

Л.Н.Домнин

**ЭЛЕМЕНТЫ
ТЕОРИИ ГРАФОВ**

Пенза 2004

Книга является учебным пособием и состоит из пяти разделов. В первом даны основные понятия и определения теории графов, рассмотрены виды графов и способы их описания. Второй раздел посвящен вопросу о связности ориентированных графов. Важнейший вид графов – деревья рассмотрен в третьем разделе. Разобраны задачи описания и пересчета деревьев, а также задача о кратчайшем остове. Четвертый раздел посвящен вопросам пересчета и перечисления путей в графах. Здесь же приведены различные варианты задачи о кратчайшем пути и алгоритмы ее решения. В пятом разделе рассматриваются фундаментальные, эйлеровы и гамильтоновы циклы. Разбираются условия существования и алгоритмы поиска таких циклов в графе.

Пособие подготовлено по материалам курса лекций по теории графов, читаемого автором для студентов специальности "Прикладная математика" Пензенского государственного университета. Может быть использовано студентами других специальностей при изучении соответствующих разделов дискретной математики.

©Л.Н.Домнин

©Оформление: Л.Н.Домнин

1 Основные понятия теории графов

1.1 Определение графа

Теоретико-множественное определение графа. Пусть V непустое множество, например, $\{v_1, v_2, v_3, v_4, v_5\}$. Запишем множество всех его двухэлементных подмножеств $V^{(2)}$. Для нашего примера это множество

$$V^{(2)} = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \{v_1, v_5\}, \\ \{v_2, v_3\}, \{v_2, v_4\}, \{v_2, v_5\}, \\ \{v_3, v_4\}, \{v_3, v_5\}, \\ \{v_4, v_5\}\}.$$

Возьмем произвольно некоторое $E \subseteq V^{(2)}$, например,

$$E = \{\{v_1, v_2\}, \{v_1, v_3\}, \{v_1, v_4\}, \\ \{v_2, v_3\}, \{v_2, v_5\}, \\ \{v_3, v_4\}, \{v_4, v_5\}\}.$$

Пару $\langle V, E \rangle$ называют *неориентированным графом* G , в котором V это множество *вершин*, а E — множество *ребер*, являющееся подмножеством множества $V^{(2)}$.

В более компактной форме это определение обычно формулируется так: *пара $\langle V, E \rangle$ называется неориентированным графом, если V — непустое множество элементов, называемых вершинами, а E — множество неупорядоченных пар различных элементов из V , называемых ребрами.*

При записи различных соотношений в теории графов пользуются обозначениями VG или $V(G)$ для множества вершин и EG или $E(G)$ для множества ребер графа G .

Наглядным способом представления графа является рисунок (диаграмма), на котором вершины изображаются точками, кружочками или другими фигурками, а ребра — линиями, соединяющими изображения вершин реберной пары.¹ Форма

¹ Иногда графами называют именно эти рисунки (диаграммы).

и размеры изображения значения не имеют. Важно только, чтобы оно соответствовало множествам V и E . На рис. 1.1 даны варианты такого представления вышеописанного графа. Обратите внимание на существенное различие изображений

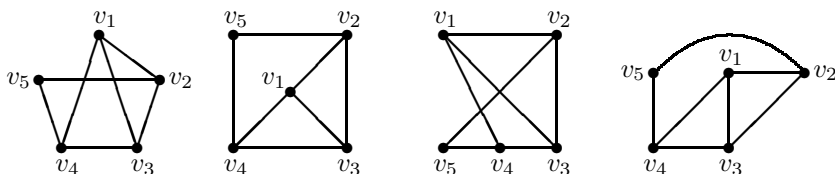


Рис. 1.1

и, вместе с тем, полное соответствие друг другу и описанию графа на стр. 3.

Основные характеристики графа и его элементов.

Две вершины, образующие ребро $\{v_i, v_j\}$, называют его концами, а про ребро говорят, что оно соединяет v_i и v_j . Говорят также, что вершины v_i и v_j *смежны*. Смежными называют и ребра с общей вершиной. Про ребро и вершину, которая является его концом, говорят, что они *инцидентны*. Так например, в графе на рис. 1.1 вершины v_1 и v_2 смежны, а вершины v_3 и v_5 не смежны. Ребра $\{v_1, v_2\}$ и $\{v_1, v_3\}$ смежны, а ребра $\{v_4, v_5\}$ и $\{v_2, v_3\}$ не смежны. Наконец, вершина v_3 и ребро $\{v_2, v_3\}$ инцидентны.

Число ребер, инцидентных вершине v , определяет *степень вершины*, которая обозначается $\deg v$. Так, в графе на рис. 1.1 $\deg v_1 = \deg v_2 = \deg v_3 = \deg v_4 = 3$, а $\deg v_5 = 2$. Вершину v называют *изолированной*, если $\deg v = 0$, и *концевой*, если $\deg v = 1$. Ребро, инцидентное концевой вершине, также называют *концевым*. Список степеней всех вершин называют *степенной последовательностью* графа.

Важнейшими количественными характеристиками графа являются: число вершин $n = |V|$, определяющее *порядок* графа, и число ребер $m = |E|$. Граф с n вершинами и m ребрами называется (n, m) -графом.

Исторически первой теоремой теории графов является утверждение, принадлежащее Эйлеру и связывающее количество

ребер, вершин и их степеней.

Теорема 1.1 *Сумма степеней вершин (n, m) -графа равна удвоенному числу его ребер: $\sum_{i=1}^n \deg v_i = 2|E| = 2m$.*

▷ Доказательство тривиально: поскольку любое ребро инцидентно двум вершинам, в сумме степеней всех вершин графа каждое ребро учитывается дважды. <

С л е д с т в и е. *В любом графе число вершин нечетной степени четно.* ▷ Пусть V_1 и V_2 – множества вершин четной и нечетной степени соответственно. Очевидно, что

$$\sum_{v \in V_1} \deg v + \sum_{v \in V_2} \deg v = 2m.$$

Имеем два слагаемых, сумма которых четное число. Первое слагаемое четно (как сумма четных чисел). Значит второе также должно быть четным, а это при суммировании нечетных чисел возможно, если только их количество четно. <

Изоморфизм графов. Обратимся вновь к рис. 1.1, где представлены четыре различных изображения одного и того же графа. Задача рисунка – показать множественность образов при графической интерпретации графа. Можно поставить другую, в некотором смысле обратную задачу. Имеются изображения графов одного порядка с одинаковым числом ребер. Необходимо установить, разные это графы или один, только по-разному изображенный. Чтобы различать подобные ситуации используют понятие изоморфизма.

Изоморфизмом называют взаимно-однозначное соответствие между множествами вершин двух графов G_1 и G_2 , сохраняющее отношение смежности, а сами графы называют изоморфными. Отображая это, пишут: $G_1 \cong G_2$ или $G_1 = G_2$.

Для графов на рис. 1.1 факт изоморфности устанавливается просто. Достаточно составить списки вершин всех графов, указав для каждой из них всех ее "соседей". Сравнив списки, легко убедиться в их идентичности – графы изоморфны. В данном случае графы действительно изоморфны, причем их эквивалентные вершины обозначены (пронумерованы)

одинаково. Задача усложняется, если эквивалентные вершины графов имеют разные номера. В качестве примера можно привести два изоморфных графа G_1 и G_2 , изображенных на рис. 1.2. Для них описанную выше процедуру работы со спис-

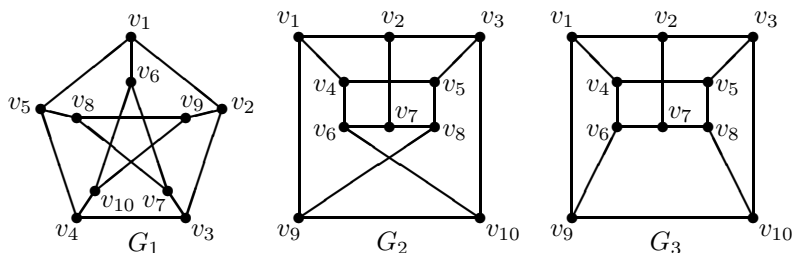


Рис. 1.2

ками придется повторять неоднократно, каждый раз меняя нумерацию вершин одного из графов, пока не будет обнаружен изоморфизм, т.е. получены идентичные списки вершин. В лучшем случае факт изоморфности графов можно установить с первой попытки, в худшем потребуется проверить $10!$ вариантов нумерации вершин. Доказать неизоморфность графов G_1 и G_3 можно, только выполнив все проверки. Существенно сократить их количество можно, если использовать инварианты.

