
$$= \bar{x}_1 x_2 x_3 \oplus x_1 \bar{x}_2 x_3 \oplus x_1 x_2 \bar{x}_3 \oplus x_1 x_2 x_3 =$$
$$= (1 \oplus x_1) x_2 x_3 \oplus x_1 (1 \oplus x_2) x_3 \oplus x_1 x_2 (1 \oplus x_3) \oplus x_1 x_2 x_3 =$$
$$= x_2 x_3 \oplus \underline{x_1 x_2 x_3} \oplus x_1 x_3 \oplus \underline{x_1 x_2 x_3} \oplus x_1 x_2 \oplus x_1 x_2 x_3 \oplus x_1 x_2 x_3 =$$
$$= x_2 x_3 \oplus x_1 x_3 \oplus x_1 x_2.$$
$$a_0 = a_1 = a_2 = a_3 = 0, \quad a_4 = a_5 = a_6 = 1, \quad a_7 = 0.$$

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

$$f = \bar{x}_1 \bar{x}_2 \bar{x}_3 \oplus \bar{x}_1 x_2 x_3 \oplus x_1 \bar{x}_2 x_3 =$$
$$= (1 \oplus x_1)(1 \oplus x_2)(1 \oplus x_3) \oplus (1 \oplus x_1) x_2 x_3 \oplus x_1 (1 \oplus x_2) x_3 =$$
$$= 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_1 x_2 \oplus x_1 x_3 \oplus x_2 x_3 \oplus x_1 x_2 x_3 \oplus$$
$$\oplus x_2 x_3 \oplus \underline{x_1 x_2 x_3} \oplus x_1 x_3 \oplus \underline{x_1 x_2 x_2} =$$
$$= 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_1 x_2 \oplus x_1 x_2 x_3;$$
$$a_0 = a_1 = a_2 = a_3 = a_4 = 1, \quad a_5 = a_6 = 0, \quad a_7 = 1.$$

3.1.4. Произвольные ДНФ.

Обобщая теоремы склеивания, поглощения, неполного склеивания для элементарных конъюнкций произвольного множества переменных, получаем операции над ДНФ.

Операция склеивания. Две конъюнкции K_1 и K_2 склеиваются, т.е. заменяются в ДНФ одной конъюнкцией K, если: а/ K_1 и K_2 содержат одинаковое множество переменных; б/ одна и только одна переменная входит в K_1 и K_2 разнотипно /т.е. в одну с инверсией, в другую без инверсии/.

Конъюнкция K получается из K_1 удалением этой разнотипной переменной:

$$K_1 \vee K_2 = x_1 \bar{x}_3 x_4 \bar{x}_6 \vee x_1 \bar{x}_3 \bar{x}_4 \bar{x}_6 = x_1 \bar{x}_3 \bar{x}_6 = K.$$

Несклеивающиеся конъюнкции:

$x_1 \bar{x}_3 x_4 \bar{x}_5 \vee x_1 \bar{x}_3 \bar{x}_4 \bar{x}_6$ - разные множества переменных.

$x_1 \bar{x}_3 x_4 \bar{x}_6 \vee x_1 x_3 \bar{x}_4 \bar{x}_6$ - две переменных разнотипны.

<u>Операция поглощения</u>. Конъюнкция K_1 поглощает K_2, т.е. конъюнкция K_2 может быть удалена из ДНФ, содержащей K_1, если все переменные K_1 входят в K_2 и каждая входит в обе конъюнкции однотипно:

$K_1 \vee K_2 = x_1 \bar{x}_4 \vee x_1 \bar{x}_2 \bar{x}_4 x_6 = x_1 \bar{x}_4 = K_1$;

$x_1 \bar{x}_4 x_5 \vee x_1 \bar{x}_2 \bar{x}_4 \bar{x}_5$, переменная x_5 входит разнотипно;

$x_1 \bar{x}_4 x_5 \vee x_1 \bar{x}_2 x_4 x_6$, переменная x_5 входит в K_1 и не входит в K_2.

<u>Операция неполного склеивания</u>. Конъюнкции K_1 и K_2 допускают операцию неполного склеивания, если все переменные K_1 входят в K_2 и одна и только одна из них входит в конъюнкции разнотипно. Тогда эта разнотипная переменная может быть удалена из K_2, т.е.

$$K_1 \vee K_2 = x_1 \bar{x}_2 \bar{x}_4 \vee x_1 x_2 x_3 \bar{x}_4 = x_1 \bar{x}_2 \bar{x}_4 \vee x_1 x_3 \bar{x}_4;$$

$x_1 \bar{x}_2 \bar{x}_4 \vee x_1 x_2 x_3 \bar{x}_5$ - переменная $\bar{x}_4$ входит в K_1 и не входит в K_2;

$x_1 \bar{x}_2 \bar{x}_4 \vee x_1 x_2 \bar{x}_3 x_4$ - разнотипны более одной переменной.

<u>Операция обобщенного склеивания</u>. Две конъюнкции K_1 и K_2 поглощают третью K_3, т.е. $K_1 \vee K_2 \vee K_3 = K_1 \vee K_2$, если в конъюнкции K_1 и K_2 одна и только одна переменная входит разнотипно, в конъюнкцию K_3 входят все переменные конъюнкций K_1 и K_2, кроме разнотипной

$$x_1 \bar{x}_2 \bar{x}_4 x_5 \vee x_1 x_2 x_3 x_6 \vee x_1 x_3 \bar{x}_4 x_5 x_6 = x_1 \bar{x}_2 \bar{x}_4 x_5 \vee x_1 x_2 x_3 x_6.$$

<u>Пример 3.1</u> использования рассмотренных операций для упрощения ДНФ.

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

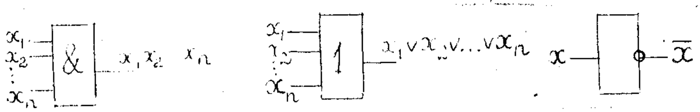
Проведем склеивание конъюнкций в совершенной ДНФ:

$$f(x_1 x_2 x_3) = \bar{x}_1 \bar{x}_2 x_3 \vee \bar{x}_1 x_2 \bar{x}_3 \vee$$
$$\vee \bar{x}_1 x_2 x_3 \vee x_1 \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_2 x_3 \vee x_1 x_2 \bar{x}_3 =$$
$$= \underbrace{\bar{x}_2 x_3}_{K_1} \vee \bar{x}_1 x_2 \vee \underbrace{x_1 \bar{x}_3}_{K_2} \vee \underbrace{x_1 \bar{x}_2}_{K_3}.$$

Теперь выполним операцию обобщенного склеивания:

$$f(x_1 x_2 x_3) = \bar{x}_2 x_3 \vee \bar{x}_1 x_2 \vee x_1 \bar{x}_3.$$

Булевы функции и задача синтеза (проектирования) комбинационных схем. Дискретные устройства строятся на микросхемах, алгоритмы работы которых описываются булевыми функциями. Совокупность применяемых при синтезе микросхем составляет элементный базис проектирования. Синтез схем дискретных устройств в различных базисах проектирования изучают в дисциплине "Прикладная теория цифровых автоматов". Простейшим элементным базисом /редко используемым/ служит базис И, ИЛИ, НЕ, который в схемотехнике имеет следующее обозначение /рис. 3.4/.



Элемент И Элемент ИЛИ Элемент НЕ

Рис. 3.4

В этом базисе по совершенной и упрощенной ДНФ булевой функции $f(x_1,x_2,x_3)$ /пример 3.I/ вполне естественно синтезируются следующие двухуровневые комбинационные схемы /рис. 3.5/.

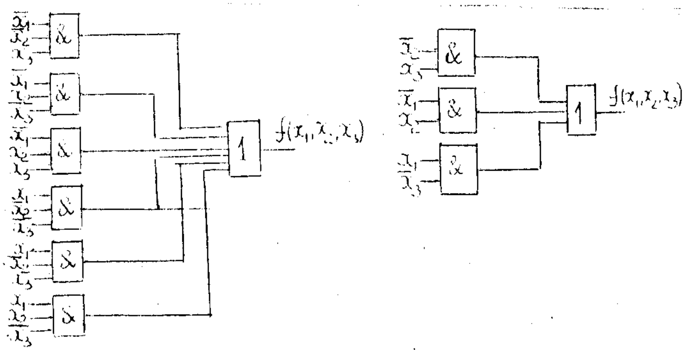


Рис. 3.5

Приведенный пример позволяет сделать два следующих вывода: представление булевой функции в ДНФ довольно полно отражает структуру двухуровневой комбинационной схемы, упрощение ДНФ /сокращение конъюнкций и числа букв в них/ соответствует упрощению схемы /уменьшению числа элементов и числа входов на них/.

Пример. Для функции $f(x_1,x_2,x_3,x_4)$ с использованием операции склеивания и неполного склеивания проведем упрощение ДНФ:

$$
\begin{aligned}
f(x_1,\dots,x_n) &= \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3\bar{x}_4 \vee \bar{x}_1\bar{x}_2x_3x_4 \vee \\
&\quad x_1x_2x_3x_4 \vee \bar{x}_1x_2x_3x_4 \vee x_1\bar{x}_2x_3\bar{x}_4 \vee x_1x_2x_3\bar{x}_4 \vee x_1\bar{x}_2x_3x_4 = \\
&= \bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_1x_2x_3 \vee x_2\bar{x}_3x_4 \vee x_1x_3\bar{x}_4 \vee x_1\bar{x}_2x_3 = \\
&= \bar{x}_2\bar{x}_3\bar{x}_4 \vee \bar{x}_2x_3 \vee x_2x_3x_4 \vee x_1x_3\bar{x}_4 = \\
&= \bar{x}_2\bar{x}_4 \vee \bar{x}_2x_3 \vee x_3x_4 \vee x_1x_3\bar{x}_4 = \\
&= \underbrace{\bar{x}_2\bar{x}_4}_{K_1} \vee \underbrace{\bar{x}_2x_3}_{K_3} \vee \underbrace{x_3x_4}_{K_2} \vee x_1x_3 = \bar{x}_2\bar{x}_4 \vee x_3x_4 \vee x_1x_3 .
\end{aligned}
$$

Дизъюнктивной нормальной формой булевой функции называется дизъюнкция элементарных конъюнкций. В частности, совершенная ДНФ – одна из них. Во многих случаях из-за громоздкости совершенной ДНФ булевы функции исходно задаются в ДНФ. Например, наиболее естественная ДНФ функции дизъюнкции

$$f(x_1,\dots,x_n) = x_1 \vee \dots \vee x_n .$$

3.1.5. Скобочная форма.

Аксиомы дистрибутивности булевой алгебры позволяют при преобразованиях форм булевых функций использовать операции раскрытия скобок и вынесения за скобки. Вынесение за скобки во многих случаях приводит к получению более компактной формы представления булевой функции, чем ДНФ /содержащей меньше символов/. Например, $f = \bar{x}_2\bar{x}_4 \vee \bar{x}_2x_3 \vee x_3x_4 \vee x_1x_3 = \bar{x}_2\bar{x}_4 \vee x_3(x_1 \vee \bar{x}_2 \vee x_4)$. Вынесение за скобки сократило число символов с 8 до 6.

Дизъюнкция конъюнкций, часть из которых может содержать своими сомножителями скобочные формы, называется скобочной формой представления булевых функций.

Классический пример построения скобочных форм:

$$f = x_1x_2 \vee x_2x_4 \vee x_4\bar{x}_5 \vee x_3x_4 \vee x_1x_3x_5 = x_1(x_2 \vee x_3x_5) \vee x_4(x_2 \vee x_3 \vee \bar{x}_5).$$

Однако сокращение с 11 до 8 символов для данного примера не является пределом. Если конъюнкцию x_3x_4 разложить по переменной x_5, то возможно получение скобочной формы, содержащей семь символов.

$$
\begin{aligned}
f &= x_1x_2 \vee x_2x_4 \vee x_4\bar{x}_5 \vee \cancel{x_3x_4\bar{x}_5} \vee x_3x_4x_5 \vee x_1x_3x_5 = \\
&= x_1(x_2 \vee x_3x_5) \vee x_4(x_2 \vee x_3x_5) \vee x_4\bar{x}_5 = (x_2 \vee x_3x_5)(x_1 \vee x_4) \vee x_4\bar{x}_5 .
\end{aligned}
$$

Этот пример показывает, что построение простейших скобочных форм – очень сложная задача.

Широко используется следующий очевидный приближенный алгоритм.

I. Исходная ДНФ считается очередной преобразуемой ДНФ.

2. Рассчитывается для каждой буквы / x_i и $\bar{x}_i$ - отдельно/ число вхождений этой буквы в очередную рассматриваемую ДНФ. Выбирается буква с большим числом вхождений и выносится за скобки. ДНФ разбивается на две: одна в скобках, другая - вне их. Каждая из этих ДНФ должна рассматриваться как очередная преобразуемая величина, если в ней есть символы с числом вхождений больше одного.

3.I.6. Вопросы и задания к подразд. 3.I.

А. Построить пороговые булевы функции четырех переменных. Пороговой называется булева функция, принимающая единичное значение на наборе $\sigma_1 \ldots \sigma_n$, на которых сумма весов $V_i \sigma_i$ /$1 \leq i \leq n$/ переменных не меньше порога T, т.е.

$$f(\sigma_1, \ldots, \sigma_n) = \begin{cases} 1 & \text{при } \sum\limits_{i=1}^{n} V_i \sigma_i \geq T; \\ 0 & \text{при } \sum\limits_{i=1}^{n} V_i \sigma_i < T. \end{cases}$$

Например, $\underset{8}{V_1}\ \underset{4}{V_2}\ \underset{2}{V_3}\ \underset{1}{V_4}$, $T = 10$.

Примечание. Для данного примера вес V_{4-i} равен весу i -го разряда в двоичной позиционной системе счисления, поэтому $\sum\limits_{i=1}^{n} V_i \sigma_i$ представляет собой десятичный эквивалент набора $\sigma_1, \ldots, \sigma_n$, и булева функция f = I на тех наборах, десятичные эквиваленты которых не представляются одной цифрой.

x_1	x_2	x_3	x_4	f	$\sum V_i \sigma_i$
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	2
0	0	1	1	0	3
0	1	0	0	0	4
0	1	0	1	0	5
0	1	1	0	0	6
0	1	1	1	0	7
1	0	0	0	0	8
1	0	0	1	0	9
1	0	1	0	1	10
1	0	1	1	1	11
1	1	0	0	1	12
1	1	0	1	1	13
1	1	1	0	1	14
1	1	1	1	1	15

Варианты пороговых булевых функций определены следующей таблицей.

N	V_1	V_2	V_3	V_4	T	N	V_1	V_2	V_3	V_4	T	N	V_1	V_2	V_3	V_4	T
1	1	1	1	2	3	11	4	2	1	3	5	21	3	4	5	2	7
2	1	2	2	1	4	12	4	3	2	2	6	22	1	1	4	5	6
3	2	1	2	2	4	13	1	1	4	3	5	23	2	5	5	2	7
4	3	1	2	1	4	14	2	2	4	4	6	24	1	2	5	4	6
5	2	3	1	1	4	15	4	3	2	3	6	25	7	4	2	1	7
6	3	2	2	1	5	16	4	3	3	1	6	26	1	2	5	6	6
7	2	3	2	3	6	17	3	4	2	3	6	27	2	3	6	5	7
8	2	2	2	2	5	18	1	4	4	4	7	28	5	5	4	3	8
9	3	3	1	1	5	19	2	3	4	5	7	29	5	4	4	5	9
10	3	2	3	3	6	20	1	5	3	4	6	30	5	3	5	3	8

Б. Проверить аксиомы сравнением табличных заданий:

I/ $xy = yx$; 2/ $\overline{xy} = \bar{x} \vee \bar{y}$; 3/ $\overline{x \vee y} = \bar{x}\bar{y}$.

В. Построить совершенные алгебраические формы для булевых функций п.А.

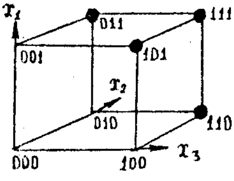
Г. Упростить совершенные ДНФ функций, построенных в п.В, используя операции склеивания, поглощения, неполного и обобщенного склеивания.

Д. Построить скобочные формы для функций предыдущего задания.

3.2. Геометрические формы представления булевых функций

3.2.I. n-мерный булев куб.

Булева функция определяется заданием ее значений на наборах. Если рассмотреть булево пространство n двоичных переменных, то получим дискретное пространство, состоящее из 2^n точек. Каждой точке соответствует один из наборов. На рис. 3.6 показано булево пространство трех и четырех переменных /единичное значение булевой функции обозначается жирной точкой/, заданы функции P_i и пороговая функция, описанная в п.3.I.6.А.



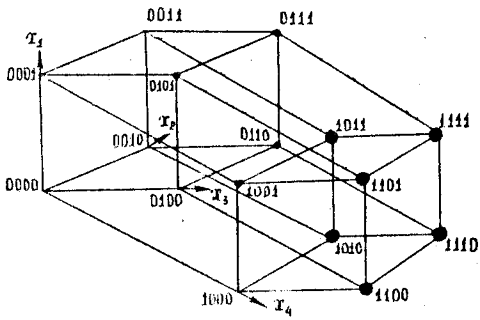


Рис. 3.6

Как видно из рис. 3.6 булево пространство n переменных изображается в виде n-мерного куба. Представление n-мерного куба громоздко и при $n>4$ теряет свою наглядность. Тем не менее геометрическая интерпретация булевых функций в некоторых приложениях оказывается полезной /например, в теории пороговых булевых функций/.

3.2.2. Матричные формы /диаграма Вейча и карта Карно/.

Матричная /двумерная/ форма наиболее удобна при работе на листе бумаги. Для сопоставления 2^n наборов элементам матрицы множество x переменных разбивается на два подмножества младших и старших переменных:

$$x=\{x_1,\ldots,x_n\}=\{x_1,\ldots,x_{]\frac{n}{2}[}\}\cup\{x_{]\frac{n}{2}[+1},\ldots,x_n\},$$

где $]z[$ - наименьшее целое, большее или равное z .

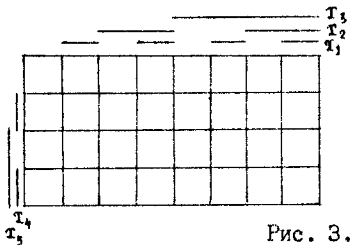
Наборы значений множества $\{x_1,\ldots,x_{]\frac{n}{2}[}\}$ младших переменных сопоставляются столбцами матрицы, наборы старших - строками. Данное сопоставление можно провести упорядочением множества наборов. Широко распространены два варианта упорядочения: использование последовательного двоичного кода и использование кода Грея. Рассмотрим принцип построения кода Грея. Последовательность начинается с нулевого набора /$0\ldots0$/, при переходе от очередного набора к следующему изменяется значение только одной переменной - той младшей переменной, которая приводит к получению нового кода по отношению ко всем ранее построенным. Например, при $n=4$ и упорядочении переменных $x_4x_3x_2x_1$ получается такая последовательность наборов при счете в коде Грея: 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000.

Если при перечислении наборов применяется позиционный код, матрица называется диаграммой Вейча, если используется код Грея - картой Карно. Принято условное обозначение наборов: единичное значение переменной изображается чертой над столбцом /у строки/, нулевое - отсутствием черты. Матричные формы при $n=5$ показаны на рис. 3.7.

17*

Матрица в позиционном коде (диаграмма Вейча)

Матрица в коде Грея (карта Карно)



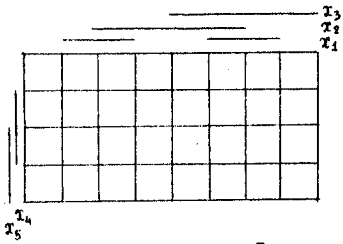


Рис. 3.7

При $n=5$ имеем 8 столбцов и 4 строки, итого 8 х 4 = 32 = 2^5 элементов матричной формы. Каждому элементу соответствует совокупный набор значений младших и старших переменных. Условлено, что единичное значение булевой функции на соответствующем наборе обозначается точкой в элементе матрицы, нулевое - отсутствием точки. Например, функции рис. 3.6 изображены на матрицах в коде Грея на рис. 3.8.

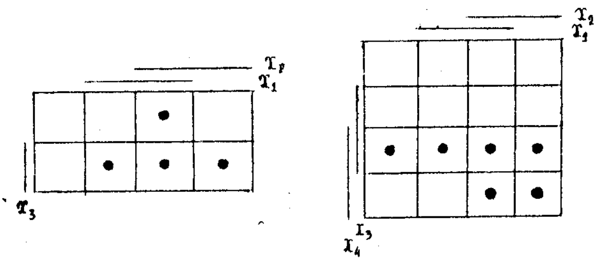


Рис. 3.8

На рис. 3.9 изображена функция $f(x_1,\dots,x_5)=x_1\oplus, x_2\oplus x_3\oplus x_4\oplus x_5$ /функция нечетности числа единиц в наборе/.

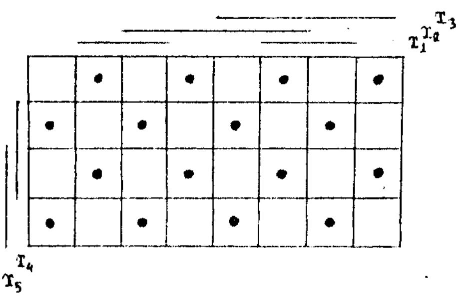


Рис. 3.9

Матричная форма в позиционном коде удобна при решении задач на ЭВМ. Матричная форма в коде Грея применяется при ручном решении задач, она обладает свойством симметрии, которое используется при упрощении ДНФ на основе матричной формы /подразд. 3.5/.

3.2.3. Интервальные формы.

По аналогии с высшей математикой можно определить понятие интервала в булевом пространстве как упорядоченной совокупности наборов, заданной минимальным A и максимальным B наборами: интервал $/A,B/$ - множество наборов σ, для которых $A \leq \sigma \leq B$. Необходимо еще раз подчеркнуть векторное определение упорядоченности наборов: $A \leq B$, если и только если $a_i \leq b_i$ $1 \leq i \leq n$. Пример. Интервал $(0010, 0010)$ содержит всего один набор 0010; интервал $(0010, 0110)$ - два набора: 0010 и 0110; интервал $/0010, 1011/$ - четыре набора, $0010, 0011, 1010, 1011$.

