**Л.В. Командина**

# ИССЛЕДОВАНИЕ ОПЕРАЦИЙ. ЗАДАЧИ НА ГРАФАХ И СЕТЯХ

***Методические рекомендации для студентов специальности «Прикладная математика»***



*2010*

УДК 519.8(075)
ББК 22.18я73
К63

Автор: доцент кафедры прикладной математики и механики УО «ВГУ им. П.М. Машерова», кандидат физико-математических наук **Л.В. Командина**

Р е ц е н з е н т ы:
доцент кафедры информатики и информационных технологий УО «ВГУ им. П.М. Машерова», кандидат биологических наук *А.А. Чиркина*; старший преподаватель кафедры прикладной математики и механики УО «ВГУ им. П.М. Машерова» *О.Г. Казанцева*

Научный редактор: заведующий кафедрой кафедры прикладной математики и механики УО «ВГУ им. П.М. Машерова», кандидат физико-математических наук, доцент **Л.В. Маркова**

Настоящие методические рекомендации содержат изложение и обоснование основных алгоритмов решения задач на графах и сетях: построение остовного дерева минимального веса, задачу построения кратчайшего пути, задачу о максимальном потоке, задачу о назначениях. Кроме этого в него включены параграфы с основными понятиями теории графов. Также оно содержит задачи для проведения практических и лабораторных работ.

Предназначено для студентов, обучающихся по специальности «Прикладная математика», может быть использовано студентами математических и экономических специальностей.

**УДК 519.8(075)**
**ББК 22.18я73**

## СОДЕРЖАНИЕ

Введение ..... 4
§ 1. Неориентированные графы ..... 4
§ 2. Эйлеровы и полуэйлеровы графы. Гамильтоновы графы ..... 11
§ 3. Деревья и их свойства. ..... 17
§ 4. Остовное дерево графа. Алгоритмы Прима и Краскала. ..... 19
§ 5. Ориентированные графы. ..... 24
§ 6. Задача построения кратчайшего пути из заданной вершины ..... 27
§ 7. Общая задача построения кратчайшего пути ..... 33
§ 8. Задача о максимальном потоке ..... 42
§ 9. Некоторые обобщения задачи о максимальном потоке ..... 53
§ 10. Задача о назначениях ..... 56
§ 11. Задания к лабораторным работам ..... 65
11.1. Лабораторная работа «Построение остовного дерева минимального веса» ..... 65
11.2. Лабораторная работа «Нахождение кратчайшего пути на сети» ..... 67
11.3. Лабораторная работа «Нахождение на сети потока максимальной величины» ..... 70
11.4. Лабораторная работа «Нахождение оптимального назначения» ..... 70
Литература ..... 71

# Введение

Теория графов – один из разделов современной математики. Важным стимулом ее развития являются многочисленные приложения. В частности, в терминах теории графов удобно представлять задачи по перевозке грузов, по перекачке воды или нефти по трубопроводам, задачи управления крупномасштабными проектами и т.п., что предполагает применение теории графов при исследовании задач оптимизации.

В предлагаемом учебном издании излагаются основные понятия теории графов. В четырех первых параграфах рассматриваются неориентированные графы и задача построения остовного дерева минимального веса. Затем рассматриваются ориентированные графы и задача построения кратчайшего пути (алгоритм Дейкстры и алгоритм Флойда), задача о максимальном потоке (алгортим Форда-Фалкерсона) и ее некоторые обобщения, классическая задача о назначениях и некоторые ее варианты. В предложенном перечне литературы [1–5,7] можно найти дополнительный материал по указанным задачам.

## § 1. Неориентированные графы

Чтобы составить наглядное представление о графе, достаточно вообразить некоторое множество точек плоскости или пространства и множество отрезков кривых или прямых линий, соединяющих все или некоторые из этих точек. Формально же граф $G$ определяется заданием двух множеств $I$ и $U$, обозначается $G=(I,U)$.

Первое множество $I$ есть множество элементов произвольной природы, которые принято называть ***вершинами*** (***узлами***). Чаще всего их будем обозначать буквами $i$, $j$.

Второе множество $U$ определяется следующим образом

$$U = \{\, \{i,j\},\ i,j \in I \,\},$$

т.е. каждый элемент множества $U$ определятся двумя элементами множества $I$. Элементы множества $U$ принято называть ***ребрами***. Кроме обозначения ребра как двухэлементного множества $\{i,j\}$ используют и обозначение их буквами $u_1, u_2, \ldots, u_m$. Вершины, опреде-

ляющие ребро $u$, называются ***концевыми вершинами*** этого ребра. Говорят, что ребро инцидентно своим концевым вершинам, и наоборот.

Если концевые вершины совпадают, то ребро называют ***петлей***. На графе могут существовать ребра с одинаковыми концевыми вершинами. Такие ребра называются ***параллельными***.

Граф $G$ , все элементы множества $U$ которого являются ребрами, называется ***неориентированным***. Геометрическое изображение графа в виде совокупности точек, соединенных отрезками прямых или кривых, бывает весьма наглядным. Заметим, что не всегда точки пересечения ребер являются вершинами графа. Примерами неориентированных графов могут служить схемы железных или шоссейных дорог, схемы связи поставщиков и потребителей, структурные формулы молекул и т.д. В приложениях обычно используются графы, в которых множества $I$ и $U$ состоят из конечного числа элементов. Такие графы называются ***конечными***.

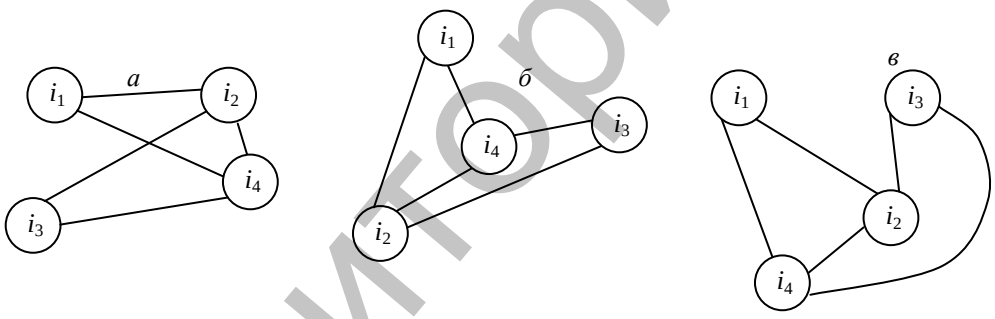


Рис. 1.1.

Поскольку при изображении графа его вершины можно располагать произвольно и по своему усмотрению выбирать форму соединяющих их линий, то один и тот же граф может предстать в различных видах. Так, на рис.1.1 по существу трижды изображен один и тот же граф, так как во всех вариантах содержится одна и та же информация. О таких графах говорят, что они ***изоморфны***. Два графа $G_1$ и $G_2$ изоморфны, если между множествами их вершин существует такое взаимно однозначное соответствие, при котором в одном из графов отрезками соединены вершины в том и только том случае, если в другом графе отрезками соединены соответствующие вершины.

Две различные вершины графа называются ***смежными***, если существует ребро, соединяющее эти вершины. Если два ребра имеют

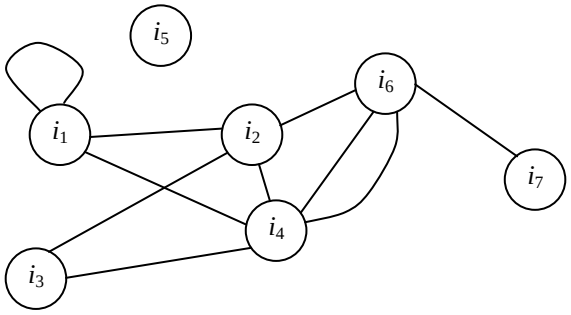


Рис. 1.2.

общую концевую вершину, они также называются ***смежными***.

Вершины в графе могут отличаться друг от друга количеством ребер, которым они инцидентны (принадлежат). ***Степенью*** $P(i_k)$ ***вершины*** $i_k$ называется число ребер графа $G$ , инцидентных данной вершине. Например, на рис. 1.1*а* вершина $i_2$ имеет степень $P(i_2) = 3$, на рис. 1.2 для вершины $i_4$ степень равна $P(i_4) = 5$, для вершины $i_5$ степень равна $P(i_5) = 0$, а на рис. 1.3 для вершины $i_6$ степень равна $P(i_6) = 1$.

Встречается другое обозначение степени вершины: $\deg i$ .

Вершина, степень которой равна нулю, называется ***изолированной***. Вершина, степень которой равна единице, называется ***висячей***, инцидентное ей ребро также называется ***висячим***.

Граф называется ***простым***, если он не содержит петель и параллельных ребер (рис. 1.1). Простой граф называется ***полным***, если в нем каждая пара вершин смежна. Граф, содержащий хотя бы два параллельных ребра, называется ***мультиграфом*** (рис. 1.2, рис. 1.3). Граф, содержащий хотя бы одно ребро вида петля, называется ***псевдографом*** (рис. 1.2) .

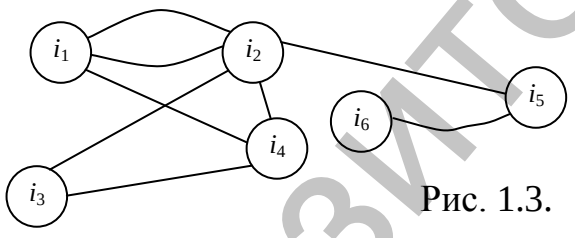


Рис. 1.3.

Каждое ребро графа соединяет две вершины, поэтому в простом графе сумма степеней всех вершин равна удвоенному числу ребер, тем самым является четным числом. Последнее утверждение известно как «лемма о рукопожатиях». Действительно, возьмем граф, все вершины которого изолированы, т.е. в графе нет ребер. Для такого графа сумма степеней всех вершин равна нулю. Прибавляя любое ребро, которое связывает две различные вершины, увеличиваем сумму всех степеней на две единицы. Значит, сумма всех степеней вершин четна. Непосредственным (и очевидным) ее следствием является следующая теорема.

*Теорема 1.1* (*Эйлера*). В любом неориентированном графе число вершин, степень каждой из которых является нечетным числом, четно.

В неориентированном графе последовательность ребер, в которой каждые два соседних ребра смежны, называется ***маршрутом*** (в некоторых учебниках – ***путем***). Задать маршрут можно либо последовательностью его вершин:

$$i_1, i_2, \ldots, i_k , \qquad (1.1)$$

либо последовательностью ребер:

$$u_1, u_2, \ldots, u_{k-1} , \qquad (1.2)$$

где $u_1 = \{i_1, i_2\}$, $u_2 = \{i_2, i_3\}$, …, $u_{k-1} = \{i_{k-1}, i_k\}$ , либо последовательностью и вершин и ребер

$$i_1, u_1, i_2, u_2, i_3, \ldots, u_{k-1}, i_k .$$

Говорят, что маршрут соединяет вершины $i_1$ и $i_k$ . Маршрут, все ребра которого различны, называется ***цепью***. Цепь, все вершины которой различны, называется ***простой цепью***.

Маршрут (1.1), у которого $i_1 = i_k$ , называется ***циклическим маршрутом.*** Если (1.1) является цепью (простой цепью), то при условии $i_1 = i_k$ имеем ***цикл*** (***простой цикл***).

На рис. 1.2 последовательность $i_3, i_2, i_6, i_4, i_2, i_6, i_7$ является маршрутом, последовательность $i_1, i_2, i_3, i_4, i_2, i_6$ является цепью, а последовательность $i_1, i_2, i_3, i_4$ является простой цепью.

Нетрудно убедиться в справедливости следующих утверждений.

*Лемма 1.1*. Если в последовательности (1.1) вершины $i_1$ и $i_k$ различны, то маршрут их соединяющий содержит простую цепь.

*Доказательство*. Если в последовательности (1.1) все вершины различны, то соответствующий маршрут не может иметь одинаковых ребер, т.е. является простой цепью. Пусть в последовательности (1.1) все вершины, кроме некоторой вершины $i_s$ , различны, а вершина $i_s$ встречается дважды:

$$i_1, \ldots, i_s, i_{s+1}, \ldots, i_{s+t}, i_s, \ldots, i_k .$$

Удалим из этой последовательности ребра $\{i_s, i_{s+1}\}$ , $\{i_{s+1}, i_{s+2}\}$ ,…, $\{i_{s+t}, i_s\}$ и вершины $i_{s+1}, \ldots, i_{s+t}$ . Тогда оставшаяся последовательность вершин и ребер является простой цепью. Рассуждения не изменяются, если вершина $i_s$ встречается более двух раз, или если таких вершин несколько.

*Лемма 1.2*. Любой циклический маршрут содержит простой цикл.

*Лемма 1.3*. Если в конечном графе степень каждой вершины не меньше двух, то в этом графе существует простой цикл.

*Доказательство*. Пусть $i_0$ – некоторая произвольная вершина графа. Так как $P(i_0) \geq 2$, то существует ребро $\{i_0, i_1\}$ . Далее для вершины $i_1$ степень $P(i_1) \geq 2$, поэтому существует ребро $\{i_1, i_2\} \neq \{i_0, i_1\}$ . Продолжим аналогичные рассуждения для последующих вершин и получим последовательность ребер, связанных с последовательностью вершин

$$i_0 , i_1 ,\ldots, i_k , \ldots .$$

При этом для любой вершины $i_k$ этой последовательности инцидентные ей ребра различны: $\{i_{k-1}, i_k\} \neq \{i_k, i_{k+1}\}$. Поскольку в графе число вершин конечное, то через некоторое число шагов какая-то вершина повторится. По определению часть маршрута между двумя повторениями этой вершины есть простой цикл. Лемма доказана.

Важным понятием является ***связность*** графа. Неориентированный граф называется ***связным***, если любые две его вершины можно соединить маршрутом (а согласно лемме 1.1 их можно соединить простой цепью), и ***несвязным*** в противном случае.

Граф $G'=(I',U')$ называется ***подграфом*** графа $G=(I,U)$, если $I' \subset I$ , $U' \subset U$ . Из определения следует, что в графе $G$ любой маршрут, любая цепь, любой цикл являются подграфом графа $G$ . ***Компонентой связности*** (или ***связной компонентой***) называется любой максимальный (по включению) связный подграф.

*Лемма 1.4*. Если в конечном связном графе удалить ребро из цикла или висячую вершину вместе с инцидентным ей висячим ребром, то новый граф также будет связным.

*Лемма 1.5*. Если в конечном связном графе удалить ребро, не принадлежащее циклу, то новый граф имеет ровно две компоненты связности.

Рассмотрим способы задания графа. Один из них – задание графа рисунком, что является удобным и наглядным при относительно небольшом числе вершин и ребер. В противном случае наглядность теряется. Другим способом задания графа является матричный способ.

Пусть $G$ – простой граф, в котором $|I| = n$ , $|U| = m$ . Графу $G$

ставят в соответствие либо матрицу смежности, либо матрицу инцидентности.

Матрица ***смежности*** – это квадратная матрица $A$ порядка $n$ (строки и столбцы этой матрицы соответствуют вершинам графа $G$ ), элементы которой определяются следующим образом:

$$a_{ij} = \begin{cases} 1\text{ , если вершины } i \text{ и } j \text{ – смежные,} \\ 0 \quad \text{в противном случае.} \end{cases}$$

Из определения вытекает, что матрица смежности неориентированного графа – это симметрическая матрица, все элементы главной диагонали равны нулю. Кроме этого число единиц в строке (столбце) равно степени соответствующей вершины. В случае мультиграфа ненулевой элемент матрицы равен числу ребер, соединяющих вершины $i$ и $j$ . Очевидно, что любой симметрической матрице с целыми неотрицательными элементами можно поставить в соответствие мультиграф, а если ненулевые элементы матрицы равны 1, то – простой граф.

Матрицы смежности графов, представленных на рис.1.1*а* и рис.1.3, имеют вид:

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 2 & 0 & 1 & 0 & 0 \\ 2 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

Матрица ***инцидентности*** – это $n \times m$-матрица $B$ , строки этой матрицы соответствуют вершинам графа $G$ , а столбцы – ребрам. Если ребра графа обозначить через $u_k$ , то элементы матрицы определяются следующим образом:

$$b_{ik} = \begin{cases} 1\text{ , если вершина } i \text{ инцидентна ребру } u_k, \\ 0 \quad \text{в противном случае.} \end{cases}$$

Из определения вытекает, что каждый столбец матрицы инцидентности состоит из нулей и двух единиц. По матрице инцидентности можно определить степень вершины – это число единиц в соответствующей строке.

Составим матрицы инцидентности для графов, представленных на рис.1.1*а* и рис.1.3. Слева от матрицы перечислим вершины, а сверху – ребра графа. Получим

$$
\begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ i_4 \end{array}
\begin{array}{c}
\begin{array}{ccccc} \{i_1i_2\} & \{i_1i_4\} & \{i_2i_3\} & \{i_2i_4\} & \{i_3i_4\} \end{array} \\
\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}
\end{array}
\qquad
\begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \\ i_4 \\ i_5 \\ i_6 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} \{i_1i_2\} & \{i_1i_2\} & \{i_1i_4\} & \{i_2i_3\} & \{i_2i_4\} & \{i_2i_6\} & \{i_3i_4\} & \{i_5i_6\} \end{array} \\
\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}
\end{array}
$$

Из определения матриц смежности и инциденций видно, что они чаще всего относятся к так называемым слабо заполненным матрицам (среди элементов матриц большой процент нулевых элементов). С точки зрения программирования этот способ задания графов является расточительным. В микроэлектронике исследуются графы, которые имеют до 2000 вершин, средняя степень которых 6 – 8. Матрица смежности содержит 4000000 элементов, из которых только примерно 16000 равны единице, а остальные – нули [5].

Существует еще один способ задания графа – ***списки смежности***. Каждой вершине $i$ графа ставится в соответствие список $S(i)$ вершин, смежных с вершиной $i$. Например, для графа, представленного на рис.1.1*а*, списки смежности следующие:

$S(i_1) = (i_2, i_4)$, $S(i_2) = (i_1, i_3, i_4)$, $S(i_3) = (i_2, i_4)$, $S(i_4) = (i_1, i_2, i_3)$;

для графа, представленного на рис. 1.3:

$$S(i_1) = (i_2, i_2, i_4),\ S(i_2) = (i_1, i_1, i_3, i_4, i_5),\ S(i_3) = (i_2, i_4),$$
$$S(i_4) = (i_1, i_2, i_3),\ S(i_5) = (i_2, i_6),\ S(i_6) = (i_5).$$

Задания

1. Доказать теорему Эйлера.
2. Доказать лемму 1.2.
3. Доказать лемму 1.4.
4. Доказать лемму 1.5.
5. Изобразить граф по заданным спискам смежности:

$$S(1) = (3, 5),\ S(2) = (3, 4),\ S(3) = (1, 2, 4, 5),$$
$$S(4) = (2, 3, 5);\ S(5) = (1, 3, 4).$$

6. Составить матрицу смежности, матрицу инцидентности и списки смежности для графа G на рис.1.4.

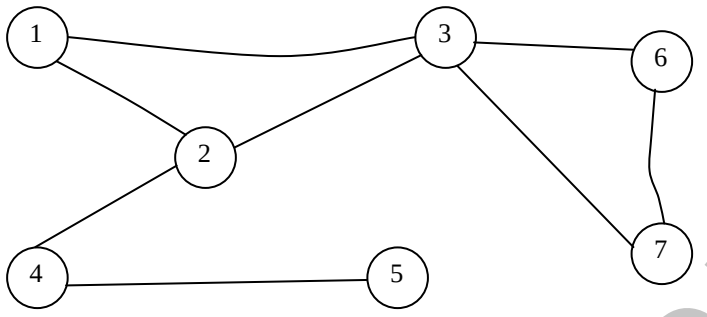


Рис. 1.4.

7. Изобразить графы по заданным матрицам смежности:

$$\text{а)}\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}, \text{б)}\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}.$$

8. Изобразить графы по заданным матрицам инцидентности:

$$\text{а)}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}, \quad \text{б)}\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

## § 2. Эйлеровы и полуэйлеровы графы. Гамильтоновы графы

Именно с задач, поставленных и решенных в этом параграфе, началась теория графов. Философ Эммануил Кант поставил задачу (1736 г.), известную в математике как *задача о семи мостах*. Части города Кенигсберга (ныне Калининграда), расположенные по бере-

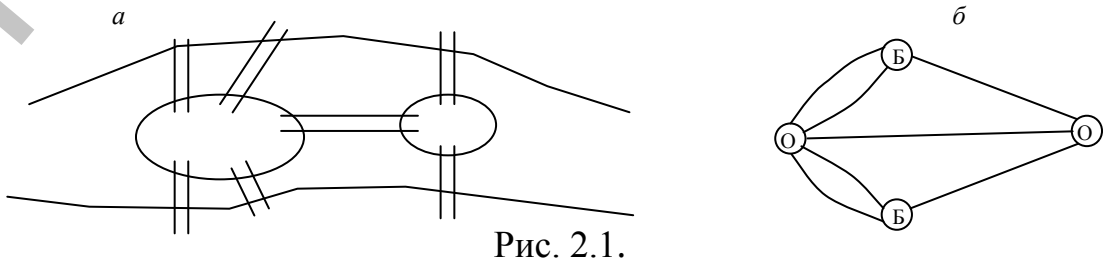


Рис. 2.1.

гам реки Прегель и на двух ее островах, соединены семью мостами. На рис. 2.1*а* изображена схема этих мостов. Задача была следующая: можно ли, выйдя из дома, вернуться назад, пройдя по всем мостам, причем по каждому мосту только один раз,. На рис. 2.1*б* изображен мультиграф, соответствующий этой схеме (каждый берег реки и острова – это вершины графа, а мосты – его ребра).

Сформулируем задачу о семи мостах на языке теории графов: существует ли в мультиграфе цикл, включающий все ребра графа. Знаменитый математик Леонард Эйлер решил эту задачу.

Введем некоторые определения. Цикл, содержащий все ребра графа, называется ***эйлеровым***. Связный граф, имеющий эйлеров цикл, также называется ***эйлеровым***. Связный граф называется ***полуэйлеровым***, если существует цепь, содержащая все ребра графа. На

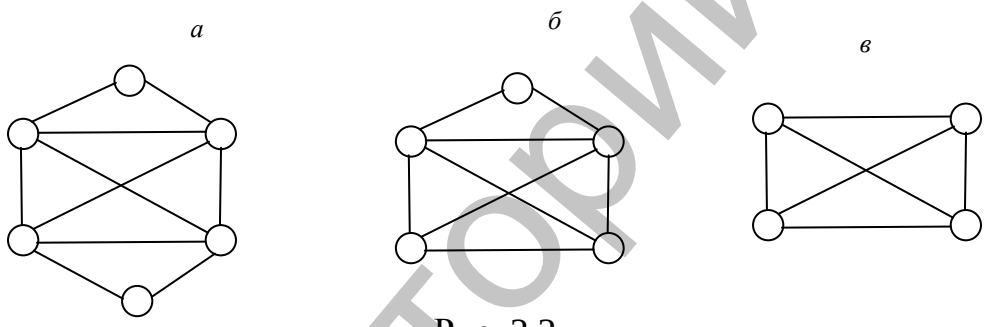


Рис. 2.2.