Инвариантом называют некоторую числовую характеристику графа G , которая принимает одно и то же значение для любого графа, изоморфного G .

Анализ графов на изоморфность целесообразно начинать со сравнения значений инвариантов. И только при их совпадении переходить к перебору допустимых вариантов нумерации вершин. Тривиальными инвариантами графа являются $|V|$ и $|E|$. К числу инвариантов относится и степенная последовательность графа. Если, например, при совпадении степенных последовательностей двух графов в каждом из них есть одна вершина степени d , обозначенная в первом графе как v_5 , то из всех возможных нумераций вершин второго графа следует рассмотреть только те, в которых вершина этой степени имеет тот же номер.

Есть множество других инвариантов, однако не известна (возможно и не существует) система инвариантов, позволяющая решать задачу изоморфизма для всех видов графов.

Количество графов. При изучении объектов любой природы одним из первых возникает вопрос об их количестве. Определим число графов порядка n .

Ясно, что на множестве из n вершин можно образовать C_n^2 различных неупорядоченных пар, соответствующих всем возможным ребрам графа. Поэтому любому n -вершинному графу можно сопоставить C_n^2 -разрядный двоичный код, каждый разряд которого соответствует определенному ребру: если разряд равен 1, граф содержит это ребро, если 0 – не содержит. Следовательно, количество графов порядка n можно определить как $l_n = 2^{C_n^2}$. При $n=10$ имеем $l_{10} = 2^{C_{10}^2}$, что составляет внушительную величину 2^{45} или $\simeq 3,518 \cdot 10^{13}$. Однако, если не учитывать разметку вершин, которая принципиального значения не имеет, а лишь позволяет описать связи между вершинами, не все эти графы различны. Например, существует $l_3 = 2^{C_3^2} = 8$ трехвершинных графов. Все они представлены на рис. 1.3. Легко заметить, что фактически

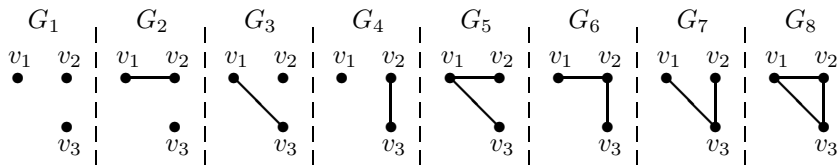


Рис. 1.3

есть лишь четыре различных трехвершинных графа, поскольку $G_2 \cong G_3 \cong G_4$ и $G_5 \cong G_6 \cong G_7$ и, если убрать метки вершин, различия между графами в этих тройках исчезнут. Поэтому естественно появление такого понятия как *непомеченный (абстрактный) граф*, в отличие от помеченного (имеющего метки вершин). Фактически, любой непомеченный граф представляет собой множество изоморфных помеченных графов.

Задача определения количества непомеченных графов g_n существенно сложнее задачи о количестве помеченных. В [2]

приведены данные о числе непомеченных графов до 24-го порядка включительно. Так число графов с десятью вершинами $g_{10}=12\,005\,168$. Эта величина на шесть порядков меньше, чем l_{10} , но все еще весьма значительна.

Существует *формула Поля* $g_n \sim 2^{C_n^2}/n!$, дающая асимптотическую оценку числа непомеченных графов. Согласно формуле количество непомеченных графов приблизительно в $n!$ раз меньше количества помеченных. Действительно, в большинстве случаев (но не всегда) разметку вершин непомеченного графа можно выполнить $n!$ способами. При малых n формула "не работает". Так, для $n=4$ получаем $2^{C_4^2}/4!=8/3$, тогда как имеется 11 четырехвершинных графов, т.е. налицо четырехкратное занижение их числа. При $n=10$ формула дает 9 696 000, что уже в 1,238 меньше фактического числа 12 005 168, но ошибка еще существенна. При $n=12$ получаем $\sim 1,54 \cdot 10^{11}$, вместо $\sim 1,65 \cdot 10^{11}$, т.е. в 1,072 раза меньше. Наконец, при $n=15$ разница сокращается до одного процента.

1.2 Подграфы

Граф $G'(V', E')$ называют *подграфом* графа $G(V, E)$, если $V' \subseteq V$ и $E' \subseteq E$, причем такие, что ребро $\{v, u\}$ содержится в E' только тогда, когда $v \in V'$ и $u \in V'$. В свою очередь, граф G по отношению к своему подграфу G' является *надграфом*.

Остовным называют подграф $G'(V, E')$, включающий в себя все вершины надграфа $G(V, E)$. В качестве примера на

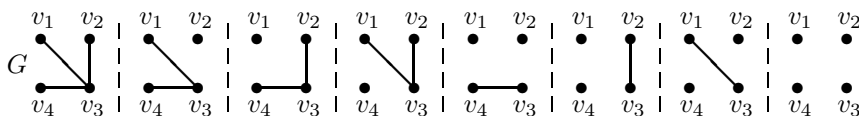


Рис. 1.4

рис. 1.4 показаны все остовные подграфы графа G . Нетрудно видеть, что любой остовный подграф $G(V, E')$ графа $G(V, E)$ может быть получен путем удаления подмножества $E \setminus E'$ ребер надграфа.

Подграфы другого типа получаются, если выбрать некоторое подмножество V' вершин надграфа и присоединить к ним все ребра, концы которых принадлежат V' . Это *вершинно-порожденные* подграфы. На рис. 1.5 изображены все подграфы этого типа для графа G . Легко убедиться, что любой

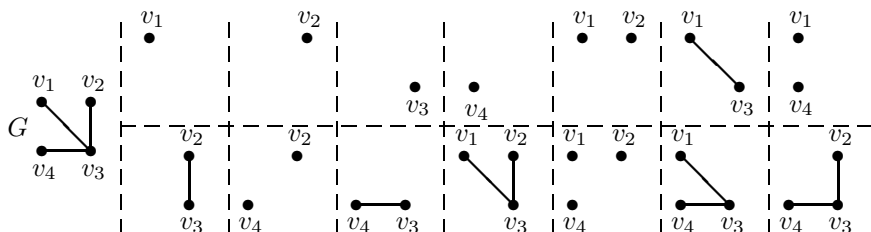


Рис. 1.5

вершинно-порожденный подграф $G(V', E')$ графа $G(V, E)$ можно получить путем удаления подмножества $V \setminus V'$ вершин и всех инцидентных им ребер надграфа.

Наконец, *реберно-порожденным* будем называть подграф $G(V', E')$, полученный на основе произвольно выбранного подмножества E' ребер надграфа, причем множество V' вершин подграфа составляют концы только этих ребер. Все реберно-порожденные подграфы графа G изображены на рис. 1.6.

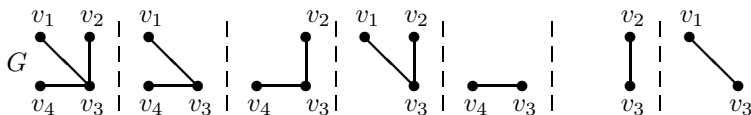


Рис. 1.6

Любой реберно-порожденный подграф $G(V', E')$ можно сформировать, если в надграфе сперва удалить подмножество ребер $E \setminus E'$, а потом в получившемся остовном подграфе удалить все изолированные вершины.

Приведенная здесь терминология соответствует принятой в [1,8,12]. В [6,10] остовный подграф называется *суграфом*, реберно-порожденный – *частичным графом*, а вершинно-порожденный – *подграфом*.

1.3 Виды графов

Обратимся к рис. 1.7, заимствованному (с небольшими изменениями и дополнениями) из книги [1]. Здесь представлены все возможные абстрактные графы четвертого порядка.

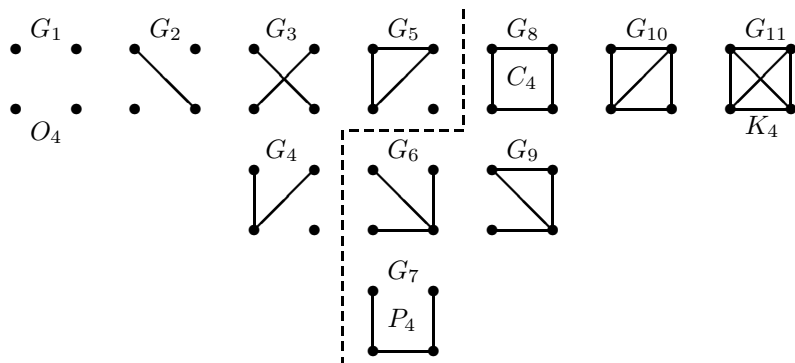


Рис. 1.7

Несвязные и связные графы. Очевидно, что изображенные на рисунке графы можно поделить на две группы (разграничены штриховой линией): *несвязные* ($G_1 - G_5$) и *связные* ($G_6 - G_{11}$).