Переменные, которые принимают одинаковые значения в минимальном A и максимальном B наборах, очевидно имеют одинаковое значение во всех наборах σ интервала (A,B). Эти переменные называются внешними переменными интервала. Остальные переменные называются внутренними и в каждом наборе σ имеют свой набор значений, который принимает в наборах интервала все 2^K возможных значений, где K - число внутренних переменных. Отсюда можно сформулировать второе определение интервала: совокупность 2^K наборов, в которых $n-K$ внешних переменных принимают одинаковые для каждого набора значение и K внутренних переменных - все возможные наборы значений.

Пример.

x_1 x_2 x_3 x_4

$0\ 0\ 1\ 0$ - все переменные внешние;

$\left.\begin{matrix} 0\ 0\ 1\ 0 \\ 0\ 1\ 1\ 0 \end{matrix}\right\}$ - переменная x_2 - внутренняя, остальные - внешние;

$\left.\begin{matrix} 0\ 0\ 1\ 0 \\ 0\ 0\ 1\ 1 \\ 1\ 0\ 1\ 0 \\ 1\ 0\ 1\ 1 \end{matrix}\right\}$ - переменные x_1 и x_4 - внутренние, переменные x_2 и x_3 - внешние; они принимают на всех четырех наборах значения $x_2 = 0$, $x_3 = 1$.

Аналогично операции склеивания конъюнкции можно проводить операцию склеивания наборов, соответствующих этим конъюнкциям: склеиваются два набора, если множества их переменных совпадают и значения одной и только одной переменной различается.

Результат склеивания двух наборов изображается в виде набора, у которого вместо значения различающейся переменной проставлена черточка /-/.

Пример.
$$0010 \cup 0011 = 001-,$$
$$0-10 \cup 0-11 = 0-1-.$$

Очевидно, что наборы интервала склеиваются, причем после всех склеиваний получается один набор, в котором внешние переменные принимают заданные значения, а значения внутренних заменены черточками:

$$\left.\begin{array}{l}\left.\begin{array}{c}0\,0\,1\,0\\0\,0\,1\,1\end{array}\right\}=001-\\ \left.\begin{array}{c}1\,0\,1\,0\\1\,0\,1\,1\end{array}\right\}=101-\end{array}\right\}=-01-.$$

Такое изображение интервала широко используется. Черточка означает, что переменная может принимать значения 0 и 1, конкретный набор значений K внутренних переменных определяет один из 2^K внутренних наборов интервала. Интервалу соответствует конъюнкция внешних переменных, каждая из них входит в конъюнкцию без инверсии, если в интервале есть значение единица, и с инверсией - если нуль. Интервальное представление используется при кодировании ДНФ в ЭВМ. Например, функции

$$f = (x_1, x_2, x_3, x_4) = x_1\bar{x}_2 \vee x_1x_3 \vee x_2x_4 \vee \bar{x}_3 x_4$$

соответствует совокупность четырех интервалов:

x_1	x_2	x_3	x_4
1	0	–	–
1	–	1	–
–	1	–	1
–	–	0	1

3.2.4. Интервальное представление на матричной форме.

По представлению булевой функции в ДНФ достаточно просто получить матричное представление этой функции. Все наборы каждого интервала, соответствующего конъюнкции ДНФ, отмечаются на матрице точками. Для этого нужно выделить столбцы, определяемые значениями младших внешних переменных интервала, и строки, определяемые значениями старших внешних переменных. Пересечение выделенных строк и столбцов выделит все элементы интервала.

Пример.

$$f(x_1, x_2, x_3, x_4) = x_1\bar{x}_2 \vee x_1 x_3 \vee x_2 x_4 \vee x_3 x_4 .$$

$$\qquad x_1 \; x_2 \; x_3 \; x_4$$

$$x_1\bar{x}_2 \sim 1 \;\; 0 \underbrace{\;-\;\;-\;}$$

означает выбор всех строк

$$x_1 x_3 \sim 1 \;\; - \;\; 1 \;\; -$$

интервалы пересекаются

$$x_2 x_4 \sim - \;\; 1 \;\; - \;\; 1$$

$$\bar{x}_3 x_4 \sim - \;\; - \;\; 0 \;\; 1$$

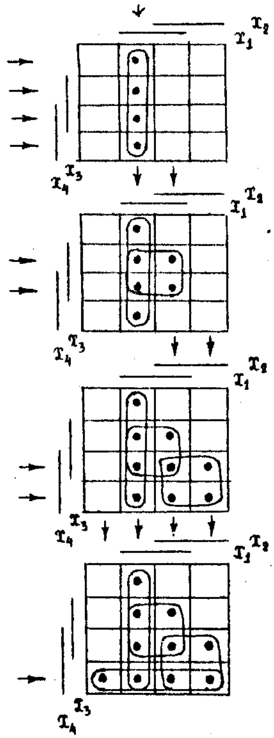


Рис. 3.10

Для функции пяти переменных соответствующие преобразования проведены на рис. 3.11.

$$f = (x_1, \dots, x_5) = \bar{x}_1 \bar{x}_2 x_4 \vee x_2 \bar{x}_5 \vee \bar{x}_3 x_4 x_5 \vee \bar{x}_1 \bar{x}_4 x_5$$

	x_1	x_2	x_3	x_4	x_5
$\bar{x}_1 \bar{x}_2 x_4 \sim$	0	0	–	1	–
$x_2 \bar{x}_5$	–	1	–	–	0
$\bar{x}_3 x_4 x_5$	–	–	0	1	1
$\bar{x}_1 \bar{x}_4 x_5$	0	–	–	0	1

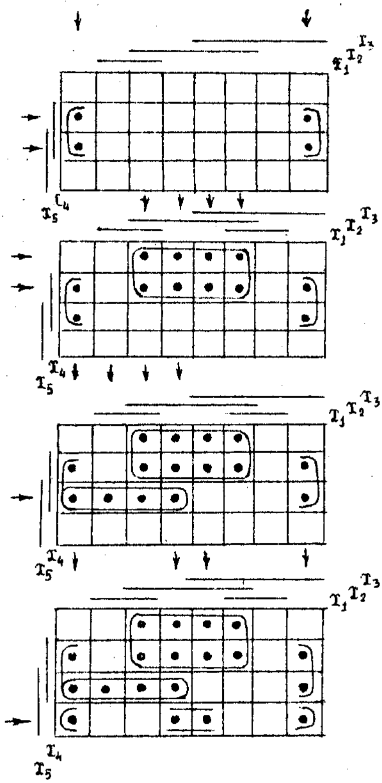


Рис. 3.II

3.2.5. Задания к подразд. 3.2.

А. Изобразить на четырехмерном булевом кубе пороговые булевы функции задания п.3.I.6.А.

Б. Изобразить эти функции на матричных формах в позиционном коде и коде Грея.

В. Получить интервальное представление булевых функций по ДНФ задания п.3.I.6.Г.

Г. Изобразить это интервальное представление на матричной форме в коде Грея.

3.3. Специальные классы булевых функций и задача о функциональной полноте

3.3.1. Симметрические булевы функции.

Булева функция симметрична по переменным x_i, x_j, если $f(\ldots, x_i, \ldots, x_j, \ldots) = f(\ldots, x_j, \ldots, x_i, \ldots)$. Например по переменным x_1, x_3 симметрическая функция $x_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3$. Булева функция называется симметрической, если она симметрическая по всем парам переменных. Очевидно, симметрическая булева функция - дизъюнкция n переменных. Симметрической является функция $f = x_1 x_2 \vee x_1 x_3 \vee x_2 x_3$.

Теорема. Симметрическая булева функция допускает любые перестановки переменных.

Теорема 3.10 /Шеннон/. Булева функция является симметрической, если для нее можно определить множество $A \subseteq \{0, 1, \ldots, n\}$, так что

$$f(\sigma) = \begin{cases} 1 & \text{при } \nabla\sigma \in A, \\ 0 & \text{при } \nabla\sigma \notin A, \end{cases}$$ где $\nabla\sigma$ - вес

набора σ, т.е. число единиц в этом наборе. Множество A называется множеством чисел симметрии и симметрическую булеву функцию с множеством $A = \{a_1, \ldots, a_k\}$ записывают в виде

$$S_{a_1 \ldots a_k}(x_1, \ldots, x_n).$$

Доказательство. Перестановка переменных при преобразовании каждого набора не изменяет его веса,поэтому, если на каком либо наборе заданного веса функция равна единице, то для допустимости любых перестановок необходимо, чтобы на всех наборах этого веса функция не изменялась, т.е. тоже была равна единице. Аналогично рассуждение о нулевых значениях функции. Тогда веса наборов, на которых функция равна единице, вводим в множество A, веса наборов, на которых функция равна 0, не вводим в A, что и определяет утверждение теоремы.

Число различных наборов, содержащих k единиц, равно C_n^k, что непосредственно следует из векторного представления i - элементных подмножеств n -элементного множества.

Алгоритм проверки булевой функции на симметричность можно построить на основе анализа множества M_1 этой функции.

1. Выписываем множество M_1 наборов, на которых функция равна единицы.

2. Разбиваем множество на классы M_1^i : к подмножеству M_1^i относим все наборы множества M_1 веса $i \ (0 \le i \le n)$.

3. Проверяем соотношение $$|M_1^i| = \begin{cases} 0, \\ C_n^i. \end{cases}$$

Если это соотношение выполняется для всех $i\ (0 \le i \le n)$, то функция симметрическая, иначе – несимметрическая.

Проведем проверку симметричности булевой функции /рис. 3.12/. При выписывании наборов заранее разбиваем их на классы:

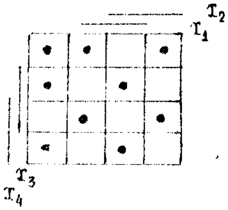


Рис. 3.12

Функция симметрическая.

x_1	x_2	x_3	x_4	
0	0	0	0	$\lvert M_1^0\rvert = 1 = C_4^0$,
1	0	0	0	
0	1	0	0	
0	0	1	0	$\lvert M_1^1\rvert = 4 = C_4^1$,
0	0	0	1	$\lvert M_1^2\rvert = 0$
1	1	1	0	
1	1	0	1	$\lvert M_1^3\rvert = 4 = C_4^3 = C_4^1$,
1	0	1	1	
0	1	1	1	$\lvert M_1^4\rvert = 0.$

Замечание. Проверку симметричности булевой функции можно непосредственно проверить по матричной форме, вычислив для каждой точки вес набора, рассчитав число наборов каждого веса i и сравнив его с C_n^i.

Для того же примера /рис. 3.12/ веса наборов проставлены непосредственно на матрице /рис. 3.13/:

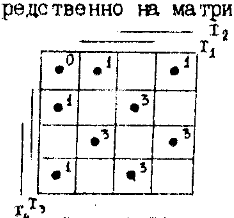


Рис. 3.13

$$|M_1^0| = 1 = C_4^0,\ |M_1^1| = 4 = C_4^1,$$
$$|M_1^2| = 0,\ |M_1^3| = 4 = C_4^3,$$
$$|M_1^4| = 0.$$

Функция симметрическая:

$$f(x_1, x_2, x_3, x_4) = S_{0,1,3}(x_1, x_2, x_3, x_4).$$

Расширение класса симметрических булевых функций

Булева функция симметрическая в расширенном смысле, если симметрическая она сама или булева функция, полученные инвертированием некоторых переменных.

Пример.

x_1	x_2	x_3	x_4	
1	0	0	0	
1	1	1	0	$\left.\right\}\ \lvert M_1^3\rvert \ne C_4^3$
1	1	0	1	
1	0	1	1	
0	0	0	1	
0	0	1	0	
0	1	0	0	

Исходно функция, заданная множеством M_1, несимметрическая, так как мощность подмножества M_1^3 равна 3, а не 4. Если инвертировать значение переменной x_1, то получим симметрическую булеву функцию.

$$\begin{array}{cccc}\bar{x}_1 & x_2 & x_3 & x_4\\ 0&0&0&0\\ 0&1&1&0\\ 0&1&0&1\\ 0&0&1&1\\ 1&0&0&1\\ 1&0&1&0\\ 1&1&0&0\end{array}\quad \begin{array}{l}\}\ |M_1^0| = 1 = C_4^0 \\ \\ \left.\right\}\ |M_1^2| = 6 = C_4^2\end{array}$$

Рассматриваемую функцию $f(x_1, x_2, x_3, x_4)$ можно записать так $S_{0,2}(\bar{x}_1, x_2, x_3, x_4)$.

Определим переменные, которые нужно инвертировать, чтобы получить симметрическую булеву функцию в расширенном смысле. Введем величины P_i и q_i . P_i - число единичных значений переменной x_i в наборах множества M_1 , q_i - число нулевых значений переменной x_i в этих наборах. Очевидно, $q_i = |M_1| - P_i$.

Для симметрических функций отношение $\frac{P_i}{q_i}$ или $\frac{q_i}{P_i}$ есть величина постоянная для всех i . Инвертировать нужно те переменные, которых меньше, чтобы постоянной величиной было только одно соотношение P_i/q_i /или q_i/P_i /.

Для данного примера:

	x_1	x_2	x_3	x_4
	1	0	0	0
	1	1	1	0
	1	1	0	1
	1	0	1	1
	0	0	0	1
	0	0	1	0
	0	1	0	0
P_i	4	3	3	3
q_i	3	4	4	4

Для переменных x_2, x_3 и x_4 $P_i/q_i = \frac{3}{4}$ для одной переменной x_1 $q_1/P_1 = \frac{3}{4}$, поэтому инвертируем переменную x_1 .

<u>Алгоритм проверки булевой функции на симметричность в расширенном смысле</u>.

Выписываем множество M_1 и подсчитываем величины P_i и q_i $(1 \le i \le n)$. Если соотношения P_i/q_i или q_i/P_i величины не постоянные для всех i , то функция несимметрическая, иначе находим меньшее подмножество переменных, инвентирование которых приводит к постоянству только одного из соотношений $(P_i/q_i$ или $q_i/P_i)$ и преобразованное инвертированием этих переменных множества M_1^* проверяем на симметричность.

<u>Замечание</u>. Алгоритм не дает критерий выбора подмножества переменных, подлежащих инвентированию, при $P_i = q_i$ $(1 \le i \le n)$.

Основные свойства симметрических булевых функций

Теорема 3.II. Существует 2^{n+1} различных симметрических булевых функций n переменных. Эта теорема - следствие теоремы Шеннона. Симметрические булевы функции различаются множеством A. Каждому множеству A можно сопоставить двоичный $n+1$-разрядный набор B :

$$b_i = \begin{cases} 0, & i \in A, \\ 1, & i \in A, \quad 0 \le i \le n. \end{cases}$$

Число таких наборов равно 2^{n+1}.

Теорема 3.I2. Дизъюнкция симметрических функций - симметрическая функция.

Теорема 3.I3. Коньюнкция симметрических функций - симметрическая функция.

Теорема 3.I4. Инверсия симметрической функции - симметрическая функция.

3.3.2. Монотонные и однородные булевы функции.

Определение и свойства монотонных булевых функций

Булева функция называется положительной по переменной x_i, если найдется такая ДНФ этой функции, в которую переменная входит только без инверсии. Булева функция называется монотонной, если она по всем переменным положительна.

Замечание. Исходная ДНФ не всегда дает ответ о монотонности функции, если ДНФ содержит в некоторых коньюнкциях переменные с инверсией, это еще не означает, что функция не монотонна. Функция не монотонна, если ее нельзя преобразовать к ДНФ без инверсий.

Пример. $f = \underline{x_1 x_2} \vee \underline{\bar{x}_1 x_2 x_3} \vee x_2 \bar{x}_3 x_4 =$

$$= x_1 x_2 \vee \underline{x_2 x_3} \vee \underline{x_2 \bar{x}_3 x_4} = x_1 x_2 \vee x_2 x_3 \vee x_2 x_4 .$$

Преобразованная ДНФ позволяет заключить, что функция монотонна.

Теорема 3.I5. Булева функция монотонна, если при векторном возрастании набора значение булевой функции не убывает.

Формулировка этой теоремы похожа на определение монотонной функции в высшей математике.

Приведем конструктивное доказательство теоремы, позволяющее сформулировать алгоритм построения ДНФ без инверсий переменных, либо определить, что функция не монотонна. Доказательство использует понятие интервала, внешние переменные которого равны единице, т.е. этому интервалу соответствует коньюнкция без инверсий. Итак, пусть на некотором наборе A $f(A)=1$. Из условия теоремы следует, что на всех наборах $\sigma \geq A$ функция не убывает, т.е. равна единице, тем более она равна единице на наборе II...I. Тогда набор A вместе с набором II...I определяют интервал, на всех наборах σ которого $f(\sigma)=1$ и все внешние переменные которого равны единице. Для любого набора $A \in M_1$ существует интервал, соответствующая которому конъюнкция не содержит инверсий переменных. Дизъюнкция таких интервалов составляет ДНФ, все конъюнкции которой не содержат инверсий переменных, что соответствует определению монотонности булевой функции.

Алгоритм проверки булевой функции на монотонность и построение монотонной ДНФ. Чтобы строить ДНФ без поглощаемых конъюнкций, алгоритм проверки булевой функции на монотонность должен включать в себя выбор минимальных наборов.

I. Строится множество M_1 и все наборы считаются неотмеченными.

2. Среди неотмеченных наборов выбирается минимальный в векторном смысле. Пусть выбранный набор имеет K нулей. Отмечается этот набор и все большие его наборы. Если число отмеченных наборов не равно 2^K , то делается вывод, что функция не монотонна, иначе по выбранному набору строится конъюнкция переменных, которые в этом наборе имеют значение единица. Если есть еще неотмеченные наборы, то выполняется п.2, иначе монотонное представление булевой функции построено.

Пример.

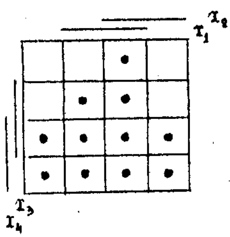


Рис.3.14

x_1	x_2	x_3	x_4			
I	I	0	0		W $k=2$	
I	0	I	0			WW $k=2$
I	I	I	0		W	WW
0	0	I	I	V		
I	0	I	I	V		WW
I	I	I	I	V	W	WW
0	I	I	I	V		$4=2^3$
0	0	0	I	V $k=3$		
I	0	0	I	V		
I	I	0	I	V	W	
0	I	0	I	V	$4=2^2$	
				$8=2^3$		

$f = x_4 \vee x_1 x_2 \vee x_1 x_3$. Функция монотонна. Пороговые функции являются подклассом монотонных функций.

Определение и алгоритм проверки булевой функции на однородность. Обобщим понятие монотонных функций. Булева функция называется отрицательной по переменной x_i, если найдется такая ДНФ функции, в конъюнкции которой переменная x_i входит только с инверсией.

Булева функция называется однородной, если она положительна или отрицательна по каждой переменной. Например, однородна следующая функция $f(x_1,x_2,x_3,x_4)=x_1x_3 \vee x_1\bar{x}_2x_4 \vee \bar{x}_2x_3 \vee \bar{x}_3x_4$. Каждая переменная в эту ДНФ входит либо только с инверсией, либо только без инверсии.

Алгоритм проверки булевой функции на однородность и построение однородной ДНФ

Если в однородной функции произвести замену $\bar{x}_i = y_i$, то получим монотонную ДНФ относительно множества $X \cup Y$. Таким образом, алгоритм проверки булевой функции на однородность состоит из двух этапов.

1. Определение переменных, относительно которых функция может быть отрицательна, и инверсирование этих переменных.

2. Проверка преобразованной булевой функции на монотонность на множестве $X \cup Y$ и построение однородного представления функции.

Алгоритм реализации второго этапа уже известен, таким образом необходимо найти критерий отрицательности функции по переменной x_i. Если во всех наборах множества M переменная x_i принимает значение 0, то уже совершенная ДНФ отрицательна по x_i. Если в наборах множества M_1 переменная x_i принимает значения 1 и 0, то для того, чтобы построить ДНФ, в которую переменная x_i входит только с инверсией, необходимо те конъюнкции совершенной ДНФ, которые положительны по x_i, склеить по переменной x_i. Это возможно только в том случае, если в M_1 число наборов, на которых переменная x_i равна 1, меньше, чем число наборов, на которых x_i равно 0. Это и служит критерием для выделения переменных x_i, относительно которых функция может быть отрицательной.

Алгоритм выделения переменных, которые могут быть отрицательными.

1. Для каждой переменной x_i рассчитываем величины p_i и q_i – соответственно числа единичных и нулевых значений переменной x_i в наборах множества M_i.

2. Те переменные, для которых $p_i < q_i$, инвертируем $y_i = \bar{x}_i$ и получаем множество M_1^* наборов, определенных на множестве $X \cup Y$ переменных.

Пример проверки булевой функции на однородность показан на рис. 3.15.

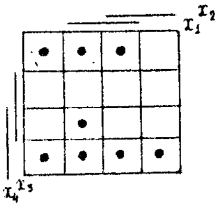


Рис. 3.15

	x_1	x_2	x_3	x_4
	0	0	0	0
	I	0	0	0
	I	I	0	0
	I	0	I	I
M_1	0	0	0	I
	I	0	0	I
	I	I	0	I
	0	I	0	I
p_i	5	3	I	5
q_i	3	5	7	3

	x_1	x_2	x_3	x_4	
	0	I	I	0*	V
	I	I	I	0	V W
	I	0	I	0	W
M_1^*	I	I	0	I	⊕
	0	I	I	I	V W
	I	I	I	I	V W W +
	I	0	I	I	W W
	0	0	I	I	W

$$f^* = y_2 y_3 \vee x_1 y_3 \vee y_3 x_4 \vee x_1 y_2 x_4,$$

$$f = \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_3 \vee \bar{x}_3 x_4 \vee x_1 \bar{x}_2 x_4.$$ Функция однородна.

3.3.3. Самодвойственные булевы функции.