рис. 2.2*а* изображен эйлеров граф, на рис. 2.2*б* – полуэйлеров граф и на рис. 2.2*в* – граф, не являющийся ни эйлеровым, ни полуэйлеровым.

*Теорема 2.1*. Для того чтобы данный связный граф был эйлеровым, необходимо и достаточно, чтобы степени всех вершин были четными.

*Доказательство*. *Необходимость.* Пусть граф является эйлеровым. Тогда в нем имеется эйлеров цикл, двигаясь по которому, в любую вершину входим по одному ребру и покидаем ее по другому, так как каждое ребро должно использоваться только один раз (т.е. каждый “заход” в вершину и “выход” из нее дает 2 степени вершины). Таким образом, сумма степеней всех вершин должна быть четной (равна удвоенному числу “заходов” в эту вершину при обходе эйлерова цикла).

*Достаточность*. Пусть в данном связном графе степень любой вершины четна (т.е. степень больше или равна 2, так как нулевая сте-

пень приводит к несвязному графу). Докажем, что в нем имеется эйлеров цикл. Доказательство проведем индукцией по числу вершин. Связный граф, в котором только две вершины, и обе они имеют четную степень, является мультиграфом. Примеры такого графа представлены на рис. 2.3. Очевидно, что в этом случае в графе имеется эйлеров цикл.

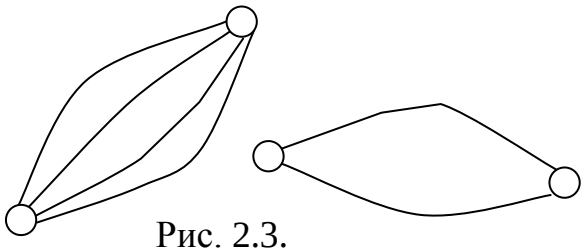

Рис. 2.3.

Предположим, что наше утверждение верно для всех связных графов, число вершин в которых строго меньше $n$, и докажем его для графа, имеющего $n$ вершин. Так как степень всех вершин больше или равна 2, то в графе существует простой цикл (лемма 1.3). Если этот цикл содержит все ребра, то он и является эйлеровым циклом, а граф эйлеровым. Если этот простой цикл содержит не все ребра, то из графа удалим все ребра этого цикла и те вершины, которые после удаления ребер стали иметь нулевую степень. Тогда получим новый граф, который может быть несвязным, но в этом новом графе все вершины обязательно имеют *четную* степень, так как при удалении ребер цикла степень каждой вершины, входящей в этот цикл, уменьшается на 2. Новый граф распадается на компоненты связности, причем каждая из них имеет общую вершину с удаленным циклом (в противном случае исходный граф не был бы связным). Степени всех вершин каждой компоненты связности четны и число вершин в каждой из них строго меньше $n$, тогда по индукционному предположению каждая компонента имеет эйлеров цикл. Теперь можем построить эйлеров цикл в исходном графе. Начнем движение с произвольной вершины удаленного цикла. Если мы пришли в вершину, общую для цикла и какой-либо компоненты связности, то обходим эту компоненту по ее эйлерову циклу, возвращаемся при этом в вершину удаленного цикла, идем по этому циклу дальше. Тем самым все ребра исходного графа будут пройдены и каждое только один раз.

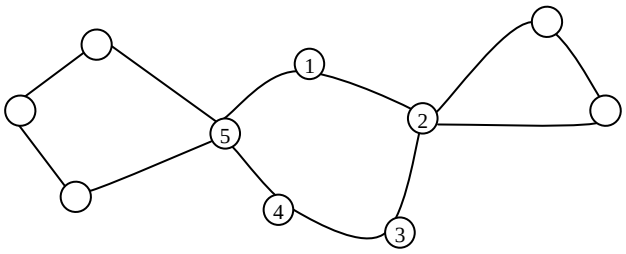


Рис. 2.4.

Все это схематично изображено на рис. 2.4: начинаем обходить цикл 1, 2, 3, 4, 5. Пройдя ребро {1,2}, проходим «правую» компоненту и возвращаемся в вершину 2, далее идем по ребрам {2,3}, {3,4}, {4,5}, обходим «левую» компоненту и т.д.

Теорема доказана.

*Теорема 2.2*. Связный граф будет полуэйлеровым тогда и только тогда, когда степени двух вершин будут нечетными, а степени остальных вершин – четными.

*Доказательство*. *Необходимость*. Пусть имеем полуэйлеров граф. Это значит, что он имеет эйлерову цепь, начинающуюся в одной вершине и заканчивающуюся в другой. Видно, что обе эти вершины должны иметь нечетную степень, а степень остальных четная.

*Достаточность*. Пусть в связном графе вершины $k$ и $p$ имеют нечетную степень, а остальные вершины – четную. Тогда возможны 2 случая: 1) вершины $k$ и $p$ связаны ребром, 2) вершины $k$ и $p$ не связаны ребром.

В случае 1) удалим ребро $\{k, p\}$. В новом графе степень всех вершин станет четной. На рис. 2.5 *а*, *б* приведены примеры таких графов.

Если новый граф – связный граф (рис. 2.5*а*), то согласно теореме 2.1 этот граф содержит эйлеров цикл. Начнем обход с вершины $k$

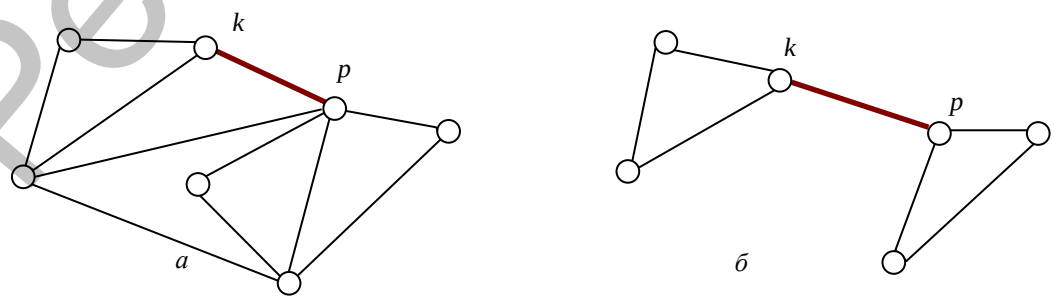


Рис. 2.5.

(или с вершины $p$) и закончим его в этой же вершине, далее добавим ребро $\{k, p\}$. Получим эйлеров путь, который начался в одной из вершин с нечетной степенью и закончился в другой.

Если новый граф – несвязный граф (рис. 2.5*б*), то он имеет две компоненты связности, между которыми удаляемое ребро было мостом (вершины $k$ и $p$ находятся в разных компонентах). В каждой компоненте степень всех вершин четная, поэтому в каждой имеется эйлеров цикл. Начнем обход, например, с вершины $k$, пройдем одну компоненту по ее эйлерову циклу, закончив обход в вершине $k$, затем добавим удаленное ребро $\{k, p\}$ (мост), пройдем его, попадем в вершину $p$, и пройдем вторую компоненту связности по ее эйлерову циклу. Снова получим эйлеров путь, который начался в одной из вершин с нечетной степенью и закончился в другой.

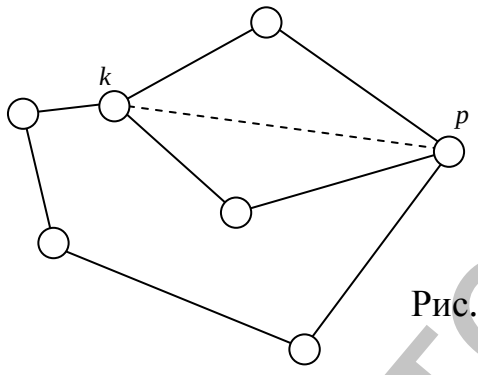

Рис. 2.6.

Рассмотрим случай, когда вершины $k$ и $p$ не связаны ребром (рис. 2.6). Добавим к графу ребро $\{k, p\}$. В новом графе степень всех вершин станет четной.

Очевидно, что новый граф – связный граф, тогда согласно теореме 2.1 этот граф содержит эйлеров цикл. Начнем обход с вершины $k$ (или с вершины $p$), начав с ребра $\{k, p\}$, и закончим его в этой же вершине, далее удалим ребро $\{k, p\}$. Получим эйлеров путь, который начался в одной из вершин с нечетной степенью и закончился в другой.

Теорема доказана.

Заметим, что три из четырех вершины мультиграфа (рис. 2.1*б*), соответствующего мостам Кенигсберга, имеют степень 3, а четвертая имеет степень, равную 5. Поэтому эйлеров цикл или путь невозможны.

Заметим, что если граф содержит $2k$ вершин нечетной степени, то его можно разбить на $k$ полуэйлеровых графов. Доказательство аналогично доказательству теоремы Эйлера.

Имеется простой алгоритм (так называемый *алгоритм Флери*) для нахождения эйлерова цикла (конечно, если этот цикл существует), который состоит в следующем: *начинаем с любой вершины и* "*стираем*" *пройденные ребра. При этом по мосту (перешейку) проходим только, если нет других возможностей.*

Очевидно, что для того, чтобы построить эйлеров путь, достаточно использовать алгоритм Флери, который надо начать с вершины, имеющей нечетную степень.

Рассмотрим некоторые приложения эйлеровых графов.

Пусть имеется некоторый связный граф, ребрам которого приписаны некоторые числа, называемые ***весами*** ребер (например, вес ребра – это его длина). Требуется найти такой цикл, в который каждое ребро входит, *по крайней мере, один раз* и суммарный вес всех ребер, вошедших в цикл, минимален. Заметим, что если граф является эйлеровым, то любой эйлеров цикл решает поставленную задачу (для эйлерова графа веса роли не играют).

Эта задача имеет много приложений, например, поливка улиц одной машиной (здесь ребра графа – дороги, а перекрестки – вершины; веса – это длины дорог), а также сбор мусора, доставка почты или даже наилучший маршрут для осмотра музея или уборка помещений и коридоров в больших учреждениях.

Кратко рассмотрим проблему, связанную с возможным обходом всех вершин в графе: существует ли в данном графе маршрут, проходящий через каждую вершину только один раз.

Если в графе существует цикл, содержащий каждую вершину графа ровно один раз, то граф называется ***гамильтоновым***. Очевидно, что указанный цикл является простым и называется также ***гамильтоновым.***

Несмотря на сходство постановки задач для гамильтоновых графов с эйлеровыми графами, «хорошего» решения для гамильтоновых графов нет. Вообще, о гамильтоновых графах известно очень мало. Очевидно, что если граф гамильтонов, то он связный (необходимое условие). Из достаточных условий отметим следующие утверждения.

*Теорема Оре*. Если в графе $G$ , имеющем $n$ вершин, для любой пары вершин $i$ и $j$ выполняется условие

$$\deg i + \deg j \geq n,$$

то $G$ – гамильтонов граф.

*Теорема Дирака*. Если в графе $G$ , имеющем $n$ вершин, степень каждой вершины не меньше $n/2$ , то $G$ – гамильтонов граф.

## § 3. Деревья и их свойства

Граф, не имеющий циклов, называется ***лесом*** (рис. 3.1). Связный граф, не имеющий циклов, называется ***деревом***.

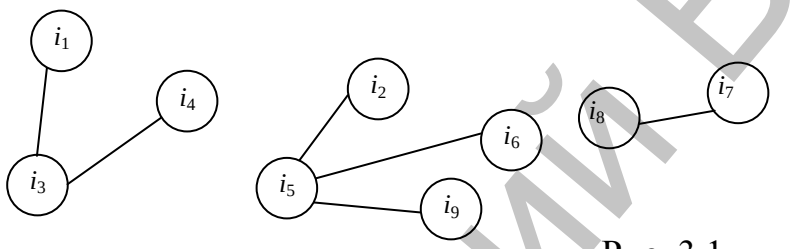


Рис. 3.1.

В случае конечных графов лес состоит из конечного числа деревьев. При этом каждое дерево является компонентой связности леса.

*Теорема 3.1*. Пусть $G=(I,U)$ , где $|I| = n$ , $|U| = m$ . Тогда следующие утверждения эквивалентны:

1) $G$ – дерево.
2) $G$ не содержит циклов и $m = n - 1$.
3) $G$ – связный граф и $m = n - 1$.
4) Любые две вершины соединяет единственная простая цепь.
5) $G$ не содержит циклов, и если соединить ребром любую пару несмежных вершин, то новый граф содержит ровно один цикл.

*Доказательство*. Схема: $1) \Leftrightarrow 2)$; $1) \Rightarrow 5) \Rightarrow 4) \Rightarrow 1)$; $1) \Leftrightarrow 3)$.

$1) \Rightarrow 2)$. Имеем связный граф без циклов. Согласно лемме 1.3 в нем существует вершина, степень которой равна 1, т.е. висячая вершина. Ей инцидентно висячее ребро. Удалим и вершину, и ребро из графа. Новый граф также связный и не содержит циклов (лемма 1.4). Повторим процедуру удаления. Через $n - 2$ шага останутся две вершины и одно ребро им инцидентное. Таким образом, в исходном графе число ребер на единицу меньше числа вершин, т.е. $m = n - 1$.

2) $\Rightarrow$ 1). Имеем граф $G$ без циклов, в котором $m = n - 1$. Предположим, что он несвязный. Тогда в нем можно выделить компоненты связности. Пусть для определенности их две. Каждая из них по определению является деревом, а значит, в каждой из них число ребер на единицу меньше числа вершин. Тогда в графе $G$ число вершин $n$ больше числа ребер $m$ на 2. Противоречие.

1) $\Rightarrow$ 5). Имеем связный граф $G$ без циклов. Пусть $i$ , $j$ – две несмежные вершины графа $G$ . В силу связности в $G$ существует простая цепь, соединяющая эти две вершины. Добавим ребро $\{i, j\}$ , тем самым образуется цикл, который обозначим через С. Докажем, что он

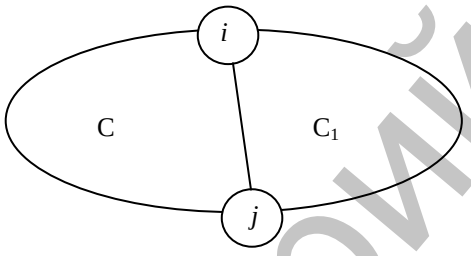


Рис. 3.2.

единственный методом от противного. Пусть кроме цикла С образовался еще один цикл $C_1$ (рис. 3.2), понятно, что он одержит ребро $\{i, j\}$ , т.к. в исходном графе циклов не было. Очевидно, что тогда образуется третий цикл: вершина $i$, затем обход по циклу С до вершины $j$ , затем обход по циклу $C_1$ до вершины $i$ . Заметим, что этот цикл не содержит ребра $\{i, j\}$ , а значит, существует в исходном графе $G$. Противоречие.

5) $\Rightarrow$ 4). Имеем граф $G$ без циклов. Пусть $i$ , $j$ – две несмежные вершины графа $G$ . Добавим ребро и тем самым образуем единственный цикл, который начинается в вершине $j$ и заканчивается ребром $\{i, j\}$ . Но это означает, что в графе $G$ существует единственная простая цепь, соединяющая эти две вершины.

4) $\Rightarrow$ 1). Имеем граф $G$ , в котором любые две вершины соединяет единственная простая цепь. Значит, граф $G$ – связный и не содержит циклов (ибо в противном случае для несмежных вершин существовали бы, по крайней мере, две цепи, их соединяющих). Другими словами, $G$ – дерево.

3) $\Rightarrow$ 1). Пусть $G$ – связный граф и $m = n - 1$. Предположим, что в $G$ существует цикл. В любом цикле число вершин равно числу

ребер. Поэтому в силу второго условия в графе имеются вершины и ребра, не входящие в рассматриваемый цикл. Найдем все висячие вершины и инцидентные им висячие ребра, удалим их. Для нового графа $G_1$ имеем соотношение: $m_1 = n_1 - 1$, где $m_1$ – число ребер, $n_1$ – число вершин в графе $G_1$. Если в графе имеется еще один цикл, то удалим из него ребро, не принадлежащее рассматриваемому графу. Получим граф $G_2$, для которого $m_2 = m_1 - 1$, $n_2 = n_1$. Продолжим этот процесс, через конечное число шагов получим граф, состоящий из рассматриваемого цикла, для которого $m_k = n_k$. Таким образом, в исходном графе число ребер не меньше числа вершин: $m \geq n$. Противоречие. Итак, $G$ – связный граф без циклов, т.е. дерево.

1) $\Rightarrow$ 3). (Самостоятельно).

Теорема доказана.

*Теорема 3.2*. В любом дереве $G=(I,U)$, для которого $|U| \geq 2$, не менее двух висячих вершин (рис. 3.3).

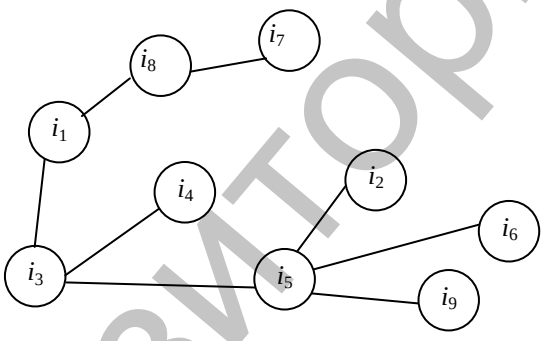


Рис. 3.3.

Задания

1. Доказать, что в теореме 3.1 что из утверждения 1) следует утверждение 3).
2. Доказать теорему 3.2.

## § 4. Остовное дерево графа. Алгоритмы Прима и Краскала

Пусть $G=(I,U)$ – связный граф. Любой связный подграф $G'$, содержащий все вершины графа $G$ и не имеющий циклов, называется ***остовным*** деревом или деревом графа $G$. Из определения следует, что любой связный граф имеет остовное дерево.

Опишем процесс построения остовного дерева для заданного

графа, основанный на лемме 1.4. Если заданный связный граф не является деревом, то в нем можно выделить цикл. Удалим любое ребро из цикла. При этом получим новый граф, который является связным, и число вершин которого равно числу вершин заданного графа. Если в новом графе нет циклов, то он является остовным деревом заданного графа. В противном случае повторим операцию удаления ребра из цикла. Понятно, что остовное дерево для заданного графа, вообще говоря, определяется неоднозначно.

Пусть в графе $G$ каждому ребру $\{i, j\}$ поставлено в соответствие некоторое неотрицательное число $\ell_{ij}$, которое назовем ***весом*** ребра. В этом случае граф называется ***взвешенным***. Если некоторые две вершины соединены маршрутом, то ***весом маршрута*** называется сумма весов всех ребер маршрута. Если в графе построено некоторое остовное дерево, то ***весом дерева*** называется сумма весов всех его ребер.

***Задача***. В заданном взвешенном графе $G=(I,U)$, $|I| = n$, среди всех его остовных деревьев найти остовное дерево минимального веса.

Рассмотрим два алгоритма, решающих эту задачу.

Алгоритм Прима.

***Шаг* 0**. Строим подграф $G_0 = (I,U_0)$, где $U_0 = \varnothing$. (Граф $G_0$ несвязный, все вершины изолированные).

***Шаг* 1**. Находим ребро с минимальным весом $\ell_{i_1 j_1}$. Строим подграф $G_1 = (I, U_1)$, где $U_1 = U_0 \cup \{i_1, j_1\}$.

***Шаг* k**. Находим ребро$\{i_k, j_k\}$, удовлетворяющее трем условиям:

1) ребро $\{i_k, j_k\}$ является смежным некоторому ребру графа $G_{k-1}$,
2) ребро $\{i_k, j_k\}$ не образует цикла при добавлении к графу $G_{k-1}$,
3) ребро $\{i_k, j_k\}$ имеет минимальный вес среди тех ребер, которые удовлетворяют первым двум условиям.

Строим подграф $G_k = (I, U_k)$, где $U_k = U_{k-1} \cup \{i_k, j_k\}$.

Если $k = n - 1$, то построено минимальное остовное дерево $G_{n-1}$.

Следующая теорема является обоснованием алгоритма Прима.

*Теорема 4.1*. Пусть $G=(I,U)$ – связный граф, у которого $|I| = n$, и подграф $G_{n-1}$ построен по алгоритму Прима. Тогда $G_{n-1}$ – остовное дерево минимального веса.

*Доказательство*. Прежде всего, покажем, что подграф $G_{n-1}$ может быть построен по правилам алгоритма. Для этого достаточно показать выполнимость любого шага *k*. Для каждого подграфа $G_{k-1}$ число вершин равно *n*, число ребер равно $|U_{k-1}| = k - 1 < n - 1$, следовательно, в подграфе $G_{k-1}$ имеется изолированная вершина. Исходный граф $G$ – связный, поэтому всегда найдется хотя бы одно ребро $\{i_k, j_k\} \in U \setminus U_{k-1}$, которое в подграфе $G_{k-1}$ связывает изолированную вершину с неизолированной вершиной.

Далее, по построению подграф $G_{n-1}$ – связный, и для него $|U_{n-1}| = n - 1$, т.е. выполняются условия теоремы 3.2 (п.3). Следовательно, $G_{n-1}$ – дерево, причем остовное.

Покажем, что $G_{n-1}$ имеет минимальный вес. В конечном графе существует конечное число остовных деревьев. Пусть $\bar{G} = (I, \bar{U})$ – минимальное по весу остовное дерево графа $G$. Обозначим через $\ell(G_{n-1})$ и $\ell(\bar{G})$ вес построенного остовного дерева и вес минимального. Тогда по предположению

$$\ell(G_{n-1}) \geq \ell(\bar{G}) . \qquad (4.1)$$

Пусть $\{i_1, j_1\}$, $\{i_2, j_2\}$,…, $\{i_{n-1}, j_{n-1}\}$ – последовательность ребер, которая образовала дерево $G_{n-1}$ (по шагам алгоритма), и пусть ребра, $\{i_1, j_1\}$, $\{i_2, j_2\}$,…,$\{i_s, j_s\}$, $s < n - 1$, общие и для дерева $G_{n-1}$ и для дерева $\bar{G}$.

Рассмотрим ребро $\{i_{s+1}, j_{s+1}\} \in U_{n-1}$ и $\{i_{s+1}, j_{s+1}\} \notin \bar{U}$. Если это ребро добавить к минимальному дереву $\bar{G}$, то получим новый граф, который содержит ровно один цикл (теорема 3.2 (п. 5)). Заметим, во-первых, что среди ребер цикла найдется ребро $\{i', j'\} \notin U_{n-1}$, ибо в противном случае в графе $G_{n-1}$ был бы цикл. Во-вторых, ребро $\{i_{s+1}, j_{s+1}\}$ по построению является смежным некоторому ребру подграфа $G_s$, поэтому вес ребра $\{i_{s+1}, j_{s+1}\}$ не больше веса ребра $\{i', j'\}$ согласно требованию 3) шага *k*. Заменим ребро $\{i', j'\}$ в дереве $\bar{G}$ на ребро $\{i_{s+1}, j_{s+1}\}$. Получим новое остовное дерево $\bar{G}' = (I, \bar{U}')$, для которого

$$\ell(\bar{G}') \leq \ell(\bar{G}). \qquad (4.2)$$

Далее рассмотрим ребро $\{i_{s+2}, j_{s+2}\} \in U_{n-1}$ и $\{i_{s+2}, j_{s+2}\} \notin \bar{U}'$. Рассуждая аналогично, построим остовное дерево $\bar{G}'' = (I, \bar{U}'')$ такое, что

$$\ell(\bar{G}'') \leq \ell(\bar{G}'). \qquad (4.3)$$

Продолжим аналогичные построения, и через конечное число шагов получим остовное дерево, состоящее из ребер $\{i_1, j_1\}$, $\{i_2, j_2\}$,…, $\{i_{n-1}, j_{n-1}\}$, т.е. дерево $G_{n-1}$. При этом из (4.2), (4.3) и аналогичных им неравенств следует, что

$$\ell(G_{n-1}) \le \ell(\bar{G}). \qquad (4.4)$$

Тогда из (4.1) и (4.4) получаем $\ell(G_{n-1}) = \ell(\bar{G})$, т.е. $G_{n-1}$ имеет минимальный вес. Теорема доказана.