Граф несвязен, если множество его вершин распадается на два (или более) непересекающихся подмножества, таких, что нет ни одного ребра, концы которого принадлежат разным подмножествам. Иными словами, несвязный граф состоит из двух и более частей, не соединенных ребрами. Эти части называются *компонентами связности*. Так, на рис. 1.7 граф G_1 имеет четыре компоненты, G_2 – три компоненты, G_3 , G_4 и G_5 по две, а остальные графы по одной компоненте.

Среди несвязных графов, изображенных на рис. 1.7, особое положение занимает G_1 , вообще не имеющий ребер и называемый *пустым*. Для таких графов часто используется обозначение O_n , где n – число вершин.

Среди связных графов $G_6 - G_{11}$ также можно выделить граф G_{11} , имеющий максимальное для графа четвертого порядка число ребер. Это так называемый *полный* граф. Для та-

ких графов принято обозначение K_n . Ясно, что число ребер в полном графе равно количеству двухэлементных подмножеств множества V , поэтому имеем $|E(K_n)| = C_n^2 = \frac{n(n-1)}{2}$.

Деревья и цепи. Связные графы с минимальным количеством ребер ($|EG|=n-1$) образуют класс так называемых *деревьев*. В нашем примере это G_6 и G_7 . Рассмотрению деревьев посвящен третий раздел пособия. Здесь же отметим обозначенный на рисунке как P_4 граф G_7 , являющийся частным случаем дерева и называемый *простой цепью*.

Вообще *цепь* – это чередующаяся последовательность вершин и ребер $(v_0, e_1, v_1, e_2, v_2, \dots, v_{k-1}, e_k, v_k)$, где v_{i-1} и v_i являются концами ребра e_i . В компактной форме эта запись выглядит так: $(v_0, v_1, \dots, v_{k-1}, v_k)$ или $(e_1, e_2, \dots, e_{k-1}, e_k)$. В отличие от простой цепи, цепь общего вида может содержать повторяющиеся вершины. Например, в графе на рис. 1.1 $(v_1, v_2, v_5, v_4, v_3)$ – простая цепь, а $(v_1, v_2, v_5, v_4, v_1, v_3)$ – цепь. Обычно цепь рассматривается не как самостоятельный граф, а как часть некоторого другого графа. Длиной цепи называют количество составляющих ее ребер. Ясно, что длина простой цепи не может превышать порядок содержащего ее графа, а длина цепи общего вида – числа ребер этого графа.

Понятие цепи широко используется в теории графов. Например, *связный граф можно определить как граф, в котором любая пара вершин соединена хотя бы одной цепью*.

Циклы. Циклом (*простым циклом*) называется замкнутая цепь (*простая цепь*). Примером простого цикла является граф G_8 . В общем случае граф, представляющий собой простой цикл, обозначается как C_n . Основным интерес, как и в случае цепей, представляет рассмотрение циклов не как самостоятельных графов, а как частей некоторого графа.

Дополнения. Рассмотрим пары графов: G_1 и G_{11} , G_2 и G_{10} , G_3 и G_8 , G_4 и G_9 , G_5 и G_6 . Если в каждой из них изображение одного из графов "наложить" определенным образом на изображение другого, то получится полный граф, т.е. один граф дополняет другой до K_4 . Это пары взаимно дополнительных графов. Вообще *дополнением* $\bar{G}$ графа G называется

граф, имеющий то же множество вершин, что и G , в котором любые две вершины смежны тогда и только тогда, когда они не смежны в G . Особое место среди множества представленных на рис. 1.7 графов занимает G_7 . Это так называемый *самодополнительный* граф, являющийся своим собственным дополнением, в чем нетрудно убедиться, если изобразить его так $\overline{\overline{X}}$. Для самодополнительных графов $\overline{\overline{G}}=G$, тогда как в общем случае $\overline{\overline{G}}=G$.

Регулярные графы. Для графов G_1, G_3, G_8, G_{11} характерным является то, что в каждом из них все вершины имеют одинаковые степени. Такие графы называют *регулярными* или *однородными*. На рис. 1.8 приведены примеры регулярных восьмивершинных графов третьей, четвертой и пятой степени.

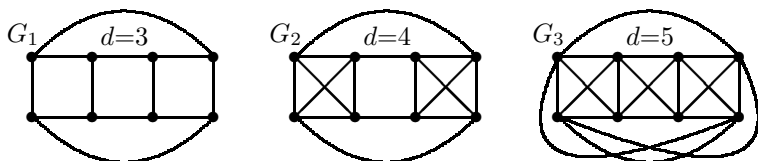


Рис. 1.8

Отметим, что графы третьей степени называют *кубическими*. Это графы G_{11} на рис. 1.7 и G_1 на рис. 1.8. Очевидно, что число ребер в регулярном графе степени d равно $m=\frac{1}{2}nd$. Из этого следует, что при нечетном числе вершин регулярный граф может иметь только четную степень, а при нечетной степени только четное число вершин. Поэтому любой кубический граф имеет четное число вершин.

Двудольные графы. Завершая обзор графов, представленных на рис. 1.7, охарактеризуем класс *двудольных* графов.

Граф называется *двудольным*, если существует такое разбиение множества его вершин V на два подмножества (две доли) V_1 и V_2 , что концы любого ребра принадлежат разным подмножествам. Среди графов на рис. 1.7 под определение двудольного подпадают графы $G_1 - G_4, G_6$ и G_7 , причем однозначное разбиение множества вершин на доли возможно только для связных графов G_6 и G_7 . Более наглядные

примеры двудольных графов даны на рис. 1.9². Несмотря на очевидное несходство изображений, G_1 и G_2 изоморфны, т.е. это фактически один и тот же граф, что легко проверить.

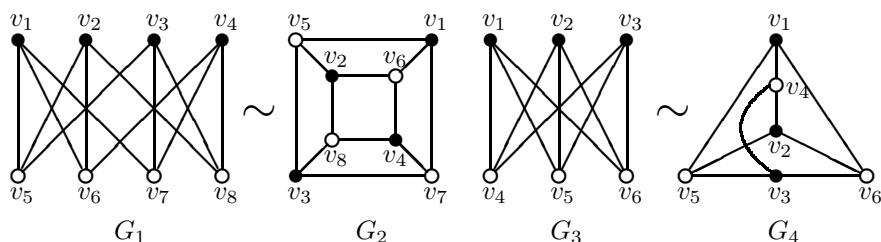


Рис. 1.9

То же самое относится к паре G_3 и G_4 . Кроме того, здесь каждая вершина одной доли $\{v_1, v_2, v_3\}$ смежна с каждой вершиной другой доли $\{v_4, v_5, v_6\}$. Такие графы называют *полными двудольными* и обозначают в общем случае K_{n_1, n_2} , где $n_1 = |V_1|$, $n_2 = |V_2|$. Очевидно, что $|E(K_{n_1, n_2})| = n_1 n_2$.

Существует простой способ распознавания двудольности графа, основанный на следующей теореме Кенига.

Теорема 1.2 *Граф является двудольным, тогда и только тогда, когда он не содержит циклов нечетной длины.*

В основу доказательства (и алгоритма распознавания) может быть положена следующая процедура разметки графа.

▷ Выбираем некоторую вершину v и помечаем ее знаком "+". Далее для каждой помеченной на предыдущем шаге вершины отыскиваем всех ее "соседей" и помечаем их противоположным знаком. Продолжаем эту операцию до тех пор,

– пока все вершины графа не будут помечены без каких-либо коллизий т.е. метки любой вершины со стороны ее соседей совпадают. Это означает, что все цепи, соединяющие любую вершину графа с v , состоят только из четного или только из нечетного числа дуг и значит в совокупности образуют циклы четной длины. Вместе с тем, согласованность меток всех вершин свидетельствует о возможности "двудольного

²Вершины, относящиеся к разным долям, изображены по-разному.