Булева функция $\bar{f}(x_1,\dots,x_n)$ называется двойственной функцией $f(x_1,\dots,x_n)$. Например:

$$f(x_1, x_2, x_3) = x_1 \vee x_2 \vee x_3;\ \bar{f}(\bar{x}_1, \bar{x}_2, \bar{x}_3) = \bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 = x_1 x_2 x_3.$$

Функции дизъюнкции и конъюнкции – двойственны.

Булева функция называется самодвойственной, если двойственная ей функция равна ей самой, т.е. если

$$f(x_1, \dots, x_n) = \bar{f}(\bar{x}_1, \dots, \bar{x}_n). \qquad /3.3/$$

Следствие. Самодвойственная булева функция принимает инверсные значения на инверсных наборах.

Это следствие непосредственно вытекает из формулы /3.3/, являясь ее формулировкой. Одновременно это следствие служит основой алгоритма проверки булевой функции на самодвойственность.

Например, на матричной форме трех переменных взаимно инверсные значения должны принимать элементы, в которых на рис. 3.16 показаны одинаковые буквы.

a	b	c	d
$\bar{c}$	$\bar{d}$	$\bar{a}$	$\bar{b}$

Рис. 3.16

Например, если произвести замену $a=0$, $b=1$, $c=1$, $d=1$, то однозначно можно определить булеву функцию /рис. 3.17/, которая является самодвойственной.

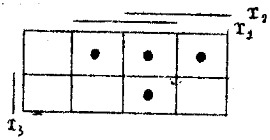


Рис. 3.17

Проверим ее самодвойственность

$$f(x_1,x_2,x_3)=x_1\bar{x}_2\bar{x}_3\vee x_1x_2\bar{x}_3\vee\bar{x}_1x_2\bar{x}_3\vee x_1x_2x_3.$$

Чтобы сократить процесс проверки, упростим ДНФ, проведя склеивания:

$$f(x_1,x_2,x_3)=x_1\bar{x}_3\vee x_2\bar{x}_3\vee x_1x_2;$$

$$\bar{f}(\bar{x}_1,\bar{x}_2,\bar{x}_3)=\overline{\bar{x}_1x_3\vee\bar{x}_2x_3\vee\bar{x}_1\bar{x}_2}=\overline{\bar{x}_1x_3}\;\overline{\bar{x}_2x_3}\;\overline{\bar{x}_1\bar{x}_2}=$$

$$=(x_1\vee\bar{x}_3)(x_2\vee\bar{x}_3)(x_1\vee x_2)=(x_1x_2\vee\bar{x}_3)(x_1\vee x_2)=$$

$$=x_1x_2\vee x_1\bar{x}_3\vee x_2\bar{x}_3=f(x_1,x_2,x_3).$$

Теорема 3.16. Существует $2^{2^{n-1}}$ различных самодвойственных булевых функций.

Теорема вытекает из следствия. Множество 2^n наборов разбивается на 2^{n-1} пар взаимно инверсных наборов. Для каждой пары достаточно определить значение функции на одном наборе. Число возможных доопределений $2^{2^{n-1}}$.

3.3.4. Линейные, сохраняющие 0 и 1 булевы функции.

Булева функция называется линейной, если ее совершенная полиномиальная форма имеет вид

$$f(x_1,\ldots,x_n)=a_0\oplus a_1x_1\oplus\ldots\oplus a_nx_n,\qquad a_i\in\{0,1\}.$$

Алгоритм проверки линейности функции сводится к построению ее совершенной полиномиальной нормальной формы.

Теорема 3.17. Существует 2^{n+1} различных линейных булевых функций. 2^{n+1} – это число различных наборов значений коэффициентов $a_0,a_1,\ldots,a_n$, каждый из которых определяет свою линейную булеву функцию.

Пример.

Набор $a_0 a_1 a_2 a_3 a_4$	Функция
0 I I 0 I	$f(x_1,\dots,x_4) = x_1 \oplus x_2 \oplus x_4$,
I I 0 I I	$f(x_1,\dots,x_4) = 1 \oplus x_1 \oplus x_3 \oplus x_4$.

Булева функция называется сохраняющей 0 /сохраняющей I/, если на наборе 0 0...0 она равна 0 /на наборе II...I она равна I/.

Значит, значение функции на единственном наборе /0...0 или II...I/ однозначно определяет ее принадлежность классу /сохраняющих 0 или сохраняющих I/ функций.

Теорема 3.I8. Существует 2^{2^n-1} различных сохраняющих 0 /сохраняющих I/ булевых функций.

3.3.5. Задача о функциональной полноте системы элементарных булевых функций.

Постановка задачи. Задается некоторая совокупность булевых функций, которые объявляются элементарными.

Примеры совокупностей:

$$\{\vee,\wedge\},\{\vee,-\},\{\wedge,-\},\{\bar{\vee}\},\{\bar{\wedge}\},\{\vee,\wedge,-\},\{\oplus,\wedge\},\dots$$

Возникает вопрос, можно ли выразить любую булеву функцию через суперпозицию заданных элементарных функций? Эта задача называется проверкой функциональной полноты системы /совокупности/ элементарных булевых функций.

Уже доказано, например, что система $\{\vee,\wedge,-\}$ - функционально полна /теорема о совершенной ДНФ или совершенной КНФ/. Функционально полна, например, одна операция $\bar{\wedge}$ /теорема о совершенной Шефферовской НФ/. Для общего решения задачи о функциональной полноте необходимо сформулировать критерии проверки, совокупности свойств элементарных булевых функций системы, удовлетворение которых позволяет судить о их функциональной полноте.

Теорема 3.I9 о функциональной полноте. Для того, чтобы система элементраных функций была функционально полной среди функций этой системы, должно быть хотя бы по одной функции следующих пяти классов: не монотонная, не линейная, не самодвойственная, не сохраняющая нуль и не сохраняющая единицы. Необходимость условий теоремы очевидна, поскольку каждый класс /например, монотонных функций/ не содержит всех функций, а именно, если в системе элементарных функций все они принадлежат одному классу /все монотонны/, то суперпозиция функций этого класса порождает только функции этого же класса.

Примеры применения теоремы: функции совокупности $\{\vee, \wedge\}$ сохраняют 0 и I, эта совокупность не является функционально полной. Среди совокупности $\{\wedge, -\}$ есть функции: I/ не монотонная /$-$/; 2/ не линейная /$\wedge$/; 3/ не самодвойственная /$\wedge$/; 4/ не сохраняющая 0 и I /$-$/. Таким образом, система $\{\wedge, -\}$ функционально полна.

3.3.6. Задачи к подразд. 3.3.

А. Проверить на симметричность булевы функции /с.147/.

Б. Проверить на монотонность булевы функции задания п.3.I.6.А.

В. Проверить на однородность булевы функции /с.148/.

Г. Проверить на линейность булевы функции /с.149/.

Д. Проверить на самодвойственность булевы функции /с.150/.

Е. Проверить на полноту следующие системы элементарных функций $\{1, \wedge, \oplus\}, \{\vee, -\}, \{\wedge, \oplus\}$.

3.4. Минимизация булевых функций в классе ДНФ. Метод Квайна – Мак-Класки

3.4.I. Постановка задачи и основные определения.

Минимизация булевых функций в классе ДНФ – это задача получения наиболее простой из возможных ДНФ булевой функции.

В п. 3.I.4 рассмотрены операции упрощения ДНФ на основе алгебраических преобразований, проводимых интуитивно. Приведенные примеры преобразований проиллюстрировали возможность упрощения ДНФ при таких преобразованиях. Не всегда простыми рассуждениями можно доказать, что полученные результаты – наилучшие. В целом разработаны десятки методов упрощения ДНФ булевых функций, их можно разбить на два класса:

точные методы минимизации ДНФ, в которых гарантируется, что полученная ДНФ – наилучшая;

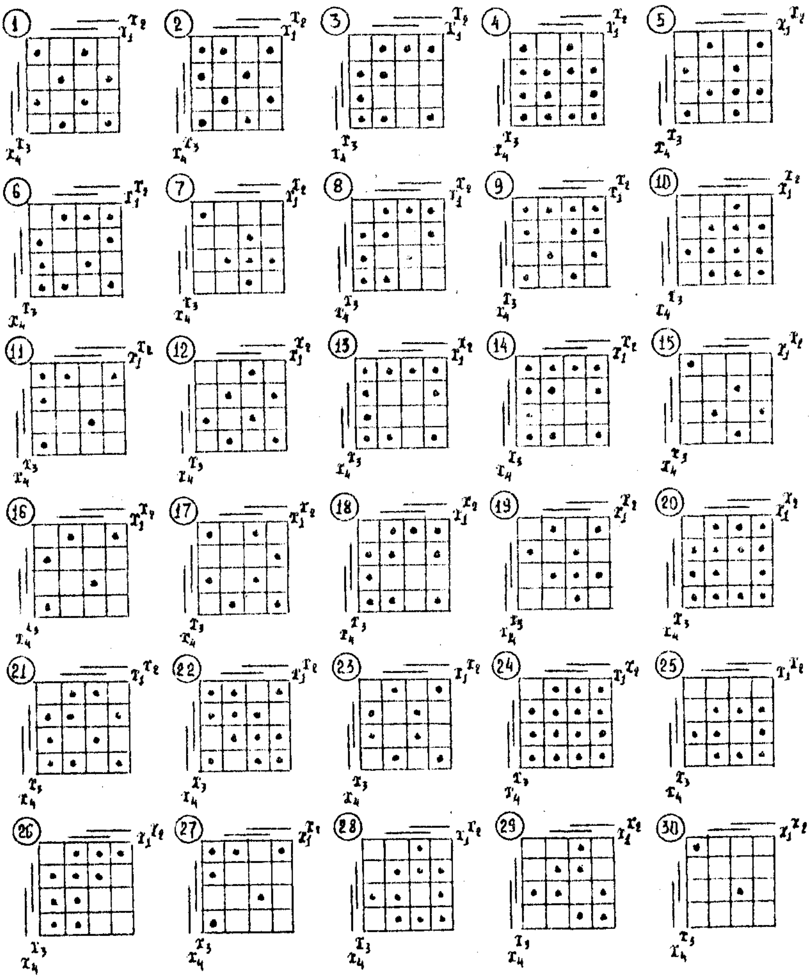
приближенные методы упрощения ДНФ, в которых такой гарантии нет – в лучших из них обычно приводятся интуитивные или статистические обоснования того, что упрощенная ДНФ достаточно близка к наилучшей.

В данном подразделе рассматривается один из наиболее характерных и известных точных методов минимизации – метод Квайна – Мак-Класки, в следующем – метод упрощения ДНФ, использующий матричную форму представления булевых функций и достаточно удобный при ручном построении упрощенной ДНФ – метод Закревского.

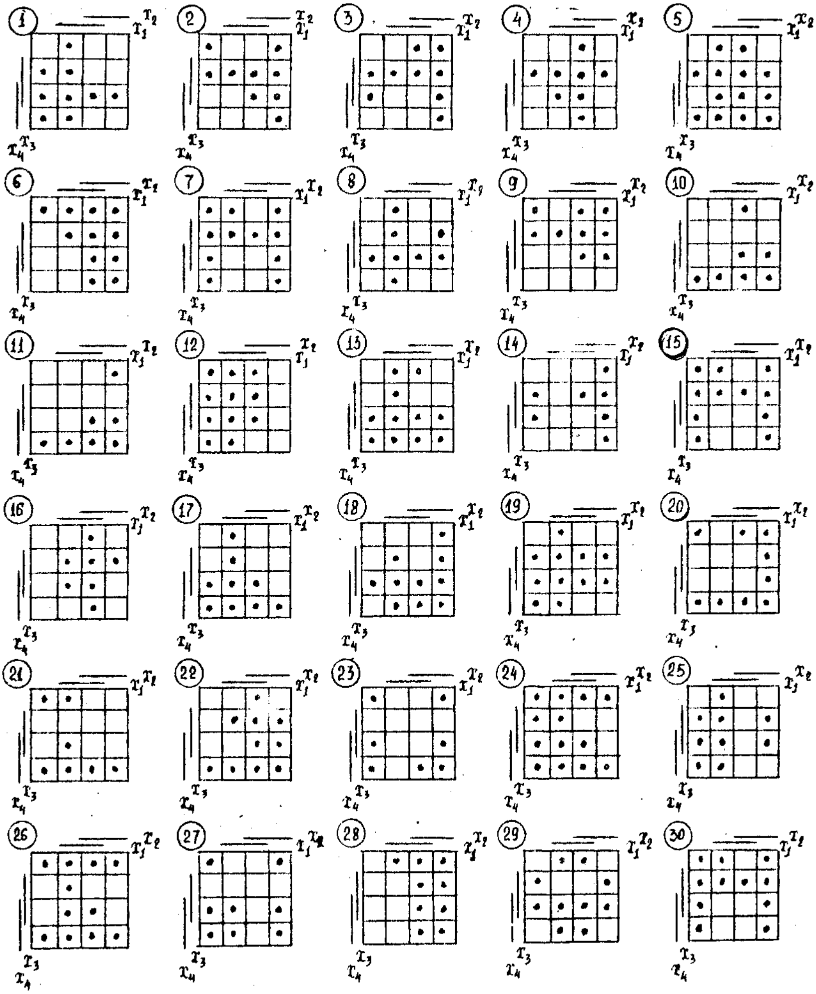
Основные определения:

I. Конъюнкция К принадлежит /поглощается/ булевой функции, если ее разложение по всем, не вошедшим в нее переменным, дает только конъюнкции, принадлежащие совершенной ДНФ функции.

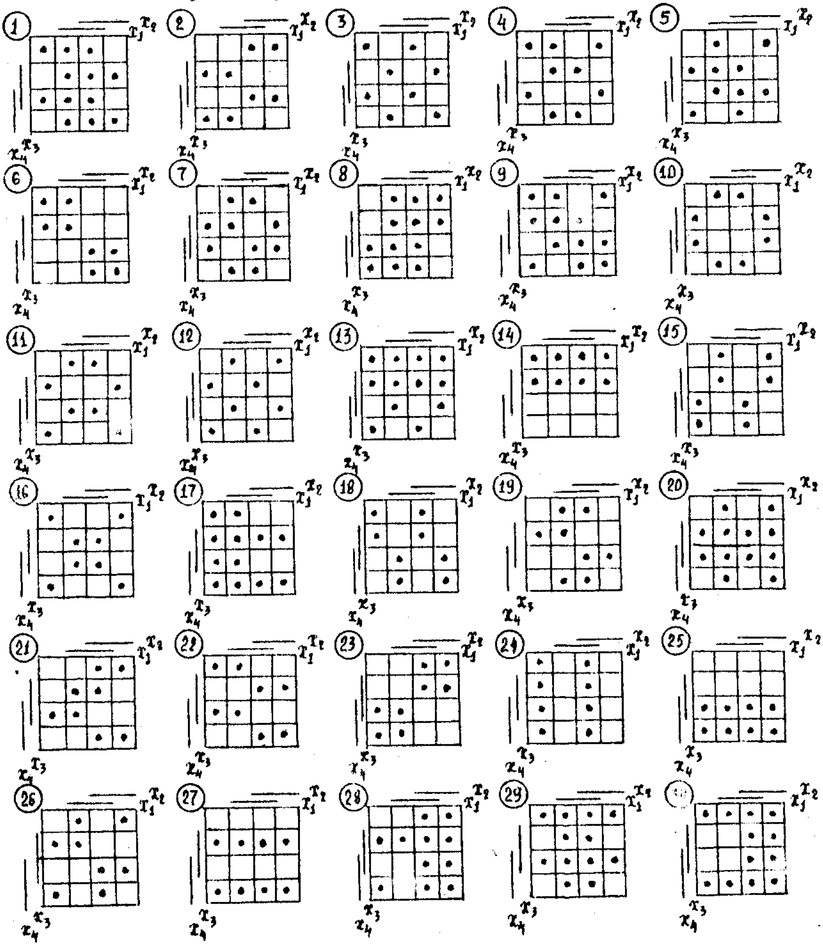
Является ли булева функция симметрической?



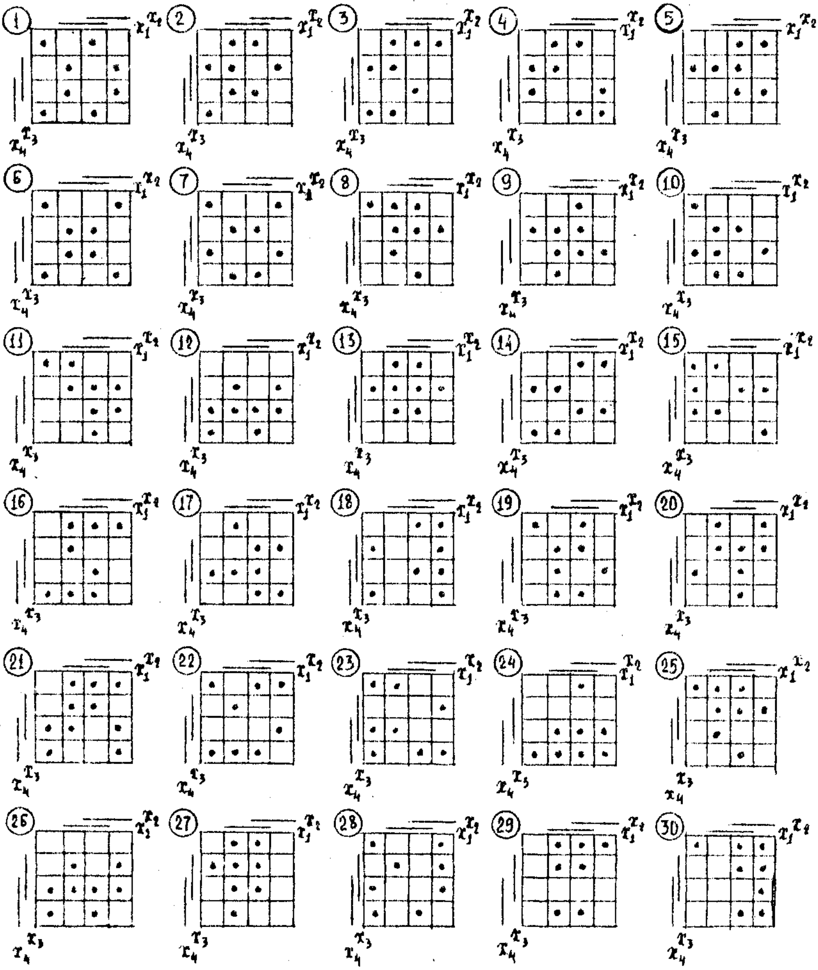
Является ли булева функция однородной?



Является ли булева функция линейной?



Является ли булева функция самодвойственной?



Примеры. Совершенная ДНФ функции $f(x_1,x_2,x_3,x_4) = \underset{1}{\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4} \vee$

$\vee \underset{2}{x_1\bar{x}_2\bar{x}_3\bar{x}_4} \vee \underset{3}{x_1\bar{x}_2 x_3\bar{x}_4} \vee \underset{4}{x_1 x_2 x_3\bar{x}_4} \vee \underset{5}{x_1\bar{x}_2 x_3 x_4} \vee$

$\vee \underset{6}{x_1 x_2 x_3 x_4} \vee \underset{7}{\bar{x}_1 x_2 x_3 x_4} \vee \underset{8}{x_1 x_2\bar{x}_3 x_4} \vee \underset{9}{\bar{x}_1\bar{x}_2 x_3 x_4}.$

$K_1 = \bar{x}_2\bar{x}_3\bar{x}_4$. Разложение $\bar{x}_2\bar{x}_3\bar{x}_4(x_1 \vee \bar{x}_1) = \underset{1}{\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4} \vee \underset{2}{x_1\bar{x}_2\bar{x}_3\bar{x}_4}$ содержит конъюнкции I и 2 совершенной ДНФ, поэтому K_1 принадлежит булевой функции.

$K_2 = x_1 x_3$. Разложение $x_1 x_3(x_2 \vee \bar{x}_2)(x_4 \vee \bar{x}_4) = \underset{3}{x_1\bar{x}_2 x_3\bar{x}_4} \vee$

$\vee \underset{5}{x_1\bar{x}_2 x_3 x_4} \vee \underset{4}{x_1 x_2 x_3\bar{x}_4} \vee \underset{6}{x_1 x_2 x_3 x_4}.$

Следовательно, K_2 тоже принадлежит булевой функции.

$K_3 = \underset{8}{x_1 x_2\bar{x}_3 x_4}$ принадлежит булевой функции.

$K_4 = x_2 x_4$ не принадлежит булевой функции, так как ее разложение

$$x_2 x_4 = \underset{?}{\bar{x}_1 x_2\bar{x}_3 x_4} \vee \underset{7}{\bar{x}_1 x_2 x_3 x_4} \vee \underset{8}{x_1 x_2\bar{x}_3 x_4} \vee \underset{6}{x_1 x_2 x_3 x_4}$$

содержит конъюнкцию $\bar{x}_1 x_2\bar{x}_3 x_4$, которая не принадлежит совершенной ДНФ функции.

2. ДНФ представляет функцию, если входящие в нее конъюнкции принадлежат функции и для каждой конъюнкции K соверш. ДНФ, входящей в совершенную ДНФ, найдется хотя бы одна конъюнкция ДНФ, разложение которой содержит конъюнкцию K соверш.ДНФ.

Пример.

Функция $f = \bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1 x_3 \vee x_1 x_2\bar{x}_3 x_4 \vee \bar{x}_1 x_3 x_4$ представляет собой рассматриваемую булеву функцию, так как, во-первых, все конъюнкции принадлежат функции /последняя $\bar{x}_1 x_3 x_4 = \underset{9}{\bar{x}_1\bar{x}_2 x_3 x_4} \vee \underset{7}{\bar{x}_1 x_2 x_3 x_4}$ / и, во-вторых, совокупность разложений конъюнкций этой ДНФ содержит все конъюнкции /I-9/ совершенной ДНФ функции.

3. Конъюнкция K <u>называется простой импликантой</u>, если она принадлежит функции, и ее нельзя сократить ни на одну букву без нарушения этого свойства, т.е. удаление каждой из букв приводит к получению конъюнкции, не принадлежащей функции.