Алгоритм Краскала.

***Шаг* 0**. Строим подграф $G_0 = (I, U_0)$, где $U_0 = \varnothing$. (Граф $G_0$ несвязный, все вершины изолированные).

***Шаг* 1**. Находим ребро $\{i_1, j_1\}$ с минимальным весом $\ell_{i_1 j_1}$. Строим подграф $G_1 = (I, U_1)$, где $U_1 = U_0 \cup \{i_1, j_1\}$.

***Шаг* k**. Находим ребро $\{i_k, j_k\}$, удовлетворяющее двум условиям:

1) ребро $\{i_k, j_k\}$ не образует цикла при добавлении к графу $G_{k-1}$,
2) ребро $\{i_k, j_k\}$ имеет минимальный вес среди тех ребер, которые удовлетворяют первому условию.

Строим подграф $G_k = (I, U_k)$, где $U_k = U_{k-1} \cup \{i_k, j_k\}$.

Если $k = n - 1$, то построено минимальное остовное дерево $G_{n-1}$.

Обоснование алгоритма Краскала дает следующая теорема.

*Теорема 4.2*. Пусть $G = (I,U)$ – связный граф, у которого $|I| = n$, и подграф $G_{n-1}$ построен по алгоритму Краскала. Тогда $G_{n-1}$ – остовное дерево минимального веса.

*Доказательство*. Возможность построения подграфа $G_{n-1}$ по правилам алгоритма следует из первой части теоремы 4.1, так как условия алгоритма Краскала менее жесткие.

Далее, по построению подграф $G_{n-1}$ не содержит циклов, и для него $|U_{n-1}| = n - 1$, т.е. выполняются условия теоремы 3.2 (п. 2). Следовательно, подграф $G_{n-1}$ – остовное дерево.

Доказательство того, что $G_{n-1}$ – остовное дерево минимального веса, аналогично соответствующей части теоремы 4.1.

*Пример*. Рассмотрим граф, представленный на рис. 4.1. Каждому ребру графа поставлен в соответствие вес. Построить остовное дерево минимального веса согласно алгоритмам Прима и Краскала.

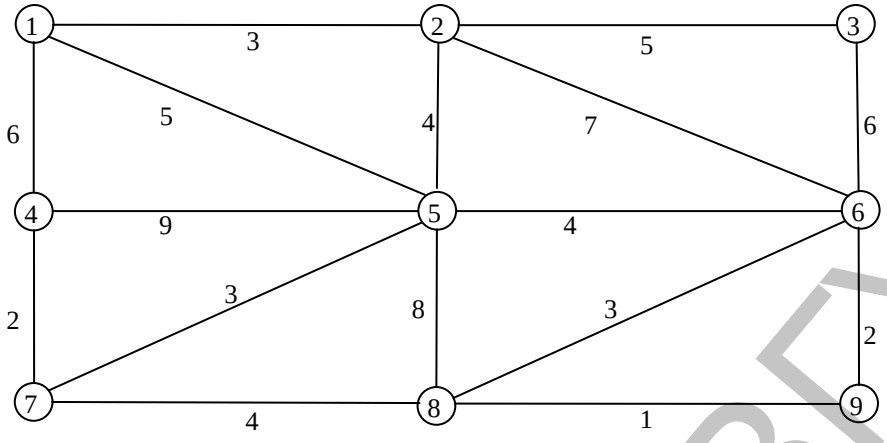


Рис. 4.1.

1) Согласно алгоритму Прима остовное дерево должно включать все вершины заданного графа. На первом шаге добавляем ребро {8,9}, имеющее минимальный вес, равный 1. Далее добавляем ребро {6,9} с весом 2. На третьем шаге выбор следует сделать из ребер {2,6}, {3,6}, {5,6}, {5,8}, {6,8}, {7,8}, которые смежные ребрам {6,9} и {8,9}. Ребро {6,8} образует цикл, поэтому его из списка исключаем. Среди остальных минимальный вес, равный 4, имеют два ребра. Выберем ребро {7,8}. В дальнейшем следует последовательно добавить: ребра {4,7} с весом 2, {5,7} с весом 3, {2,5} с весом 4, {1,2} с весом 3 и ребро {2,3} с весом 5. Полученное дерево представлено на рис. 4.2. Его вес равен 24.

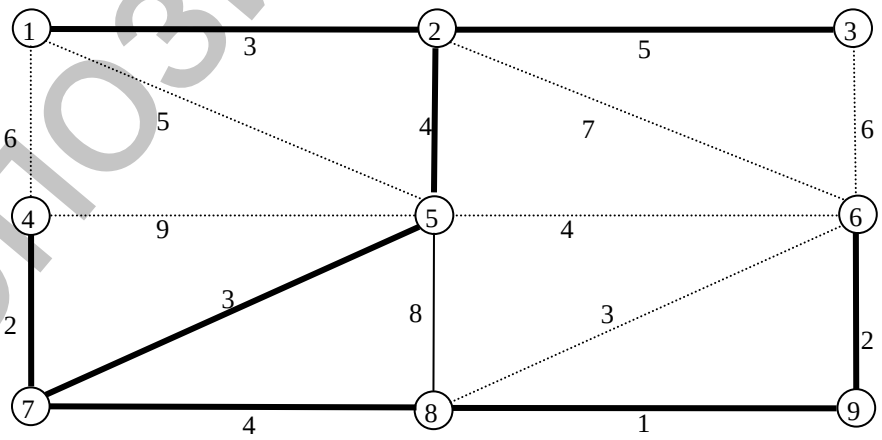


Рис. 4.2.

2) Остовное дерево, построенное согласно алгоритму Краскала приведено на рис. 4.3. К несвязному графу $G_0$ последовательно добавлены ребра: {8,9}, {4,7}, {6,9}, {1,2}, {5,7}, {2,5}, {5,6}, {2,3}. Это

остовное дерево отличается от дерева, построенного по алгоритму Прима, но его вес также равен 24.

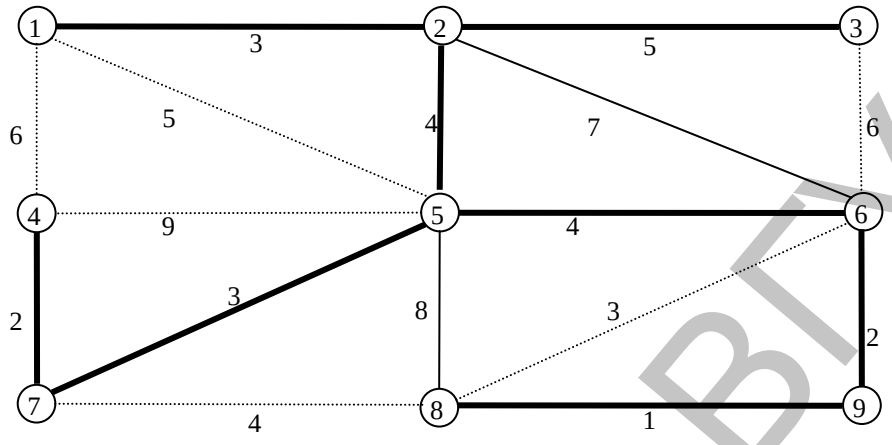


Рис.

Задания

1. Используя алгоритмы Прима и Краскала, построить остовное дерево графа на рисунке 4.1, веса ребер которого заданы в следующей таблице

| {1,2} | {1,4} | {1,5} | {2,3} | {2,5} | {2,6} | {3,6} | {4,5} | {4,7} | {5,6} | {5,7} | {5,8} | {6,9} | {6,8} | {7,8} | {8,9} |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 5 | 3 | 6 | 4 | 8 | 1 | 4 | 10 | 2 | 3 | 5 | 2 | 9 | 5 | 3 |

## § 5. Ориентированные графы

***Ориентированный*** граф $G$ (или ***орграф***), как и неориентированный, определяется заданием двух множеств $I$ и $U$ и обозначается также: $G=(I,U)$ .

Элементы множества $I$ , как и прежде, называются ***вершинами***. Будем их обозначать буквами $i$ , $j$. Вершины изображают точками плоскости или пространства.

Множество $U$ есть подмножество множества $I \times I$ (декартово произведение множества $I$ на себя). Элементами множества $U$ являются упорядоченные пары вершин, их называются ***дугами***. Другими словами, для двух рассматриваемых вершин указано, какая из них первая, ее называют ***началом*** дуги, какая вторая, ее называют ***концом*** дуги. Обозначать их будем либо буквами $u_1, u_2, \ldots, u_m$ , либо парами вида $(i, j)$, а изображать отрезками кривых или прямых линий, со стрелкой

у вершины, которая является концом дуги.

Понятия ***петли***, ***параллельных*** дуг, ***смежных*** вершин и ***смежных*** дуг графа вводятся также как аналогичные понятия в неориентированном графе.

Для ориентированного графа вводятся понятия ***полустепени захода*** $P^+(i_k)$ ***вершины*** $i_k$ – количество дуг, заходящих в $i_k$, и ***полустепени исхода*** $P^-(i_k)$ – количество дуг, исходящих из $i_k$. Понятно, что $P^+(i_k) + P^-(i_k) = P(i_k)$. На рис. 5.1*а* для вершины $i_2$ имеем: $P^+(i_2) = 1$, $P^-(i_2) = 2$, $P(i_2) = P^+(i_2) + P^-(i_2) = 3$.

Как и прежде, вершина, степень которой равна нулю, называется ***изолированной***. Вершина, степень которой равна единице, называется ***висячей***, инцидентная ей дуга также называется ***висячей***.

Если в графе $G$ множество $U$ содержит как ребра, так и дуги, то граф называется ***смешанным***. Смешанный граф преобразуется в орграф путем замены ребра $\{i, j\}$ на две дуги $(i, j)$ и $(j, i)$. На рис. 5.1*б* представлен пример смешанного графа. После замены ребра $\{i_1, i_2\}$ на дуги $(i_1, i_2)$ и $(i_2, i_1)$, ребра $\{i_2, i_4\}$ на дуги $(i_2, i_4)$ и $(i_4, i_2)$ получим орграф, который представлен на рис. 5.1*в*.

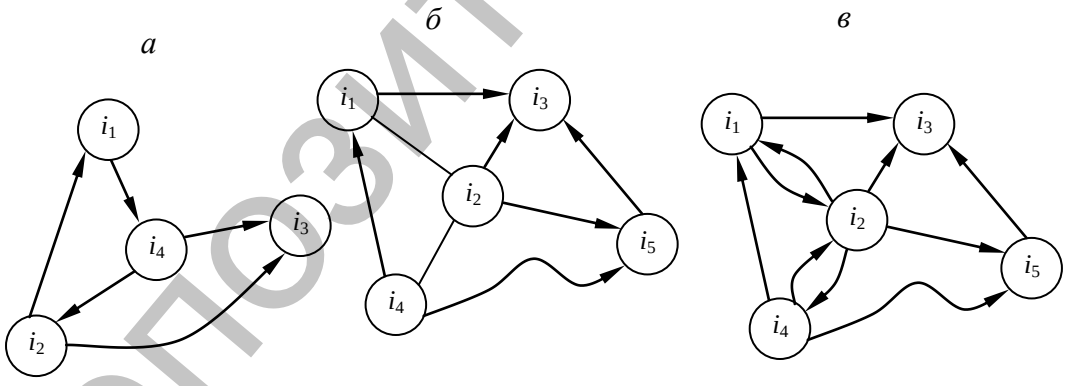


Рис. 5.1.

***Ориентированным маршрутом*** в орграфе называется последовательность дуг, в которой конец каждой предыдущей дуги совпадает с началом следующей:

$$(i_1, i_2), (i_2, i_3), \ldots, (i_{n-2}, i_{n-1}), (i_{n-1}, i_n). \qquad (5.1)$$

Задать маршрут (5.1) можно и последовательностью его вершин:

$$i_1, i_2, \ldots, i_n, \qquad (5.2)$$

где $i_1$ – начало маршрута, $i_n$ – конец маршрута. Говорят, что

вершина $i_n$ достижима из вершины $i_1$. ***Ориентированная цепь*** – ориентированный маршрут, все дуги которого различны. Ориентированная цепь, все вершины которой различны, называется ***путем***.

Если в ориентированном маршруте (ориентированной цепи, пути) начало и конец совпадают: $i_1 = i_n$, то получаем ***циклический ориентированный маршрут*** (***ориентированный цикл, контур***).

На рис.5.1в последовательность вершин $i_2, i_4, i_1, i_2, i_5, i_3$ задает ориентированную цепь; последовательность вершин $i_2, i_4, i_5, i_3$ задает путь, который можно записать как последовательность дуг

$$(i_2, i_4), (i_4, i_5), (i_5, i_3),$$

последовательность дуг $(i_1, i_2), (i_2, i_4), (i_4, i_1)$ задает контур.

Нетрудно убедиться, что справедлива

*Лемма 5.1*. Любой ориентированный маршрут содержит путь.

Ориентированные графы могут быть связными и несвязными. Любому орграфу может быть поставлен в соответствие неориентированный граф, который получается заменой дуг в орграфе на ребра. Орграф называется ***связным***, если соответствующий ему неориентированный граф является связным.

Для орграфа существует еще понятие ***сильной связности***. Говорят, что орграф сильно связный, если между любыми двумя его вершинами существует хотя бы один путь.

Так же как и для неориентированного графа вводятся понятие подграфа и понятие компоненты связности. Очевидно, что для ориентированного графа остается верной лемма 1.4.

Рассмотрим способы задания ориентированного графа. Как и неориентированный граф его можно задать рисунком или с помощью матрицы. Для простого орграфа $G$, в котором $|I| = n$, $|U| = m$, матрица ***смежности*** – это квадратная матрица $A$ порядка $n$ (строки и столбцы этой матрицы соответствуют вершинам графа $G$). Элементы матрицы определяются следующим образом:

$$a_{ij} = \begin{cases} 1, \text{ если } (i, j) \in U, \\ 0 \quad \text{в противном случае.} \end{cases}$$

Из определения вытекает, что матрица смежности орграфа не является симметрической матрицей. Число единиц в строке равно степени исхода соответствующей вершины, а число единиц в столбце равно степени захода соответствующей вершины.

Матрица ***инцидентности*** – это $n\times m$-матрица $B$ (строки этой матрицы соответствуют вершинам графа $G$ , а столбцы – дугам). Элементы матрицы определяются следующим образом:

$$b_{ik}=\begin{cases}-1\text{ , если вершина } i - \text{ начало дуги } u_k,\\ 1\text{ , если вершина } i - \text{ конец дуги } u_k,\\ 0 \quad \text{в противном случае.}\end{cases}$$

Из определения вытекает, что каждый столбец матрицы инцидентности содержит ровно два ненулевых элемента: 1 и –1. Степень вершины равна числу ненулевых элементов строки, число единиц в соответствующей строке равно степени захода, а число отрицательных элементов есть степень исхода.

Матрицы смежности и инцидентности графа, представленного на рис.5.1*а* имеют вид:

$$\begin{pmatrix}0&0&0&1\\1&0&1&0\\0&0&0&0\\0&1&1&0\end{pmatrix},\qquad \begin{array}{c}\\ i_1\\ i_2\\ i_3\\ i_4\end{array}\begin{pmatrix}\{i_1i_4\}&\{i_2i_1\}&\{i_2i_3\}&\{i_4i_2\}&\{i_4i_3\}\\ -1&1&0&0&0\\ 0&-1&-1&1&0\\ 0&0&1&0&0\\ 1&0&0&-1&-1\end{pmatrix}.$$

Еще один способ задания орграфа – ***списки смежности***. Каждой вершине $i$ графа ставится в соответствие список $S(i)$ вершин, недостижимых из вершины $i$ посредством одной дуги. Например, для графа, представленного на рис.5.1*а*, списки смежности следующие:

$$S(i_1)=(i_4),\ S(i_2)=(i_1,\ i_3),\ S(i_3)=\varnothing,\ S(i_4)=(i_2,\ i_3).$$

Задания

1. Доказать лемму 5.1.
2. Составить матрицу смежности и матрицу инцидентности орграфа, представленного на рис.5.1*в*.

## § 6. Задача построения кратчайшего пути из заданной вершины

Пусть $G=(I,U)$ – конечный связный орграф. Пусть в графе $G$ каждой дуге $(i, j)$ поставлено в соответствие некоторое неотрицательное число $\ell_{ij}$ , которое здесь назовем ***длиной*** (***весом***) дуги. В этом случае орграф называется ***взвешенным*** графом или ***сетью***.

Постановка задачи. Задана сеть $G=(I,U)$ , $|I|=n$ , в которой

выделены две вершины $s$ и $t$. Найти путь минимальной длины из заданной вершины $s$ в заданную вершину $t$.

Для каждой вершины $i \in I$ определим множество вершин, соседних справа:

$$I^+(i) = \{j \in I: \exists (i, j) \in U\},$$

т.е. это те вершины, которые достижимы из вершины $i$ посредством одной дуги.

Предлагаемый ниже алгоритм основан на упорядочении вершин сети по возрастанию расстояния от вершины $s$. Через $\ell(s, i)$ обозначим минимальное расстояние от вершины $s$ до вершины $i$.

*Поиск 1-ой ближайшей вершины $i_1$ к вершине $s$.*

1) Определим множество $I^+(s)$ (множество вершин соседних справа для вершины $s$).
2) Определим вершину $i_1$ из условия:

$$\ell_{si_1} = \min\{\ell_{sj}, j \in I^+(s)\}.$$

Другими словами, среди вершин, удовлетворяющих условию 1), найдем самую близкую к вершине $s$. Таким образом, $\ell(s, i_1) = \ell_{si_1}$.

*Поиск 2-ой ближайшей вершины $i_2$ к вершине $s$* (см. рис. 6.1).

1) Среди вершин множества $I \setminus \{s, i_1\}$ определяем вершину, ближайшую к вершине $i_1$. Для этого
   - определим множество $I^+(i_1) \setminus \{s\}$,
   - определим вершину $i_*$, для которой

$$\ell_{i_1 i_*} = \min\{\ell_{i_1 j}, j \in I^+(i_1) \setminus \{s\}\}.$$

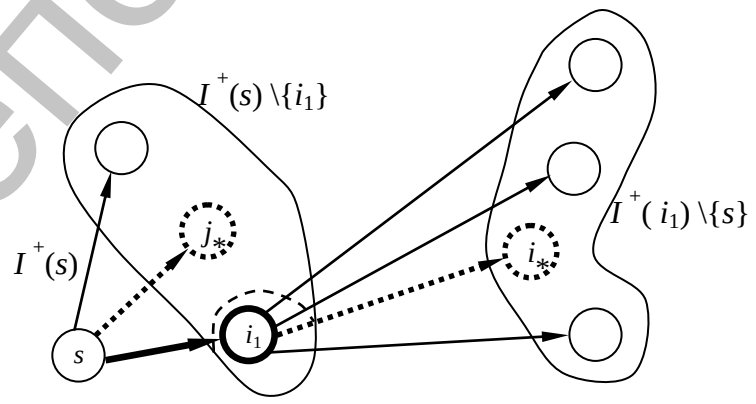


Рис. 6.1.

2) Среди вершин множества $I^+(s)\setminus\{i_1\}$ определяем вершину $j_*$, ближайшую к вершине $s$: $\ell_{sj_*} = \min\{\ell_{sj}, j \in I^+(s)\setminus\{i_1\}\}$ .

3) Определим вершину $i_2$:

если $\ell_{si_1} + \ell_{i_1 i_*} \geq \ell_{sj_*}$, то $i_2 = j_*$ и $\ell(s, i_2) = \ell_{sj_*}$;

если $\ell_{si_1} + \ell_{i_1 i_*} < \ell_{sj_*}$, то $i_2 = i_*$ и $\ell(s, i_2) = \ell_{si_1} + \ell_{i_1 i_*}$.

Далее находим 3-ю и т.д. ближайшие вершины к вершине $s$ .

Пусть определены $m$-1 ближайших вершин к вершине $s$ :

$$i_1, i_2, i_3, \ldots, i_{m-1},$$

и для них вычислены минимальные расстояния $\ell(s, i_p)$, $p = \overline{1, m-1}$.

*Поиск m-ой ближайшей вершины $i_m$ к вершине $s$.*

1) Для каждой из вершин $i_p$, $p = \overline{1, m-1}$, определим множество вершин, соседних справа, и исключим из каждого из них вершины $s, i_1, i_2, i_3, \ldots, i_{m-1}$ (если они попали в них):

$$\overline{I}^+(i_p) = I^+(i_p)\setminus\{s, i_1, i_2, i_3, \ldots, i_{m-1}\}.$$

Затем определим вершины $(i_p)_*$, $p = \overline{1, m-1}$, для которых

$$\ell_{i_p i_{(p)*}} = \min\{\ell_{i_p j},\ j \in \overline{I}^+(i_p)\}\ , p = \overline{1, m-1}.$$

2) Среди вершин множества $I^+(s)\setminus\{i_1, i_2, \ldots, i_{m-1}\}$ определяем вершину $j_*$, ближайшую к вершине $s$:

$$\ell_{sj_*} = \min\{\ell_{sj}, j \in I^+(s)\setminus\{i_1, i_2, \ldots, i_{m-1}\}\} \quad .$$

3) Среди вершин $\{j_*, (i_p)_*, p = \overline{1, m-1}\}$ определим вершину $i_m$ по правилу:

$$\ell(s, i_m) = \min\left\{\ \ell_{sj_*};\ \ell(s, i_p) + \ell_{i_p i_{(p)*}}, p = \overline{1, m-1}\right\}.$$

Поскольку заданная сеть является конечной, то количество элементов множеств вида $I^+(s)\setminus\{i_1, i_2, \ldots, i_{m-1}\}$ и $\overline{I}^+(i_p)$ на некотором шаге начнет убывать. Если путь из вершины $s$ в вершину $t$ существует, то через конечное число шагов определим и число $\ell(s, t)$. Если же путь из вершины $s$ в вершину $t$ не существует, то на каком-то шаге указанные выше множества станут пустыми (вершины $s, i_1, i_2, \ldots$ не будут иметь соседних справа среди остальных вершин сети).

Изложенная идея упорядочения вершин по их расстоянию от вершины $s$ лежит в основе алгоритма Дейкстры. В ходе его выполнения всем вершинам сети присваиваются метки – временные или по-

стоянные. Временная метка – это число, дающее верхнюю оценку минимального пути из вершины $s$ в рассматриваемую вершину. Постоянная метка – число, равное длине минимального пути.

**Алгоритм Дейкстры**

***Шаг* 0**. Каждой вершине $i$ сети $G$ ставим в соответствие метку $\ell(i)$ :

$$\ell(i) = \begin{cases} 0 \text{ , если } i = s \text{ ,} \\ \infty \text{ , если } \ i \in I \setminus \{s\}. \end{cases}$$

$\ell(s)$ – постоянная метка, остальные метки временные.