разбиения" множества вершин графа (собственно распределение знаков и определяет это разбиение); или

– пока не возникнет ситуация, когда некоторая вершина u окажется помеченной знаком "+" со стороны одной соседней вершины и знаком "-" со стороны другой. Это значит, что в графе есть две простые цепи, соединяющие v и u , одна из которых состоит из четного числа дуг, а другая – из нечетного, в совокупности образующие цикл нечетной длины. Вместе с тем, разные знаки у вершины u не позволяют выполнить "двудольное разбиение" множества вершин графа. $\triangleleft$

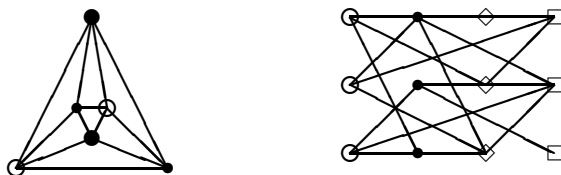


Рис. 1.10

Аналогично двудольным определяются трех-, четырех- и, вообще, k -дольные графы. Примеры трех- и четырехдольного графов даны на рис. 1.10. К сожалению, простых критериев выявления k -дольности графа при $k > 2$ не существует.

Мульти- и псевдографы. Согласно определению, совокупность ребер графа является множеством. Поэтому смежные вершины соединяет только одно ребро. Если допустить

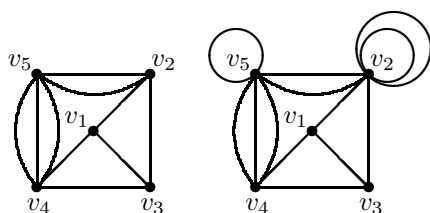


Рис. 1.11

кратные (параллельные) ребра, то получим мультиграф. Иными словами, мультиграф это пара $\langle V, E \rangle$, где V – непустое множество элементов (вершин), а E – семейство подмножеств из $V^{(2)}$ (ребер). Наконец, если на-

ряду с кратными разрешить ребра, связывающие вершину саму с собой, (*петли*), то получим *псевдограф*. Мультиграф и псевдограф изображены на рис. 1.11.

1.4 Матрицы графов

Матрица смежности. Определим *матрицу смежности* как симметричную квадратную матрицу $A=[a_{i,j}]$ порядка n , в которой элемент $a_{i,j}$ равен 1, если в графе есть ребро $\{v_i, v_j\}$, т.е. v_i и v_j смежны, и 0, если такого ребра нет.

Из определения следует, что $\sum_{j=1}^n a_{i,j}=d(v_i)$ при любом i , $\sum_{i=1}^n a_{i,j}=d(v_j)$ при любом j и $\sum_{i=1}^n \sum_{j=1}^n a_{i,j}=2m$, т.е. количество единиц в любой строке или столбце матрицы смежности равно степени соответствующей вершины графа, а общее количество единиц равно удвоенному числу его ребер.

В качестве примера на рис. 1.12 представлены граф, его матрица смежности и столбец степенной последовательности.

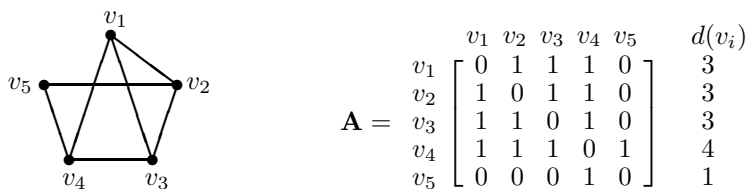


Рис. 1.12

Охарактеризуем матрицы смежности некоторых видов графов. Очевидно, что матрица смежности пустого графа состоит из одних нулей, т.е. $A(O_n)=0$. Матрица смежности полного графа состоит из одних единиц, кроме диагональных элементов, которые равны 0. Запишем это так $A(K_n)=1_n-I$, где I — единичная матрица порядка n . Если граф несвязен и имеет

$$A = \begin{bmatrix} A_{11} & 0 & \dots & 0 \\ 0 & A_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & A_{ss} \end{bmatrix}$$

s компонент, то путем перестановки строк и столбцов его матрица смежности может быть приведена к блочно-диагональному виду, где каждый диагональный блок A_{ii} является матрицей смежности соответствующей компоненты.

В случае k -дольного графа его матрица смежности может быть приведена к блочному виду, когда по главной диагонали идут только "нулевые" блоки.

$$\mathbf{A} = \left[\begin{array}{c|c|c|c} \mathbf{0} & \mathbf{A}_{12} & \dots & \mathbf{A}_{1k} \\ \hline \mathbf{A}_{21} & \mathbf{0} & \dots & \mathbf{A}_{2k} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{A}_{k1} & \mathbf{A}_{k2} & \dots & \mathbf{0} \end{array} \right]$$

Блоки $\mathbf{A}_{ij}$, где $i \neq j$, могут рассматриваться как приведенные³ матрицы смежности двудольных вершинно-порожденных подграфов k -дольного графа, образованных вершинами i и j долей, причем $\mathbf{A}_{ij} = \mathbf{A}_{ji}^T$.

В качестве примера на рис.1.13 дан трехдольный граф с долями $\{v_1, v_2\}$, $\{v_3, v_4, v_5\}$, $\{v_6, v_7\}$ и его матрица смежности.

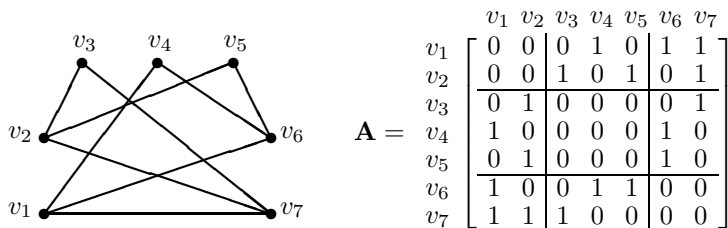


Рис. 1.13

Своеобразным аналогом матрицы смежности является *матрица Кирхгофа* $\mathbf{K} = [k_{i,j}]$, определяемая как квадратная матрица порядка n , элементы которой

$$k_{i,j} = \begin{cases} -1, & \text{если вершины } v_i \text{ и } v_j \text{ смежны,} \\ 0, & \text{если вершины } v_i \text{ и } v_j \text{ не смежны,} \\ \deg v_i, & \text{если } i=j. \end{cases}$$

Связь между матрицей смежности $\mathbf{A}$ и матрицей Кирхгофа $\mathbf{K}$ имеет вид $\mathbf{K} = \mathbf{D} - \mathbf{A}$, где $\mathbf{D} = \text{diag}(\deg v_1, \deg v_2, \dots, \deg v_n)$, т.е. является матрицей, диагональные элементы которой равны степеням соответствующих вершин. Важная отличительная особенность матрицы Кирхгофа (как и любой другой матрицы, у которой сумма элементов каждой строки и каждого столбца равна нулю) состоит в том, что алгебраические дополнения всех элементов матрицы равны между собой.

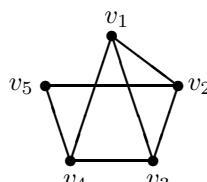
Поскольку изоморфные графы различаются лишь размет-

³В приведенной матрице смежности двудольного графа строки соответствуют вершинам одной доли, а столбцы - вершинам другой

кой (нумерацией) вершин соответствующего абстрактного графа, ясно, что их матрицы смежности (матрицы Кирхгофа) могут быть получены одна из другой путем некоторой перестановки строк и столбцов.

Матрица инцидентности. Определим *матрицу инцидентности* графа как прямоугольную матрицу $\mathbf{B}=[b_{i,j}]$ размера $n \times m$, в которой элемент $b_{i,j}$ равен 1, если вершина v_i инцидентна ребру e_j , и 0 в противном случае.

Строки матрицы $\mathbf{B}$ называют *векторами инцидентности* и обозначают как $\mathbf{B}_i$. На рис. 1.14 изображен тот же граф, что и на рис. 1.12 и приведена его матрица инцидентности.



$$\mathbf{B} = \begin{matrix} & \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix} = \begin{bmatrix} \mathbf{B}_1 \\ \mathbf{B}_2 \\ \mathbf{B}_3 \\ \mathbf{B}_4 \\ \mathbf{B}_5 \end{bmatrix}$$

Рис. 1.14

Как следует из определения, общее количество единиц в матрице инцидентности равно удвоенному числу ребер графа, количество единиц в любой строке равно степени соответствующей вершины, а столбцы содержат по две единицы. Поэтому между строками матрицы существует простая зависимость: элементы любой строки могут быть получены сложением одноименных элементов остальных строк по модулю 2. Используя понятие вектора инцидентности, можно записать $\mathbf{B}_i = \sum \mathbf{B}_j$, где $1 \leq j \leq n$ и $j \neq i$. Так, для приведенной выше матрицы $\mathbf{B}_1 = \mathbf{B}_2 \oplus \mathbf{B}_3 \oplus \mathbf{B}_4 \oplus \mathbf{B}_5 = [1, 0, 0, 1, 1, 0, 0] \oplus [0, 1, 0, 1, 0, 1, 0] \oplus [0, 0, 1, 0, 1, 1, 1] \oplus [0, 0, 0, 0, 0, 0, 1] = [1, 1, 1, 0, 0, 0, 0]$.