<u>Примеры</u>: $K_1 = \bar{x}_2\bar{x}_3\bar{x}_4$ принадлежит функции, имеется три варианта сокращения: $\bar{x}_2\bar{x}_3$, разложение содержит конъюнкцию $x_1\bar{x}_2\bar{x}_3 x_4$, не входящую в совершенную ДНФ;

$\bar{x}_2\bar{x}_4$ – разложение содержит конъюнкцию $\bar{x}_1\bar{x}_2 x_3\bar{x}_4$, не входящую в совершенную ДНФ;

$\bar{x}_3\bar{x}_4$ – разложение содержит конъюнкцию $x_1x_2\bar{x}_3\bar{x}_4$, не входящую в совершенную ДНФ.

Таким образом, $\bar{x}_2\bar{x}_3\bar{x}_4$ – простая импликанта.

$K_2 = x_1x_3$ – тоже простая импликанта, так как два возможных ее сокращения содержат конъюнкции, не входящие в совершенную ДНФ.

x_1 – содержит конъюнкцию $x_1\bar{x}_2\bar{x}_3x_4$;

x_3 – содержит конъюнкцию $\bar{x}_1\bar{x}_2x_3\bar{x}_4$.

Конъюнкция $K_3 = x_1x_2\bar{x}_3x_4$ может быть сокращена, например, до $x_1x_2x_4$, принадлежащей функции /разложение $x_1x_2x_4 = \underset{8}{x_1x_2\bar{x}_3x_4} \vee \underset{6}{x_1x_2x_3x_4}$ /, поэтому K_3 не является простой импликантой.

4. Дизъюнкция всех простых импликант булевой функции называется <u>сокращенной ДНФ</u>.

Очевидно, сокращенная ДНФ представляет собой булеву функцию. Однако несмотря на название /сокращенная/, эта ДНФ не простейшая, более того сокращенная ДНФ многих функций очень сложна. Существуют булевы функции, сокращенная ДНФ которых содержит $\frac{3^n}{n}$ конъюнкции, что уже при $n \geq 4$ намного больше 2^n, т.е. максимальной сложности совершенной ДНФ.

Сокращенная ДНФ содержит все простейшие, не сокращаемые, принадлежащие булевой функции конъюнкции, в этом ее ценность для точных методов минимизации.

5. Минимальной ДНФ называется представляющая функцию ДНФ, имеющая наименьшую сумму букв ее конъюнкций.

<u>Теорема 3.20</u>. Конъюнкции минимальной ДНФ – простые импликанты.

6. Кратчайшей ДНФ называется представляющая булеву функцию ДНФ с наименьшим числом конъюнкций.

Обычно, кратчайшую ДНФ строят тоже из простых импликант.

7. <u>Безызбыточной ДНФ</u> называется представляющая булеву функцию ДНФ, удаление любой конъюнкции из которой приводит к получению ДНФ, уже не представляющей функции.

<u>Теорема 3.21</u>. Минимальные и кратчайшие ДНФ – безызбыточны.

Общая методика точных методов минимизации булевых функций. Алгоритм точных методов можно разделить на два этапа:

построение сокращенной ДНФ, т.е. всех простых импликант, всех простейших, принадлежащих булевой функции конъюнкций;

выбор совокупности простых импликант /безызбыточной, минимальной, кратчайшей/, покрывающей все конъюнкции совершенной ДНФ. Этот этап, очевидно, сводится к задаче о покрытии /безызбыточном, минимальном, кратчайшем/ простыми импликантами конъюнкций совершенной ДНФ согласно их разложениям.

Точные методы различаются в основном реализацией первого этапа минимизации. Примеры минимальных, кратчайших ДНФ приведены далее.

3.4.2. Метод Квайна – Мак-Класки минимизации булевых функций.

Теорема 3.22 /Квайна/. Если, исходя из совершенной ДНФ булевой функции, провести все возможные операции склеивания конъюнкций этой формы, затем всех возможных результатов склеивания, наконец выполнять все поглощения, получится сокращенная ДНФ.

Мак-Класки предложил метод сокращения перебора при проведении операций склеивания и поглощений на основе использования интервальной формы и разбиения множества M_1 на классы M_1^i $(0 \le i \le n)$. Напоминаем, что M_1^i содержит наборы веса i. Основу метода составляют пять следующих очевидных положений:

склеиваться могут наборы только соседних классов M_1^i и M_1^{i+1} ; $(0 \le i \le n-1)$.

если организовать процесс склеиваний так, что результаты склеиваний наборов помещаются в новое множество, то наборы исходного множества и нового множества – не склеиваются; при склеивании наборов множеств M_1^i и M_1^{i+1} получается новое множество M_1^{iH} наборов, содержащих i единиц;

если наборы склеиваются, то они могут участвовать в других склеиваниях, но должны считаться поглощенными;

если наборы исходного множества не склеивались, то они уже не будут поглощаться, т.е. соответствующие им конъюнкции являются простыми импликантами и должны быть включены в сокращенную ДНФ;

новое множество результатов склеиваний может в следующей итерации считаться исходным и с наборами этого множества снова нужно производить все возможные склеивания.

Алгоритм построения множества наборов, соответствующих конъюнкциям сокращенной ДНФ /назовем эти наборы максимальными/.

1. Выписывается множество M_1 наборов и разбивается на классы M_1^i $(0 \le i \le n)$. Исходное множество M_p максимальных наборов пусто.

2. Производятся все возможные склеивания наборов множеств M_1^i и M_1^{i+1} . Склеивающиеся наборы отмечаются, результат склеивания записывается с приведением подобных в множество M_1^{iH} . Пункт 2 выполняется для $i = 0, 1, \dots, n-1$.

3. Если во множествах $M_1^i (0 \le i \le n)$ есть неотмеченные наборы, то они вносятся во множество M_p максимальных наборов.

4. Если множества M_1^{iH} не пусты, то индекс H уничтожается и для них выполняются пп.2–4, иначе множество максимальных наборов построено.

Пример 3.2 /рис.3.18/.

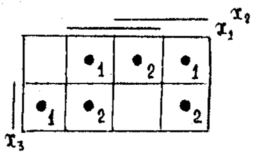


Рис. 3.18

	x_1	x_2	x_3	N		x_1	x_2	x_3	
M_1^1	I	0	0	I V		I	–	0	(I,4)
	0	I	0	2 V		I	0	–	(I,5)
	0	0	I	3 V	M_1^{1H}	–	I	0	(2,4)
M_1^2	I	I	0	4 V		0	I	–	(2,6)
	I	0	I	5 V		–	0	I	(3,5)
	0	I	I	6 V		0	–	I	(3,6)

Чтобы проиллюстрировать процесс склеивания, наборы множества M_1 пронумерованы и рядом с результатами склеиваний записаны пары наборов, участвующие в склеивании. Так как все наборы множества M_1 склеивались, то все они отмечены / V /. Так как в результате склеиваний получилось только одно множества M_1^{1H}, то дальнейший цикл склеиваний не приведет к получению новых результатов склеиваний, таким образом, множество M_p содержит шесть наборов.

$$M_p = \{ I-0,\ I0-,\ -I0,\ 0I-,\ -0I,\ 0-I \}.$$

Сокращенная ДНФ содержит шесть конъюнкций.

$$f_{сокр.ДНФ} = x_1\bar{x}_3 \vee x_1\bar{x}_2 \vee x_2\bar{x}_3 \vee \bar{x}_1 x_2 \vee \bar{x}_2 x_3 \vee \bar{x}_1 x_3 .$$

Пример 3.3 /рис. 3.19/.

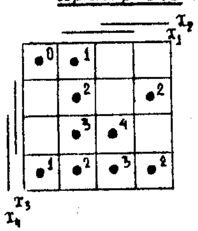


Рис. 3.19

M_1:

x_1	x_2	x_3	x_4	N
0	0	0	0	I V
I	0	0	0	2 V
0	0	0	I	3 V
I	0	I	0	4 V
0	I	I	0	5
I	0	0	I	6 V
0	I	0	I	7 V
I	0	I	I	8 V
I	I	0	I	9 V
I	I	I	I	I0 V

M_1^H:

x_1	x_2	x_3	x_4	
–	0	0	0	(I,2) $\tilde{I}$ V
0	0	0	–	(I,3) $\tilde{2}$ V
I	0	–	0	(2,4) $\tilde{3}$ V
I	0	0	–	(2,6) $\tilde{4}$ V
–	0	0	I	(3,6) $\tilde{5}$ V
0	–	0	I	(3,7) $\tilde{6}$ V
I	0	I	–	(4,8) $\tilde{7}$ V
I	0	–	I	(6,8) $\tilde{8}$ V
I	–	0	I	(6,9) $\tilde{9}$ V
–	I	0	I	(7,9) $\widetilde{I0}$ V
I	–	I	I	(8,I0) $\widetilde{II}$ V
I	I	–	I	(9,I0) $\widetilde{I2}$ V

M_1^{HH}:

x_1	x_2	x_3	x_4	
–	0	0	–	($\tilde{I}$,$\tilde{5}$),($\tilde{2}$,$\tilde{4}$)
I	0	–	–	($\tilde{3}$,$\tilde{8}$), ($\tilde{4}$,$\tilde{7}$)
–	–	0	I	($\tilde{5}$,$\widetilde{I0}$),($\tilde{6}$,$\tilde{9}$)
I	–	–	I	($\tilde{9}$,$\widetilde{II}$),($\tilde{8}$,$\widetilde{I2}$)

$$M_p = \{0II0,\ -00-,\ I0--,\ I--I,\ --0I\}$$

Наборы множества $M_1^{нн}$ уже не склеиваются, каждый из них получается дважды.

Общая формулировка метода Квайна – Мак-Класки. Как все классические методы минимизации булевых функций, метод Квайна – Мак-Класки выполняется в два этапа.

I. Построение сокращенной ДНФ /множества M_p максимальных наборов/.

2. Построение таблицы покрытий, решение задачи о покрытии и выбор наилучшего решения в зависимости от критерия /всех минимальных или всех кратчайших, одного минимального или одного кратчайшего/.

Реализация второго этапа. Столбцы таблицы покрытий ставятся в соответствие наборам множества M_1, строки – наборам множества M_p. Элементы T_{ij} таблицы покрытий вычисляются по следующему правилу:

$$T_{ij} = \begin{cases} \text{I, если значения внешних переменных набора } m_i \in M_p \text{ совпадают с} \\ \quad \text{набором } m_j \in M_1 \text{ ;} \\ \text{0, если не совпадают.} \end{cases}$$

Иначе говоря, T_{ij} = I, если набор $m_j \in M_1$ входит в разложение набора $m_i \in M_p$.

Для примеров 3.2 и 3.3.

	IOO	OIO	OOI	IIO	IOI	OII
I-O	I			I		
IO-	I				I	
-IO		I		I		
OI-		I				I
-OI			I		I	
O-I			I			I

	0000	1000	0001	1010	0110	1001	0101	1011	1101	1111
OIIO					I					
-OO-	I	I	I			I				
IO--		I		I		I		I		
I--I						I		I	I	I
--OI			I			I	I		I	

Первая таблица, очевидно, циклическая и решение получается методом разложения по столбцу, строки второй таблицы ядерные и имеется одно минимальное /оно же кратчайшее/ решение

$$f = \bar{x}_1 x_2 x_3 \bar{x}_4 \vee \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_2 \vee x_1 x_4 \vee \bar{x}_3 x_4 .$$

Проведем разложение /рис. 3.20/ первой таблицы с получением всех минимальных покрытий /вес строки - число внешних переменных наборов множества M_p/.

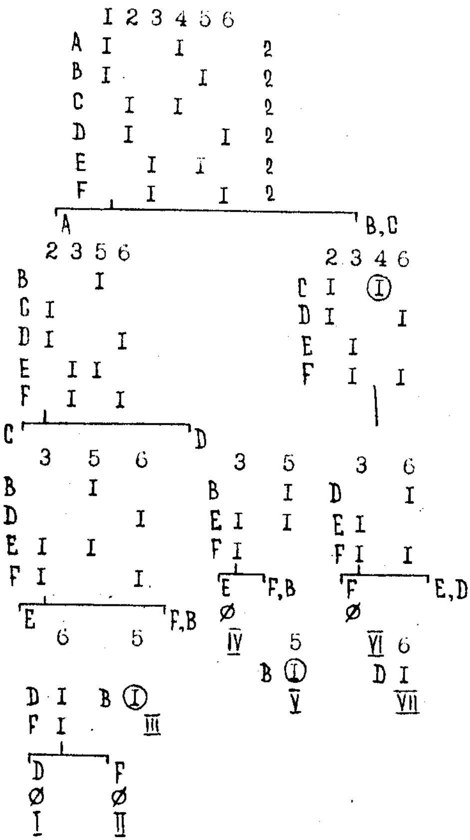


Рис. 3.20

Получено семь покрытий: I: $A, C, E, D (8)$; $\underline{II}$: $A, C, E, F (8)$; $\underline{III}$: $A, C, F, B (8)$; $\underline{IV}$: $A, D, E (6)$; $\underline{V}$: $A, D, F, B (8)$; $\underline{VI}$: $B, C, F (6)$; $\underline{VII}$: $B, C, E, D (8)$, из них – два минимальных /IУ и УI/. Этим покрытиям соответствуют две минимальные ДНФ:

$$f_{\underline{IV}} = x_1\bar{x}_3 \vee \bar{x}_1 x_2 \vee \bar{x}_2 x_3 \ ; \quad f_{\underline{VI}} = x_1\bar{x}_2 \vee x_2\bar{x}_3 \vee \bar{x}_1 x_3 .$$

Рассмотрим еще один пример, иллюстрирующий метод Квайна – Мак-Класки в целом. Получим хотя бы одну минимальную ДНФ /рис.3.21/.

x_1	x_2	x_3	x_4	
I	0	0	0	V
0	I	0	0	V
0	0	I	0	V
0	0	0	I	V
I	I	0	0	V
I	0	I	0	V
0	I	I	0	V
0	0	I	I	V
I	0	0	I	V
0	I	I	I	V
I	I	0	I	V
I	I	I	I	V

x_1	x_2	x_3	x_4	
I	–	0	0	V
I	0	–	0	
I	0	0	–	V
–	I	0	0	
0	I	–	0	
–	0	I	0	
0	–	I	0	V
0	0	I	–	V
0	0	–	I	
–	0	0	I	
I	I	0	–	V
0	I	I	–	V
0	–	I	I	V
I	–	0	I	V
–	I	I	I	
I	I	–	I	

x_1	x_2	x_3	x_4
I	–	0	–
0	–	I	–

	1 1000	2 0100	3 0010	4 0001	5 1100	6 1010	7 0110	8 0011	9 1001	10 0111	11 1101	12 1111	
A I – 0 –	I				I				I		I		2
B 0 – I –			I				I	I		I			2
C I 0 – 0	I					I							3
D – I 0 0		I			I								3
E 0 I – 0		I					I						3
F – 0 I 0			I			I							3
G 0 0 – I				I				I					3
H – 0 0 I				I					I				3
I – I I I										I		I	3
J I I – I											I	I	3

A, E, F, G, I

C, B, D, H, J

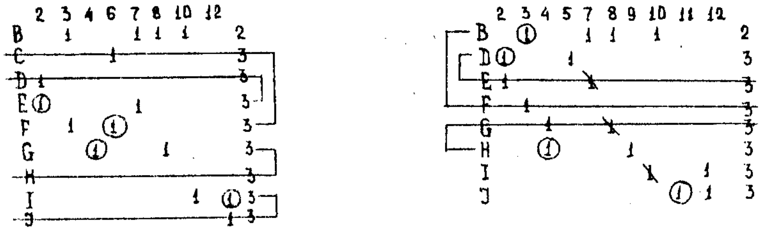


Рис. 3.21

Получены два минимальных покрытия, т.е. две минимальные ДНФ:

$$f = x_1\bar{x}_3 \vee \bar{x}_1 x_2 \bar{x}_4 \vee \bar{x}_2 x_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_4 \vee x_2 x_3 x_4 =$$
$$= x_1\bar{x}_2\bar{x}_4 \vee \bar{x}_1 x_3 \vee x_2\bar{x}_3\bar{x}_4 \vee \bar{x}_2\bar{x}_3 x_4 \vee x_1 x_2 x_4 .$$

3.4.4. Неполностью определенные булевы функции и обобщение метода Квайна-Мак-Класки.

Булева функция называется <u>неполностью определенной</u>, если на некоторых наборах ее значение неопределено /равно $X \in \{0,1\}$/ /рис. 3.22/.

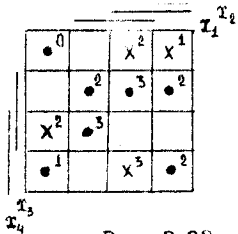


Рис. 3.22

Существуют две причины неопределенности булевой функции: значение функции на данном наборе безразлично в конкретном применении, набор в конкретном приложении не может быть сформирован.

Неопределенные значения можно использовать при минимизации булевых функций для упрощения ДНФ. При построении сокращенной ДНФ неопределенные элементы можно склеивать, получая максимальные наборы с большим множеством внутренних переменных. При решении задачи покрытия можно эти наборы специально не покрывать. Обозначив через M_x множество наборов, на которых функция неопределена, можно сформулировать <u>обобщение метода Квайна - Мак-Класки для неполностью определенных булевых функций</u>.

1. При построении множества M_p максимальных наборов исходным рассматриваем множество $M_1 \cup M_x$.

2. При построении таблицы покрытий столбцам сопоставляем только элементы множества M_1, строкам – элементы множества M_p.

Пример /рис. 3.22/.

$x_1 x_2 x_3 x_4$		
0 0 0 0 V	0 – 0 0 V	0 – 0 –
0 I 0 0 V	0 0 0 – V	– I – 0
0 0 0 I V	– I 0 0 V	– I 0 –
I I 0 0 V	0 I – 0 V	
I 0 I 0 V	0 I 0 – V	
0 I I 0 V	0 0 – I	
0 0 I I V	0 – 0 I V	
0 I 0 I V	I I – 0 V	
0 I I 0 V	I I 0 – V	
I 0 I I V	I – I 0	
I I 0 I V	I 0 I –	
I I I 0 V	– I I 0 V	
	– 0 I I	
	– I 0 I V	

	0000	1010	1110	0110	1011	0001	0101	
00-1						1		3
1-10		1	1					3
101-		1			1			3
-011					1			3
0-0-	1					1	1	2
-1-0			1	1				2
-10-							1	2

$$f = x_1\bar{x}_2x_3 \vee \bar{x}_1\bar{x}_3 \vee x_2\bar{x}_4 .$$

3.4.5. Задания к подразд. 3.4.

А. Минимизировать методом Квайна – Мак-Класки пороговые булевы функции п. 3.I.6.А, сравнить полученные ДНФ с результатом подразд. 3.2.

Убедиться, что пороговые функции имеют минимальную ДНФ, совпадающую с сокращенной ДНФ.

Б. Убедиться, что линейная булева функция /рис. 3.23/ не минимизируется.

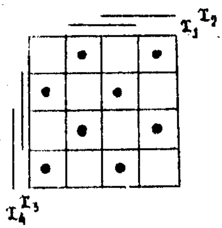
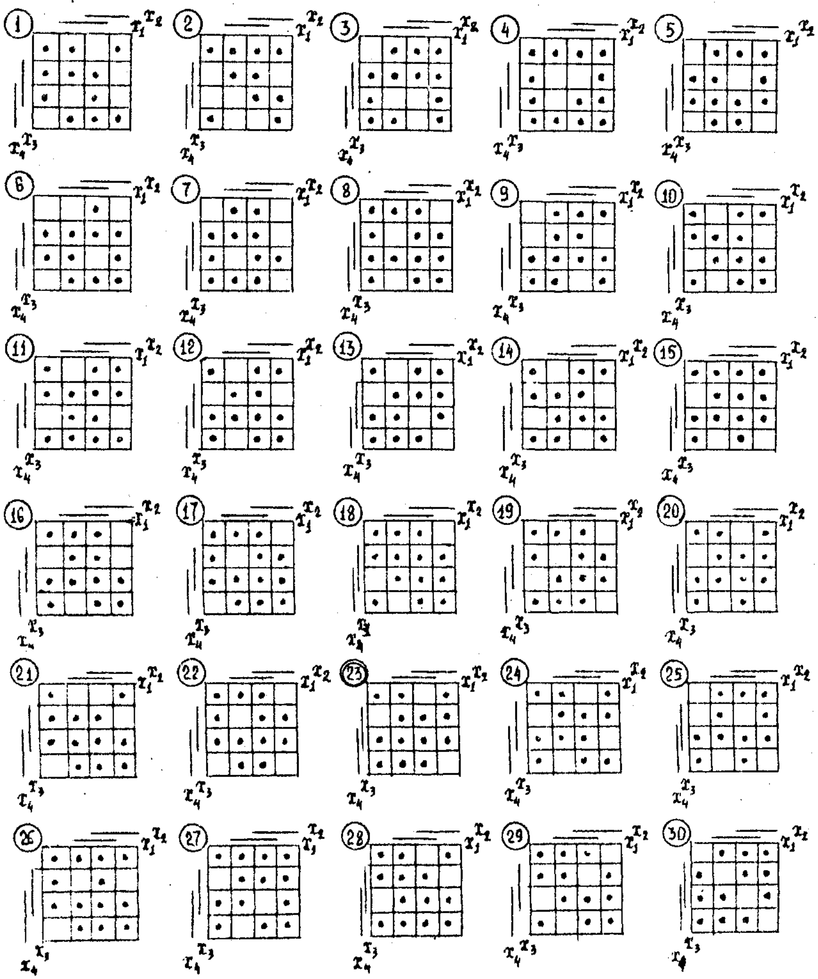


Рис. 3.23

В. Минимизировать методом Квайна - Мак-Класки следующие булевы функции.



3.5. Упрощение ДНФ булевых функций по матричной форме /метод Закревского/

3.5.1. Построение максимальных интервалов для заданного элемента матричной формы.

В подразд. 3.2 приведены приемы изображения интервалов на матричной форме, строящихся по векторному интервальному заданию. В данном разделе рассматриваются обратная и более сложная задача – выделение интервалов по матричному заданию булевой функции. Симметрия матричной формы в коде Грея поможет построить интервалы.

Оси симметрии и зоны их действия.