***Шаг* $k$**. Определены вершины с постоянными метками: $s, i_1, i_2, \ldots, i_{k-1}$, и $p$ – последняя вершина, получившая постоянную метку.

*1 этап* (этап обновления меток).

1) Находим $I^+(p)$ – множество вершин, соседних справа для вершины $p$.
2) Находим $\overline{I}^+(p)$ : из множества $I^+(p)$ исключаем вершины, имеющие постоянные метки, т.е. $\overline{I}^+(p) = I^+(p) \setminus \{ s, i_1, i_2, \ldots, i_{k-1}\}$.
3) Для каждой вершины $i \in \overline{I}^+(p)$ пересчитываем метки по правилу:

$$\ell'(i) = \min \{ \ell(i); \ \ell(p) + \ \ell_{pi} \} ,$$

где $\ell(i)$ – прежняя временная метка вершины $i$, $\ell'(i)$ – новая временная метка вершины $i$, $\ell(p)$ – постоянная метка вершины $p$, $\ell_{pi}$ – длина дуги $(p, i)$.

*2 этап* (перевод одной из временных меток в постоянную метку).

1) Формируем множество $I_k$ – множество вершин с временными метками.
2) Находим вершину $i_k \in I_k$ с минимальной временной меткой:

$$\ell(i_k) = \min \{ \ell(i), i \in I_k \}. \qquad (6.1)$$

3) Метка $\ell(i_k)$ – становится постоянной, вершина $p = i_k$ .

***Выход из алгоритма*** (принцип останова).

Если $p = t$ , то $\ell(p) = \ell(t)$ – минимальная длина пути из заданной вершины $s$ в заданную вершину $t$.

Если $\overline{I}^+(p) = \varnothing$ и вершина $t$ имеет временную метку, равную $\infty$, то в сети не существует пути из вершины $s$ в вершину $t$ .

*Замечание 1*. Алгоритм может быть модифицирован таким образом, что кроме значения минимального пути будет получен и сам путь, например, в виде последовательности вершин. Для этого метку вершины на шаге обновления меток дополним информацией о вершине, через которую пересчитывается минимум в (6.1), т.е. метка вершины будет состоять из двух частей. Как только вершина $t$ получит постоянную метку, по второй части метки определим предшествующую ей вершину. По второй части метки последней определим ей предшествующую вершину и т.д., пока не доберемся до вершины $s$ .

*Замечание 2*. Алгоритм может быть использован для решения задачи о кратчайшем пути из заданной вершины до любой другой вершины. В этом случае изменяется принцип останова.

Если все вершины получили постоянные метки, то любая вершина достижима из вершины $s$ . Значение постоянной метки вершины есть длина минимального пути из заданной вершины $s$ в эту вершину.

Если на некотором шаге $\bar{I}^+(p) = \varnothing$ и некоторые вершины имеют временную метку, равную $\infty$, то в сети не существует пути из вершины $s$ до этих вершин.

Пример. На сети, представленной на рис. 6.1, найти кратчайший путь из вершины $s$ до вершины $t$ .

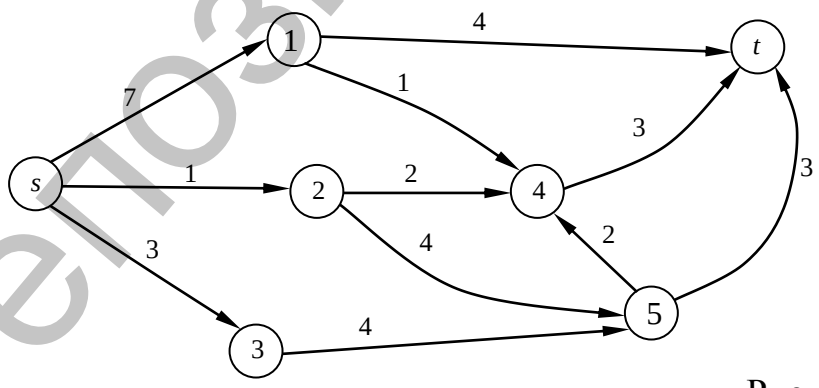


Рис. 6.1.

Итерация 0. $\ell(s) = 0$, $\ell(i) = \infty$, $i \in \{1, 2, 3, 4, 5, t\}$, $p = s$.

Дальнейшие вычисления на каждой итерации оформим в виде таблиц.

Итерация 1.

| $p$ | $I^+(p)$ | $\overline{I}^+(p)$ | первая часть метки $\ell'(i)$ | вторая часть метки | Результат сравнения | Вид метки |
|---|---|---|---|---|---|---|
| $s$ | 1, 2, 3 | 1<br>2<br>3 | $\ell'(1) = \min\{\infty;0+7\}=7,$<br>$\ell'(2) = \min\{\infty;0+1\}=1,$<br>$\ell'(3) = \min\{\infty;0+3\}=3.$ | $s$<br>$s$<br>$s$ | (min) | врем.<br>пост.<br>врем. |

Итерация 2.

| $p$ | $I^+(p)$ | $\overline{I}^+(p)$ | первая часть метки $\ell'(i)$ | вторая часть метки | Результат сравнения | Вид метки |
|---|---|---|---|---|---|---|
| соседние справа для вершины $p$ | | | | | | |
| 2 | 4, 5 | 4<br>5 | $\ell'(4) = \min\{\infty;1+2\}=3,$<br>$\ell'(5) = \min\{\infty;1+4\}=5.$ | 2<br>2 | (min) | врем.<br>врем. |
| другие вершины с временными метками | | | | | | |
| | | 1<br>3 | $\ell'(1) = 7,$<br>$\ell'(3) = 3.$ | $s$<br>$s$ | (min) | врем.<br>пост. |

Замечание. Выбор вершины $p$ неоднозначен, можно выбрать либо вершину 4, либо вершину 3. Полагаем $p = 3$.

| Итерация 3. | | | | | | |
|---|---|---|---|---|---|---|
| $p$ | $I^+(p)$ | $\overline{I}^+(p)$ | первая часть метки $\ell'(i)$ | вторая часть метки | Результат сравнения | Вид метки |
| соседние справа для вершины $p$ | | | | | | |
| 3 | 5 | 5 | $\ell'(5) = \min\{5;3+4\}=5.$ | 2 | | врем. |
| другие вершины с временными метками | | | | | | |
| | | 1<br>4 | $\ell'(1) = 7,$<br>$\ell'(4) = 3.$ | $s$<br>2 | (min) | врем.<br>пост. |

Итерация 4.

| $p$ | $I^+(p)$ | $\overline{I}^+(p)$ | первая часть метки $\ell'(i)$ | вторая часть метки | Результат сравнения | Вид метки |
|---|---|---|---|---|---|---|
| соседние справа для вершины $p$ | | | | | | |
| 4 | $t$ | $t$ | $\ell'(t) = \min\{\infty;3+3\}=6.$ | 4 | | врем. |
| другие вершины с временными метками | | | | | | |
| | | 1<br>5 | $\ell'(1) = 7,$<br>$\ell'(5) = 5.$ | $s$<br>2 | (min) | врем.<br>пост. |

Итерация 5.

| $p$ | $I^+(p)$ | $\overline{I}^+(p)$ | первая часть метки $\ell'(i)$ | вторая часть метки | Результат сравнения | Вид метки |
|---|---|---|---|---|---|---|
| соседние справа для вершины $p$ | | | | | | |
| 5 | 4, $t$ | $t$ | $\ell'(t) = \min\{6;5+3\}=6$ | 4 | (min) | пост. |
| другие вершины с временными метками | | | | | | |
| | | 1 | $\ell'(1) = 7$ | $s$ | | врем. |

Так как вершина $t$ получила постоянную метку, то решение закончено. Длина кратчайшего пути из заданной вершины $s$ в вершину $t$ равна 6. По второй части метки вершин восстанавливаем этот путь: $t \leftarrow 4 \leftarrow 2 \leftarrow s$.

*Замечание 3*. Алгоритм неприменим к сетям, где есть отрицательные веса дуг. На рис. 6.2 представлена сеть, где $\ell_{1t} = -2$. Согласно алгоритму Дейкстры минимальный путь из вершины $s$ в вершину $t$ равен 3 (вершина $t$ является 1-ой ближайшей). На самом деле минимальный путь проходит через вершину 1, его длина равна 2.

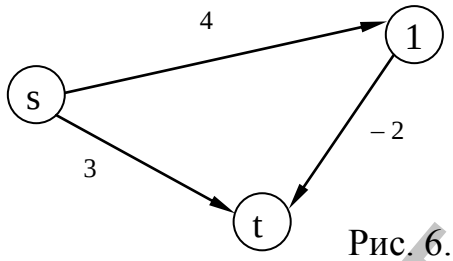

Рис. 6.2.

## § 7. Общая задача построения кратчайшего пути

Пусть $G=(I,U)$ , $|I|=n$ , – конечный связный орграф. Как и в предыдущем параграфе, каждой дуге $(i, j)$ поставлен соответствие вес дуги $\ell_{ij}$ (здесь это число произвольное).

Постановка задачи. В заданной сети $G=(I,U)$ , найти путь минимальной длины между любой парой вершин или установить отсутствие пути.

Заметим, что если $\ell_{ij} \geq 0$, то для решения можно использовать алгоритм Дейкстры, задав в качестве вершины $s$ по очереди все имеющиеся вершины, т.е. алгоритм следует применить $n$ раз.

В случае, когда $\ell_{ij}$ есть произвольное по знаку число, следует применять модифицированный алгоритм Дейкстры. В этом случае следует протестировать сеть на наличие контуров отрицательного ве-

са, так как по алгоритму Дейкстры метки вершин такого контура будут меняться в сторону уменьшения до минус бесконечности. Произойдет зацикливание алгоритма.

В дальнейшем предполагаем, что в сети нет контуров отрицательного веса.

Указанную выше задачу хорошо решает алгоритм, предложенный Флойдом. Договоримся, что вершины сети перенумеруем элементами множества $\{1, 2, \dots, n\}$.

**Алгоритм Флойда**

***Шаг* 0**. Формируем две $n \times n$-матрицы:

$L^{(0)} = \left( \ell_{ij}^{(0)}, i, j = \overline{1,n} \right)$ – матрица расстояний, где

$$\ell_{ij}^{(0)} = \begin{cases} \ell_{ij} \text{ , если } (i, j) \in U, \\ \infty \text{ , если } (i, j) \notin U, \\ 0 \text{ , если } i = j . \end{cases} \qquad (7.1)$$

$S^{(0)} = \left( s_{ij}^{(0)}, i, j = \overline{1,n} \right)$ – справочная матрица, где $s_{ij}^{(0)} = i, \ j = \overline{1,n}$ .

***Шаг* k**. Известны матрицы $L^{(k-1)}$ и $S^{(k-1)}$. Находим матрицы $L^{(k)} = ( \ell_{ij}^{(k)} \ i, j = \overline{1,n} )$, и $S^{(k-1)} = ( s_{ij}^{(k)}, i, j = \overline{1,n} )$:

1) в матрице $L^{(k-1)}$ выделяем $k$-ый столбец и $k$-ую строку (назовем их *базовыми*),
2) вычисляем элементы новых матриц по правилу, используя элементы базовых строки и столбца:

$$\text{если } \ell_{ij}^{(k-1)} > \ell_{ik}^{(k-1)} + \ell_{kj}^{(k-1)}, \text{ то } \begin{cases} \ell_{ij}^{(k)} = \ell_{ik}^{(k-1)} + \ell_{kj}^{(k-1)}, \\ s_{ij}^{(k)} = s_{kj}^{(k-1)}, \end{cases} \qquad (7.2)$$

$$\text{если } \ell_{ij}^{(k-1)} \le \ell_{ik}^{(k-1)} + \ell_{kj}^{(k-1)}, \quad \text{то } \begin{cases} \ell_{ij}^{(k)} = \ell_{ij}^{(k-1)}, \\ s_{ij}^{(k)} = s_{ij}^{(k-1)}. \end{cases} \qquad (7.3)$$

***Выход из алгоритма*** (принцип останова).

Если $k = n + 1$, то найденная матрица $L^{(n)} = \left( \ell_{ij}^{(n)}, i, j = \overline{1,n} \right)$ является матрицей искомых расстояний. Справочная матрица $S^{(n)} = \left( s_{ij}^{(n)}, i, j = \overline{1,n} \right)$ служит для определения вершин, через которые пройдет соответствующий минимальный путь. Значение $s_{ij}^{(n)}$ показывает номер $t$ вершины, предшествующей вершине $j$ на пути из вершины $i$. Далее значение $s_{it}^{(n)}$ показывает номер вершины, предшествующий вершине $t$ на пути из вершины $i$. И так далее, пока значение эле-

мента $i$-ой строки не будет равно $i$.

Поясним смысл шага 1. Базовыми являются первая строка и первый столбец. Рассмотрим элемент $\ell_{ij}^{(0)}$ матрицы $L^{(0)}$, где $i, j = \overline{2,n}$. Согласно формулам (7.2) и (7.3) сравниваются две величины: $\ell_{ij}^{(0)}$ – длина дуги непосредственно из вершины $i$ в вершину $j$, и $\ell_{i1}^{(0)} + \ell_{1j}^{(0)}$ – длина «обходного» пути из вершины $i$ в вершину $j$ через вершину 1 (см. рис.7.1). Элементу $\ell_{ij}^{(1)}$ присваивается значение, равное меньшему из сравниваемых величин. Очевидно, что при $\ell_{i1}^{(0)} = \infty$ или $\ell_{1j}^{(0)} = \infty$ (не существует «обходного» пути из вершины $i$ в вершину $j$ через вершину 1) значение $\ell_{ij}^{(1)} = \ell_{ij}^{(0)}$. В случае, когда $\ell_{ij}^{(1)} < \ell_{ij}^{(0)}$, изменяется соответствующий элемент справочной матрицы: $s_{ij}^{(1)} = 1$. Так фиксируется вершина, через которую проходит «обходной» путь.

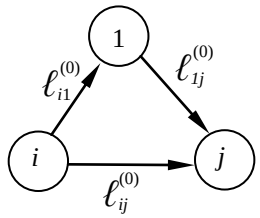

Рис. 7.1.

*Теорема 1*. Пусть $G=(I,U)$ – взвешенный орграф, в котором нет контуров отрицательной длины. Если в матрице $L^{(k)}$ элемент $\ell_{ij}^{(k)} \neq \infty$, то

а) $\ell_{ij}^{(k)}$ есть длина минимального пути из вершины $i$ в вершину $j$, где промежуточными вершинами являются вершины из $\{1, 2, \ldots, k\}$;

б) в матрице $S^{(k)}$ элемент $s_{ij}^{(k)}$ – номер вершины, которая непосредственно предшествует вершине $j$ на пути из $i$ в $j$.

*Доказательство*. Пусть $k$ – некоторое целое неотрицательное число, не большее $n = |I|$. В соответствие ему поставим множество вершин $\{1, 2, \ldots, k\}$.

Построим матрицу $W(k)$ истинно кратчайших расстояний из каждой вершины $i$ в каждую вершину $j$, $i = \overline{1,n}$, $j = \overline{1,n}$, где промежуточными вершинами пути являются вершины из $\{1, 2, \ldots, k\}$, следующим образом.

Если $k = 0$, то множество промежуточных вершин пусто, положим $W(0) = L^{(0)}$.

Пусть $k > 0$. Для пары вершин $i$ и $j$, $i, j = \overline{1,n}$, строим все пути (если они существуют) из $i$ в $j$ такие, что промежуточными вершинами могут быть лишь вершины из множества $\{1, 2, \ldots, k\}$, сравниваем длины всех построенных путей, выбираем наименьшее значение. Это

значение и есть элемент $w_{ij}(k)$ матрицы $W(k)$. Если же указанного выше пути не существует, то полагаем $w_{ij}(k) = \infty$.

Докажем, что $W(k) = L^{(k)}$, где матрица $L^{(k)}$ определена на $k$-том шаге алгоритма Флойда.

Воспользуемся методом математической индукции по числу $k$, $k = \overline{1,n}$.

Пусть $k = 1$. Тогда для пары вершин $i$ и $j$ соответствующий элемент матрицы $W(k)$ есть длина минимального пути из вершины $i$ в вершину $j$, где промежуточной вершиной может быть вершина 1. Другими словами, следует рассмотреть пути вида $i \to j$ или $i \to 1 \to j$. Следовательно, элемент матрицы $W(k)$ совпадет с соответствующим элементом матрицы $L^{(k)}$ (см. пояснения к шагу 1 и рис.7.1).

Пусть утверждение справедливо для $k = m-1$.

Докажем его для $k = m$. Согласно алгоритму Флойда элементы матриц $L^{(m)}$ и $S^{(m)}$ находим либо по формуле (7.2):

$$\ell_{ij}^{(m)} = \ell_{im}^{(m-1)} + \ell_{mj}^{(m-1)}, \quad s_{ij}^{(m)} = s_{mj}^{(m-1)}, \tag{7.4}$$

либо по формуле (7.3):

$$\ell_{ij}^{(m)} = \ell_{ij}^{(m-1)}, \quad s_{ij}^{(m)} = s_{ij}^{(m-1)}. \tag{7.5}$$

Базовыми являются $m$-ая строка и $m$-ый столбец. Сравниваем две величины: $\ell_{ij}^{(m-1)}$ – длина пути из вершины $i$ в вершину $j$, и $\ell_{im}^{(m-1)} + \ell_{mj}^{(m-1)}$ – длина «обходного» пути из вершины $i$ в вершину $j$ через вершину $m$ (см. рис. 7.1). Причем по индуктивному предположению числа $\ell_{ij}^{(m-1)}$, $\ell_{im}^{(m-1)}$, $\ell_{mj}^{(m-1)}$ – длины кратчайших путей, где промежуточными вершинами могут быть вершины из множества $\{1, 2, \ldots, m-1\}$. Понятно, что $\ell_{im}^{(m-1)} + \ell_{mj}^{(m-1)}$ – длина самого короткого «обходного» пути из вершины $i$ в вершину $j$ через вершины из множества $\{1, 2, \ldots, m\}$. Поэтому формулы (7.4) или (7.5) дают значение $\ell_{ij}^{(m)}$ кратчайшего пути из вершины $i$ в вершину $j$ через вершины из множества $\{1, 2, \ldots, m\}$.

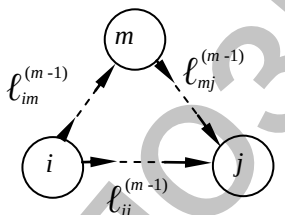

Рис. 7.1.

Если значение $\ell_{ij}^{(m)}$ определено по формуле (7.4), то изменяется соответствующий элемент справочной матрицы: $s_{ij}^{(m)} = s_{mj}^{(m-1)}$. Таким способом фиксируется вершина, которая предшествует вершине $j$ на

новом «обходном» пути.

Ясно, что при $\ell_{im}^{(m-1)} = \infty$ или $\ell_{mj}^{(m-1)} = \infty$ не существует «обходного» пути из вершины $i$ в вершину $j$ через вершину $m$.

Итак, $\ell_{ij}^{(m)} = w_{ij}(m)$, т.е. $W(m) = L^{(m)}$.

Теорема доказана.

*Замечание*. Если при формировании матрицы $L^{(0)}$ положить $\ell_{ii}^{(0)} = \infty$, то на $k$-том шаге алгоритма Флойда элемент $\ell_{ii}^{(n)} \neq \infty$ матрицы $L^{(k)}$ равен длине кратчайшего контура (если он существует), проходящего через вершину $i$ и вершины множества $\{1, 2,\ldots, k\}$. Поэтому алгоритм Флойда можно использовать для нахождения кратчайших контуров (если они существуют), проходящих через вершины данного орграфа. Значения $\ell_{ii}^{(n)} \neq \infty$ и есть искомые величины. По справочной матрице $S^{(n)}$ находим вершины, через которые проходит контур. Понятно, что если в исходном графе существуют контуры отрицательной длины, то на некотором шаге появятся отрицательные диагональные элементы.

Итак, алгоритм Флойда может быть использован для нахождения контуров отрицательного веса. Если же поставлена задача нахождения пути минимальной длины между любой парой вершин, и на некотором шаге появился отрицательный диагональный элемент, то это признак того, что исходная задача не корректна, решение должно быть прекращено. Понятно, что если некоторый маршрут содержит контур отрицательной длины, то его длину можно сделать как угодно малой.

Пример 1. На сети, представленной на рис. 7.2, найти кратчайший путь (если он существует) между любой парой вершин.

Итерация 0.Строим матрицы $L^{(0)}$ и $S^{(0)}$:

$$L^{(0)} = \begin{pmatrix} 0 & 7 & 3 & 8 & \infty & \infty \\ \infty & 0 & 3 & 1 & \infty & 2 \\ \infty & \infty & 0 & 9 & 4 & \infty \\ \infty & \infty & \infty & 0 & 2 & 3 \\ \infty & \infty & 5 & \infty & 0 & 6 \\ \infty & 4 & \infty & \infty & \infty & 0 \end{pmatrix} = L^{(1)}, \quad S^{(0)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix} = S^{(1)}.$$

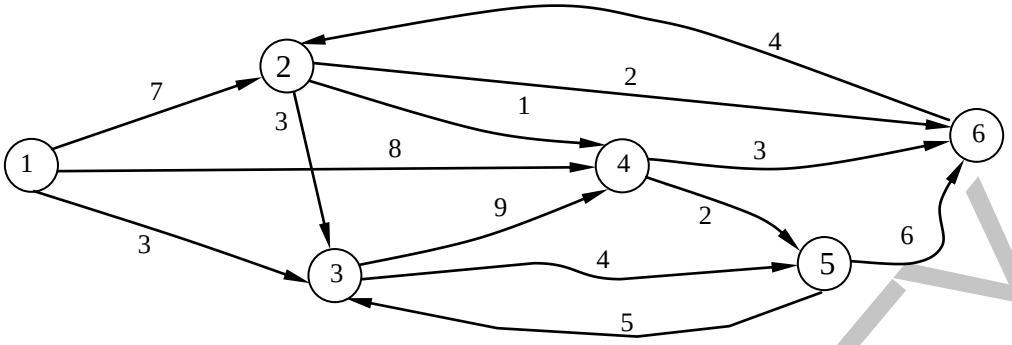


Рис. 7.2.

Итерация 1. В полученной матрице $L^{(0)}$ выделяем первую строку и первый столбец. Для всех элементов матрицы $L^{(0)}$, где $i, j = \overline{2,6}$, имеем $\ell_{i1}^{(0)} = \infty$. Поэтому $L^{(0)} = L^{(1)}$ и $S^{(0)} = S^{(1)}$.