Матрица инцидентности несвязного графа, как и матрица смежности, может быть приведена к блочно-диагональному виду, где каждый диагональный блок является матрицей инцидентности некоторой компоненты связности.

Поскольку в графе отсутствуют параллельные ребра, то скалярное произведение любой пары векторов инцидентности

$B_i B_j$ равно единице, если вершины v_i и v_j смежны, и нулю, если эти вершины несмежны. Следовательно, произведение BB^T должно совпадать с матрицей смежности графа за исключением диагональных элементов, которые равны степеням соответствующих вершин. Это означает, что матрица смежности и матрица инцидентности графа связаны соотношением $A=BB^T-D$, где $D=\text{diag}(\deg v_1, \deg v_2, \dots, \deg v_n)$.

Применительно к матрицам инцидентности изоморфных графов также справедливо утверждение о том, что они могут быть получены одна из другой путем некоторой перестановки строк и столбцов.

Кроме рассмотренных для представления графов используют и другие матрицы. Некоторые из них описаны в следующих разделах.

При решении многих задач наряду с матричным удобным оказывается представление графов с помощью различных списков. В первую очередь это *список ребер* (*список инцидентности*) и *список вершин* (*список смежности*). Так, для графа на рис. 1.14 эти списки имеют вид:

Список ребер

e_1	e_2	e_3	e_4	e_5	e_6	e_7
v_1	v_1	v_1	v_2	v_2	v_3	v_4
v_2	v_3	v_4	v_3	v_4	v_4	v_5

Список вершин

v_1	v_2, v_3, v_4
v_2	v_1, v_3, v_4
v_3	v_1, v_2, v_4
v_4	v_1, v_2, v_3, v_5
v_5	v_2

В списке ребер каждое ребро представлено парой конечных вершин, а в списке смежности для каждой вершины указаны все смежные с ней. Можно считать, что список ребер является компактной записью матрицы инцидентности, если в каждом столбце заменить обе единицы обозначением соответствующих вершин (строк), а нули отбросить. Аналогично из матрицы смежности можно получить список вершин, если единицы в каждой строке заменить обозначением соответствующей вершины (столбца) и отбросить нули.

1.5 Диаметр, радиус и центр графа

Пусть v и w любые две вершины связного графа G . Назовем *расстоянием* между v и w длину самой короткой цепи, соединяющей эти вершины, и обозначим как $d(v, w)$. Тогда каждая вершина графа v может быть охарактеризована величиной $e(v) = \max_{w \in VG} d(v, w)$, которая называется *эксцентриситетом*. Нетрудно убедиться, что в графе G_2 на рис. 1.9 эксцентриситет всех вершин равен трем, тогда как в графе на рис. 1.14 имеем: $e(v_1) = e(v_2) = e(v_3) = e(v_5) = 2$, $e(v_4) = 1$. Очевидно, что в полном графе все вершины имеют эксцентриситет равный 1, а в простом цикле — $\lfloor |VC_n|/2 \rfloor$.

Максимальный эксцентриситет $d(G) = \max_{v \in VG} e(v)$ носит название *диаметра* графа, а минимальный $r(G) = \min_{v \in VG} e(v)$ — *радиуса* графа G . Диаметр графа на рис. 1.14 равен двум, радиус — единице, а в графе G_2 на рис. 1.9 и диаметр и радиус равны трем. В полном графе имеем $d(K_n) = r(K_n) = 1$, а в простом цикле — $d(C_n) = r(C_n) = \lfloor |VC_n|/2 \rfloor$.

Для любого графа справедливо соотношение $r(G) \leq d(G)$.

Вершина v называется *центральной*, если $e(v) = r(G)$. В графе на рис. 1.14 центральной является v_4 . Граф может иметь несколько таких вершин, а в некоторых графах все вершины оказываются центральными, например как в графах на рис. 1.9. В простой цепи при нечетном числе вершин только одна является центральной, а при четном их числе таких вершин две. В полном графе центральными являются все вершины. То же самое справедливо и для простого цикла. Множество всех центральных вершин называется *центром графа*.

Понятия центральной вершины и центра графа появились в связи с задачами оптимального размещения пунктов массового обслуживания, таких как больницы, пожарные части, пункты охраны общественного порядка и т.п., когда важно минимизировать наибольшее расстояние от любой точки некоторой сети до ближайшего пункта обслуживания.

1.6 Ориентированные графы

Наряду с неориентированными в теории графов рассматриваются *ориентированные графы* или *орграфы*.

Теоретико-множественное определение орграфа.

Пусть V непустое множество, например, $\{v_1, v_2, v_3, v_4, v_5\}$. Запишем его декартов квадрат $V^2 = V \times V$. Для нашего примера это множество всех упорядоченных пар вида (v_i, v_j) , $i, j = \overline{1, 5}$:

$$V^2 = \{(v_1, v_1), (v_1, v_2), (v_1, v_3), (v_1, v_4), (v_1, v_5), \\ (v_2, v_1), (v_2, v_2), (v_2, v_3), (v_2, v_4), (v_2, v_5), \\ (v_3, v_1), (v_3, v_2), (v_3, v_3), (v_3, v_4), (v_3, v_5), \\ (v_4, v_1), (v_4, v_2), (v_4, v_3), (v_4, v_4), (v_4, v_5), \\ (v_5, v_1), (v_5, v_2), (v_5, v_3), (v_5, v_4), (v_5, v_5)\}.$$

Возьмем произвольно некоторое $A \subseteq V^2$, например, $A = \{(v_1, v_2), (v_1, v_3), (v_2, v_5), (v_3, v_2), (v_4, v_1), (v_4, v_3), (v_5, v_4)\}$.⁴

Пару $\langle V, A \rangle$ называют *ориентированным графом* G , в котором V – множество *вершин*, A – множество *дуг*, являющееся подмножеством V^2 . Окончательно сформулируем определение так: *пара $\langle V, A \rangle$ называется ориентированным графом, если V – непустое множество элементов, называемых вершинами, A – множество упорядоченных пар различных элементов из V , называемых дугами.* Тогда для нашего примера имеем $G(V, A) = \langle \{v_1, v_2, v_3, v_4, v_5\}, \{(v_1, v_2), (v_1, v_3), (v_2, v_5),$

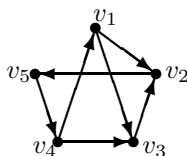


Рис. 1.15

$(v_3, v_2), (v_4, v_1), (v_4, v_3), (v_5, v_4)\}$, где (v_i, v_j) – дуга, у которой v_i – начальная, а v_j – конечная вершины. Изображение этого графа приведено на рис. 1.15. Очевидно, что множество дуг, соединяющих элементы множества вершин, отображает это множество само в себя. Поэтому орграф можно рассматривать как пару $\langle V, \Gamma \rangle$, где Γ – отображение, заданное на множестве V . Для графа на рис. 1.15 имеем

$$\Gamma(v_1) = \{v_2, v_3\}, \quad \Gamma(v_2) = \{v_5\}, \quad \Gamma(v_3) = \{v_2\}, \\ \Gamma(v_4) = \{v_1, v_3\}, \quad \Gamma(v_5) = \{v_4\}.$$

⁴В записи для V^2 элементы иножества A подчеркнуты.