Осью симметрии по переменной x_i называется линия смены значения этой переменной на матричной форме. Для некоторых переменных несколько осей симметрии. На матричной форме /рис. 3.24/ выделены оси симметрии всех переменных. Показаны для выделенных элементов $(0, \Delta)$ симметричные элементы по всем осям симметрии.

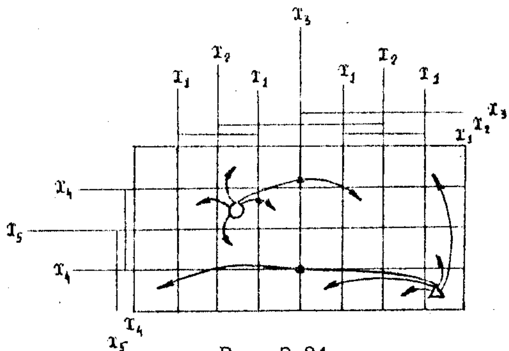
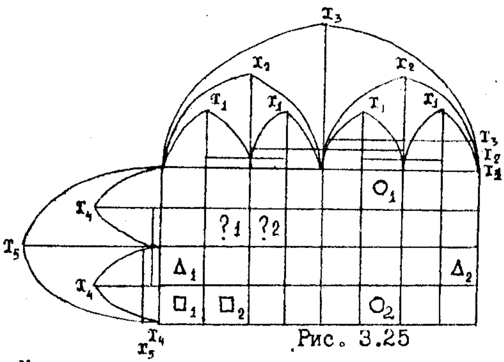


Рис. 3.24

Оси симметрии не наносят, их представляют мысленно. Каждая ось симметрии имеет зону своего действия, если ось одна и зона ее действия – вся матрица, если две – половина матрицы около оси и т.д. Зоны действия осей изображены на рис. 3.25.

Утверждение. Элементы матричной формы, соответствующие склеивающимся наборам, расположены симметрично относительно оси симметрии переменной, по которой наборы склеиваются, и в зоне ее действия /рис. 3.26/. Это утверждение выражает основное свойство матричной формы в коде Грея, которое помогает при поиске интервалов на матрице.



?1 I 0 0 I 0
?2 I I 0 I 0 } ось x2

Δ1 0 0 0 I I
Δ2 0 0 I I I } ось x3

□1 0 0 0 0 I
□2 I 0 0 0 I } ось x1

O1 I I I 0 0
O2 I I I 0 I } ось x5

Δ1 0 0 0 I I
□1 0 0 0 0 I } ось x4

Рис. 3.25

Новая трактовка понятия интервала.

Дадим определение интервала с учетом свойства симметрии. Интервалом называется совокупность 2^K элементов матричной формы, симметрично расположенных по K осям внутренних переменных в зоне их действия. Интервал принадлежит функции, если на всех его элементах функция равна единице или не определена /для не полностью определенных функций/. Интервал называется максимальным, если он принадлежит функции, но при симметрировании его по каждой из осей внешних переменных происходит поглощение элементов, на которых функция равна 0. На рис. 3.26 показаны примеры максимальных интервалов с выделением осей внутренних переменных, по которым элементы интервалов составляют симметричную фигуру.

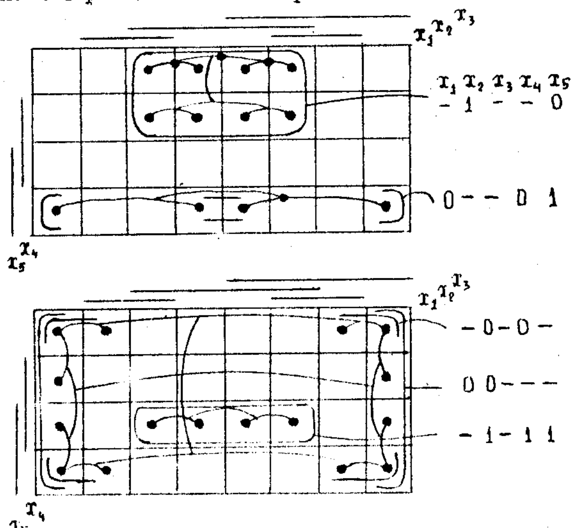


Рис. 3.26

На рис. 3.27 изображена совокупность 2^2 элементов, которая не является интервалом, так как эта фигура не симметрична относительно осей симметрии элементов в зоне их действия.

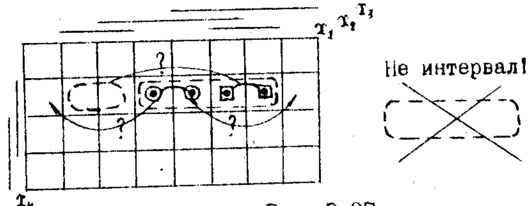
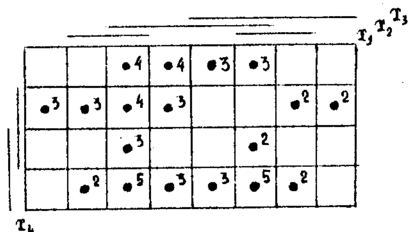


Рис. 3.27

Элементы, обозначенные значком ○ , симметричны по оси x_3 , поэтому остальные элементы, чтобы формировать интервал, должны быть симметричными по этой же оси, однако это не так. Элементы □ симметричны по оси x_2 , поэтому элементы ○ должны иметь в интервале симметричные элементы, причем для каждого по той оси x_2 , в зоне действия которой этот элемент находится.

Построение максимальных интервалов.

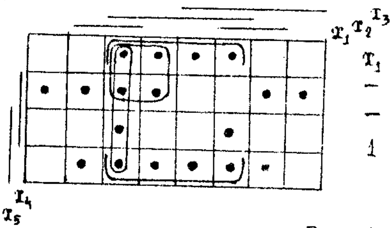
Прежде всего необходимо определить множества $X^m_{пвп}$ потенциально внутренних переменных для выделенного элемента $m \in M_1$, т.е. тех переменных, по осям симметрии которых для исходной точки симметрично расположены точки или неопределенные элементы (X) . Будем называть такие симметрично расположенные точки или неопределенные элементы соседями. На рис. 3.28 рассчитано число соседей для всех точек и показаны множества $X^m_{пвп}$ для некоторых из них.



Код m точки;	$X_{пвп}$;
I I 0 0 0	x_1 x_3 x_4 x_5
0 0 0 I 0	x_1 x_2 x_3
I 0 I I 0	x_1 x_3
I 0 0 0 I	x_2 x_3
I I 0 0 I	x_1 x_2 x_3 x_4 x_5

Рис. 3.28

Потенциально внутренние переменные называются так потому, что исходная точка может склеиться, образовав интервал с каждым из своих соседей, и тогда соответствующая переменная станет внутренней. Однако не всегда все потенциально внутренние переменные одновременно могут быть внутренними в одном интервале. Например, для точки 11000 четыре переменных потенциально внутренние /рис. 3.29/, однако максимальные интервалы, покрывающие эту точку, имеют каждый внутренними только некоторое подмножество этих переменных /рис. 3.29/.



x_1	x_2	x_3	x_4	x_5	Внутренние переменные
–	1	–	0	–	$x_1\ x_3\ x_5$
–	1	0	–	0	$x_1\ x_4$
1	1	0	–	–	$x_4\ x_5$

Рис. 3.29

Для определения максимальных интервалов, содержащих заданную точку, необходимо находить максимальные совместимые подмножества переменных множеств $X_{пвп}$ этой точки. Очевидно, в таком поиске нужно использовать алгоритм граничного перебора на выпуклом множестве, трансформировав процедуру проверки совместимости переменных: переменные совместимы, если симметричный по ним интервал будет принадлежать функции.

Формулировка алгоритма построения максимальных интервалов для заданной точки

1. Строится множество $X^{m}_{пвп}$ для заданной точки. Переменные этого множества будут участвовать в граничном переборе. Точка m рассматривается как текущий интервал.

2. Методом граничного перебора выбирается очередная переменная, по которой проверяется возможность симметрирования текущего интервала по оси симметрии выбранной переменной. Если симметрируемый интервал попадает только на элементы точка или X , то расширение интервала возможно, переменная вводится в текущее множество внутренних переменных и расширенный интервал рассматривается в качестве текущего. Если граничный перебор предлагает для рассмотрения следующую переменную, то выполняется п.2, иначе, если множество внутренних переменных не поглощается ранее построенными, получаем очередной максимальный интервал, векторное представление которого строится заменой в коде m исходной точки значений внутренних переменных на черточку /-/.

3. Процедура граничного перебора при переходе к построению следующего интервала удаляет несколько переменных из построенного множества внутренних переменных. При удалении каждой переменной текущий интервал сокращается симметрично по оси соответствующей переменной так, чтобы

по-прежнему содержать исходную точку. Если граничный перебор предусматривает дальнейшие построения, то формируется текущее множество внутренних переменных и выполняется п.2 алгоритма, иначе процесс заканчивается.

При реализации алгоритма используется векторное представление подмножеств внутренних переменных интервала. Рассмотрим пример /рис. 3.30/.

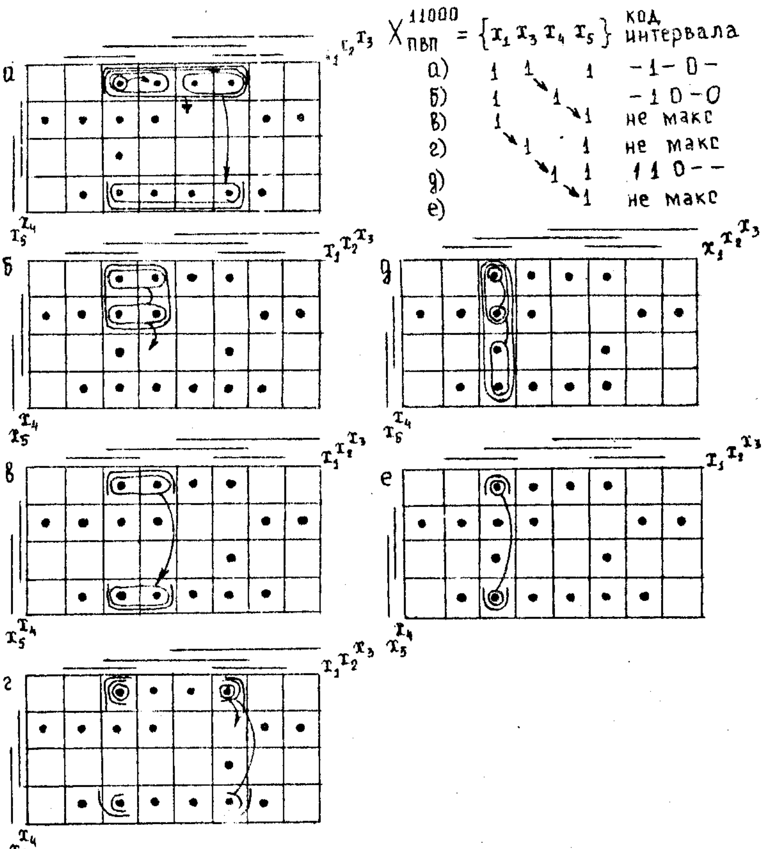


Рис. 3.30

На рис. 3.30 показана вся последовательность как расширения, так и сужения текущего интервала согласно алгоритму граничного перебора. Стрелками показаны факты невозможности расширения текущего интервала по соответствующей переменной. Естественно, что при решении не нужно перечерчивать матрицу, текущие интервалы и их изменения нужно хранить в памяти.

Приведем еще пример, по-прежнему перечерчивая матрицы /рис. 3.31/.

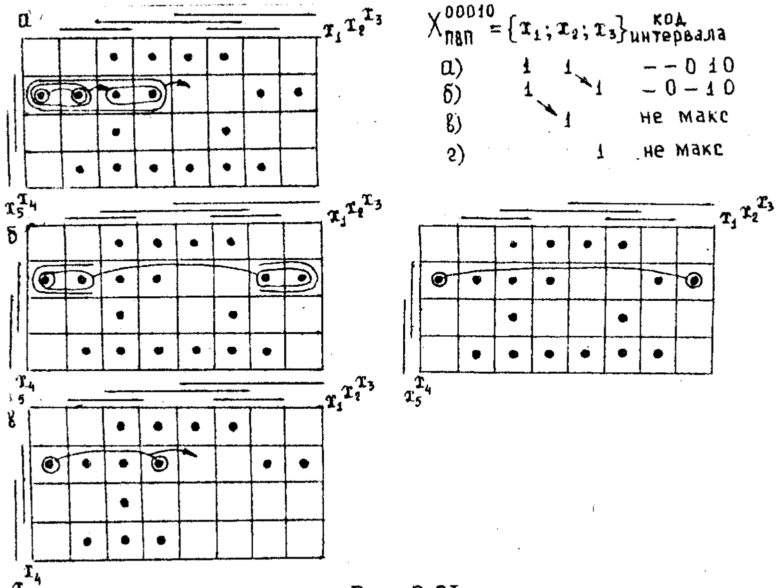


Рис. 3.31

Очевидно, что предложенный алгоритм строит все максимальные интервалы для заданной точки, что гарантируется применением процедуры граничного перебора. В случае небольшого числа потенциально внутренних переменных, все построения можно выполнять мысленно, по крайней мере при $|X^m_{пап}| \in \{0,1,2\}$.

3.5.2. Алгоритм упрощения ДНФ.

Формулировка алгоритма:

1. Для каждого элемента $m \in M_1$ рассчитывается число K соседних элементов множества $M_1 \cup M_x$.

2. Выбирается элемент $m \in M_1$ с меньшим значением K /если таких элементов несколько, то для определенности алгоритма выбирается первый/.

3. Находятся максимальные интервалы, содержащие элемент m , и среди них выбирается один, который покрывает наибольшее число остающих-

ся в множестве M_1 элементов /уточняющие правила выбора приведены в замечаниях/.

4. Конъюнкция, соответствующая выбранному максимальному интервалу, записывается в результирующую ДНФ; все элементы множества M_1, покрытые выбранным интервалом, переводятся в M_x /• заменяется на х /.

5. Если на матрице еще есть элементы множества M_1, то выполняется п.2, иначе - следующий пункт.

6. На матрицу наносятся все интервалы, выбранные при построении упрощенной ДНФ, и если хотя бы для одного из них все точки покрыты другими выбранными интервалами, то такой интервал удаляется и вычеркивается соответствующая конъюнкция из ДНФ. Проверка безызбыточности оставшейся совокупности интервалов проводится до тех пор, пока в каждом интервале не окажется хотя бы по одной точке, покрытой только этим интервалом. Эта процедура гарантирует безызбыточность построенной ДНФ.

Замечания к алгоритму

1. О самопроверке. Процедура п.6 алгоритма позволяет проверить правильность решения, построенная ДНФ должна представлять функцию /все элементы множества M_1 /точки/ покрыты хотя бы одним из интервалов, ни один из элементов множества M_0 /пустые элементы матрицы/ не покрыт ни одним из интервалов/.

2. О точности метода Закревского. Точный метод Квайна - Мак-Класки использует для гарантии построения оптимальных решений таблицу покрытий. В разд. I приведен приближенный метод построения покрытий, метод минимального столбца - максимальной строки. Метод Закревского статистически аналогичен этому приближенному методу по точности построения результата: выбор элемента с меньшим числом соседей статистически эквивалентен выбору минимального столбца /чем меньше потенциально-внутренних переменных, тем статистически меньше число максимальных совместимых подмножеств и, значит, максимальных интервалов, покрывающих данную точку/; выбор среди интервалов, покрывающих данную точку, интервала с большим числом элементов множества M_1, полностью эквивалентен выбору максимальной строки, имеющей единицу в рассматриваемом столбце. Однако метод Закревского не связан с построением всех максимальных интервалов таблицы покрытий, и поэтому значительно проще.

3. О ядерных интервалах. В точном методе значительное сокращение процесса решения достигается выделением ядерных строк таблицы покрытий, т.е. строк, содержащих единицу хотя бы в одном таком столбце, в котором эта единица - единственная. Так как строка соответствует интервалу, а столбец - элементу множества M_1, то ядерный интервал - это такой

интервал, который содержит хотя бы один элемент множества M_1, который покрывается только одним интервалом. Для такого элемента все потенциально-внутренние переменные составляют совместимое подмножество, т.е. все соседи элемента входят в один интервал. Это последнее свойство – удобный критерий выделения ядерных интервалов. При расчете числа соседей можно проверять, нельзя ли образовать интервал, покрывающий всех соседей данной точки одновременно. Такая проверка при числе соседей $K = 0,1,2,3$ не занимает много времени. Если для некоторой точки оказывается, что все соседи ее покрываются одним интервалом, то найден ядерный интервал и его нужно сразу же ввести в решение, все покрытые им точки перевести в множество M_x и не считать для них соседей. Выделение ядерных интервалов повышает точность метода Закревского.

4. О возможных сокращениях процесса решения в методе Закревского:

а/ число K соседей необходимо рассчитывать на начальных этапах освоения алгоритма, либо при программировании алгоритма для ЭВМ. По мере освоения, приобретения навыка решения такой подсчет можно не производить, достаточно в п.2 алгоритма выделять интуитивно наименее симметричную точку;

б/ при построении максимальных интервалов для выделенной точки граничный перебор при числе соседей $K = 0,1,2$ необходимо даже при освоении алгоритма вести устно, и только при $K = 3,4...$ – письменно. Так как выбираются для рассмотрения точки с меньшим числом соседей, то выполнять перебор письменно приходится редко. По мере приобретения навыка и этот перебор тоже нужно вести устно – интуитивно находить максимальные интервалы. Поэтому А.Д.Закревский свой метод назвал визуальным.

5. О выборе интервалов для рассматриваемой точки. Рекомендуется среди максимальных интервалов, содержащих рассматриваемую точку, выбирать тот, который покрывает наибольшее число элементов, оставшихся принадлежать множеству M_1 /не переведенных еще в множество M_x /. Если таких интервалов несколько, то предпочтение нужно отдать тому, который содержит точки с меньшим числом соседей, если таких интервалов тоже несколько, то тому, который покрывает большее число элементов, исходно принадлежащих множеству M_1 .

Рассмотрим несколько примеров.

Для каждого элемента множества M_1 на рис. 3.32 указано число K соседей. Точка 1011 с $K = 0$ покрывается единственным интервалом, содержащим только этот элемент. Для выбранной точки 0000/$K = 2$/ существуют два максимальных интервала – 000 и 000 –, первый содержит два элемента множества M_1 , второй – один. Выбирается интервал – 000. Текущая ДНФ $f = x_1\bar{x}_2x_3x_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4 \vee ...$.

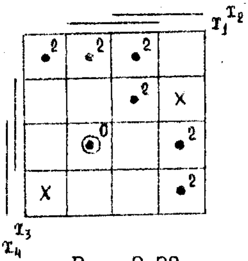


Рис. 3.32

Преобразованная функция изображена на рис. 3.33. Для оставшихся точек аналогичные рассуждения приводят к выбору интервалов, показанных на рис.3.34.

$$f = x_1 x_2 x_3 x_4 \vee \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee x_1 x_2 \bar{x}_4 \vee \bar{x}_1 x_2 x_4 .$$

На рис. 3.34 изображено покрытие точек выбранной совокупностью интервалов. Она безызбыточна, построенная ДНФ – также безызбыточна.

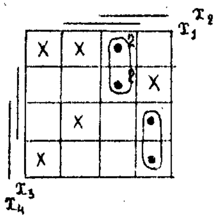


Рис. 3.33

Немного изменим булеву функцию /рис. 3.35/. Для элемента 0000/К = 2/ два возможных интервала – 000 и 000–, оба содержат по два элемента множества M_1. Выбор случаен – пусть выбран интервал – 000. Для следующей точки II00 снова два варианта I–00 и II–0, оба содержат только один элемент множества /II00/, однако интервал I–00 содержит два элемента первоначального множества M_1, поэтому он выбирается.

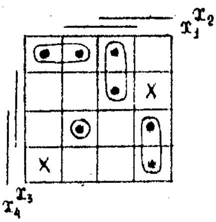


Рис. 3.34

Оставшиеся элементы /рис. 3.36/ покрываются при аналогичных рассуждениях: элементы 0II0 и 0III интервалом 0II–, покрывающим два элемента множества M_1, элемент 000I – интервалом 000–, покрывающим два элемента исходного множества M_1:

$$f = \bar{x}_2 \bar{x}_3 \bar{x}_4 \vee x_1 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 .$$

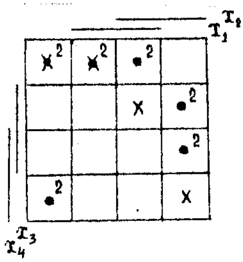


Рис. 3.35

Покрытие элементов множества M_1 выбранной совокупностью интервалов показано на рис. 3.37. Очевидно, интервал – 000 избыточен – оба его элемента покрываются другими выбранными интервалами. После его ударения совокупность интервалов становится безызбыточной /рис. 3.38/:

$$f = x_1 \bar{x}_3 \bar{x}_4 \vee \bar{x}_1 x_2 x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3 .$$

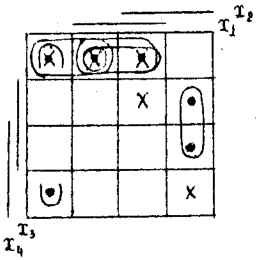


Рис. 3.36

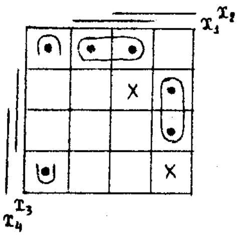


Рис. 3.37

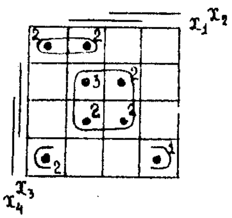


Рис. 3.38

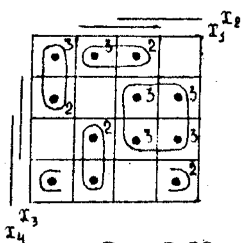


Рис. 3.39

Рассмотрим несколько примеров без комментариев. Процесс решения зафиксирован в последовательности перечисления конъюнкций в ДНФ /рис. 3.38/

$$f = \bar{x}_1\bar{x}_3x_4 \vee \bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_3.$$

Последовательность выбора интервалов полностью соответствует основным особенностям метода Закревского.