Итерация 2. В матрице $L^{(1)}$ выделяем вторую строку и второй столбец. Поскольку во втором столбце $\ell_{i2}^{(1)} = \infty$, $i = \overline{3,5}$, то строки с этими номерами не изменятся. Аналогично, во второй строке $\ell_{2j}^{(1)} = \infty$, $j = 1, 5$, поэтому не изменятся элементы первого и пятого столбцов. Значит, следует пересчитать элементы $\ell_{13}^{(1)}$, $\ell_{14}^{(1)}$, $\ell_{16}^{(1)}$, $\ell_{63}^{(1)}$, $\ell_{64}^{(1)}$, $\ell_{66}^{(1)}$:

$$\ell_{13}^{(2)} = \min(3;\ 7+3) = 3 = \ell_{13}^{(1)};$$
$$\ell_{14}^{(2)} = \min(8;\ 7+1) = 8 = \ell_{14}^{(1)};$$
$$\ell_{16}^{(2)} = \min(\infty;\ 7+2) = 9 \neq \ell_{16}^{(1)}, \qquad s_{16}^{(2)} = 2;$$
$$\ell_{63}^{(2)} = \min(\infty;\ 4+3) = 7 \neq \ell_{63}^{(1)}, \qquad s_{63}^{(2)} = 2;$$
$$\ell_{64}^{(2)} = \min(\infty;\ 4+1) = 5 \neq \ell_{64}^{(1)}, \qquad s_{64}^{(2)} = 2;$$
$$\ell_{66}^{(2)} = \min(0;\ 4+2) = 0 = \ell_{66}^{(1)}.$$

Таким образом,

$$L^{(2)} = \left(\begin{array}{cc:c:cc:c} 0 & 7 & 3 & 8 & \infty & 9 \\ \infty & 0 & 3 & 1 & \infty & 2 \\ \hdashline \infty & \infty & 0 & 9 & 4 & \infty \\ \hdashline \infty & \infty & \infty & 0 & 2 & 3 \\ \infty & \infty & 5 & \infty & 0 & 6 \\ \infty & 4 & 7 & 5 & \infty & 0 \end{array}\right), \quad S^{(2)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 2 & 2 & 6 & 6 \end{pmatrix}$$

Итерация 3. Базовыми являются третья строка и третий столбец. Не изменяются базовые строка и столбец, а также первый, второй и шестой столбцы и четвертая строка. Результат пересчета остальных

элементов – матрицы $L^{(3)}$ и $S^{(3)}$:

$$L^{(3)} = \left(\begin{array}{cccc:cc} 0 & 7 & 3 & 8 & 7 & 9 \\ \infty & 0 & 3 & 1 & 7 & 2 \\ \infty & \infty & 0 & 9 & 4 & \infty \\ \hdashline \infty & \infty & \infty & 0 & 2 & 3 \\ \hdashline \infty & \infty & 5 & 14 & 0 & 6 \\ \infty & 4 & 7 & 5 & 11 & 0 \end{array}\right), \quad S^{(3)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 2 \\ 2 & 2 & 2 & 2 & 3 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 3 & 5 & 5 \\ 6 & 6 & 2 & 2 & 3 & 6 \end{pmatrix}$$

Итерация 4. Базовыми являются четвертая строка и четвертый столбец. Не изменяются базовые строка и столбец, а также первый, второй, третий столбцы. Результат пересчета остальных элементов – матрицы $L^{(4)}$ и $S^{(4)}$:

$$L^{(4)} = \left(\begin{array}{cccc:cc} 0 & 7 & 3 & 8 & 7 & 9 \\ \infty & 0 & 3 & 1 & 3 & 2 \\ \infty & \infty & 0 & 9 & 4 & 12 \\ \infty & \infty & \infty & 0 & 2 & 3 \\ \hdashline \infty & \infty & 5 & 14 & 0 & 6 \\ \hdashline \infty & 4 & 7 & 5 & 7 & 0 \end{array}\right), \quad S^{(4)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 2 \\ 2 & 2 & 2 & 2 & 4 & 2 \\ 3 & 3 & 3 & 3 & 3 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 3 & 5 & 5 \\ 6 & 6 & 2 & 2 & 4 & 2 \end{pmatrix}.$$

Итерация 5. Базовыми являются пятая строка и пятый столбец. Не изменяются базовые строка и столбец, а также первый, второй столбцы. Результат пересчета остальных элементов – матрицы $L^{(5)}$ и $S^{(5)}$:

$$L^{(5)} = \left(\begin{array}{cccc:cc} 0 & 7 & 3 & 8 & 7 & 9 \\ \infty & 0 & 3 & 1 & 3 & 2 \\ \infty & \infty & 0 & 9 & 4 & 10 \\ \infty & \infty & 7 & 0 & 2 & 3 \\ \infty & \infty & 5 & 14 & 0 & 6 \\ \hdashline \infty & 4 & 7 & 5 & 7 & 0 \end{array}\right), \quad S^{(5)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 2 \\ 2 & 2 & 2 & 2 & 4 & 2 \\ 3 & 3 & 3 & 3 & 3 & 5 \\ 4 & 4 & 5 & 4 & 4 & 4 \\ 5 & 5 & 5 & 3 & 5 & 5 \\ 6 & 6 & 2 & 2 & 4 & 6 \end{pmatrix}.$$

Итерация 6. Базовыми являются шестая строка и шестой столбец. Не изменяются базовые строка и столбец, а также первый столбец. Результат пересчета остальных элементов – матрицы $L^{(6)}$ и $S^{(6)}$:

$$L^{(6)} = \begin{pmatrix} 0 & 7 & 3 & 8 & 7 & 9 \\ \infty & 0 & 3 & 1 & 3 & 2 \\ \infty & 14 & 0 & 9 & 4 & 10 \\ \infty & 7 & 7 & 0 & 2 & 3 \\ \infty & 10 & 5 & 11 & 0 & 6 \\ \infty & 4 & 7 & 5 & 7 & 0 \end{pmatrix}, \quad S^{(6)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 2 \\ 2 & 2 & 2 & 2 & 4 & 2 \\ 3 & 6 & 3 & 3 & 3 & 5 \\ 4 & 6 & 5 & 4 & 4 & 4 \\ 5 & 6 & 5 & 2 & 5 & 5 \\ 6 & 6 & 2 & 2 & 4 & 6 \end{pmatrix}.$$

Матрицы $L^{(6)}$ и $S^{(6)}$ являются искомыми. Например, кратчайшее расстояние из вершины 6 в вершину 3 равно $\ell_{63}^{(6)} = 7$. Путь проходит через вершины (указываем в обратном порядке) 3, 2 (элемент $s_{63}^{(6)}$), 6 (элемент $s_{62}^{(6)}$).

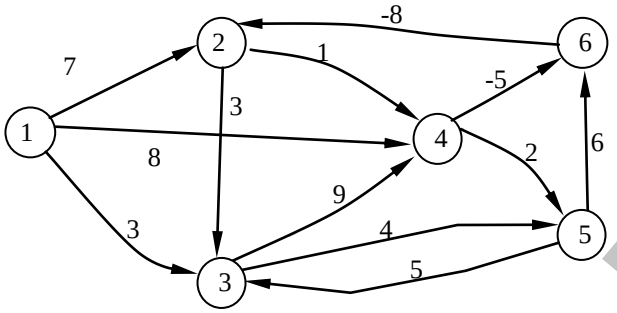


Рис. 7.3.

Пример 2. На сети, представленной на рис. 7.3, найти контуры отрицательной длины, если они существуют.

Итерация 0.Строим матрицы $L^{(0)}$ и $S^{(0)}$:

$$L^{(0)} = \left(\begin{array}{c|c|cccc} 0 & 7 & 3 & 8 & \infty & \infty \\ \hline \infty & 0 & 3 & 1 & \infty & 2 \\ \hline \infty & \infty & 0 & 9 & 4 & \infty \\ \infty & \infty & \infty & 0 & 2 & 3 \\ \infty & \infty & 5 & \infty & 0 & 6 \\ \infty & 4 & \infty & \infty & \infty & 0 \end{array}\right) = L^{(1)},\ S^{(0)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix} = S^{(1)}.$$

Итерация 1. Как и в предыдущем примере, для всех элементов матрицы $L^{(0)}$, где $i, j = \overline{2,6}$, имеем $\ell_{i1}^{(0)} = \infty$. Поэтому $L^{(0)} = L^{(1)}$ и $S^{(0)} = S^{(1)}$.

Итерация 2.

$$L^{(2)} = \left(\begin{array}{cc|c|cc} \infty & 7 & 3 & 8 & \infty & \infty \\ \infty & \infty & 3 & 1 & \infty & \infty \\ \hline \infty & \infty & \infty & 9 & 4 & \infty \\ \hline \infty & \infty & \infty & \infty & 2 & -5 \\ \infty & \infty & 5 & \infty & \infty & 6 \\ \infty & -8 & -5 & -7 & \infty & \infty \end{array}\right),\ S^{(2)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 2 & 2 & 6 & 6 \end{pmatrix}.$$

Итерация 3.

$$L^{(3)} = \left(\begin{array}{cccc:c:c} \infty & 7 & 3 & 8 & 7 & \infty \\ \infty & \infty & 3 & 1 & 7 & \infty \\ \infty & \infty & \infty & 9 & 4 & \infty \\ \hdashline \infty & \infty & \infty & \infty & 2 & -5 \\ \hdashline \infty & \infty & 5 & 14 & 9 & 6 \\ \infty & -8 & -5 & -7 & -1 & \infty \end{array}\right), \quad S^{(3)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 1 \\ 2 & 2 & 2 & 2 & 3 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 3 & 3 & 5 \\ 6 & 6 & 2 & 2 & 3 & 6 \end{pmatrix}.$$

Итерация 4.

$$L^{(4)} = \left(\begin{array}{cccc:c:c} \infty & 7 & 3 & 8 & 7 & 3 \\ \infty & \infty & 3 & 1 & 3 & -4 \\ \infty & \infty & \infty & 9 & 4 & 4 \\ \infty & \infty & \infty & \infty & 2 & -5 \\ \hdashline \infty & \infty & 5 & 14 & 9 & 6 \\ \hdashline \infty & -8 & -5 & -7 & -5 & -12 \end{array}\right), \quad S^{(4)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 4 \\ 2 & 2 & 2 & 2 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 3 & 3 & 5 \\ 6 & 6 & 2 & 2 & 4 & 4 \end{pmatrix}.$$

Поскольку $\ell_{66}^{(4)} = -12$, то через вершину 6 проходит контур отрицательной длины, причем вершины этого контура из множества {2, 4}. Действительно, так как $s_{66}^{(4)} = 4$, $s_{64}^{(4)} = 2$, $s_{62}^{(4)} = 6$, то контур состоит из дуг (6,2), (2,4), (4,6).

Итерация 5.

$$L^{(5)} = \left(\begin{array}{cccc:c:c} \infty & 7 & 3 & 8 & 7 & 3 \\ \infty & \infty & 3 & 1 & 3 & -4 \\ \infty & \infty & 9 & 9 & 4 & 4 \\ \infty & \infty & 7 & 16 & 2 & -5 \\ \infty & \infty & 5 & 14 & 9 & 6 \\ \hdashline \infty & -8 & -5 & -7 & -5 & -12 \end{array}\right), \quad S^{(5)} = \begin{pmatrix} 1 & 1 & 1 & 1 & 3 & 4 \\ 2 & 2 & 2 & 2 & 4 & 4 \\ 3 & 3 & 5 & 3 & 3 & 4 \\ 4 & 4 & 5 & 5 & 4 & 4 \\ 5 & 5 & 5 & 3 & 3 & 5 \\ 6 & 6 & 2 & 2 & 4 & 4 \end{pmatrix}.$$

Итерация 6.

$$L^{(6)} = \begin{pmatrix} \infty & -5 & -2 & -4 & -2 & 3 \\ \infty & -12 & -9 & -11 & -9 & -4 \\ \infty & -4 & -1 & -3 & -1 & 4 \\ \infty & -13 & -10 & -12 & -10 & -5 \\ \infty & -2 & 1 & -1 & 1 & 6 \\ \infty & -8 & -5 & -7 & -5 & -12 \end{pmatrix}, \quad S^{(6)} = \begin{pmatrix} 1 & 6 & 6 & 6 & 6 & 4 \\ 2 & 6 & 6 & 6 & 6 & 4 \\ 3 & 6 & 6 & 6 & 6 & 4 \\ 4 & 6 & 6 & 6 & 6 & 4 \\ 5 & 6 & 6 & 6 & 6 & 4 \\ 6 & 6 & 2 & 2 & 4 & 4 \end{pmatrix}.$$

В сети имеется контур отрицательной длины, который проходит через вершину 3, поскольку $\ell_{33}^{(6)} = -1$.

## § 8. Задача о максимальном потоке

Теория потоков возникла первоначально в связи с разработкой методов решения задач, связанных с рациональной перевозкой грузов. Схема доставки груза представлялась в виде графа, по ребрам которого проходит подлежащий максимизации поток груза. Позднее обнаружилось, что к задачам о максимальном потоке сводятся и другие важные оптимизационные практические задачи.

**1. Стационарный поток. Разрез сети**. Пусть $G=(I,U)$ – ориентированный граф без петель и без параллельных дуг. Каждой дуге $(i, j)$ поставлен соответствие вес дуги $d_{ij}$ – некоторое неотрицательное число, которое назовем ***пропускной способностью*** дуги. Среди вершин множества $I$ выделены две вершины: $s$ – источник и $t$ – сток.

***Стационарным потоком*** $x$ ***величины*** $v$ на сети $G=(I,U)$ назовем совокупность чисел $x = (x_{ij}, (i, j) \in U)$, которые удовлетворяют двум условиям:

$$0 \le x_{ij} \le d_{ij}\ ,\ (i, j) \in U, \tag{8.1}$$

$$\sum_{j \in I_i^+(U)} x_{ij} - \sum_{j \in I_i^-(U)} x_{ji} = \begin{cases} v, \text{ если } i = s, \\ -v, \text{ если } i = t, \\ 0, \text{ если } i \in I,\ i \ne s,\ i \ne t. \end{cases} \tag{8.2}$$

Здесь $I_i^+(U) = \{j \in I: \exists (i, j) \in U\}$ – множество вершин сети $G$, соседних справа для вершины $i$, $I_i^-(U) = \{j \in I: \exists (j, i) \in U\}$ – множество вершин сети $G$, соседних слева для вершины $i$. Число $x_{ij}$ называется ***дуговым потоком*** или потоком по дуге.

Условие (8.1) означает, что дуговой поток является неотрицательным и не должен превышать пропускной способности дуги. Условие вида (8.2) записывается для каждой вершины сети. Для заданного потока величина $v$ является некоторой константой. Левая часть равенства (8.2) для вершины $i \in I$, $i \ne s$, $i \ne t$ означает, что суммарный «вытекающий» из вершины $i$ поток и суммарный «поступающий» в вершину $i$ поток равны. Условие (8.2) называют ***условием баланса***, или условием неразрывности потока.

Пример 1. На сети, представленной на рис. 8.1, первое число, указанное на дуге есть пропускная способность, второе – величина дуго-

вого потока. Выяснить, образует ли совокупность дуговых потоков стационарный поток на сети.

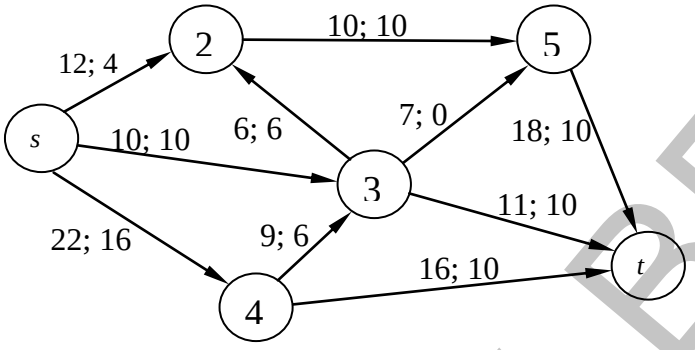


Рис. 8.1.

Проверка условия (8.1). На каждой дуге второе число, т.е. дуговой поток, является неотрицательным и не превосходит первого числа – пропускной способности дуги.

Проверка условия (8.2). Для вершины 2: выходящий поток равен 10 по дуге (2,5), суммарный входящий поток по дугам (s, 2) и (3,2) также равен 10. В вершине 3 имеем равенство:

$$x_{32} + x_{35} + x_{3t} - x_{s3} - x_{43} = 6 + 0 + 10 - 10 - 6 = 0.$$

Непосредственной проверкой убеждаемся, что в вершинах 4 и 5 также выполняется условие баланса. Для вершин $s$ и $t$ имеем равенства:

$x_{s2} + x_{s3} + x_{s4} = 4 + 10 + 16 = 30$, $\quad - x_{3t} - x_{4t} - x_{5t} = -10 - 10 - 10 = -30$.

Таким образом, условие (8.2) выполняется для всех вершин. Величина потока равна $v = 30$.

Пусть $I_1$ – подмножество $I$ такое, что $s \in I_1$, $t \notin I_1$. ***Разрезом*** на сети G, отделяющим источник $s$ от стока $t$ называется множество таких дуг, что начало каждой из них есть вершина из $I_1$, а конец – вершина из $\overline{I}_1 = I \setminus I_1$.

Согласно определению $I_1 \cap \overline{I}_1 = \varnothing$, $I_1 \cup \overline{I}_1 = I$, другими словами, каждая вершина попадает либо в $I_1$, либо в $\overline{I}_1$. Обозначать разрез будем как пару $(I_1, \overline{I}_1)$. Рассмотрим сеть на рис. 8.1. Пусть $I_1 = \{s, 4, 2\}$, $\overline{I}_1 = \{3, 5, t\}$. Тогда имеем разрез $(I_1, \overline{I}_1) = \{(s, 3), (2,5), (4, 3), (4, t)\}$.

*Лемма 8.1*. Пусть $G=(I,U)$ – сеть с источником $s$ и стоком $t$. Любой ориентированный маршрут (путь) из источника в сток содержит хотя бы одну дугу произвольного разреза.

*Доказательство*. Пусть $s$, $i_1$, $i_2$, …, $i_k$, $t$ – последовательность вершин некоторого ориентированного маршрута (пути). $(I_1, \bar{I}_1)$ – произвольный разрез, отделяющий источник $s$ от стока $t$. Если вершина $i_1 \notin I_1$, то $i_1 \in \bar{I}_1$, тогда дуга $(s, i_1)$ – дуга разреза, и лемма доказана. В противном случае для вершины $i_2$ повторим рассуждения. Если все вершины $i_1, i_2, \ldots, i_k$ попали в $I_1$, то разрезу принадлежит дуга $(i_k, t)$, поскольку $t \in \bar{I}_1$.

*Следствие*. Пусть $G=(I,U)$ – сеть с источником $s$ и стоком $t$, и $(I_1, \bar{I}_1)$– произвольный разрез, отделяющий источник $s$ от стока $t$. Если из $G$ удалить все дуги разреза, то в новой сети не существует путей из вершины $s$ в вершину $t$.

***Пропускной способностью*** разреза называется число

$$d(I_1, \bar{I}_1) = \sum_{(i,j) \in (I,\bar{I}_1)} d_{ij}$$

*Лемма 8.2*. Пусть $x = (x_{ij}, (i,j) \in U)$ – произвольный поток величины $v$ на сети $G = (I,U)$, и $(I_1, \bar{I}_1)$ – произвольный разрез, отделяющий источник $s$ от стока $t$. Тогда величина потока не превосходит пропускной способности разреза, т.е.:

$$v \le d(I_1, \bar{I}_1). \qquad (8.3)$$

*Доказательство*. Для потока $x$ в любой вершине выполняется условие (8.2). Выберем равенства для вершин множества $I_1$ и сложим их почленно. Слагаемое $x_{ij}$ , где $i \in I_1$ и $j \in I_1$, присутствует в левой части двух равенств: в равенстве для вершины $i$ со знаком «плюс», а в равенстве для вершины $j$ со знаком «минус». При сложении они в сумме дают нуль. Останутся слагаемые $x_{ij}$ со знаком плюс, если $i \in I_1$ и $j \in \bar{I}_1$, и слагаемые $x_{ij}$ со знаком минус, если $i \in \bar{I}_1$ и $j \in I_1$. Значит, получим:

$$\sum_{i \in I_1,\, j \in \bar{I}_1} x_{ij} - \sum_{i \in \bar{I}_1,\, j \in I_1} x_{ij} = v. \qquad (8.4)$$

Поскольку $x_{ij} \ge 0$, то

$$\sum_{i\in I_1,\, j\in\overline{I_1}} x_{ij} - \sum_{i\in\overline{I_1},\, j\in I_1} x_{ij} \le \sum_{i\in I_1,\, j\in\overline{I_1}} x_{ij} = \sum_{(i,j)\in(I,\overline{I_1})} x_{ij}. \qquad (8.5)$$

Далее в силу того, что $x_{ij} \le d_{ij}$, имеем

$$\sum_{(i,j)\in(I,\overline{I_1})} x_{ij} \le \sum_{(i,j)\in(I,\overline{I_1})} d_{ij}. \qquad (8.6)$$

Из цепочки равенств и неравенств (8.4)-(8.6) получаем (8.3).

**2. Постановка задачи о максимальном потоке. Теорема Форда-Фалкерсона**. Задана сеть $G=(I,U)$, в которой выделены две вершины $s$ и $t$ – источник и сток и заданы пропускные способности дуг. Найти поток из вершины $s$ в вершину $t$ максимальной величины. Таким образом, математическая постановка задачи имеет вид:

$$\left.\begin{array}{l} v \to \max, \\ \displaystyle\sum_{j\in I_i^+(U)} x_{ij} - \sum_{j\in I_i^-(U)} x_{ji} = \begin{cases} v, \text{ если } i = s, \\ -v, \text{ если } i = t, \\ 0, \text{ если } i\in I,\ i \ne s,\ i \ne t. \end{cases} \\ 0 \le x_{ij} \le d_{ij},\ (i,j)\in U. \end{array}\right\} \qquad (8.7)$$

Данная задача является задачей линейного программирования. Но применение общих методов решения задач ЛП к задаче (8.7) не рационально, поскольку матрица условий является слабо заполненной. Математиками Фордом (Ford L.R.) и Фалкерсоном (Fulkerson D.R.) был предложен новый способ решения. Он основа на следующей теореме, которая носит имя ее авторов.

*Теорема Форда-Фалкерсона*. Пусть $G=(I,U)$ – сеть с источником $s$ и стоком $t$. Величина $v$ максимального потока в сети $G$ равна минимальной пропускной способности из всех разрезов, отделяющих источник $s$ от стока $t$.

*Доказательство*. Пусть $x = (x_{ij}, (i,j)\in U)$ – максимальный поток, и его величина равна $v$. Заметим, что такой поток существует. Почему? Во-первых, в сети $G$ существует, по крайней мере, нулевой поток: $(x_{ij} = 0, (i,j)\in U)$ с нулевой величиной. Во-вторых, согласно лемме 8.2 величина потока ограничена сверху пропускной способностью любого разреза.

По максимальному потоку построим разрез $(I_1, \overline{I_1})$. Множество

$I_1$ формируем согласно следующим правилам:

а) $s \in I_1$ (правило следует из определения разреза);

б) если вершина $i \in I_1$ и существует дуга $(i, j) \in U$, где $j \notin I_1$ и $x_{ij} < d_{ij}$, то вершину $j$ включаем в множество $I_1$ (см. рис.8.1);

в) если вершина $i \in I_1$ и существует дуга $(j, i) \in U$, где $j \notin I_1$ и $x_{ij} > 0$, то вершину $j$ включаем в множество $I_1$ (см. рис. 8.2).

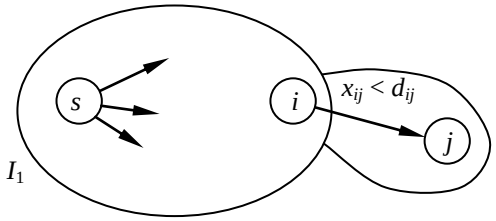


Рис. 8.1.