Поскольку $\Gamma(v_i)$ обозначает множество конечных вершин всех дуг, исходящих из v_i , то через $\Gamma^{-1}(v_i)$ естественно обозначить множество начальных вершин всех дуг, входящих в v_i . Тогда описание графа, изображенного на рис. 1.15, может быть выполнено так:

$$\Gamma^{-1}(v_1) = \{v_4\}, \quad \Gamma^{-1}(v_2) = \{v_1, v_3\}, \quad \Gamma^{-1}(v_3) = \{v_1, v_4\}, \\ \Gamma^{-1}(v_4) = \{v_5\}, \quad \Gamma^{-1}(v_5) = \{v_2\}.$$

О количестве орграфов. Любому простому графу G можно поставить в соответствие ориентированный, придав каждому ребру одну из двух возможных ориентаций. Всего имеется 2^m различных вариантов ориентации множества из m ребер, т.е. из одного простого помеченного графа в общем случае можно получить 2^m ориентированных. Кроме того, вместо одной дуги можно взять две параллельные с противоположной ориентацией. Варьируя количество параллельных дуг и выбирая их в различных комбинациях, получаем дополнительное и весьма существенное расширение числа графов, производных от графа G .

Характеристики орграфов. При описании орграфов используют те же понятия, термины и характеристики, что и для неориентированных графов, но есть и своя специфика, обусловленная наличием ориентации дуг. Например, для орграфов сохраняются понятия смежности и инцидентности, определенные в разделе 1.1, но наряду с такой характеристикой, как степень вершины $\deg v$, используют еще две: *полустепень исхода* – число исходящих из v дуг $\deg^+ v = |\Gamma(v)|$ и *полустепень захода* – число входящих в v дуг $\deg^- v = |\Gamma^{-1}(v)|$, связанные очевидными соотношениями $\deg v = \deg^+ v + \deg^- v$ и $\sum_{i=1}^n \deg^- v_i = \sum_{i=1}^n \deg^+ v_i = |A| = m$, где m – число дуг в графе. Некоторые специфические понятия и характеристики орграфов рассмотрены в последующих разделах.

Матрицы орграфов. Матрица смежности орграфа определяется как квадратная матрица $A = [a_{i,j}]$ порядка n , в которой элемент $a_{i,j}$ равен 1, если в графе есть дуга (v_i, v_j) , и 0, если такой дуги нет. В качестве примера на рис. 1.16 пред-

ставлены оргграф, его матрица смежности и две степенные последовательности $\deg^+ v_i$ и $\deg^- v_i$.

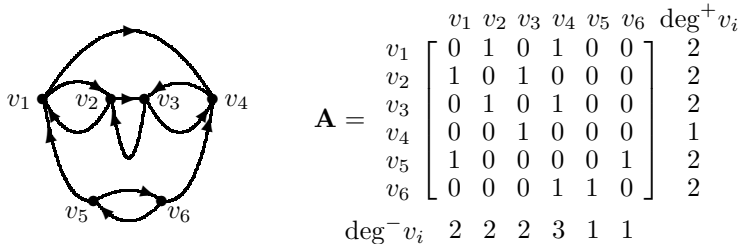


Рис. 1.16

Из определения следует (пример подтверждает это), что

- в общем случае матрица не симметрична, так как наличие дуги (v_i, v_j) совсем не означает, что в графе присутствует также и обратная дуга (v_j, v_i) ;
- количество единиц в любой строке равно полустепени исхода соответствующей вершины: $\sum_{j=1}^n a_{i,j} = \deg^+ v_i, \quad i = \overline{1, n}$;
- количество единиц в любом столбце равно полустепени захода соответствующей вершины: $\sum_{i=1}^n a_{i,j} = \deg^- v_j, \quad j = \overline{1, n}$;
- общее количество единиц в матрице равно числу дуг в графе: $\sum_{i=1}^n \sum_{j=1}^n a_{i,j} = m$.

Матрица инцидентности оргграфа определяется как прямоугольная матрица $\mathbf{B} = [b_{i,j}]$ размера $n \times m$, в которой

$$b_{i,j} = \begin{cases} 1, & \text{если } v_i \text{ является начальной вершиной дуги } a_j; \\ -1, & \text{если } v_i \text{ является конечной вершиной дуги } a_j; \\ 0, & \text{если вершина } v_i \text{ и дуга } a_j \text{ не инцидентны.} \end{cases}$$

В соответствии с этим определением матрица инцидентности, например, для графа на рис. 1.16 имеет вид

$$\mathbf{B} = \begin{array}{c} \begin{array}{cccccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 & e_{10} & e_{11} \end{array} \\ \begin{array}{l} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \left[\begin{array}{cccccccccccc} 1 & 1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 \end{array} \right] \end{array}$$

Свойства матриц инцидентности орграфа и неорграфа практически одинаковы, хотя есть и отличия. Так, в случае орграфа характер зависимости строк матрицы инцидентности чисто линейный, т.е. вектор инцидентности любой строки матрицы инцидентности связного орграфа равен простой сумме векторов инцидентности остальных строк.

Наряду с матрицами для описания орграфов используются и списки. Фактически пары $\langle V, \Gamma \rangle$, $\langle V, \Gamma^- \rangle$ и являются соответственно прямым и инвертированным списками вершин. В списке дуг должна быть отражена их ориентация.

Виды орграфов. Как и простые графы, ориентированные могут быть связными и несвязными, полными, k -дольными, регулярными и т.п. Существуют ориентированные аналоги цепей, циклов, деревьев. Типичным примером подобной аналогии являются так называемые *турниры* – ориентированные графы, в которых каждая пара вершин соединена только одной дугой. Любой турнир получается из простого полного графа, если каждому его ребру дать некоторую ориентацию. Название этого вида графов объясняется тем, что они позволяют наглядно отобразить результаты любого соревнования, когда каждый участник должен встретиться со всеми остальными и запрещены ничьи, например, как в большом теннисе или в волейболе. При этом участникам соответствуют вершины, а результатам встреч – дуги, направленные от победителя к побежденному. На рис. 1.17 изображены все непомечен-

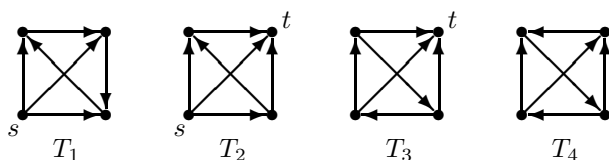


Рис. 1.17

ные турниры четвертого порядка. Опишем некоторые свойства турниров. В любом турнире может быть не более одной вершины s с нулевой полустепенью захода и не более одной вершины t с нулевой полустепенью исхода, называемых соответственно *исток*ом и *сток*ом. Действительно, наличие,

например, двух истоков означает, что в некотором реальном турнире есть два абсолютных победителя. Присутствие стока или истока существенно сокращает количество циклов в графе, так как такие вершины не могут принадлежать ни одному циклу. Поэтому число циклов в турнире $T(n)$, имеющем, как графы T_1 и T_3 на рис. 1.17, сток или исток, равно числу циклов в турнире $T(n-1)$, получающемся при удалении из $T(n)$ стока (истока) и всех инцидентных ему дуг. Проведем подобную операцию с графами T_1 и T_3 , убеждаемся, что "остаток" имеет один цикл. Наконец, в турнире, имеющем, как граф T_2 , и исток s и сток t , число циклов то же, что и в турнире, полученном после удаления s и t .

В приложениях часто встречаются *ациклические орграфы*, т.е. орграфы, не содержащие циклов. Один такой граф есть и на рис. 1.17, это T_2 . Более наглядные примеры даны на рис. 1.18. Понятно, что любой граф этого вида должен иметь

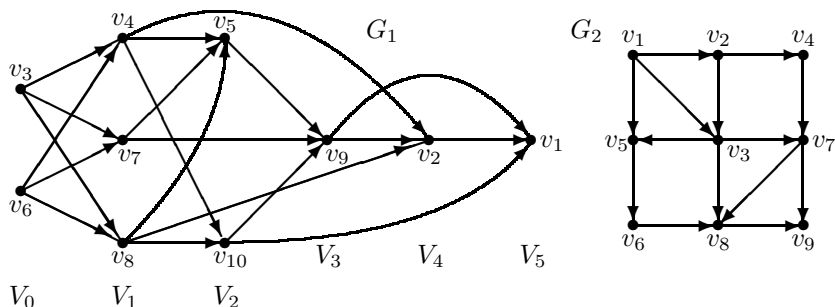


Рис. 1.18

по крайней мере один сток и один исток. Если к тому же разметка вершин графа такова, что для любой дуги (v_i, v_j) справедливо соотношение $i < j$, то говорят, что граф *топологически отсортирован*. Это значит, что, если вершины графа отображают промежуточные цели при выполнении некоторого комплекса работ, а дуги – действия, направленные на достижение этих целей, то топологическая сортировка определяет последовательность достижения целей. Так, из двух графов, изображенных на рис. 1.18, G_2 отсортирован, а G_1 – нет. Однако граф G_1 очень просто сортируется, если учесть отчетли-

во выраженное разбиение его вершин на подмножества, такое, что начальная вершина дуги всегда принадлежит подмножеству с меньшим индексом по сравнению с конечной вершиной. Достаточно пронумеровать сперва все вершины первого подмножества, потом второго, третьего и т.д. При этом внутри подмножества порядок произвольный, а это значит, что существует несколько вариантов такой разметки.