Очень не характерный /трудный/ для метода Закревского пример /рис. 3.39/. Такое хорошее решение получено скорее интуитивно:

$$f = \bar{x}_1\bar{x}_2x_4 \vee x_1\bar{x}_3\bar{x}_4 \vee x_1\bar{x}_2x_4 \vee \bar{x}_1\bar{x}_3x_4 \vee x_2x_3.$$

$$f = \bar{x}_2\bar{x}_3\bar{x}_4 \vee x_1x_3 \vee \bar{x}_1x_2 .$$

Решение получено в полном соответствии с методом Закревского. Кстати, все интервалы – ядерные /рис. 3.40/.

Этот пример решается с использованием понятия ядерных интервалов /рис. 3.41/. Число соседей рассчитано только для тех точек, которые интуитивно мало симметричны /и поэтому с большой вероятностью покрываются ядерными интервалами/

$$f = x_1\bar{x}_2x_3x_5 \vee \bar{x}_1x_4\bar{x}_5 \vee \bar{x}_1x_2x_3 \vee x_2\bar{x}_4 \vee x_1\bar{x}_2\bar{x}_3\bar{x}_5.$$

Заметим, что правильный выбор последнего интервала 100-0 / неядерного/ был обусловлен предварительным выбором ядерных интервалов /рис.3.42/.

$$f = x_1\bar{x}_4\bar{x}_5 \vee \bar{x}_1x_4 \vee x_1\bar{x}_2\bar{x}_3x_5 \vee x_2\bar{x}_3\bar{x}_4x_5.$$

<u>Замечание</u>. Матричная форма широко используется в большинстве курсов про-

ектирования дискретных устройств /прикладная теория цифровых автоматов, теория и проектирование ЦВМ, схемотехника и т.д./ при иллюстрации приемов и методов проектирования. Чтобы хорошо усваивать особенность этих приемов, необходимо свободное владение преобразованиями интервалов на матричной форме. В большинстве современных монографий советских и зарубежных авторов для этих целей тоже используется матричная форма в коде Грея. Поэтому необходимо самостоятельное решение достаточного количества примеров /10-15 различных функций 4-5 переменных/.

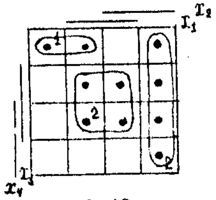


Рис. 3.40

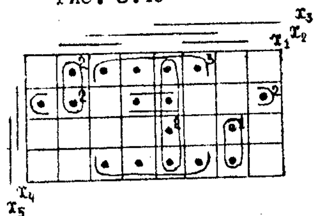


Рис. 3.4I

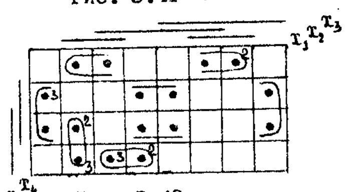


Рис. 3.42

3.5.3. Вопросы и задания к подразд. 3.5.

А. По матричным формам /с.171/ построить максимальные интервалы для выделенных точек.

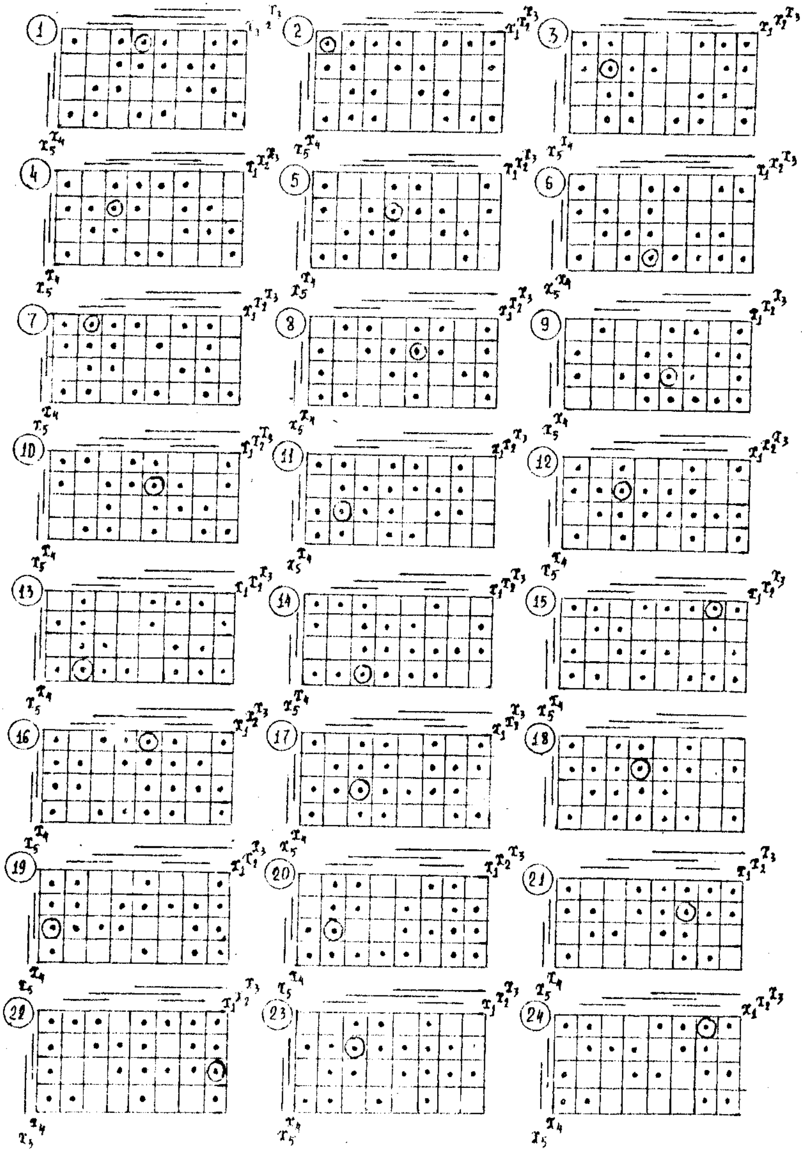
Б. Построить упрощенные ДНФ пороговых булевых функций /задание 3.I.6.A/.

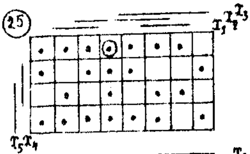
В. Построить упрощенные ДНФ булевых функций четырех переменных /с. 160 /.

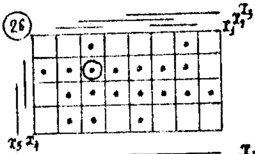
Г. Построить упрощенные ДНФ булевых функций пяти переменных /с. 172 /

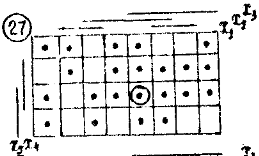
Д. Построить упрощенные ДНФ булевых функций шести переменных /с.173/.

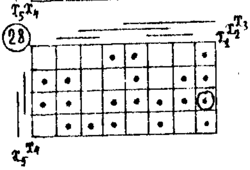
Булевы функции пяти переменных

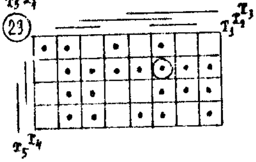


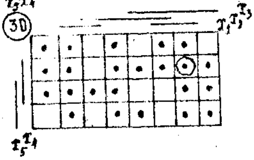
25


26


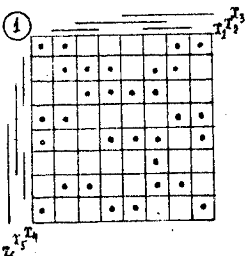
27


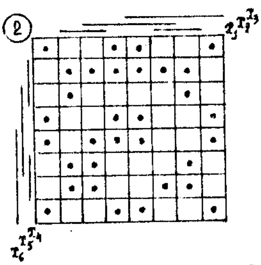
28


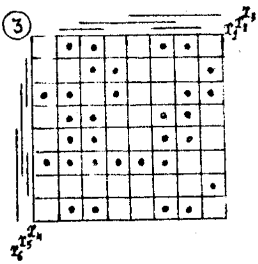
29


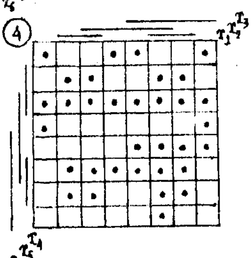
30


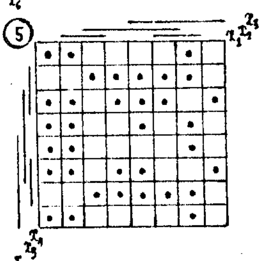
Булевы функции шести переменных

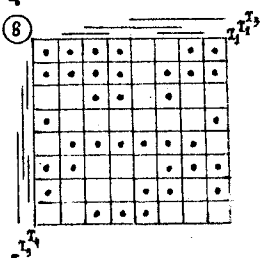
1


2


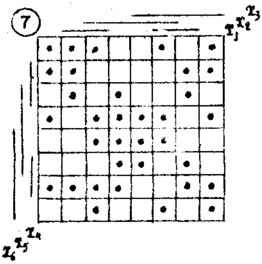
3


4


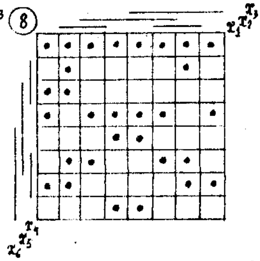
5


6


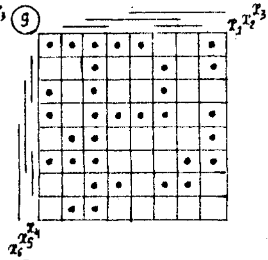
7



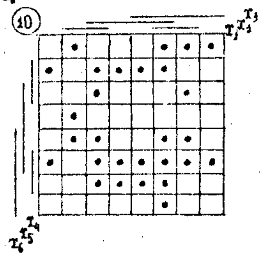
8



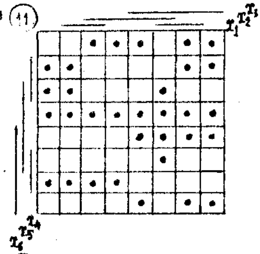
9



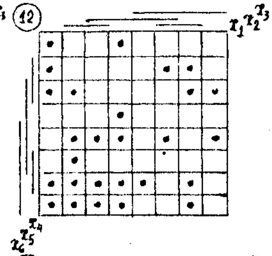
10



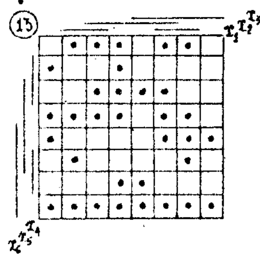
11



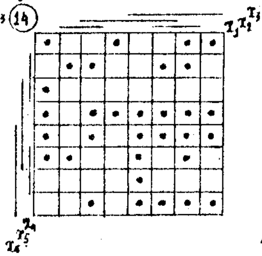
12



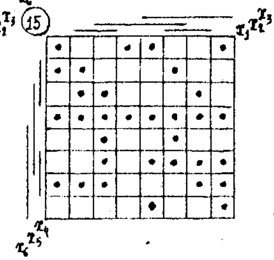
13



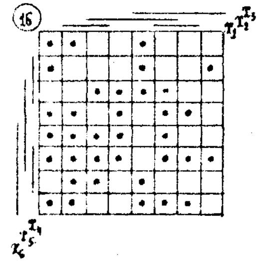
14



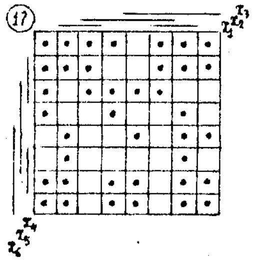
15



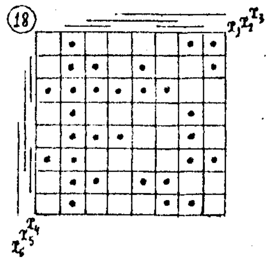
16

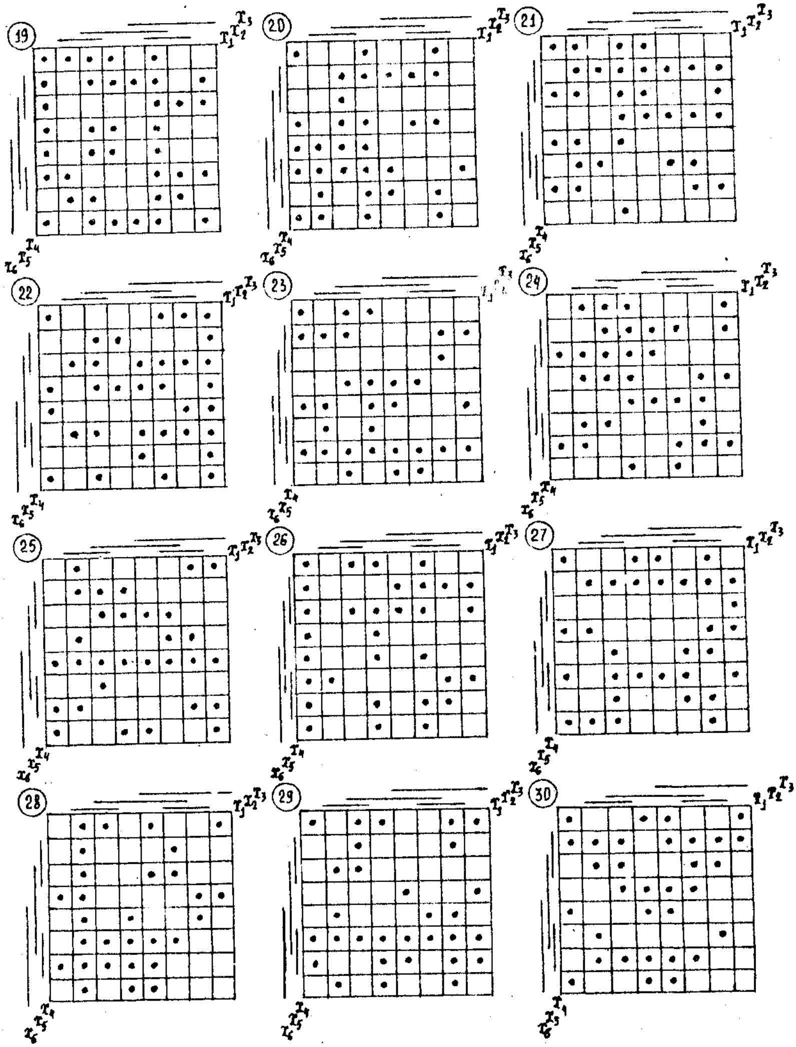


17



18




19
20
21
22
23
24
25
26
27
28
29
30

3.6. Минимизация систем булевых функций

3.6.I. Определения.

В большинстве приложений используются не отдельные булевы функции, а системы вида

$$
\begin{array}{l}
f_1(x_1,\dots,x_n)\\
f_2(x_1,\dots,x_n)\\
\dots\dots\dots\dots\\
f_m(x_1,\dots,x_n).
\end{array}
$$

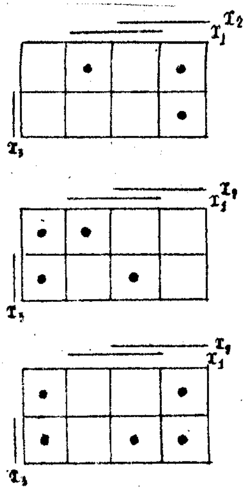


Рис. 3.43

На рис. 3.43 показан конкретный пример. Если минимизировать каждую функцию в отдельности, то

$$
\begin{aligned}
f_1 &= x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1 x_2,\\
f_2 &= \bar{x}_2\bar{x}_3 \vee \bar{x}_1\bar{x}_2 \vee x_1 x_2 x_3, \qquad /3.4/\\
f_3 &= \bar{x}_1 \vee x_2 x_3.
\end{aligned}
$$

Сложность системы ДНФ оценивается семью различными конъюнкциями /или I5 символами/. Можно предложить другую форму, построенную с учетом того, что функции составляют единую систему

$$
\begin{aligned}
f_1 &= x_1\bar{x}_2\bar{x}_3 \vee \bar{x}_1 x_2,\\
f_2 &= \bar{x}_1\bar{x}_2 \vee x_1\bar{x}_2\bar{x}_3 \vee x_1 x_2 x_3, \qquad /3.5/\\
f_3 &= \bar{x}_1\bar{x}_2 \vee x_1 x_2 x_3 \vee \bar{x}_1 x_2,
\end{aligned}
$$

сложность которой оценивается в четыре различные конъюнкции, содержащие I0 символов.

Определение. Кратчайшей /минимальной/ ДНФ системы булевых функций называется такая система ДНФ, которая содержит наименьшее число различных конъюнкций, составляющих эти ДНФ /наименьшее число символов, составляющих эти различные конъюнкции/.

Каждая конъюнкция, кроме составляющих ее переменных, характеризуется тем, в ДНФ каких функций эта конъюнкция входит. Множество функций, приписываемое каждой конъюнкции, назовем ее ярлыком. Систему ДНФ можно однозначно представить списком конъюнкций с ярлыками:

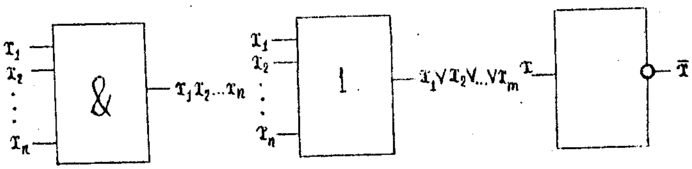
для системы ДНФ /3.4/ – $x_1\bar{x}_2\bar{x}_3 f_1, \bar{x}_1 x_2 f_1, \bar{x}_1\bar{x}_2 f_2, \bar{x}_2\bar{x}_3 f_2, x_1 x_2 x_3 f_2, \bar{x}_1 f_3, x_2 x_3 f_3$;
для системы ДНФ /3.5/ – $x_1\bar{x}_2\bar{x}_3 f_1 f_2, \bar{x}_1 x_2 f_1 f_3, x_1 x_2 x_3 f_2 f_3, \bar{x}_1\bar{x}_2 f_2 f_3$.

Определение. Простой импликантой системы булевых функций называется конъюнкция с ярлыком такая, что нельзя сократить конъюнкцию или увеличить ярлык, поскольку тогда конъюнкция перестает принадлежать хотя бы одной функции ярлыка.

Для рассматриваемой системы булевых функций простыми импликантами являются все перечисленные в приведенном ранее списке для /3.5/ конъюнкции с ярлыками. Существует еще ряд простых импликант – $\bar{x}_2\bar{x}_3 f_2$, $\bar{x}_1 f_3$, $x_2 x_3 f_3$. Таким образом, простыми импликантами системы являются и $\bar{x}_1 x_2 f_1 f_3$ и $\bar{x}_1 f_3$: в первой – больше ярлык, во второй – меньше символов в конъюнкции /больше соответствующий интервал булева пространства/.

3.6.2. Пример использования системы булевых функций для синтеза комбинационных дискретных устройств.

Цифровая техника строится на микросхемах, алгоритмы работы которых описываются булевыми функциями. Совокупность применяемых при проектировании конкретного устройства микросхем определяет элементный базис проектирования. Простейший элементный базис /правда, реально редко используемый/ – это базис И, ИЛИ, НЕ, который в схемотехнике имеет обозначение, показанное на рис. 3.44.



Элемент И Элемент ИЛИ Элемент НЕ

Рис. 3.44

В этом базисе системы ДНФ /3.4/ и /3.5/ реализуются следующими схемами /рис. 3.45/.

Сравнение схем на рис. 3.45,а и б наглядно иллюстрирует преимущество совместной минимизации систем булевых функций и подтверждает целесообразность введенных определений кратчайшей и минимальной ДНФ системы.

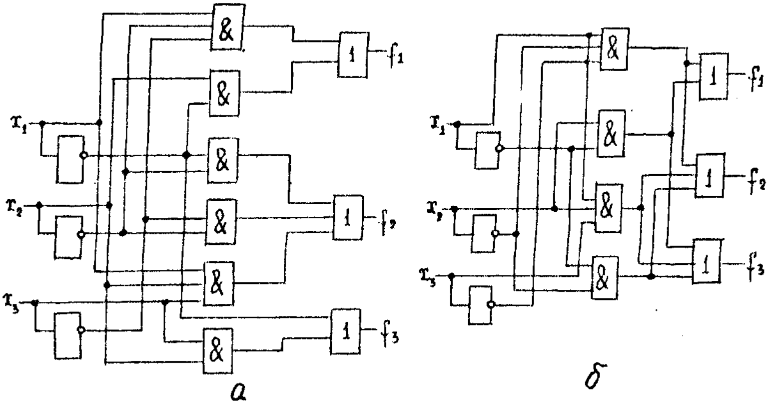


Рис. 3.45

3.6.3. Точный метод Барти – Полянского минимизации систем булевых функций.

Метод является обобщением метода Квайна – Мак-Класки и основан на преобразованиях конъюнкций вместе с ярлыками.

При минимизации систем булевых функций склеиваются наборы, которые допускают склеивание и ярлыки которых имеют непустое пересечение; результату склеивания приписывается это пересечение ярлыков.

0 – I – 0 $f_1 f_3$	0 – I – 0 $f_2 f_3$
I – I – 0 $f_1 f_2$	I – I – 0 $f_1 f_4$
– – I – 0 f_1	Не склеиваются

Каждый из склеивающихся наборов отмечается только тогда, когда ярлык результата склеивания и ярлык этого набора равны.

Формулировка метода

I. Выписывается множество M_1 системы булевых функций, в это множество входят наборы, на которых хотя бы одна булева функция равна единице. Каждому набору приписывается ярлык – множество функций, равных единице на этом наборе.

2. Множество M_1 разбивается на классы M_1^i $(0 \leq i \leq n)$.

3. Выполняются все возможные склеивания наборов соседних классов M_1^i и M_1^{i+1} с учетом особенностей склеиваний наборов для систем

булевых функций. Формируется множество M_1^{iH} результатов склеиваний $(0 \le i \le n-1)$.

4. Не отмеченные наборы множества M_1 запоминаются как максимальные наборы системы булевых функций.

5. Пункты 3, 4 выполняются для наборов множества M_1^H, если оно не пусто.