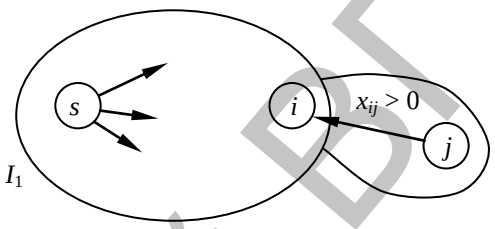


Рис. 8.2.

Затем находим множество $\overline{I}_1$. Покажем, что $t \in \overline{I}_1$. Предположим противное: $t \in I_1$. Тогда среди элементов множества $I_1$ можно выделить последовательность вершин

$$s, i_1, i_2, \ldots, i_k, t, \tag{8.8}$$

в которой две соседние вершины являются смежными. Другими словами, последовательность (8.8) определяет (неориентированную) простую цепь из $s$ в $t$, на прямых дугах которой дуговой поток не является насыщенным (правило б)), а на обратных дугах не является нулевым (правило в)). Найдем положительное число

$$\varepsilon = \min \{\varepsilon_1, \varepsilon_2\}, \tag{8.9}$$

где

$\varepsilon_1 = \min \{ d_{ij} - x_{ij}$ , если $(i, j)$ – прямая дуга в (8.8)$\}$,

$\varepsilon_2 = \min \{ x_{ij}$ , если $(i, j)$ – обратная дуга в (8.8)$\}$.

Сформируем совокупность чисел $\overline{x} = (\overline{x}_{ij}, (i, j) \in U)$:

$$\overline{x}_{ij} = \begin{cases} x_{ij} + \varepsilon, \text{ если } (i, j) \text{ – прямая дуга маршрута,} \\ x_{ij} - \varepsilon, \text{ если } (i, j) \text{ – обратная дуга маршрута} \\ x_{ij} \text{ , если } (i, j) \text{ не является дугой маршрута.} \end{cases}$$

Очевидно, что для чисел $\overline{x}_{ij}, (i, j) \in U$, выполняются условия (8.1). Проверим выполнение условий (8.2). Если $i = s$, и маршрут (8.8) включает дугу $(s, i_1)$ , то уравнение имеет вид

$$\sum_{j\in I_s^+(U)} \bar{x}_{sj} - \sum_{j\in I_s^-(U)} \bar{x}_{js} = \sum_{j\in I_s^+(U^*)} \bar{x}_{sj} + \bar{x}_{si_1} - \sum_{j\in I_s^-(U)} \bar{x}_{js} =$$

$$= \sum_{j\in I_s^+(U^*)} x_{sj} + x_{si_1} + \varepsilon - \sum_{j\in I_s^-(U)} x_{js} = \sum_{j\in I_s^+(U)} x_{sj} - \sum_{j\in I_s^-(U)} x_{js} + \varepsilon = v + \varepsilon,$$

где $I_s^+(U^*) = I_s^+(U) \setminus i_1$. Аналогичный результат получаем, если маршрут (8.8) включает дугу $(i_1, s)$ .

Если $i = t$, и маршрут (8.8) включает дугу $(i_k, t)$ , то уравнение имеет вид

$$\sum_{j\in I_t^+(U)} \bar{x}_{tj} - \sum_{j\in I_t^-(U^*)} \bar{x}_{jt} - \bar{x}_{i_k t} = \sum_{j\in I_t^+(U)} x_{tj} - \sum_{j\in I_t^-(U^*)} x_{jt} - (x_{i_k t} + \varepsilon) =$$

$$= \sum_{j\in I_t^+(U)} x_{tj} - \sum_{j\in I_t^-(U)} x_{jt} - \varepsilon = -v - \varepsilon,$$

где $I_t^-(U^*) = I_t^-(U) \setminus i_k$. Аналогичный результат получаем, если маршрут (8.8) включает дугу $(t, i_k)$ .

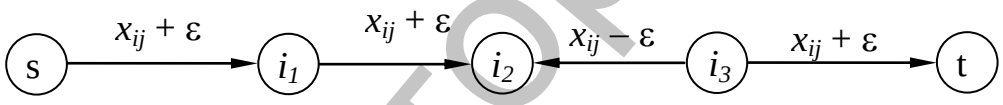


Рис. 8.3.

Для вершин, не входящих в последовательность (8.8), уравнения на совокупности $\bar{x}$ совпадает с уравнением баланса на потоке $x$. Для любой вершины из множества $\{i_1, i_2, \ldots, i_k\}$ в уравнении присутствуют ровно два слагаемых, соответствующих дугам маршрута. При этом возможны три различные ситуации: обе дуги выходят из вершины, обе дуги входят в вершину или одна является выходящей, а вторая – входящей (см. рис. 8.3). Непосредственный подсчет показывает, что условия баланса имеют место для любой из этих вершин.

Следовательно, $\bar{x} = (\bar{x}_{ij}, (i, j) \in U)$ – поток на сети, величина которого равна $v + \varepsilon > v$, что противоречит максимальности потока $x = (x_{ij}, (i, j) \in U)$.

Итак, $t \notin I_1$, а значит, $(I_1, \bar{I}_1)$ – разрез. По построению дуговой поток на дугах разреза насыщенный, т.е. если $(i, j) \in (I_1, \bar{I}_1)$ , то $x_{ij} = d_{ij}$. Если дуга $(i, j)$ такова, что $i \in \bar{I}_1$ , $j \in I_1$, то $x_{ij} = 0$. Поэтому из (8.4) и

(8.5) следует

$$v = \sum_{i \in I_1,\, j \in \bar{I}_1} x_{ij} - \sum_{i \in \bar{I}_1,\, j \in I_1} x_{ij} = \sum_{i \in I_1,\, j \in \bar{I}_1} d_{ij} - 0 = d(I_1, \bar{I}_1). \quad (8.10)$$

Доказано, что величина потока равна пропускной способности построенного разреза.

Осталось показать, что разреза с меньшей пропускной способностью не существует. От противного. Пусть имеется такой разрез $(I_2, \bar{I}_2)$, что $d(I_2, \bar{I}_2) < d(I_1, \bar{I}_1)$. Тогда, используя лемму 8.2 и выше доказанное равенство (8.10), имеем

$$v \le d(I_2, \bar{I}_2) < d(I_1, \bar{I}_1) = v.$$

Полученное неравенство $v < v$ говорит, что построенный разрез имеет минимальную пропускную способность. Теорема доказана.

Последовательность (8.8), для которой число ε, вычисленное по правилу (8.9), является положительным, называется ***увеличивающей цепью***.

*Следствие* из теоремы Форда-Фалкерсона. Если в сети построен максимальный поток, то в ней нет увеличивающих цепей.

**3. Алгоритм Форда-Фалкерсона**. Основная идея алгоритма следует из теоремы Форда-Фалкерсона: чтобы найти максимальный поток, надо построить минимальный разрез (найти на сети узкое место, которое не дает увеличить поток). Считаем, что начальный поток на сети задан, в частности им может быть нулевой поток.

Каждая итерация состоит из двух частей. На первом этапе строится увеличивающий путь (если его построить не удается, то в сети построен максимальный поток), а на втором этапе осуществляется увеличение потока. И первое, и второе реализуется с помощью присвоения вершинам сети специальных меток. В ходе алгоритма вершина может быть в одном из трех состояний: непомеченная (это начальное состояние всех вершин), помеченная, помеченная и просмотренная. Метка вершины содержит информацию о формируемом пути (первая часть метки) и о величине потока, который можно дополнительно пропустить по этому пути из источника до рассматриваемой вершины.

**I этап**. Все вершины не помечены.

*Шаг 1*. Вершина $s$ получает метку $[\varnothing; \varepsilon_s = \infty]$, что означает, что у вершины $s$ нет предшествующих вершин, и мощность источни-

ка неограниченна.

*Шаг 2*. Пусть имеется некоторое множество помеченных вершин. Произвольно выбираем из них вершину $i$, и начинаем *процесс* ее *просмотра*. Определяем все смежные с ней, но еще непомеченные вершины. Пусть вершина $j$ – одна из них. Вершина $j$ получает метку в двух следующих случаях:

а) если существует дуга $(i, j) \in U$, причем $x_{ij} < d_{ij}$, то метка вершины имеет вид $[i^{+}; \varepsilon_j]$. Первая часть метки означает, что вершина $j$ помечена через вершину $i$, причем дуга $(i, j)$ является прямой дугой формируемого пути. Вторая часть метки вычисляется по правилу

$$\varepsilon_j = \min \{ \varepsilon_i,\ d_{ij} - x_{ij} \},$$

где $\varepsilon_i$ – вторая часть метки вершины $i$.

б) если существует дуга $(j, i) \in U$, причем $x_{ij} > 0$, то метка вершины имеет вид $[i^{-}; \varepsilon_j]$. Первая часть метки означает, что вершина $j$ помечена через вершину $i$, причем дуга $(j, i)$ является обратной дугой формируемого пути. Вторая часть метки вычисляется по правилу

$$\varepsilon_j = \min \{ \varepsilon_i,\ x_{ij} \}.$$

После того, как для всех вершин, смежных с вершиной $i$, будет решен вопрос с метками, вершина $i$ переходит в разряд помеченных и просмотренных.

При выполнении шага 2 одна вершина становится просмотренной, какие-то вершины становятся помеченными. Поскольку рассматриваемые сети являются конечными, то на шаге 2 возможны два исхода: 1) помечена вершина $t$; 2) все помеченные вершины просмотрены, но вершина $t$ оказалась непомеченной. В первом случае переходим к II этапу. Во втором – процесс решения закончен, в сети построен максимальный поток. Чтобы найти его величину, найдем пропускную способность разреза $(I_1, \bar{I}_1)$, где $I_1$ – множество помеченных и просмотренных вершин, $\bar{I}_1$ – множество непомеченных вершин.

**II этап**. Строим увеличивающую цепь из вершины $s$ в вершину $t$. Первая часть метки вершины $t$ указывает вершину $i_k$, которая в цепи предшествует вершине $t$. Затем первая часть метки вершины $i_k$ указывает вершину $i_{k-1}$, которая предшествует вершине $i_k$, и так далее, пока первая часть метки некоторой вершины не укажет на вершину $s$. Выбирается направление в цепи: из вершины $s$ в вершину $t$. На пря-

мых дугах цепи дуговые потоки увеличиваются на величину $\varepsilon_t$, а на обратных дугах уменьшается на эту величину.

Переход к новой итерации.

Пример 2. На сети, представленной на рис. 8.1, найти максимальный поток.

Итерация 1. **I этап**. Все вершины не помечены.

*Шаг 1*. Вершина $s$ получает метку $[\varnothing; \varepsilon_s = \infty]$.

*Шаг 2*. Операции этого шага поместим в таблицу:

| Просмотр вершины $s$ | | Помеченные вершины | |
|---|---|---|---|
| Непомеченные смежные: 2, 3, 4 | 2: $[s^+; \varepsilon_2]$, $\varepsilon_2 = \min(\infty; 12 - 4) = 8$<br>3: пометить нельзя, т.к. $x_{s3} = d_{s3} = 10$<br>4: $[s^+; \varepsilon_4]$, $\varepsilon_4 = \min(\infty; 22 - 16) = 6$ | $s$ : $[\varnothing; \infty]$<br>2: $[s^+; 8]$,<br>4: $[s^+; 6]$, | просмотрена |
| Просмотр вершины 2 | | Помеченные вершины | |
| Непомеченные смежные: 3, 5 | 3: $[2^-; \varepsilon_3]$, $\varepsilon_2 = \min(8; 6) = 6$.<br>5: пометить нельзя, т.к. $x_{25} = d_{25} = 10$. | $s$ : $[\varnothing; \infty]$<br>2: $[s^+; 8]$,<br>4: $[s^+; 6]$,<br>3: $[2^-; 6]$ | просмотрена<br>просмотрена |
| Просмотр вершины 4 | | Помеченные вершины | |
| Непомеченные смежные: $t$ | $t$: $[4^+; \varepsilon_t]$, $\varepsilon_t = \min(6; 16 - 10) = 6$. | $s$ : $[\varnothing; \infty]$,<br>2: $[s^+; 8]$,<br>4: $[s^+; 6]$,<br>3: $[2^-; 6]$,<br>$t$: $[4^+; 6]$ | просмотрена<br>просмотрена<br>просмотрена |

**II этап**. Увеличивающая цепь проходит через вершины $s$ , 4, $t$. На дугах ($s$, 4) и (4, $t$) дуговые потоки увеличиваем на 6 единиц. Полу-

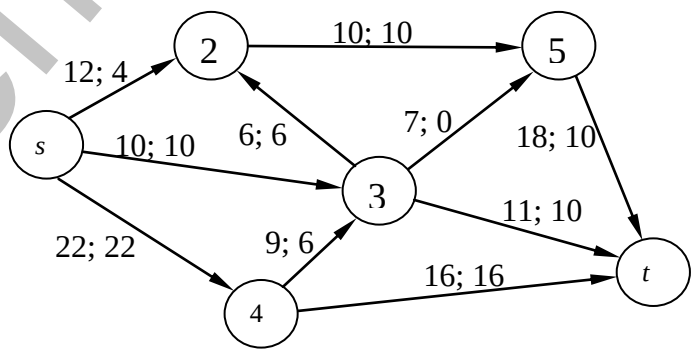


Рис. 8.4.

ченный новый поток указан на рис. 8.4.

Итерация 2. **I этап**. Все вершины не помечены.

*Шаг 1*. Вершина $s$ получает метку $[\varnothing; \varepsilon_s = \infty]$.

*Шаг 2*.

| Просмотр вершины $s$ | | Помеченные вершины | |
|---|---|---|---|
| Непомеченные смежные: 2, 3, 4 | 2: $[s^+; \varepsilon_2]$, $\varepsilon_2 = \min(\infty; 12 - 4) = 8$<br>3: пометить нельзя, т.к. $x_{s3} = d_{s3} = 10$<br>4: пометить нельзя, т.к. $x_{s4} = d_{s4} = 22$ | $s$ : $[\varnothing; \infty]$<br>2: $[s^+; 8]$, | просмотрена |
| Просмотр вершины 2 | | Помеченные вершины | |
| Непомеченные смежные: 3, 5 | 3: $[2^-; \varepsilon_3]$, $\varepsilon_2 = \min(8; 6) = 6$.<br>5: пометить нельзя, т.к. $x_{25} = d_{25} = 10$. | $s$: $[\varnothing; \infty]$<br>2: $[s^+; 8]$,<br>3: $[2^-; 6]$ | просмотрена<br>просмотрена |
| Просмотр вершины 3 | | Помеченные вершины | |
| Непомеченные смежные: 4, 5, $t$ | 4: $[3^-; \varepsilon_4]$, $\varepsilon_4 = \min(6; 6) = 6$<br>5: $[3^+; \varepsilon_5]$, $\varepsilon_5 = \min(6; 7 - 0) = 6$<br>$t$: $[3^+; \varepsilon_t]$, $\varepsilon_t = \min(6; 11 - 10) = 1$. | $s$: $[\varnothing; \infty]$<br>2: $[s^+; 8]$,<br>3: $[2^-; 6]$,<br>4: $[3^-; 6]$,<br>5: $[3^+; 6]$,<br>$t$: $[3^+; 1]$ | просмотрена<br>просмотрена<br>просмотрена |

**II этап**. Увеличивающая цепь проходит через вершины $s$ , 2, 3, $t$. На дугах ($s$, 2) и (3, $t$) дуговые потоки увеличиваем на 1 единицу, на дуге (3, 2) дуговой поток уменьшаем. Полученный новый поток указан на рис. 8.5.

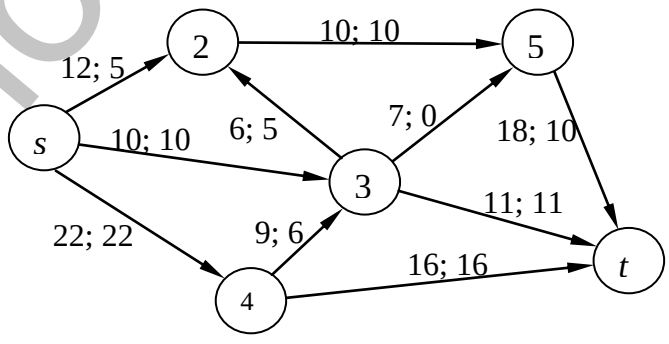


Рис. 8.5.

Итерация 3. **I этап**. Все вершины не помечены.

*Шаг 1*. Вершина $s$ получает метку $[\varnothing; \varepsilon_s = \infty]$.

*Шаг 2*.

| Просмотр вершины $s$ | | Помеченные вершины | |
|---|---|---|---|
| Непомеченные смежные: 2, 3, 4 | 2: $[s^+; \varepsilon_2]$, $\varepsilon_2 = \min(\infty; 12 - 5) = 7$<br>3: пометить нельзя, т.к. $x_{s3} = d_{s3} = 10$<br>4: пометить нельзя, т.к. $x_{s4} = d_{s4} = 22$ | $s$ : [∅; ∞]<br>2: $[s^+; 7]$, | просмотрена |
| Просмотр вершины 2 | | Помеченные вершины | |
| Непомеченные смежные: 3, 5 | 3: $[2^-; \varepsilon_3]$, $\varepsilon_2 = \min(7; 5) = 5$.<br>5: пометить нельзя, т.к. $x_{25} = d_{25} = 10$. | $s$: [∅; ∞]<br>2: $[s^+; 7]$,<br>3: $[2^-; 5]$ | просмотрена<br>просмотрена |
| Просмотр вершины 3 | | Помеченные вершины | |
| Непомеченные смежные: 4, 5, $t$ | 4: $[3^-; \varepsilon_4]$, $\varepsilon_4 = \min(5; 6) = 5$<br>5: $[3^+; \varepsilon_5]$, $\varepsilon_5 = \min(5; 7 - 0) = 5$<br>$t$: пометить нельзя, т.к. $x_{3t} = d_{3t} = 11$ | $s$: [∅; ∞]<br>2: $[s^+; 7]$,<br>3: $[2^-; 5]$,<br>4: $[3^-; 5]$,<br>5: $[3^+; 5]$ | просмотрена<br>просмотрена<br>просмотрена |
| Просмотр вершины 4 | | Помеченные вершины | |
| Непомеченные смежные: $t$ | $t$: пометить нельзя, т.к. $x_{4t} = d_{4t} = 16$ | $s$: [∅; ∞]<br>2: $[s^+; 7]$,<br>3: $[2^-; 5]$,<br>4: $[3^-; 5]$,<br>5: $[3^+; 5]$, | просмотрена<br>просмотрена<br>просмотрена<br>просмотрена |
| Просмотр вершины 5 | | Помеченные вершины | |
| Непомеченные смежные: $t$ | $t$: $[5^+; \varepsilon_t]$, $\varepsilon_t = \min(5; 18 - 10) = 5$. | $s$: [∅; ∞]<br>2: $[s^+; 7]$,<br>3: $[2^-; 5]$,<br>4: $[3^-; 5]$,<br>5: $[3^+; 5]$,<br>$t$: $[5^+; 5]$ | просмотрена<br>просмотрена<br>просмотрена<br>просмотрена<br>просмотрена |

**II этап**. Увеличивающая цепь проходит через вершины $s$, 2, 3, 5, $t$. На дугах ($s$, 2), (3, 5) и (5, $t$) дуговые потоки увеличиваем на 5 единиц, на дуге (3, 2) дуговой поток уменьшаем. Полученный новый поток указан на рис. 29.

Итерация 4. **I этап**. Все вершины не помечены.

*Шаг 1*. Вершина $s$ получает метку $[\varnothing; \varepsilon_s = \infty]$.

*Шаг 2*.

| Просмотр вершины $s$ | | Помеченные вершины | |
|---|---|---|---|
| Непомеченные смежные: 2, 3, 4 | 2: $[s^+; \varepsilon_2]$, $\varepsilon_2 = \min(\infty; 12 - 10) = 2$<br>3: пометить нельзя, т.к. $x_{s3} = d_{s3} = 10$<br>4: пометить нельзя, т.к. $x_{s4} = d_{s4} = 22$ | $s$ : $[\varnothing; \infty]$<br>2: $[s^+; 2]$, | просмотрена |
| Просмотр вершины 2 | | Помеченные вершины | |
| Непомеченные смежные: 3, 5 | 3: пометить нельзя, т.к. $x_{32} = 0$<br>5: пометить нельзя, т.к. $x_{25} = d_{25} = 10$. | $s$: $[\varnothing; \infty]$<br>2: $[s^+; 7]$ | просмотрена<br>просмотрена |

Все помеченные вершины просмотрены. Вершина $t$ не помечена. Поток, указанный на рис. 8.6 является максимальным. Определим его величину. Последней итерации соответствует разрез $(I_1; \bar{I}_1)$, где $I_1 = \{s, 2\}$, $\bar{I}_1 = \{3, 4, 5, t\}$, $(I_1; \bar{I}_1) = \{(s, 3), (s, 4), (2, 5)\}$. Его пропускная способность равна

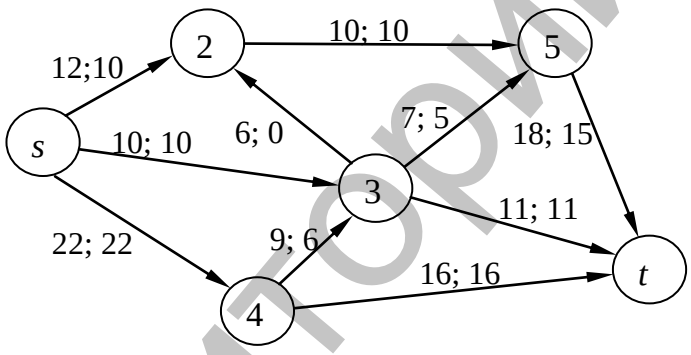


Рис. 8.6.

$$d(I_1; \bar{I}_1) = 10 + 22 + 10 = 42 = v.$$

## § 9. Некоторые обобщения задачи о максимальном потоке

Рассматриваются нестандартные задачи о максимальном потоке.

**1. Задача о максимальном потоке с несколькими источниками и несколькими стоками**. Пусть $G=(I, U)$ – ориентированный граф без петель и без параллельных дуг. Каждой дуге $(i, j)$ поставлена в соответствие пропускная способность $d_{ij} > 0$. Среди вершин множества $I$ выделены вершины: $s_1, s_2, \ldots, s_n$ – источники и вершины $t_1, t_2, \ldots, t_m$ – стоки. Мощность источников не ограничена, стоки могут

принять любой поток. Требуется найти стационарный поток на сети, величина которого максимальна.

Решение этой задачи сводится к классической задаче о максимальном потоке (§ 8) применением следующего стандартного приема. Заданную сеть $G=(I,U)$ расширяют. К множеству вершин добавляют две новые вершины: $s$ – общий источник и $t$ – общий сток. Множество дуг пополняется дугами, которые связывают общий источник с заданными источниками, и дугами, связывающие заданные стоки с общим стоком. Таким образом, получаем сеть $\bar{G} = (\bar{I},\bar{U})$, где

$$\bar{I} = I \cup \{s, t\}, \quad \bar{U} = U \cup \{(s, s_i), i = \overline{1,n}\} \cup \{(t_j, t), j = \overline{1,m}\}.$$

Новые дуги не должны стать на новой сети «узким» местом, поэтому пропускные способности новых дуг неограничены: $d_{ss_i} = \infty$, $d_{t_j t} = \infty$.