Следовательно, задача топологической сортировки сводится к отысканию разбиения множества вершин графа V на подмножества V_i (называемые уровнями, слоями, ярусами), такие, что, если вершина принадлежит подмножеству V_i , то любая следующая за ней входит в подмножество, индекс которого больше i . Иными словами, на множестве вершин необходимо найти числовую функцию, называемую *порядковой функцией графа*, которая определяется следующим образом.

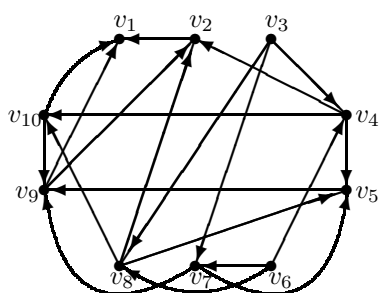
Пусть $G(V, \Gamma^{-1})$ ациклический граф, и найдены $r+1$ непесекающихся подмножеств множества V , определяемых как

$$\begin{aligned} V_0 &= \{ v \mid v \in V & \& \Gamma^{-1}(v) = \emptyset \}; \\ V_1 &= \{ v \mid v_i \in V - V_0 & \& \Gamma^{-1}(v) \subset V_0 \}; \\ V_2 &= \{ v \mid v \in V - (V_0 \cup V_1) & \& \Gamma^{-1}(v) \subset (V_0 \cup V_1) \}; \\ \dots & \\ V_r &= \{ v \mid v \in V - \bigcup_{j=0}^{r-1} V_j & \& \Gamma^{-1}(v) \subset \bigcup_{j=0}^{r-1} V_j \}, \end{aligned}$$

где r такое, что $\Gamma(v) = \emptyset$ для всех $v \in V_r$. Тогда порядковая функция графа $O(v)$, определенная на множестве вершин, сопоставляет каждой из них число, равное значению индекса подмножества V_k , элементом которого эта вершина является, т.е. если $v_i \in V_k$, то $O(v_i) = k$. В соответствии с приведенными соотношениями для получения результата следует повторно выполнять процедуру поиска вершин, у которых $\Gamma(v) = \emptyset$, сперва в заданном графе, потом в графе, полученном из заданного путем удаления вершин, найденных на первой итерации, затем в графе, полученном путем удаления вершин, найденных на первых двух итерациях, и т.д., пока не будут обработаны все вершины.

Покажем на примере способ отыскания порядковой функ-

ции, основанный на использовании матрицы смежности и известный как *алгоритм Демукрона*. Пусть граф выглядит, как показано на рис. 1.19 и имеет матрицу смежности A . Подсчи-



$$A = \begin{matrix} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10} \end{matrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{matrix}$$

Рис. 1.19

таем число единиц в каждом столбце матрицы и запишем найденные значения в первой строке табл. 1.1. Зафиксируем в столбце V_i вершины v_3, v_6 с нулевой суммой и отметим

Таблица 1.1

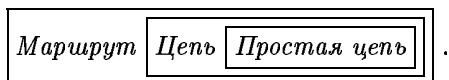
Итерации	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	V_i
1	3	3	0	2	3	0	2	2	3	2	$V_0 = \{v_3, v_6\}$
2	3	3		0	3		0	0	3	2	$V_1 = \{v_4, v_7, v_8\}$
3	3	1			0				2	0	$V_2 = \{v_5, v_{10}\}$
4	2	1							0		$V_3 = \{v_9\}$
5	1	0									$V_4 = \{v_2\}$
6	0										$V_5 = \{v_1\}$
$O(v_i)$	5	4	0	1	2	0	1	1	3	2	

третью и шестую строки матрицы. Вновь подсчитаем число единиц в столбцах, не учитывая при этом содержимое отмеченных строк. Результаты запишем во второй строке таблицы, опуская нули, полученные на первом шаге. В столбец V_i заносим вершины с нулевой суммой v_4, v_7, v_8 и отмечаем в матрице соответствующие строки. Для завершения процесса требуется еще четыре итерации (см. табл. 1.1). В результате последний столбец таблицы дает искомое разбиение⁵, а последняя строка – порядковую функцию.

⁵Изображение того же графа, отражающее полученное разбиение, дано на рис 1.18 (см. граф G_1).

1.7 Маршруты, цепи и простые цепи

В 1.3 даны понятия *цепи* и *простой цепи*, как чередующейся последовательности вершин и ребер, причем в случае цепи попарно различны все ребра, а в случае простой цепи – все вершины. *Маршрут* определяется также, за исключением ограничений на повторение вершин и ребер: допускается повторение как вершин, так и ребер. Таким образом, имеем триаду, отношения между элементами которой можно представить схемой:



Применительно к ориентированным графам существуют аналогичные понятия: *ориентированный маршрут*, *ориентированная цепь* и *ориентированная простая цепь*. Для последней часто используется другой более короткий термин *путь*. Приведенная терминология целиком соответствуют принятой, например, в [3]. Вообще же для рассматриваемых понятий используют и другие названия. Эта "терминологическая специфика" теории графов наглядно отражена в табл. 1.1.

Таблица 1.1

	Простой граф	Орграф	Библиогр.
1	Цепь	Путь	[1]
2	Простая цепь	Простой путь	
3	Элементарная цепь	Элементарный путь	
1	Маршрут	Ориентированный маршрут	[3]
2	Цепь	Ориентированная цепь	
3	Простая цепь	Простая орцепь	
1	Маршрут	Ориентированный маршрут	[6]
2	Цепь	Ориентированная цепь	
3	Путь	Ориентированный путь	
1	Маршрут	Ориентированный маршрут	[10]
2	Цепь	Цепь	
3	Простая цепь	Путь	

В дальнейшем слово *ориентированный* применительно к маршруту или цепи будем опускать, если из контекста ясно, о каком графе идет речь.

2 Связность в орграфах

2.1 Основные понятия

Говорят, что вершина w *достижима* из вершины v , если существует путь $(v \rightarrow \dots \rightarrow w)$. Если при этом v достижима из w , то о вершинах говорят, что они *взаимно достижимы*.

Орграф называется *сильно связным* или *сильным*, если любые две вершины в нем взаимно достижимы. Пример сильного орграфа приведен на рис. 2.1а. Поскольку в графе имеется орцикл $(v_1, v_4, v_2, v_3, v_5, v_1)$, включающий все вершины, то любые две из них взаимно достижимы.

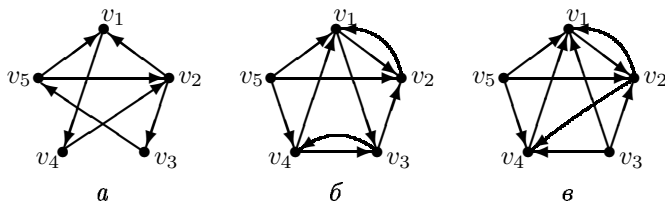


Рис. 2.1

Орграф называется *односторонне связным* или *односторонним*, если в любой паре его вершин хотя бы одна достижима из другой. Пример одностороннего графа дан на рис. 2.1б. В этом графе есть орцикл $(v_1, v_3, v_4, v_1, v_2, v_1)$, включающий четыре вершины. Все они взаимно достижимы. В отличие от этих вершин v_5 имеет нулевую полустепень захода, а это значит, что не существует путей, ведущих в v_5 . Вместе с тем из v_5 достижима любая из четырех остальных вершин.

Орграф называется *слабо связным* или *слабым*, если в нем любые две вершины соединены *полупутем*. В отличие от пути, когда все дуги одинаково ориентированы (от начальной вершины пути к конечной) в полупути могут быть и противоположно направленные дуги. Пример слабо связного орграфа приведен на рис. 2.1в. Очевидно, что в графе нет пути между вершинами v_3 и v_5 , но существует полупуть, состоящий из двух противоположно направленных дуг (v_3, v_4) и (v_5, v_4) .