6. Строится таблица покрытий, строки которой сопоставляются найденным максимальным набором, столбцы - элементам множества M_1, причем, каждый элемент повторяется столько раз, сколько булевых функций составляют его ярлык и в ярлык каждого набора входит только одна функция из ярлыка, для каждого - своя.

7. Решается задача о покрытии и строится система ДНФ. При распределении простых импликант по ДНФ функций системы учитываются ярлыки.

Минимизируем систему булевых функций /рис. 3.46/.

x_1	x_2	x_3		
0	0	0	$f_2 f_3$	V
I	0	0	$f_1 f_2$	
0	I	0	$f_1 f_3$	V
0	0	I	$f_2 f_3$	V
0	I	I	$f_1 f_3$	V
I	I	I	$f_2 f_3$	

x_1	x_2	x_3		
-	0	0	f_2	
0	-	0	f_3	V
0	0	-	$f_2 f_3$	
0	I	-	$f_1 f_3$	
0	-	I	f_3	V
-	I	I	f_3	

x_1	x_2	x_3	
0	-	-	f_3

Таблица покрытий

	000f_2	000f_3	100f_1	100f_2	010f_1	010f_3	001f_2	001f_3	011f_1	011f_3	111f_2	111f_3
I 0 0 $f_1 f_2$			I	I								
I I I $f_2 f_3$											I	I
- 0 0 f_2	I			I								
0 0 - $f_2 f_3$	I	I					I	I				
0 I - $f_1 f_3$					I	I			I	I		
- I I f_3										I		I
0 - - f_3		I				I		I		I		

Рис. 3.46

Совокупность простых импликант, выбранных в покрытие:

$$x_1\bar{x}_2\bar{x}_3 f_1 f_2, \quad x_1 x_2 x_3 f_2 f_3, \quad \bar{x}_1\bar{x}_2 f_2 f_3, \quad \bar{x}_1 x_2 f_1 f_3,$$

определяет систему ДНФ /3.5/.

Замечание. Обобщение метода Барти – Полянского для неполностью определенных булевых функций аналогично соответствующему обобщению метода Квайна – Мак-Класки.

3.6.4. Интуитивный метод упрощения системы ДНФ по матричной форме.

Точный метод Барти – Полянского достаточно громоздок для ручной минимизации булевых функций. Основываясь на матричном представлении системы булевых функций, на опыте видения максимальных интервалов, полученном при освоении метода Закревского, можно находить достаточно хорошие решения по матричной форме. Не нужно формулировать алгоритмическое построение решения, достаточно апеллировать к интуитивному выделению интервалов, сразу подходящих для нескольких функций. Во многих случаях необходимо рассмотреть несколько вариантов покрытия всех точек всех функций меньшим числом интервалов. Компактность и наглядность матричной формы позволяет выполнять это достаточно эффективно и быстро. На рис. 3.47 изображена последовательность выделения интервалов, обеспечивающая получение оптимального решения и почти очевидная.

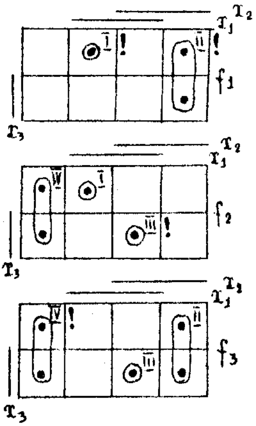


Рис. 3.47

На рис. 3.48 показан другой пример, выделенное множество интервалов которого тоже достаточно очевидно и компактно:

$$x_1\bar{x}_3\bar{x}_4f_1f_2,\ x_1x_2x_3f_1f_2,\ \bar{x}_1\bar{x}_2\bar{x}_3x_4f_1f_2,$$

$$x_3\bar{x}_4f_1,\ \bar{x}_2x_3x_4f_2;$$

$$f_1 = x_1\bar{x}_3\bar{x}_4 \vee x_1x_2x_3 \vee \bar{x}_1\bar{x}_2\bar{x}_3x_4 \vee x_3\bar{x}_4,$$

$$f_2 = \text{———————}\ \text{"}\ \text{———————} \vee \bar{x}_2x_3x_4.$$

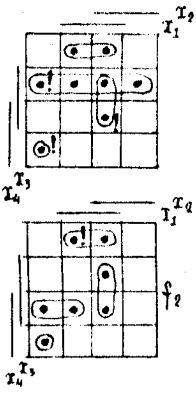


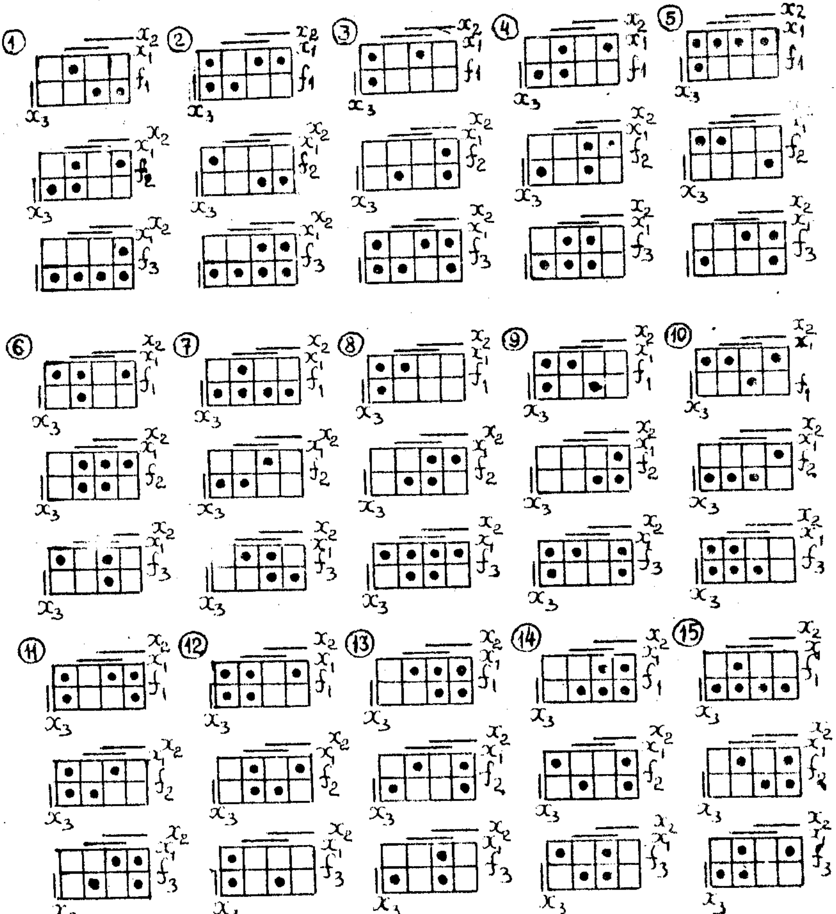
Рис. 3.48

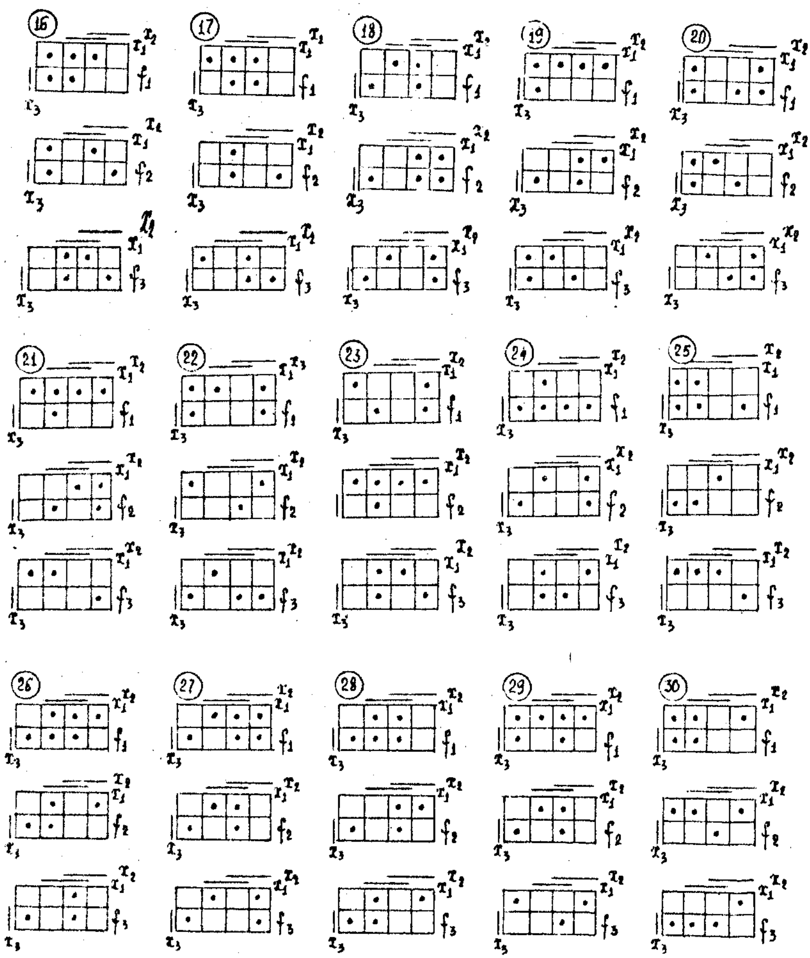
3.6.5. Задания к подразд. 3.6.

А. Минимизировать методом Барти – Полянского системы булевых функций /с.181/. Учесть специфику неполностью определенных булевых функций.

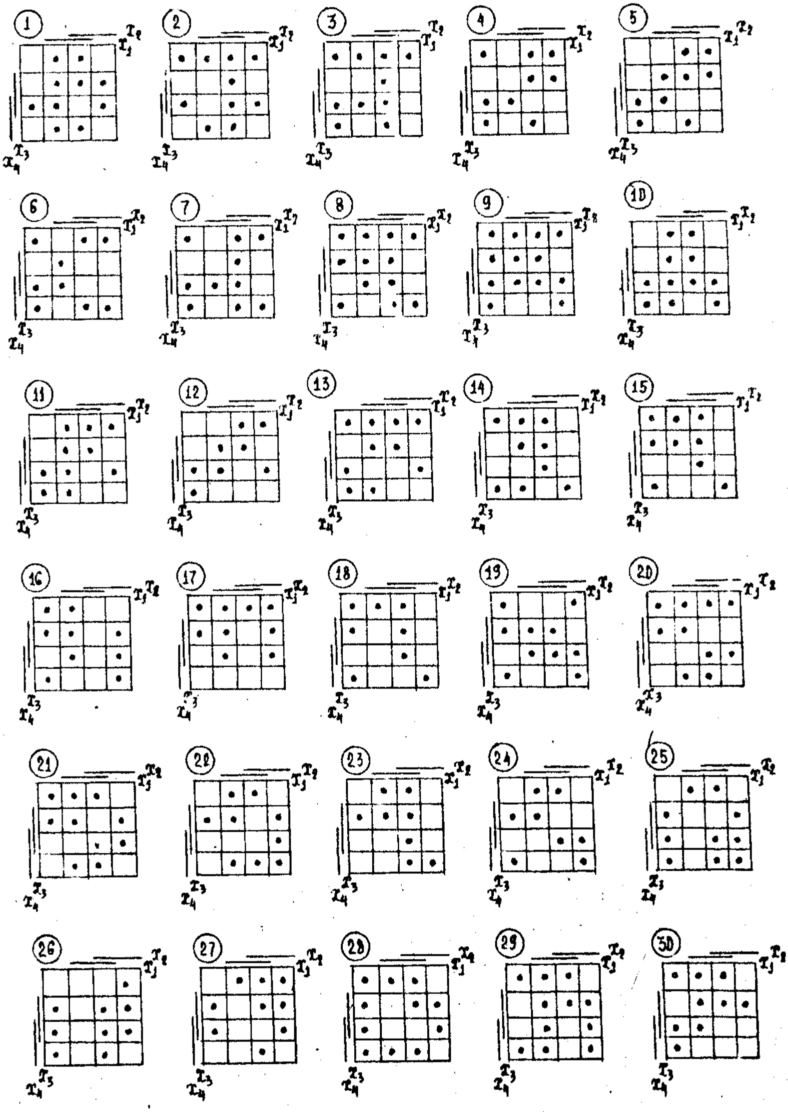
Б.Упростить систему двух булевых функций, выбирая матричные формы с номерами i и i + I /с.182/. При i = 30; i +I = I.

Варианты индивидуальных заданий. Системы трех булевых функций. Минимизация методом Барти – Полянского




16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
x_1
x_2
x_3
f_1
f_2
f_3

Система двух функций (i и i+1) четырех переменных



3.7. Интервальные формы и их преобразования

3.7.I. Интервальное представление в ЭВМ.

ЭВМ преобразуют информацию в цифровой форме, <u>векторное интервальное представление</u> систем булевых функций наиболее естественно для автоматизированного проектирования цифровых устройств. Системы булевых функций описываются списками наборов с приписанными им ярлыками. Алфавит $A_u=\{0,1,-\}$ описания наборов известен, при описании ярлыков используется алфавит $A=\{0,1,x,-\}$/$0,1,x$ – функция принимает на элементах интервала значение соответственно 0,I и x ; "-" означает, что данная функция не входит в ярлык интервала, т.е. интервал не используется для описания данной функции/. Существует несколько интерпретаций исходного задания векторного интервального представления систем булевых функций.

<u>Системы полностью определенных булевых функций.</u>

А. $A_я=\{I,-\}$. В этом случае считается, что на всех элементах булева пространства, не принадлежащих интервалам, для которых определено в задании единичное значение каждой функции, она принимает нулевое значение.

<u>Пример.</u>

x_1	x_2	x_3	f_1	f_2
1	0	–	1	1
1	–	1	1	–
0	1	–	–	1
–	0	0	1	1

Это соответствует следующему заданию булевых функций f_1 и f_2 на наборах булева пространства /рис. 3.49/.

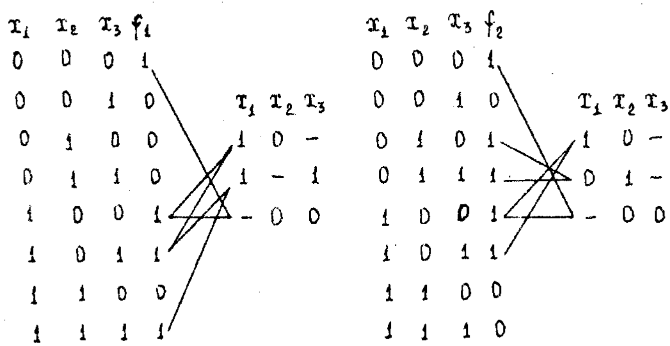


Рис. 3.49

Очевидно, что можно задать алфавит Б $A_я=\{0,-\}$, тогда на наборах, не определенных в интервальном представлении нулем /для каждой функции своих/, фиксируется единичное значение.

Системы неполностью определенных булевых функций.

Для таких систем разнообразие возможных алфавитов $A_я$ больше.

А. $A_я=\{1,x,-\}$, на наборах, для которых интервальным представлением значение функции не задано, фиксируется нулевое значение функций.

Б. $A_я=\{1,0,-\}$, на наборах, не определенных интервальным представлением, функции считываются не определенными /слабоопределенные булевы функции/.

В. $A_я=\{1,0,x,-\}$, на наборах, для которых интервальным представлением значение функций не задано, фиксируется нуль /или единица, или не определено/.

3.7.2. Основные операции над векторным интервальным представлением.

Операции формируются для векторного интервального представления и иллюстрируются на матричной форме. В подразд. 3.4 определены операции склеивания, поглощения для наборов, однако имеется ряд специфических операций, прежде всего, связанных с интерпретацией интервалов как совокупностей элементов булева пространства. Объединение интервалов /ОИ/. Эта операция заключается в совместном рассмотрении интервалов, составлении списка интервалов. Например ОИ, представляющих собой булеву функцию. Объединение интервалов рассматривается /интерпретируется/ как множество наборов, содержащихся /покрытых/ в этой совокупности интервалов.

Исследование ортогональности интервалов. Интервалы ортогональны, если они не имеют общих элементов. Наборы ортогональных интервалов расположены в разных частях булева пространства, т.е. должна существовать хотя бы одна внешняя переменная этих интервалов, общая для них обоих, принимающая значение нуль на элементах одного и единицу - на элементах другого /см. рис. 3.50/:

$$\underline{\underline{0}} - \underline{\underline{1}} - 1 \qquad\qquad \underline{\underline{0}} - 1 - 1$$

$$\underline{\underline{1}} - \underline{\underline{0}}\ 0\ 1 \qquad\qquad \underline{\underline{1}}\ 1 - - 1$$

Если интервалы не имеют общих внешних переменных или если каждая из внешних переменных принимает одинаковые значения для обоих интервалов, то эти интервалы пересекаются и не являются ортогональными:

1 - 0 0 1   1 - 0 0 1

1 1 - - 1   1 1 0 0 -

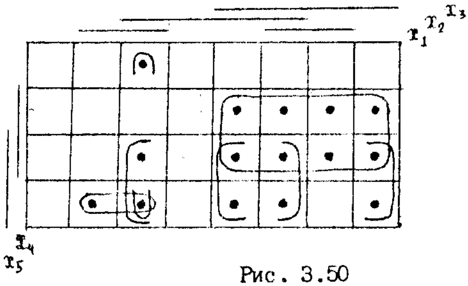


Рис. 3.50

Пересечение интервалов. Если интервалы не ортогональны, то их общие элементы обязательно образуют интервал. Выделение этого интервала составляет операцию пересечения /II/. В векторном представлении происходит объединение множества внешних переменных, пересекающихся не ортогональных интервалов /рис.3.50/:

$$\begin{array}{c} 1\;1\;0\;0\;- \\ 1\;1\;-\;-\;1 \\ \hline 1\;1\;0\;0\;1 \end{array} \qquad \begin{array}{c} 0\;-\;1\;-\;1 \\ -\;-\;1\;1\;- \\ \hline 0\;-\;1\;1\;1 \end{array}$$

Симметрирование интервала. В алгоритме построения максимальных интервалов для заданного элемента матричной формы /пп. 3.5.I/ использована операция симметрирования текущего интервала по оси одной из внешних переменных. В векторном представлении симметрирование интервала по внешней переменной сводится к инвертированию значения этой переменной: 0I -- по $x_2 \to$ 00--, 0-I0 по $x_4 \to$ 0-II.

Склеивание и поглощение интервалов. Обе эти операции уже были введены в пп. 3.2.3. Проиллюстрируем их.

$$\text{Склеивание} \quad \begin{array}{c} 0\;-\;1\;0\;- \\ 0\;-\;1\;1\;- \\ \hline 0\;-\;1\;-\;- \end{array} \; ; \qquad \text{поглощение} \quad \begin{array}{c} 0\;-\;1\;0\;- \\ 0\;1\;1\;0\;- \\ \hline 0\;-\;1\;0\;- \end{array}$$

Поглощение интервала объединением интервалов. Если все наборы интервала покрываются интервалами заданного с ОИ, то интервал поглощается ОИ. На матричной форме это очевидно /рис. 3.5I/. В векторном интервальном представлении это не так очевидно и может быть получено за счет разложения интервала по внутренним переменным с проверкой поглощения полученных наборов интервалами ОИ.

Исходный интервал I-I-;
его разложение ОИ:

1010	10-0
1011	-110
1110	11-1
1111	-011

Иллюстрация на матричной форме

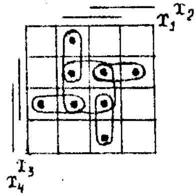


Рис. 3.5I

Показано поглощение наборов интервалами ОИ, таким образом, исходный интервал поглощается заданным ОИ. Операция разложения при большом числе внутренних переменных может привести к громоздкому множеству наборов, поэтому на ЭВМ эту операцию удобнее выполнять за счет вычитания кубов / пп. 3.7.3/.

Расширение интервала. Если при симметрировании по некоторой внешней переменной интервала результат поглощается другими интервалами ОИ, то на заданном ОИ исходный интервал может быть расширен, переводом рассматриваемой внешней переменной во внутренние. Правомерность расширения интервала в заданном ОИ проверяется следующим алгоритмом:

исходный интервал симметрируется по выбранной внешней переменной;

проверяется, поглощается ли полученный симметрированный интервал совокупностью других интервалов ОИ;

если да, то интервал расширяется, если нет - расширения не происходит.

Пример расширения интервала IOI - по переменной x_2 /рис. 3.52/

Исходное ОИ

I 0 I -
I I - 0
- I - I

Симметрированный интервал IOI- по переменной x_2:
III-
Его разложение
III0 II-0
IIII -I-I

Результат расширения
I-I-
II-0
-I-I

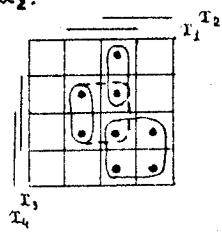
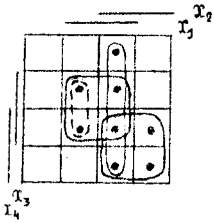


Рис. 3.52

Очевидно, интервал II-0 тоже допускает расширение по переменной x_4.

Сокращение интервала. Если в заданном ОИ при фиксировании значения /0 или I/ выбранной внутренней переменной для некоторого исходного интервала совокупность наборов, покрытых интервалами ОИ не изменяется, то такое фиксирование допустимо и называется операцией сокращения интервала. Пример.

Исходное ОИ I-I-
II--
-I-I



Исследуемый интервал I-I-
выбранная переменная x_2,
выбранное значение 0,
сокращаемый интервал III-

его разложение

1 1 1 0 ⟶ 1 1 - -
1 1 1 1

Рис.3.53

Результирующее ОИ
IOI-
II--
-I-I

На рис. 3.53 проиллюстрирован алгоритм сокращения. При заданных интервале, внешней переменной и ее значении алгоритм прост:

в исходном интервале фиксируется значение выбранной переменной, инверсное заданному;

если полученный интервал поглощается совокупностью других интервалов ОИ, то сокращение возможно, иначе не допустимо.