На расширенной сети $\bar{G} = (\bar{I},\bar{U})$ с одним источником $s$ и одним стоком $t$ решают задачу о максимальном потоке. Величина потока $v$ будет равна пропускной способности минимального разреза, в который очевидно войдут только дуги исходной сети. Кроме этого, понятно, что

$$v = \sum_{i=1}^{n} x_{ss_i} = \sum_{j=1}^{m} x_{t_j t}\ .$$

**2. Задача с нижними границами на пропускные способности дуг**. Задана сеть $G=(I,U)$, в которой выделены две вершины $s$ и $t$ – источник и сток Каждой дуге $(i, j)$ поставлены соответствие два неотрицательных числа: $d_{*ij}$ и $d^*_{ij}$ – нижняя и верхняя пропускные способности дуги. Требуется найти поток из вершины $s$ в вершину $t$ максимальной величины, если выполняются условия:

$$\sum_{j\in I_i^+(U)} x_{ij} - \sum_{j\in I_i^-(U)} x_{ji} = \begin{cases} v, \text{ если } i = s, \\ -v, \text{ если } i = t, \\ 0, \text{ если } i\in I,\ i \neq s,\ i \neq t, \end{cases}$$

$$d_{*ij} \le x_{ij} \le d^*_{ij},\ (i, j) \in U\ .$$

Для решения этой задачи применим модифицированный алгоритм Форда-Фалкерсона. Изменим в алгоритме (см. § 8) шаг 2б).

*Шаг 2*. б) если существует дуга $(j, i)\in U$, причем $x_{ij} > d_{*ij}$, то метка вершины имеет вид $[i^-; \varepsilon_j]$. Первая часть метки означает, что вершина $j$ помечена через вершину $i$, причем дуга $(j, i)$ является об-

ратной дугой формируемого пути. Вторая часть метки вычисляется по правилу

$$\varepsilon_j = \min \{ \varepsilon_i,\ x_{ij} - d_{*ij} \}.$$

Кроме этого заметим, что для данной задачи отдельно требуется решить вопрос о начальном допустимом потоке, поскольку нулевой поток допустимым не является.

**3. Задача о максимальном потоке в смешанных сетях**. Напомним, что граф называется смешанным, если в нем есть и дуги, и ребра. Поток по дуге $(i, j)$ имеет единственное направление: из вершины $i$ в вершину $j$. Наличие ребра $\{i, j\}$ в графе означает, что поток может двигаться как в одном направлении, так и в обратном направлении (понятно, что не одновременно). Потоки и по дугам, и по ребрам не превышают заданной пропускной способности дуги или ребра. Как и прежде ставиться задача нахождения максимального потока. Чтобы можно было применить известный алгоритм, преобразуем сеть. Каждое ребро $\{i, j\}$ заменим двумя дугами $(i, j)$ и $(j, i)$, положив пропускную способность каждого равной пропускной способности ребра: $d_{ij} = d_{ji}$ . На расширенной ориентированной сети с помощью алгоритма Форда-Фалкерсона решаем задачу. Затем в максимальном потоке просматриваем дуговые потоки, которые соответствуют ребрам.

Если для ребра $\{i, j\}$ по дугам $(i, j)$ и $(j, i)$, имеем потоки такие, что $x_{ij} > x_{ji}$ , то поток пойдет по ребру из вершины $i$ в вершину $j$ , и его величина будет равна $x_{ij} - x_{ji}$ . Если же $x_{ji} > x_{ij}$, то поток по ребру пойдет из вершины $j$ в вершину $i$ , и его величина будет равна $x_{ji} - x_{ij}$ .

**4. Задача о максимальном потоке в сетях с пропускными способностями вершин**. Пусть в заданной сети $G=(I,U)$ с источником $s$ и стоком $t$ задано множество $I_o \subset I$ , каждой вершине которого поставлено в соответствие положительное число $a_i$, $i \in I_o$, – пропускная способность вершины. Требуется, как и прежде, найти поток из вершины $s$ в вершину $t$ максимальной величины, если выполняются условия (8.7), (8.8) и дополнительные условия для вершин с ограниченной пропускной способностью:

$$\sum_{j \in I_i^+(U)} x_{ij} \ \le a_i,\ i \in I_o\ .$$

Как и в п.1 решение задачи сводится к классической задаче о максимальном потоке на расширенной сети. Вместо каждой вершины $k \in I_o$ вводят две вершины $k_1$ и $k_2$ и дугу $(k_1, k_2)$, которой приписывают пропускную способность, равную пропускной способности вершины: $d_{k_1k_2} = a_k$. Если две вершины $k$ и $j$ из множества $I_o$ являются смежными,

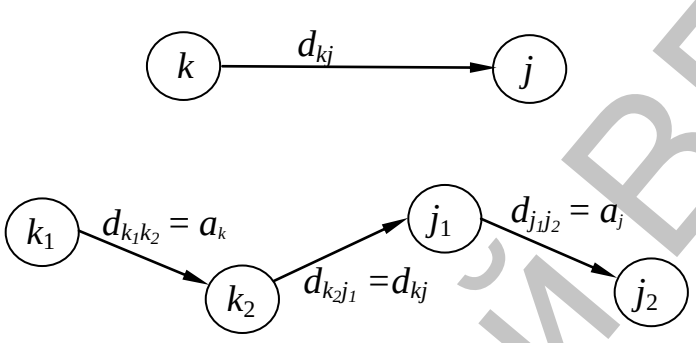


Рис. 9.1.

то дуга $(k, j)$ преобразуется в дугу $(k_2, j_1)$ (см. рис 9.1).

## § 10. Задача о назначениях

**1. Классическая задача о назначении.** В книге [2, т.1] описана конкретная ситуация, приводящая к так называемой задаче о назначениях.

**Задача фирмы «Электрон».** Фирма, выпускающая сложную электронную аппаратуру, получила заказ объемом в несколько тысяч новых изделий, собирающихся из отдельных блоков. Руководство фирмы приняло решение разместить заказы на изготовление $n$ блоков и выбрало $n$ фирм-поставщиков, которые зарекомендовали себя ранее как производители высококачественной продукции. Каждый заказ настолько велик, что фирма-поставщик не может выполнить более одного заказа. Каждому поставщику предложено определить стоимость выполнения заказа, т.е. цену, по которой он готов поставить фирме блоки. Некоторые из запрошенных цен настолько велики, что явно свидетельствуют о нежелании отдельных поставщиков принять заказы на определенные блоки. Располагая этой информацией, фирма электронной аппаратуры должна заключить $n$ контрактов на поставку

ей $n$ видов блоков, минимизировав при этом свои общие затраты на приобретение комплектующих узлов со стороны.

**Математическая модель задачи.** Пусть имеется $n$ работ и $n$ исполнителей, каждый из которых может выполнить любую работу. Известны выплаты, которые следует сделать каждому исполнителю за выполнение любой имеющейся работы. Ставится задача распределения исполнителей по работам таким образом, чтобы каждая работа выполнялась только одним исполнителем, каждый исполнитель выполнял только одну работу, и выплаты при этом были бы минимальны.

Введем обозначения. Через $I = \{1, 2,\ldots, n\}$ обозначим множество работ, через $J = \{1, 2,\ldots, n\}$ – множество исполнителей, через $c_{ij}$ – «стоимость» назначения на $i$-ую работу $j$-го исполнителя. Переменные в задаче определим следующим образом:

$$x_{ij} = \begin{cases} 1, \text{ если на } i\text{-ую работу назначен } j\text{-ый исполнитель,} \\ 0 \text{ в противном случае.} \end{cases} \quad (9.1)$$

Ограничения на переменные определяются тем, что каждая работа должна быть выполнена одним исполнителем:

$$\sum_{j \in J} x_{ij} = 1, \quad i \in I, \quad (9.2)$$

и каждый исполнитель выполняет только одну работу:

$$\sum_{i \in I} x_{ij} = 1, \quad j \in J. \quad (9.3)$$

Совокупность переменных $x = (x_{11}, x_{12},\ldots, x_{nn})$, которая удовлетворяет условиям (9.1)-(9.3), назовем ***назначением***. Очевидно, что суммарные выплаты, которые следует минимизировать, задаются функцией:

$$\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \to \min. \quad (9.4)$$

На практике встречаются задачи о назначениях, в которых элемент $c_{ij}$ выражает эффективность назначения на $i$-ую работу $j$-го исполнителя. Тогда целевая функция есть суммарная эффективность назначения, а значит, следует найти ее максимальное значение

$$\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \to \max \quad (9.5)$$

при тех же ограничениях (9.1)-(9.3).

Далее будем рассматривать задачу (9.1)-(9.4).

Совокупность $x^o = (x_{11}^o, x_{12}^o, \ldots, x_{nn}^o)$, удовлетворяющую условиям (9.1)-(9.3), на которой функция (9.4) принимает минимальное значение, назовем ***оптимальным назначением***.

Коэффициенты целевой функции (9.4) удобно задать в виде матрицы $n$-го порядка

$$C = (c_{ij},\ i \in I,\ j \in J).$$

Выполним операцию приведения матрицы $C$. Для этого в каждой строке выбираем минимальный элемент

$$u_i = \min_{j \in J} c_{ij},\ i \in I,$$

и вычитаем его из каждого элемента этой строки. В результате получим матрицу $C_1$, в каждой строке которой имеется по крайней мере один нулевой элемент. Затем в каждом столбце $j$ матрицы $C_1$ находим минимальный элемент $v_j$, и вычитаем его из каждого элемента столбца. В результате получим матрицу $C_2$, у которой в каждом столбце и каждой строке имеется хотя бы один нулевой элемент. Числа $u_i$ и $v_j$, которые вычитались, называются *константами приведения*.

*Лемма 1*. Оптимальное назначение в задаче (9.1)-(9.4) является оптимальным в задаче (9.1)-(9.3) с целевой функцией

$$\sum_{i \in I} \sum_{j \in J} (c_{ij} - u_i - v_j)\, x_{ij}\,, \qquad (9.6)$$

где $u_i$ и $v_j$ – константы приведения матрицы $C$. И обратно, оптимальное назначение в задаче (9.1)-(9.3), (9.6) является оптимальным для задачи (9.1)-(9.4).

*Доказательство*. Преобразуем сумму (9.6), раскрыв скобки и перегруппировав слагаемые,

$$\sum_{i \in I} \sum_{j \in J} (c_{ij} - u_i - v_j)\, x_{ij} = \sum_{i \in I} \sum_{j \in J} (c_{ij}\, x_{ij} - u_i\, x_{ij} - v_j\, x_{ij}) =$$

$$= \sum_{i \in I} \sum_{j \in J} c_{ij}\, x_{ij} - \sum_{i \in I} \sum_{j \in J} u_i\, x_{ij} - \sum_{i \in I} \sum_{j \in J} v_j\, x_{ij} =$$

$$= \sum_{i \in I} \sum_{j \in J} c_{ij}\, x_{ij} - \sum_{i \in I} u_i \sum_{j \in J} x_{ij} - \sum_{j \in J} v_j \sum_{i \in I} x_{ij}$$

Далее учтем ограничения (9.2) и (9.3):

$$\sum_{i\in I}\sum_{j\in J}(c_{ij}-u_i-v_j)\,x_{ij}=\sum_{i\in I}\sum_{j\in J}c_{ij}\,x_{ij}-\sum_{i\in I}u_i-\sum_{j\in J}v_j\,.$$

Из последнего равенства следует, что для любого назначения, в том числе и оптимального, значения целевых функций рассматриваемых задач отличаются на одну и ту же константу. Другими словами минимальное значение одной функции достигается на том же назначении, что и минимальное значение другой. Лемма доказана.

*Лемма 2*. Если в задаче (9.1)-(9.4) все элементы матрицы $C$ неотрицательны, и имеется такое назначение $x = (x_{11}, x_{12}, \ldots, x_{nn})$, что

$$\sum_{i\in I}\sum_{j\in J}c_{ij}\,x_{ij}=0, \qquad (9.7)$$

то это назначение оптимально.

*Доказательство*. Очевидно.

Рассмотрим ***алгоритм решения задачи о назначениях***, в котором используется алгоритм решения задачи о максимальном потоке.

**Предварительный шаг** алгоритма состоит в приведении исходной матрицы, т.е. переход от матрицы $C$ к матрице $C_2$.

**Общий шаг.** 1) На основании матрицы $C_2$ строим двудольный орграф $G = (S \cup T, U)$, где $S = \{s_i, i = \overline{1, n}\}$ – множество вершин, которые соответствуют работам, $T = \{t_j, j = \overline{1, n}\}$ – множество вершин, соответствующих исполнителям. Дуга $(s_i, t_j) \in U$, если соответствующий элемент матрицы $C_2$ равен нулю.

2) По графу $G$ строим расширенную сеть: вводим две дополнительные вершины $s$ и $t$ – источник и сток, и дополнительные дуги из источника $s$ в вершины множества $S$ и из вершин множества $T$ в сток $t$. Пропускные способности всех дуг полагаем равными единице.

3) На полученной сети решаем задачу о нахождении максимального потока из источника $s$ в сток $t$.

4) Если величина максимального потока $v = n$, то решение закончено. Оптимальное назначение соответствует дугам вида $(s_i, t_j)$, для которых величина дугового потока равна единице. Оптимальность назначения следует из леммы 2, поскольку каждой такой дуге соответствует нулевой элемент матрицы $C_2$.

5) Если величина максимального потока меньше $n$, то необходимо ввести дополнительные дуги вида $(s_i, t_j)$, чтобы увеличить мощ-

ность потока. Необходимую для этого информацию можем извлечь из последнего шага алгоритма построения максимального потока.

Пусть $S_{пом}$ – множество вершин, помеченных на последнем шаге алгоритма ($S_{пом} \subset S$), $T_{неп}$ – множество вершин, непомеченных на последнем шаге алгоритма ($T_{неп} \subset T$). Ясно, что надо ввести новую дугу из $S_{пом}$ в $T_{неп}$. В матрице $C_2$ рассмотрим элементы на пересечении строк, соответствующих вершинам из $S_{пом}$, и столбцов, соответствующих вершинам из $T_{неп}$ (на рис. 9.1 заштрихованный блок). Пусть $c$ – минимальный среди них элемент. Вычитаем число $c$ из всех элементов, соответствующих вершинам из $S_{пом}$, и прибавляем $c$ ко всем элементам столбцов, соответствующих вершинам из $T_{пом}$. Переходим к пункту 1) общего шага.

| | $T_{пом}$ | $T_{неп}$ |
|---|---|---|
| $S_{пом}$ | $-c+c$ | $-c$ |
| $S_{неп}$ | $+c$ | |

Рис. 9.1.

Заметим, что $c > 0$. В противном случае, т.е. при $c = 0$, в сети существует дуга $(s_i, t_j)$, где $s_i \in S_{пом}$ и $t_j \in T_{неп}$, чего не может быть согласно алгоритму Форда-Фалкерсона. В результате преобразования матрицы в заштрихованном блоке появится новый нулевой элемент, а значит, появится новая дуга $(s_i, t_j)$, где $s_i \in S_{пом}$ и $t_j \in T_{неп}$. Кроме этого отметим, что элементы матрицы на пересечении строк, соответствующих вершинам из $S_{неп}$, и столбцов, соответствующих вершинам из $T_{пом}$, увеличиваются на $c$. Это означает, что в новой сети может исчезнуть дуга $(s_i, t_j)$, где $s_i \in S_{неп}$ и $t_j \in T_{пом}$, но эта дуга «бесполезна», т.к. вершину $t_j$ все же пометили, но не через вершину $s_i$.

Конечность алгоритма следует из того, что добавление новой дуги приводит к тому, что в результате алгоритма Форда-Фалкерсона помечается дополнительно, по крайней мере, одна вершина. Число вершин конечно, поэтому рано или поздно произойдет увеличение потока на единицу. Общая же величина потока ограничена сверху числом $n$.

*Замечание*. Если задача о назначениях есть задача максимизации (9.1)-(9.3),(9.5), то прежде, чем применять описанный выше алгоритм, преобразуем матрицу $C$. Найдем максимальный элемент среди всех элементов матрицы: $\max\limits_{i \in I,\, j \in J} c_{ij} = m$. Далее получим матрицу

$\tilde{C} = (\tilde{c}_{ij}, i \in I, j \in J)$, где $\tilde{c}_{ij} = m - c_{ij}$. Нетрудно доказать, что оптимальное назначение, на котором целевая функция $\sum_{i \in I} \sum_{j \in J} \tilde{c}_{ij} x_{ij}$ достигает минимума, является оптимальным назначением, на котором целевая функция $\sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$ достигает максимума.

Пример 1. Найти оптимальное назначение в задаче на минимизацию целевой функции, если матрица «стоимости» назначений имеет вид

$$C = \begin{pmatrix} 21 & 12 & 6 & 8 & 3 \\ 10 & 6 & 11 & 16 & 9 \\ 5 & 3 & 13 & 7 & 6 \\ 6 & 8 & 4 & 8 & 4 \\ 1 & 2 & 1 & 3 & 5 \end{pmatrix}.$$

Решение. Выполним процедуру приведение матрицы. Константы приведения по строкам равны 3, 6, 3, 4, 1. В полученной матрице $C_1$ константы приведения по столбцам равны 0, 0, 0, 2, 0. В результате имеем матрицу $C_2$.

$$C_1 = \begin{pmatrix} 18 & 9 & 3 & 5 & 0 \\ 4 & 0 & 5 & 10 & 3 \\ 2 & 0 & 10 & 4 & 3 \\ 2 & 4 & 0 & 4 & 0 \\ 0 & 1 & 0 & 2 & 4 \end{pmatrix}, \quad C_2 = \begin{pmatrix} 18 & 9 & 3 & 3 & 0 \\ 4 & 0 & 5 & 8 & 3 \\ 2 & 0 & 10 & 2 & 3 \\ 2 & 4 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 4 \end{pmatrix}.$$

Сеть $G = (S \cup T \cup \{s, t\}, U)$, соответствующая матрице $C_2$ изображена на рис. 9.2, здесь $S = \{s_i, i = \overline{1,5}\}$, $T = \{t_j, j = \overline{1,5}\}$. Первый общий шаг

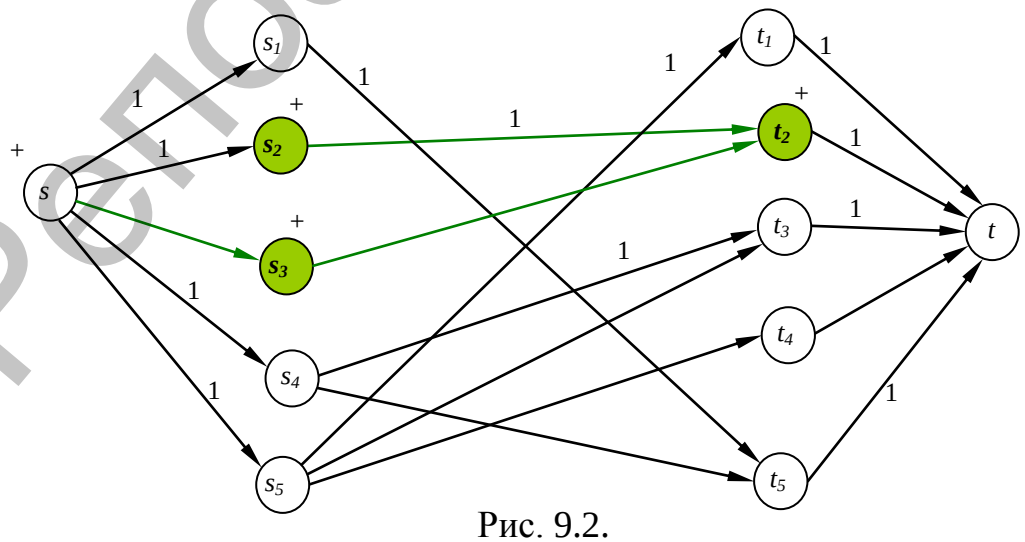


Рис. 9.2.

алгоритма приводит к построению максимального потока, величина $v$ которого равна $4 < n = 5$. На последнем шаге алгоритма Форда-Фалкерсона удалось пометить вершины $s$, $s_3$, $t_2$, $s_2$ (на рис.9.2 эти вершины отмечены знаком «плюс»). Следовательно, $S_{пом} = \{s_2, s_3\}$, $T_{неп} = \{t_1, t_3, t_4, t_5\}$. В матрице $C_2$ на пересечении строк, соответствующих вершинам из $S_{пом}$, и столбцов, соответствующих вершинам из $T_{неп}$ находятся элементы:

$$\begin{pmatrix} . & . & . & . & . \\ 4 & . & 5 & 8 & 3 \\ 2 & . & 10 & 2 & 3 \\ . & . & . & . & . \\ . & . & . & . & . \end{pmatrix},$$

минимальный элемент среди них равен $c = 2$. Вычитаем число 2 из всех элементов второй и третьей строк, и прибавляем число $c$ ко всем элементам второго столбца. Получим новую матрицу

$$C_3 = \begin{pmatrix} 18 & 11 & 3 & 3 & 0 \\ 2 & 0 & 3 & 6 & 1 \\ 0 & 0 & 8 & 0 & 1 \\ 2 & 6 & 0 & 2 & 0 \\ 0 & 3 & 0 & 0 & 4 \end{pmatrix}$$

Сеть $G = (S \cup T \cup \{s, t\}, U)$, соответствующая матрице $C_3$ изображена на рис.9.3. Общий шаг алгоритма приводит к построению максимального потока, величина которого равна $n = 5$. Дуговые потоки единич-

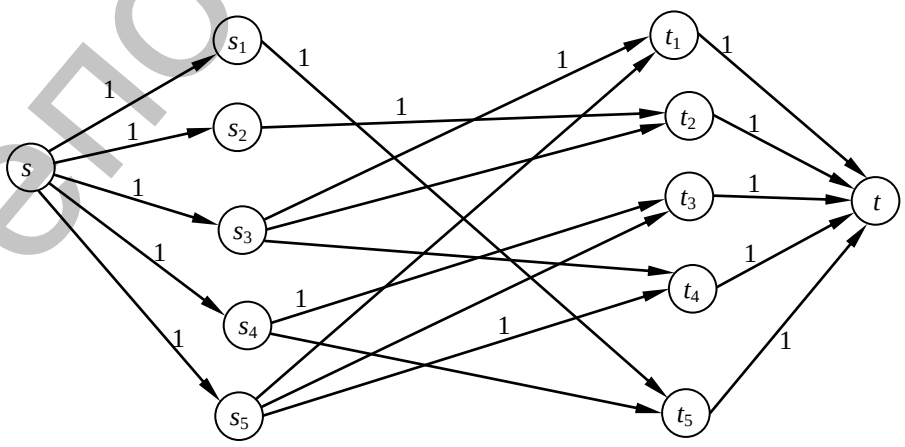


Рис. 9.3.

ной величины идут по дугам $(s_1, t_5)$, $(s_2, t_2)$,$(s_3, t_1)$, $(s_4, t_3)$, $(s_5, t_4)$. Этому потоку соответствуют назначения:

на 1-ую работу 5-ый исполнитель,
на 2-ую работу 2-ый исполнитель,
на 3-ую работу 1-ый исполнитель,
на 4-ую работу 3-ий исполнитель,
на 5-ую работу 4-ый исполнитель.