Вычитание интервалов. Операция заключается в представлении покрытия наборов уменьшаемого интервала, не покрытых вычитаемым интервалом. Очевидно, вычитание ортогонального интервала не приводит к изменению уменьшаемого, вычитание поглощающего интервала приводит к получению пустой разности. В общем случае разность не может быть представлена /покрыта/ одним интервалом /рис. 3.54/.

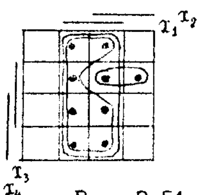


Рис. 3.54

При построении разности пересекающихся интервалов рассматривается два варианта операции вычитания:

покрытие разности объединением максимальных интервалов /обозначение операции \ /;

покрытие разности объединением ортогональных интервалов /обозначение операции # /.

Вычитание с получением покрытия максимальными интервалами.

Если интервалы не ортогональны и вычитаемый не поглощает уменьшаемый, то выполняется следующий алгоритм:

выделяется множество $X_{вн}$ внешних переменных вычитаемого интервала, не являющихся внешними уменьшаемого; пусть таких переменных будет K ;

разность представляется объединением K интервалов, каждый из которых получается сокращением уменьшаемого интервала по одной из переменных множества $X_{вн}$ со значением, инверсным ее значению в вычитаемом интервале. Пример для рис. 3.54, $X_{вн}=\{x_2, x_3, x_4\}$.

ОИ разности

$$1---\setminus-110 = \begin{matrix} 10-- \\ 1-0- \\ 1--1 \end{matrix}$$

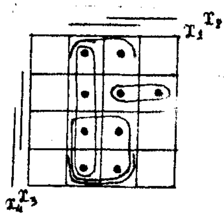


Рис. 3.55

Вычитание с получением покрытия ортогональными интервалами. Аналогично, если интервалы не ортогональны и вычитаемый не поглощает уменьшаемый, то строится множество $X_{вн}$, пусть $K=|X_{вн}|$. Разность получается как объединение K интервалов, которые строятся в такой последовательности:

1/ уменьшаемый интервал рассматривается как текущий интервал;

2/ выбирается очередная переменная из множества $X_{вн}$ и текущий интервал сокращается по ней со значением этой переменной, взятом из вычитаемого интервала; текущий интервал симметрируется по рассматриваемой переменной и таким образом получается очередной интервал результата;

3/ если еще не все переменные множества $X_{вн}$ рассмотрены, то выполняется п.2.

Пример.

ОИ разности

1---#-110 = 10--
110-
1111

Текущий интервал

11--
111-
1110

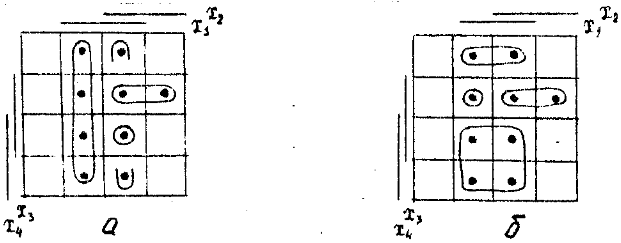


Рис. 3.56

Различные последовательности рассмотрения переменных множества $X_{вн}$ приводят к различным результатам /рис. 3.56,а и б/. Для последовательности x_2, x_3, x_4 результат проиллюстрирован на рис. 3.56,а; для последовательности x_4, x_3, x_2 – на рис.3.56,б

Построение МПИ – минимального покрывающего интервала для заданного объединения интервалов. МПИ должен содержать все наборы заданного объединения интервалов и иметь возможно меньшее число внутренних переменных. В общем случае МПИ может включать в себя наборы, не принадлежащие покрываемым интервалам /рис. 3.57,а, в/. Построение МПИ достаточно просто: необходимо взять один из интервалов совокупности и расширить его по всем тем внешним переменным, которые имеют другое значение в остальных интервалах.

Примеры:

а)
1 - 0 1
1 1 - -
1 - - -

б)
1 0 - 0
1 0 1 -
1 0 0 1
1 0 - -

в)
1 0 1 -
1 0 0 1
0 0 0 1
- 0 - -

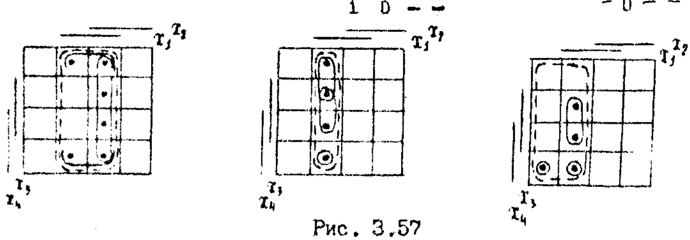


Рис. 3.57

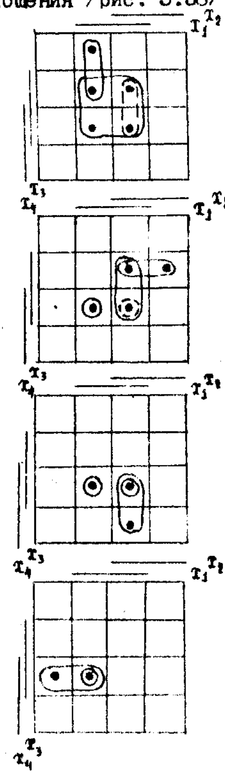
3.7.3. Использование операций при преобразованиях и исследовании интервального представления булевых функций.

Проверка покрытия интервала объединением интервалов на основе операции вычитания. Удобнее использовать операцию # . Проводится последовательное вычитание из исходного интервала элементов ОИ. Если после всех вычитаний остается пустое множество наборов, то исходный интервал покрывается объединением интервалов, иначе - не покрывается.

Снова рассмотрим функцию, изображенную на рис. 3.5I.

Исходный интервал	Заданное объединение интервалов
I- I-	I 0 - 0
	- I I 0
	I I - I
	- 0 I I

Последовательная процедура проверки поглощения /рис. 3.58/.



I - I- # IO - O = III -
IOII

III- # -IIO = IIII
IOII IOII

IIII # II-I = IOII
IOII

IOII # -OII = ∅

Рис. 3.58

Интервал I-I- поглощается заданным ОИ.

Расширение интервала в заданном ОИ до максимального. Процедура в общем случае неоднозначная, зависит от выбранной последовательности рассмотрения внешних переменных интервала:

выбирается очередная внешняя переменная расширяемого интервала;

проводится симметрирование расширенного интервала по выбранной внешней переменной и проверка поглощения симметрированного интервала заданным ОИ;

если симметрированный интервал поглощается, то в расширяемом интервале выбранная переменная переводится из внешних во внутренние, иначе не переводится. Если еще не все внешние переменные расширенного интервала рассмотрены, то преобразование выполняется сначала, иначе построенный расширенный интервал - максимальный.

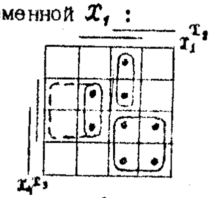
Пример /рис. 3.52/.

Исходный интервал	Исходное ОИ
IOI-	IOI-
	II-O
	-I-I

Проверка возможности расширения по переменной x_1 :

OOI- # II-O=OOI-;

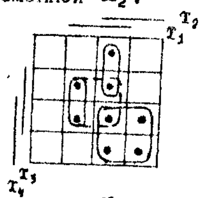
OOI- # -I-I=OOI- $\varnothing$.



Проверка возможности расширения по переменной x_2 :

III- # II-O = IIII;

IIII # -I-I = $\varnothing$.



Исходный интервал расширяется: I-I-.

Проверка возможности расширения по переменной x_3 :

I-0- # II-0 = I00-
IIOI

I00- # -I-I = I00- $\neq \phi$
IIOI

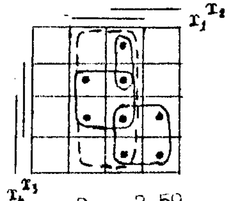


Рис. 3.59

Построен максимальный интервал I-I- /рис. 3.59/.

Рассмотрим пример, в котором проиллюстрируем неоднозначность расширения исходного интервала до максимального /рис. 3.60/.

Исходный интервал	Заданное ОИ
IO0IO	IO0IO
	I-000
	000-0
	-0I-0
	IOI-I
	I-0II

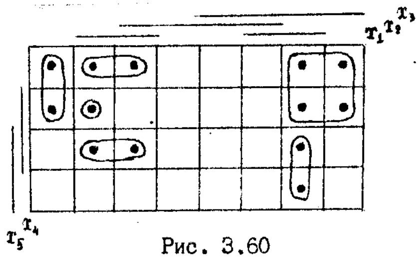


Рис. 3.60

Первая последовательность рассмотрения переменных $x_1 x_2 x_3 x_4 x_5$; вторая – $x_5 x_4 x_3 x_2 x_1$. Покажем только те преобразования, которые приводят к расширениям:

по x_1 : 000I0 # 000-0 = ϕ , расширение - 00I0;

по x_3 : -0II0 # -0I-0 = ϕ , расширение -0-I0;

по x_4 : -0-00 # I-000 = $\begin{Bmatrix} 00\text{-}00 \\ I0I00 \end{Bmatrix}$;

$\begin{Bmatrix} 00\text{-}00 \\ I0I00 \end{Bmatrix}$ # 000-0 = I0I00

I0I00 # -0I-0 = ϕ , расширение -0--0.

Заменив интервал I00I0 в ОИ на расширенный максимальный интервал, можно сократить ОИ, убрав поглощаемые интервалы /рис. 3.6I/.

Сокращенное ОИ

-	0	-	-	0
I	-	0	0	0
I	0	I	-	I
I	-	0	I	I

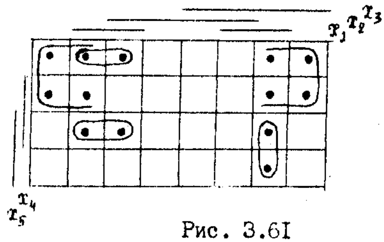


Рис. 3.6I

Для другого варианта расширения: по x_5 : I00I # I-0II = ϕ , расширение: I00I -; по x_3 : I0II- # -0I-0 = I0III,

I0III # I0I-I = ϕ , расширение I0-I-.

Замена интервала I00I0 в ОИ на расширенный максимальный интервал I0-I- к сокращению не приводит /рис. 3.62/.

I	0	-	I	-
I	-	0	0	0
0	0	0	-	0
-	0	I	-	0
I	0	I	-	I
I	-	0	I	I

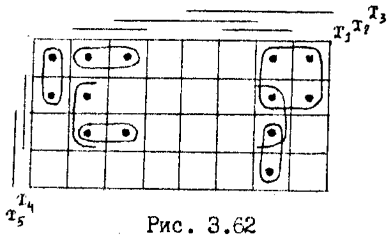


Рис. 3.62

Проверка интервала на ядерность. В п. I.I.6 введено понятие ядерной строки таблицы покрытий, которое использовано в п. 3.5.2 для определения ядерного интервала. Поиск ядерных интервалов позволяет получить более компактное ОИ для булевой функции и даже сократить сам процесс упрощения ОИ /упрощения ДНФ/. При поиске ядерного интервала по матричной форме использовано свойство: ядерный интервал должен содержать хотя бы один набор, все соседи которого находятся тоже в этом интервале. На матричной форме, пользуясь симметрией, достаточно проверить соседей точки.

Для векторного интервального представления можно несколько изменить правило проверки: должен найтись в интервале хотя бы один набор, для которого нет соседей вне данного интервала. Алгоритм проверки интервала на ядерность следующий:

выбирается очередной интервал и расширяется до максимального U /по любой последовательности переменных/, поскольку если максимальный интервал будет ядерным, то он будет единственным/. СК=И;

интервал U симметрируется по очередной внешней переменной и полученный U^c пересекается с каждым интервалом ОИ. Если пересечение не пусто, то результат пересечения $U_i^c = U^c \cap U_i$ симметрируется по рассматриваемой переменной и вычитается из СК. Такая процедура выполняется для всех $U_i \in \partial U$ и по всем внешним переменным интервала U. Если после выполнения всех преобразований СК$\neq \phi$, то интервал U – ядерный, в нем нашлись наборы, принадлежащие СИ, которые не имеют соседей вне интервала U.

Пример /рис. 3.62/ U = -0I-0.

Расширяем его до максимального: по x_2, расширение невозможно, по x_3 -00-0 # 000-0 = I00-0

I00-0 # I-000 = I00I0

I00I0 # I0-I- = ϕ расширение -0--0;

по $\tilde{x}_5$ расширение невозможно.

U = -0--0 – это максимальный интервал.

Сокращенное ОИ

I	0	-	I	-	U_1
I	-	0	0	0	U_2
-	0	-	-	0	U
I	0	I	-	I	U_3
I	-	0	I	I	U_4

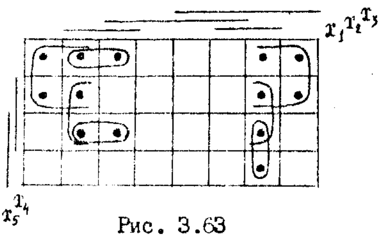


Рис. 3.63

Внешние переменные интервала U = -0--0 – это x_2 и x_5, /рис. 3.63/. СИ = -0--0

Выбираем переменную x_2: U^c = -I--0

$U^c \cap U_{2n}$ = -I--0
I-000
II000,

СИ = -0--0 # I0000 = $\begin{Bmatrix} 0\,0\,-\,-\,0 \\ I\,0\,I\,-\,0 \\ I\,0\,0\,I\,0 \end{Bmatrix}$

С остальными интервалами U_1, U_3 и U_4 пересечения пусты.

Выбираем переменную x_5: U^c = -0--I

$U^c \cap U_{1n}$ = -0--I
I0-I-
I0-II

СИ = $\begin{Bmatrix} 0\,0\,-\,-\,0 \\ I\,0\,I\,-\,0 \\ I\,0\,0\,I\,0 \end{Bmatrix}$ # I0-I0 = $\begin{Bmatrix} 0\,0\,-\,-\,0 \\ I\,0\,I\,0\,0 \end{Bmatrix}$.

$$U^c \cap U_{3n} = \begin{array}{l} -0--I \\ \underline{I0I-I} \\ I0I-0 \end{array} \qquad СИ = \left\{\begin{array}{l} 00--0 \\ I0I00 \end{array}\right\} \# \; I0I-0 = 00--0,$$

$$U^c \cap U_{4n} = \begin{array}{l} -0--I \\ \underline{I-0II} \\ I00II \end{array} \qquad СИ = 00--0 \; \# \; I00I0 = 00--0.$$

Наборы интервала 0 0 – – 0 не имеют соседей вне интервала U = = – 0 – – 0, значит, интервал U – ядерный.

Проверка избыточности интервала в ОИ. Если все наборы некоторого интервала U, входящего в ОИ, покрывается совокупностью остальных элементов ОИ, то интервал U – избыточный и может быть удален. Проверка избыточности сводится к вычитанию из U остальных интервалов и если разность оказывается пустой, то интервал U – избыточен.

Пример. Интервал U = I 0 – I – в ОИ /рис. 3.64/ – избыточен.

СИ = I 0 – I – # – 0 – – 0 = I 0 – I I.

СИ = I 0 – I I # I 0 I – I = I 0 0 I I,

СИ = I 0 0 I I # I – 0 I I = Φ , что и доказывает избыточность интервала U /рис. 3.64/.

Сокращенное ОИ

I	–	0	0	0
–	0	–	–	0
I	0	I	–	I
I	–	0	I	I

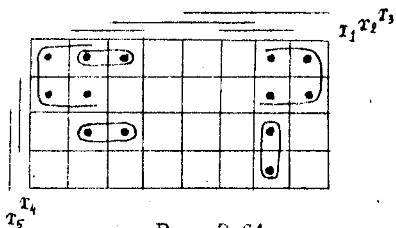


Рис. 3.64

3.7.4. Задание к подразд. 3.7.

Выполнить для одного из найденных интервалов в задании п.3.5.3.Г проверку на его избыточность, исследование на его ядерность, используя векторные операции над интервалами.

3.8. Метрические свойства ДНФ

Оценки сложности форм представления булевых функций могут быть использованы в следующих трех направлениях:

I/ оценки сложности алгоритмов преобразования форм булевых функций;

2/ оценки стоимости дискретных устройств, сложность схем которых пропорциональна сложности форм представления булевых функций;

3/ алгоритмы выполнения вычислений можно выразить через преобразования булевых функций и булев вариант алгоритма сохраняет все основные черты алгоритма решения исходной задачи. Это означает, что соответствующие построения могут служить оценкой сложности произвольных алгоритмов.

В данном подразделе приведем наиболее употребляемые оценки сложности булевых функций в ДНФ.

Совершенная ДНФ.

Максимальная сложность – $2^n \cdot n$.

Типичная сложность – $2^{n-1} \cdot n$.

Сокращенная ДНФ

Максимальная сложность – $\frac{3^n}{\sqrt{n}}$ /таким образом, если в методе Квайна – Мак-Класки строить таблицу покрытий, то она может иметь $\frac{3^n}{\sqrt{n}}$ строк и 2^{n-1} столбцов/.

Типичная сложность – $2^n n^{\log\log n}$, т.е. длина сокращенной ДНФ обычно превышает длину совершенной ДНФ.

Кратчайшая ДНФ.

Максимальная сложность – 2^{n-1} .

Типичная сложность – $2^{n-1}/\log n \log\log n$.

Отношение Y_n сложности минимальной ДНФ к кратчайшей ДНФ почти для всех функций равно единице. Поиск кратчайшей ДНФ – более простая задача /см., например, теоремы I.6–I.7 о сокращении таблицы покрытий для кратчайшего покрытия/.

Типичное отношение V_n сложности произвольной безызбыточной ДНФ и кратчайшей ДНФ оценивается следующей формулой:

$$V_n \geq \log n,$$

т.е. произвольная безызбыточная ДНФ может быть в несколько раз хуже, чем кратчайшая. Число t_n безызбыточных ДНФ для почти всех булевых функций оценивается формулой

$$t_n \simeq \left(2^{2^{n-1}}\right)^{\log n \log\log n},$$

т.е. перебор безызбыточных ДНФ при больших n – бесперспективное занятие.

Число q_n кратчайших ДНФ $q_n \geq \left(2^{2^{n-1}}\right)^{c_n\sqrt{n}}$, действительно, сокращение таблицы покрытий при поиске всех безызбыточных ДНФ не отличается от соответствующего ее сокращения при поиске всех безызбыточных ДНФ.

На основании приведенных формул можно оценить объем памяти и время реализации для алгоритмов преобразования форм булевых функций в ДНФ.

СПИСОК ЛИТЕРАТУРЫ

Раздел I

I.I. Горбатов В.А. Основы дискретной математики. - М.: Высш. шк., 1986. - 3II с.

I.2. Шихнович Ю.А. Введение в современную математику. - М. : Наука, 1965. - 376 с.

I.3. Кофман А. Введение в прикладную комбинаторику. - М.: Наука, 1975. - 480 с.

I.4. Кузнецов О.П., Адельсон-Вельский Г.М. Дискретная математика для инженеров. - М.: Энергия, 1980. - 344 с.

I.5. Поспелов Д.А. Логические методы анализа и синтеза схем. - М.: Энергия, 1974. - 368 с.

I.6. Беран А. Упорядоченные множества. - М.: Наука, 198I. - 64 с.

Раздел 2

2.I. Берж К. Теория графов и ее применения. - М.: ИЛ, 1962. - 3I9 с.

2.2. Кристофидес Н. Теория графов. Алгоритмический подход. - М.: Мир, 1978. - 432 с.

2.3. Дискретная математика и математические вопросы кибернетики.- Ч.I / Под ред. С.В.Яблонского, О.Б.Лупанова. - М.: Наука, 1974. - 3II с.

2.4. Майника Э. Алгоритмы оптимизации на сетях и графах. - М.: Мир, 198I. - 323 с.

2.5. Липский В. Комбинаторика для программистов. - М.: Мир, 1988. 2I3 с.

2.6. Лекции по теории графов / В.А.Емеличев, О.И.Мельников, В.И.Сорванов и др., - М.: Наука, 1990. - 384 с.

2.7. Свами М., Тхуласираман К. Графы, сети и алгоритмы. - М.: Мир, 1984. - 454 с.

2.8. Форд Л., Фалкерсон Д. Потоки в сетях. - М.: Мир, 1966. - 276 с.

Раздел 3

3.I. Савельев А.Я. Прикладная теория цифровых автоматов. - М.: Высш. шк., 1987. - 272 с.

3.2. Голдсуорт Б. Проектирование цифровых логических устройств. – М.: Машиностроение, 1985. – 288 с.

3.3 Миллер Р. Теория переключательных схем, Т.I. – М.: Наука, 1970. – 416 с.

3.4. Автоматизированное проектирование цифровых устройств / Под ред. С.С.Бодулина. – М.: Радио и связь, 1981. – 240 с.

3.5. Глушков В.М. Синтез цифровых автоматов. – М.: Ф.-М., 1962. – 476 с.

3.6. Яблонский С.В. Введение в дискретную математику. – М.: Наука, 1979. – 272 с.

3.7. Закревский А.Д. Логический синтез каскадных схем. – М.: Наука, 1981. – 416 с.

3.8. Киносита К., Асада К., Карацу О. Логическое проектирование СБИС. – М.: Мир, 1988. – 309 с.

3.9. Нигматуллин Р.Г. Сложность булевых функций. – М.: Наука, 1991. – 240 с.

ОГЛАВЛЕНИЕ

Навчальне видання

Новоселов Володимир Германович
Статков Олександр Володимирович

Прикладна математика
для інженерів-системотехніків.
Дискретна математика в задачах і прикладах

Навчальний посібник

Редактор Р.Г.Петрова

Коректори: М.С.Римаренко
С.М.Кушнір
Н.Ф.Слоніна

Темплан 1992, поз.555

Підп. до друку 23.09.92. Формат 60×84 1/16. Папір друк. № 3. Друк офсетний. Ум. др. арк. 6,29. Ум. фарбо-відб. 7,10
Облік.-вид. арк. 16,7. Тираж 500.
Зам. № 2-1018 Ціна 1крб 10к.

НМК ВО Міністерства освіти України
252070, Київ-70, вул. П. Сагайдачного, 37.

РОВО «Укрвузполіграф».
252151, Київ, вул. Волинська, 60.