Стоимость такого назначения равна 3 + 6 + 5 + 4 + 3 = 21.

В литературе описан также венгерский метод решения задачи о назначениях, который отличается от выше приведенного только способом определения множеств $S_{\text{пом}}$ и $T_{\text{неп}}$.

**2. Задача о максимальной занятости.** Пусть имеется $n$ исполнителей и $m$ работ. Каждый исполнитель может выполнять определенные работы из указанного перечня работ. Требуется назначить исполнителей на работы таким образом, чтобы занять максимальное количество исполнителей.

**Схема решения.** Определим $n$x$m$-матрицу назначений

$$C = (c_{ij},\ i = \overline{1, n},\ j = \overline{1, m}),$$

где элемент $c_{ij}$ равен нулю тогда и только тогда, когда $i$-ый исполнитель может выполнять $j$-ую работу, и не равен нулю в противном случае. На основании матрицы $C$ строим двудольный орграф $G = (S \cup T, U)$, где $S = \{s_i,\ i = \overline{1, n}\}$ – множество вершин, которые соответствуют исполнителям, $T = \{t_j, j = \overline{1, m}\}$ – множество вершин, соответствующих работам. Дуга $(s_i, t_j) \in U$, если соответствующий элемент матрицы $C$ равен нулю. Как и в классической задаче о назначениях, по графу $G$ строим расширенную сеть, и на полученной сети решаем задачу нахождении максимального потока из источника $s$ в сток $t$. Величина максимального потока $v$ есть максимальное число исполнителей, которые назначаются на работы. Конкретные назначения определяем по дугам вида $(s_i, t_j)$, для которых величина дугового потока равна единице.

**3. Задача о назначении на «узкие» места.** Пусть имеется $n$ работ и $n$ исполнителей. Каждый исполнитель $i$ может выполнять любую работу $j$. Требуется найти такое назначение исполнителей на работы, чтобы наименьшая эффективность среди конкретных назначений работника $i$ на $j$-ую работу была наибольшей. Предполагается, что

любой исполнитель выполняет только одну работу, и каждая работа выполняется только одним исполнителем.

Примером такой задачи может служить распределение рабочих на конвейере. Под эффективностью понимается скорость выполнения конкретным исполнителем определенной операции на конвейере. Ясно, что скорость движения конвейера определяется минимальной эффективностью.

**Схема решения.** Задана матрицу $A = (a_{ij}, i = \overline{1, n}, j = \overline{1, n})$, где $a_{ij}$ – эффективность назначения $i$-го работника на $j$-ую работу. Любое назначение можно представить в виде подстановки

$$P = \begin{pmatrix} 1 & 2 & \dots & n \\ p(1) & p(2) & \dots & p(n) \end{pmatrix},$$

$p(i)$ – номер работы, на которую назначен $i$-ый исполнитель. Произвольному назначению поставим в соответствие вектор эффективностей:

$$(a_{1,p(1)}\ a_{2,p(2)}\ \dots\ a_{n,p(n)}),$$

и определим число

$$F(P) = \min \{ a_{1,p(1)},\ a_{2,p(2)}, \dots, a_{n,p(n)} \}.$$

Ясно, что требуется найти подстановку $P^0$, для которой $F(P^0)$ будет максимальным. Заметим, что число подстановок равно $n!$, и поэтому решение перебором всех подстановок не рационально.

Пусть задано некоторое начальное назначение

$$P_0 = \begin{pmatrix} 1 & 2 & \dots & n \\ p_0(1) & p_0(2) & \dots & p_0(n) \end{pmatrix},$$

и найдено число $F(P_0)$. Определим квадратную матрицу $n$-го порядка

$$C = (c_{ij},\ i, j = \overline{1, n}),$$

где элемент $c_{ij}$ равен нулю, если $a_{ij} > F(P_0)$, и не равен нулю в противном случае. По матрице $C$ строим двудольный орграф $G = (S \cup T, U)$, где $S = \{s_i, i = \overline{1, n}\}$ – множество вершин, которые соответствуют исполнителям, $T = \{t_j, j = \overline{1, n}\}$ – множество вершин, соответствующих работам. Дуга $(s_i, t_j) \in U$, если соответствующий элемент матрицы $C$ равен нулю. Как и в классической задаче о назначениях, по графу $G$ строим расширенную сеть, и на полученной сети решаем задачу нахождении максимального потока из источника $s$ в сток $t$. Если величина максимального потока $v = n$, то по дугам вида

($s_i$, $t_j$), для которых величина дугового потока равна единице, находим новое назначение $P_1$, и соответствующее ему число $F(P_1)$. Очевидно, что $F(P_1) > F(P_0)$. Затем повторяем шаг алгоритма, определив матрицу $C$, исходя из новой оценки $F(P_1)$. Если величина максимального потока $v < n$, то имеющееся назначение оптимально.

## § 11. Задания к лабораторным работам

11.1 Лабораторная работа «Построение остовного дерева минимального веса»

Задание. Дан неориентированный граф $G=(I,U)$ . Требуется построить остовное дерево минимального веса, используя а) алгоритм Прима, б) алгоритм Краскала. Выполнить чертеж.

В таблице 1 приведены множества $I$, $U$ и веса ребер соответственно перечислению их в $U$.

Таблица 1

| | | |
|---|---|---|
| Вариант 1 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {5,6}, {5,7}, {5,8}, {5,9}, {6,9}, {7,8}, {8,9} } |
| | веса | 10, 4, 3, 9, 11, 5, 12, 6, 15, 13, 10, 4, 7, 8, 4, 8 |
| Вариант 2 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,4}, {2,3}, {2,4}, {2,5}, {2,6}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {6,8}, {6,9}, {7,8}, {8,9} } |
| | веса | 10, 5, 3, 9 , 11, 5, 10, 6, 15, 13, 10, 5, 7, 8, 4, 8 |
| Вариант 3 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {6,8}, {6,9}, {7,8}, {8,9}} |
| | веса | 10, 14, 13, 9 , 11, 15, 12, 16, 15, 13, 10, 14, 7, 8, 14, 8 |
| Вариант 4 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,4}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {5,9}, {7,8}, {8,9} } |
| | веса | 15, 14, 23, 19 , 16, 15, 14, 26, 25, 23, 20, 24, 27, 18, 14, 18 |

| | | |
|---|---|---|
| Вариант 5 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7},{5,7}, {5,8}, {5,9}, {6,8}, {6,9}, {7,9}, {8,9} } |
| | веса | 20, 15, 22, 16, 25, 17, 18, 14, 18, 14, 13, 19, 21, 23, 20, 24 |
| Вариант 6 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {5,6}, {5,7}, {5,8}, {5,9}, {6,9}, {7,8}, {8,9} } |
| | веса | 5, 15, 4, 8, 10, 4, 3, 9, 11, 13, 12, 6, 10, 4, 7, 8, |
| Вариант 7 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,4}, {2,3}, {2,4}, {2,5}, {2,6}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {6,8}, {6,9}, {7,8}, {8,9} } |
| | веса | 11, 5, 4, 7, 10, 5, 10, 6, 15, 13, 10, 5, 9, 8, 3, 8 |
| Вариант 8 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {6,8}, {6,9}, {7,8}, {8,9}} |
| | веса | 14, 14, 12, 8 , 10, 15, 13, 16, 15, 13, 11, 10, 7, 8, 14, 9 |
| Вариант 9 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,4}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {5,9}, {7,8}, {8,9} } |
| | веса | 25, 23, 20, 24, 14, 18,15, 14, 23, 19, 27, 18, 16, 15, 14, 26, |
| Вариант 10 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7},{5,7}, {5,8}, {5,9}, {6,8}, {6,9}, {7,9}, {8,9} } |
| | веса | 23, 20, 24, 17, 15, 22, 16, 18, 14, 18, 20, 14, 13, 19, 21, 25 |
| Вариант 11 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {5,6}, {5,7}, {5,8}, {5,9}, {6,9}, {7,8}, {8,9} } |
| | веса | 4, 11, 5, 12, 6, 15, 4, 8, 3, 9 , 13, 10, 7, 8, 4, 10. |
| Вариант 12 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,4}, {2,3}, {2,4}, {2,5}, {2,6}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {6,8}, {6,9}, {7,8}, {8,9} } |
| | веса | 10, 5, 3, 8, 11, 5, 10, 4, 13, 15, 10, 5, 7, 9, 7, 8 |

| | | |
|---|---|---|
| Вариант 13 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {6,8}, {6,9}, {7,8}, {8,9}} |
| | веса | 11, 15, 12, 14, 7, 8, 14, 8, 10, 14, 13, 9, 16, 15, 13, 10 |
| Вариант 14 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,4}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {4,8}, {5,6}, {5,8}, {5,9}, {7,8}, {8,9} } |
| | веса | 14, 26, 25, 23, 20, 16, 15, 23, 19, 24, 27, 18, 14, 18, 15, 14, |
| Вариант 15 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7},{5,7}, {5,8}, {5,9}, {6,8}, {6,9}, {7,9}, {8,9} } |
| | веса | 20, 14, 13, 19, 21, 15, 22, 16, 25, 23, 20, 24, 17, 18, 14, 18 |
| Вариант 16 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {{1,2}, {1,5}, {1,4}, {2,3}, {2,5}, {3,5}, {3,6}, {4,5}, {4,7}, {5,6}, {5,7}, {5,8}, {5,9}, {6,9}, {7,8}, {8,9} } |
| | веса | 10, 8, 4, 8, 4, 3, 10, 4, 5, 12, 6, 7, 9 , 11, 15, 13 |

11.2 Лабораторная работа «Нахождение кратчайшего пути на сети »

Задание. Дана сеть $G=(I,U)$ . В таблице 2 приведены множества $I$, $U$ и веса дуг соответственно перечислению их в $U$. Вес дуги – расстояние между соответствующими вершинами.

а) Найти кратчайший путь из вершины 1, в вершину 9, используя алгоритм Дейкстра.

б) По алгоритму Флойда найти кратчайшие расстояния между любой парой вершин графа. Дать комментированный ответ для двх пар вершин (например, для вершин 1 и 8, 3 и 9)

Таблица 2

| | | |
|---|---|---|
| Вариант 1 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,5), (3,6), (4,5), (4,7), (5, 1), (5,6), (5,7), (5,8), (6,9), (7,8), (8,9), (9,5) } |
| | веса | 10, 4, 3, 9, 11, 5, 12, 6, 11, 13, 10, 4, 7, 8, 6, 8 |

| | | |
|---|---|---|
| Вариант 2 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,6), (4,2), (4,5), (4,7), (4,8), (5, 6), (5,8), (5,9), (6,9), (7,8), (8,9) } |
| | веса | 10, 5, 3, 9 , 11, 5, 10, 6, 15, 13, 10, 5, 7, 8, 4 |
| Вариант 3 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (1,5), (2,3), (2,6), (3,6), (4,5), (4,7), (5,2), (5, 6), (5,7), (6,8), (7,8), (8,9) } |
| | веса | 10, 14, 13, 9 ,11, 15, 12, 16, 15, 13, 10, 14, 7, 8, |
| Вариант 4 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,4), (2,6), (3,6), (4,5), (4,7), (4,8), (5,2), (5,8), (6,5), (6,8), (6,9), (7,8), (8,9) } |
| | веса | 25, 14, 23, 19, 16, 15, 4, 22, 28, 6, 26, 24, 27, 18, 14, 18 |
| Вариант 5 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,5), (3,6), (4,5), (4,7), (5, 1), (5,6), (5,7), (5,8), (6,9), (7,8), (8,9), (9,5)} |
| | веса | 20, 15, 22, 16, 25, 17, 18, 14, 18, 14, 13, 19, 21, 23, 20, 24 |
| Вариант 6 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,6), (4,2), (4,5), (4,7), (4,8), (5, 6), (5,8), (5,9), (6,9), (7,8), (8,9) } |
| | веса | 5, 15, 4, 8, 10, 4, 3, 9, 11, 12, 6, 10, 4, 7, 8, |
| Вариант 7 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (1,5), (2,3), (2,6), (3,6), (4,5), (4,7), (5,2), (5, 6), (5,7), (6,8), (7,8), (8,9) } |
| | веса | 5, 4, 7, 10, 5, 6, 15, 13, 10, 5, 9, 8, 3, 8 |
| Вариант 8 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,4), (2,6), (3,6), (4,5), (4,7), (4,8), (5,2), (5,8), (6,5), (6,8), (6,9), (7,8), (8,9) } |
| | веса | 14, 14, 12, 8 , 10, 15, 13, 16, 15, 13, 11, 10, 7, 8, 14, 9 |
| Вариант 9 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,5), (3,6), (4,5), (4,7), (5, 1), (5,6), (5,7), (5,8), (6,9), (7,8), (8,9), (9,5)} |

| | | |
|---|---|---|
| | веса | 25, 23, 20, 24, 14, 18,15, 14, 23, 19, 27, 18, 16, 15, 14, 26, |
| Вариант 10 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,6), (4,2), (4,5), (4,7), (4,8), (5, 6), (5,8), (5,9), (6,9), (7,8), (8,9) } |
| | веса | 23, 20, 24, 17, 15, 22, 16, 14, 18, 20, 14, 13, 19, 21, 25 |
| Вариант 11 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (1,5), (2,3), (2,6), (3,6), (4,5), (4,7), (5,2), (5, 6), (5,7), (6,8), (7,8), (8,9) } |
| | веса | 4, 11, 5, 12, 6, 15, 4, 8, 3, 9, 13, 10, 7, 8 |
| Вариант 12 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,4), (2,6), (3,6), (4,5), (4,7), (4,8), (5,2), (5,8), (6,5), (6,8), (6,9), (7,8), (8,9) } |
| | веса | 10, 5, 3, 8, 11, 5, 10, 4, 13, 15, 10, 5, 7, 9, 7, 8 |
| Вариант 13 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,5), (3,6), (4,5), (4,7), (5, 1), (5,6), (5,7), (5,8), (6,9), (7,8), (8,9), (9,5)} |
| | веса | 11, 15, 12, 14, 7, 8, 14, 8, 10, 14, 13, 9, 16, 15, 13, 10 |
| Вариант 14 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,5), (3,6), (4,2), (4,5), (4,7), (4,8), (5, 6), (5,8), (5,9), (6,9), (7,8), (8,9) } |
| | веса | 14, 26, 25, 23, 20, 16, 15, 23, 19, 24, 27, 18, 14, 18, 15 |
| Вариант 15 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (1,5), (2,3), (2,6), (3,6), (4,5), (4,7), (5,2), (5, 6), (5,7), (6,8), (7,8), (8,9) } |
| | веса | 20, 14, 13, 19, 21, 15, 22, 16, 20, 24, 17, 18, 14, 18 |
| Вариант 16 | I | {1, 2, 3, 4, 5, 6, 7, 8, 9} |
| | U | {(1,2), (1,4), (2,3), (2,4), (2,6), (3,6), (4,5), (4,7), (4,8), (5,2), (5,8), (6,5), (6,8), (6,9), (7,8), (8,9) } |
| | веса | 10, 8, 4, 8, 4, 3, 10, 4, 5, 12, 6, 7, 9 , 11, 15, 13 |

11.3 Лабораторная работа «Нахождение на сети потока максимальной величины»

Задание. Дана сеть $G=(I,U)$ . В таблице 2 приведены множества $I$, $U$ и веса дуг соответственно перечислению их в $U$. Вес дуги – пропускная способность дуги. Считаем вершину 1 источником, вершину 9 – стоком.

а) Найти поток максимальной величины из источника в сток, используя алгоритм Форда-Фалкерсона.

б) Найти соответствующий найденному потоку разрез минимальной пропускной способности.

11.4 Лабораторная работа «Нахождение оптимального назначения»

Задание. В таблице 3 для каждого варианта дана квадратная матрица, которая для задания а) является матрицей стоимостей назначения на $i$-ую работу $j$-го исполнителя, для задания б) является матрицей эффективностей назначения на $i$-ую работу $j$-го исполнителя.

а) Найти оптимальное назначение, минимизирующее суммарную стоимость назначений.

б) Найти оптимальное назначение, максимизирующее суммарную эффективность назначений.

| Вариант 1 | Вариант 2 | Вариант 3 |
|---|---|---|
| $\begin{pmatrix} 6 & 9 & 1 & 7 & 3 \\ 10 & 2 & 4 & 1 & 15 \\ 13 & 4 & 6 & 5 & 13 \\ 1 & 2 & 5 & 6 & 1 \\ 7 & 8 & 4 & 2 & 8 \end{pmatrix}$ | $\begin{pmatrix} 27 & 25 & 20 & 19 & 21 \\ 19 & 13 & 12 & 20 & 16 \\ 26 & 19 & 10 & 22 & 25 \\ 10 & 14 & 10 & 8 & 12 \\ 14 & 15 & 18 & 19 & 13 \end{pmatrix}$ | $\begin{pmatrix} 27 & 25 & 20 & 19 & 21 \\ 19 & 13 & 12 & 20 & 16 \\ 26 & 19 & 10 & 22 & 25 \\ 10 & 14 & 10 & 8 & 12 \\ 14 & 12 & 18 & 19 & 13 \end{pmatrix}$ |
| Вариант 4 | Вариант 5 | Вариант 6 |
| $\begin{pmatrix} 21 & 16 & 11 & 12 & 18 \\ 9 & 11 & 13 & 19 & 13 \\ 13 & 15 & 16 & 17 & 10 \\ 9 & 12 & 15 & 14 & 11 \\ 7 & 10 & 14 & 17 & 9 \end{pmatrix}$ | $\begin{pmatrix} 18 & 16 & 15 & 17 & 19 \\ 10 & 9 & 11 & 8 & 7 \\ 17 & 15 & 10 & 11 & 16 \\ 5 & 6 & 9 & 8 & 5 \\ 8 & 5 & 6 & 7 & 10 \end{pmatrix}$ | $\begin{pmatrix} 21 & 19 & 11 & 18 & 22 \\ 8 & 9 & 12 & 13 & 8 \\ 20 & 15 & 15 & 16 & 19 \\ 21 & 16 & 18 & 22 & 20 \\ 13 & 10 & 11 & 14 & 10 \end{pmatrix}$ |
| Вариант 7 | Вариант 8 | Вариант 9 |
| $\begin{pmatrix} 14 & 2 & 4 & 1 & 15 \\ 6 & 10 & 1 & 7 & 3 \\ 13 & 4 & 6 & 5 & 12 \\ 1 & 2 & 5 & 6 & 1 \\ 7 & 7 & 4 & 2 & 8 \end{pmatrix}$ | $\begin{pmatrix} 24 & 18 & 25 & 17 & 21 \\ 31 & 22 & 23 & 24 & 30 \\ 17 & 19 & 21 & 15 & 17 \\ 30 & 20 & 20 & 25 & 28 \\ 19 & 20 & 24 & 23 & 19 \end{pmatrix}$ | $\begin{pmatrix} 2 & 4 & 1 & 15 & 10 \\ 8 & 4 & 2 & 8 & 7 \\ 2 & 5 & 6 & 1 & 1 \\ 9 & 1 & 7 & 3 & 6 \\ 4 & 6 & 5 & 13 & 14 \end{pmatrix}$ |

| Вариант 10 | Вариант 11 | Вариант 12 |
|---|---|---|
| $\begin{pmatrix} 13 & 18 & 20 & 16 & 19 \\ 19 & 10 & 22 & 25 & 26 \\ 14 & 10 & 8 & 12 & 10 \\ 12 & 12 & 19 & 13 & 14 \\ 25 & 20 & 18 & 21 & 27 \end{pmatrix}$ | $\begin{pmatrix} 16 & 14 & 13 & 15 & 17 \\ 10 & 11 & 10 & 12 & 9 \\ 15 & 13 & 12 & 13 & 14 \\ 7 & 8 & 7 & 9 & 7 \\ 8 & 7 & 9 & 10 & 11 \end{pmatrix}$ | $\begin{pmatrix} 13 & 4 & 5 & 12 & 6 \\ 1 & 2 & 6 & 1 & 5 \\ 7 & 7 & 2 & 8 & 4 \\ 15 & 3 & 2 & 16 & 5 \\ 7 & 11 & 8 & 4 & 2 \end{pmatrix}$ |
| Вариант 13 | Вариант 14 | Вариант 15 |
| $\begin{pmatrix} 12 & 13 & 8 & 7 & 9 \\ 15 & 16 & 19 & 20 & 15 \\ 18 & 22 & 20 & 21 & 16 \\ 11 & 14 & 10 & 13 & 10 \\ 10 & 17 & 21 & 20 & 15 \end{pmatrix}$ | $\begin{pmatrix} 31 & 21 & 55 & 24 & 22 \\ 25 & 19 & 48 & 35 & 41 \\ 42 & 40 & 36 & 44 & 33 \\ 37 & 35 & 27 & 29 & 51 \\ 30 & 29 & 44 & 43 & 12 \end{pmatrix}$ | $\begin{pmatrix} 13 & 15 & 18 & 11 & 19 \\ 41 & 39 & 43 & 33 & 38 \\ 37 & 31 & 29 & 33 & 35 \\ 23 & 27 & 28 & 29 & 22 \\ 46 & 42 & 48 & 44 & 45 \end{pmatrix}$ |
| Вариант 16 | Вариант 17 | Вариант 18 |
| $\begin{pmatrix} 45 & 39 & 41 & 38 & 42 \\ 54 & 48 & 52 & 49 & 55 \\ 35 & 26 & 30 & 27 & 33 \\ 48 & 36 & 42 & 37 & 39 \\ 51 & 49 & 50 & 46 & 52 \end{pmatrix}$ | $\begin{pmatrix} 13 & 18 & 12 & 15 & 11 \\ 10 & 9 & 13 & 15 & 12 \\ 17 & 15 & 11 & 14 & 18 \\ 15 & 12 & 17 & 21 & 19 \\ 18 & 22 & 14 & 19 & 16 \end{pmatrix}$ | $\begin{pmatrix} 7 & 9 & 4 & 5 & 8 \\ 10 & 11 & 12 & 14 & 15 \\ 7 & 5 & 10 & 11 & 6 \\ 13 & 12 & 8 & 7 & 13 \\ 6 & 8 & 10 & 14 & 12 \end{pmatrix}$ |

## ЛИТЕРАТУРА


1. Бахтин В.И. и др. Исследование операций. Курс лекций. – Мн.: БГУ, 2003.
2. Вагнер Г. Основы исследования операций. Т. 1, т. 2, т. 3. – М.: Мир, 1972, 1973, 1973.
3. Волков И.К., Загоруйко Е.А. Исследование операций: учеб. для вузов. – 2-е изд. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002.
4. Высшая математика для экономистов: учебник: в 3 т.– Мн.: БГЭУ, 2005. – Т. 2.
5. Костюкова О.И. Исследование операций: учеб. пособие для студ. спец.31 03 04 «Информатика» всех форм обучения. – Мн.: БГУИР, 2003. – 94 с.: ил.
6. Котов В.М., Мельников О.И.. Информатика: методы алгоритмизации: учеб. пособие для 10–11-х кл. общеобразоват. шк. с углубл. изучением информатики. – Мн.: Нар. асвета, 2000. – 221 с.: ил.
7. Кузнецов А.В., Сакович В.А., Холод Н.И. Высшая математика: Математическое программирование. – Мн.: Выш. шк., 1994.