2.2 Компоненты связности

Для орграфа определены три типа компонент связности: *сильная компонента* - максимальный сильный подграф, *односторонняя компонента* - максимальный односторонний подграф и *слабая компонента* - максимальный слабый подграф.

Понятие сильной компоненты иллюстрирует рис. 2.2. Граф

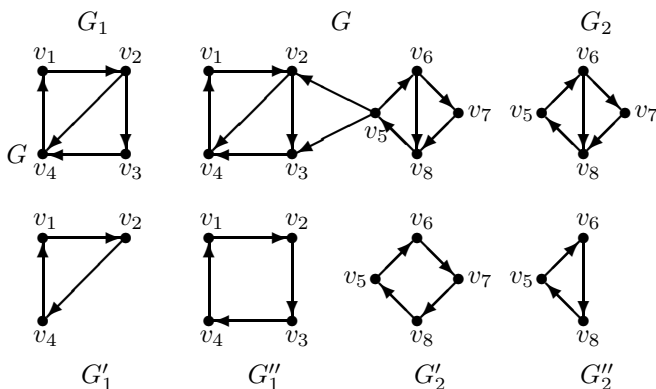


Рис. 2.2

G , который является односторонне связным, содержит шесть сильных подграфов $G_1, G'_1, G''_1, G_2, G'_2, G''_2$, из которых только два G_1 и G_2 являются сильными компонентами. Аналогично иллюстрируется понятие односторонней компоненты. В рассматриваемом случае односторонняя компонента совпадает с самим графом. Если же сменить, например, ориентацию дуги (v_4, v_1) на противоположную, то получим слабосвязный граф с двумя односторонними компонентами, одна из которых образована вершинами $v_1 - v_4$, а другая - вершинами $v_2 - v_8$.

Слабая компонента эквивалентна компоненте связности.

2.3 Конденсация орграфа

На основе сильных компонент определяется еще один вид графов - *граф конденсации*. Пусть $S_1, S_2, \dots, S_k$ - сильные компоненты графа $G(V, E)$. Конденсацией графа G называется

граф $G^*(V^*, E^*)$, множеством вершин которого V^* служит множество $\{S_1, S_2, \dots, S_k\}$ всех сильных компонент графа G , а дуга (S_i, S_j) является элементом множества E^* , если в графе G есть по крайней мере одна дуга, идущая от некоторой вершины компоненты S_i к одной из вершин компоненты S_j . Пример орграфа и его конденсации дан на рис. 2.3.

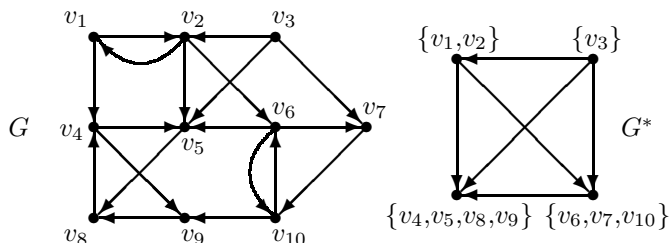


Рис. 2.3

Ясно, что граф конденсации G^* не может содержать циклов, так как все вершины любого цикла взаимно достижимы и значит принадлежат некоторой сильной компоненте, а это противоречит определению конденсации.

2.4 Отыскание сильных компонент

Пусть $R(v)$ - множество вершин (n, m) -графа, достижимых из вершины v . Тогда, поскольку $\Gamma(v)$ является множеством вершин, достижимых из v с использованием путей длины 1, $\Gamma^2(v)$ - множеством вершин, достижимых из v с использованием путей длины 2, $\Gamma^3(v)$ - с использованием путей длины 3 и т.д., то $R(v)$ можно представить в виде

$$R(v) = \{v\} \cup \Gamma(v) \cup \Gamma^2(v) \cup \dots \cup \Gamma^p(v).$$

Операции объединения выполняются последовательно слева направо, до тех пор, пока результирующее множество не перестанет увеличиваться. Количество операций зависит от графа, но, очевидно, что $p < n$.

Обобщая, можно говорить о множестве вершин $R(X)$, достижимых (в совокупности) не из единственной вершины, а

из целой группы вершин $X \subset V$, которое определяется как

$$R(X) = \bigcup_{v \in X} R(v).$$

Пусть $Q(v)$ - множество вершин (n, m) -графа, из которых достигается вершина v . Тогда, поскольку $\Gamma^{-1}(v)$ является множеством вершин, из которых v достигается с использованием путей длины 1, $\Gamma^{-2}(v)$ - множеством вершин, из которых v достигается с использованием путей длины 2, $\Gamma^{-3}(v)$ - с использованием путей длины 3 и т.д., то $Q(v)$ можно представить в виде

$$Q(v) = \{v\} \cup \Gamma^{-1}(v) \cup \Gamma^{-2}(v) \cup \dots \cup \Gamma^{-p}(v).$$

Операции объединения выполняются последовательно слева направо, до тех пор, пока результирующее множество не перестанет увеличиваться. Количество операций зависит от графа, но, очевидно, что $p < n$.

В качестве объекта, достижимого из других вершин, можно рассматривать не единственную вершину, а целую группу таковых. Тогда получаем множество

$$Q(X) = \bigcup_{v \in X} Q(v).$$

Если $R(v)$ это множество вершин, достижимых из v , а $Q(w)$ - множество вершин, из которых достигается w , то $R(v) \cap Q(w)$ является множеством вершин, принадлежащих всем путям, идущим из v в w . В том случае, когда $v = w$, пересечение $R(v) \cap Q(v)$ определяет множество (включая и v) взаимно достижимых вершин и, следовательно, некоторую сильную компоненту графа.

С учетом изложенного, можно предложить следующую процедуру отыскания всех сильных компонент графа $G(V, \Gamma)$.

1. Формируем описание графа в виде $\Gamma^+(v_i)$ и $\Gamma^-(v_i)$ для всех $v_i \in V$.
2. Выбираем некоторую вершину v и находим $R(v)$ и $Q(v)$.

3. Получаем $VS = R(v) \cap Q(v)$. Вершинно-порожденный подграф на множестве VS является сильной компонентой.
4. Если множество $V - VS$ пусто, то все сильные компоненты найдены, в противном случае строим вершинно-порожденный подграф на $V - VS$ и переходим к п.1.

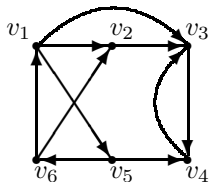


Рис 2.4

Рассмотрим пример на использование описанного подхода. Найдем сильные компоненты графа, изображенного на рис. 2.4.

Описания графа с использованием прямого и обратного отображений выглядят так:

$$\begin{aligned}
 \Gamma(v_1) &= \{v_2, v_3, v_5\}, & \Gamma(v_2) &= \{v_3\}, & \Gamma(v_3) &= \{v_4\}, \\
 \Gamma(v_4) &= \{v_3\}, & \Gamma(v_5) &= \{v_4, v_6\}, & \Gamma(v_6) &= \{v_1, v_2\}. \\
 \Gamma^{-1}(v_1) &= \{v_6\}, & \Gamma^{-1}(v_2) &= \{v_1, v_6\}, & \Gamma^{-1}(v_3) &= \{v_1, v_2, v_4\}, \\
 \Gamma^{-1}(v_4) &= \{v_3, v_5\}, & \Gamma^{-1}(v_5) &= \{v_1\}. & \Gamma^{-1}(v_6) &= \{v_5\}.
 \end{aligned}$$

Находим множества $R(v)$, $Q(v)$ и $R(v) \cap Q(v)$ для произвольно выбранной вершины, например для v_1 :

$$R(v_1) = \{v_1\} \cup \{v_2, v_3, v_5\} \cup \{v_3, v_4, v_6\} = \{v_1, v_2, v_3, v_4, v_5, v_6\},$$

$$Q(v_1) = \{v_1\} \cup \{v_6\} \cup \{v_5\} = \{v_1, v_5, v_6\},$$

$S_1 = R(v_1) \cap Q(v_1) = \{v_1, v_5, v_6\}$ - множество вершин первой сильной компоненты графа.

Множество вершин, не относящихся к найденной сильной компоненте, равно разности $V - S_1 = \{v_2, v_3, v_4\}$. Для поиска следующей сильной компоненты возьмем одну из них, например v_2 . В результате получим:

$$R(v_2) = \{v_2\} \cup \{v_3\} \cup \{v_4\} = \{v_2, v_3, v_4\}, \quad Q(v_2) = \{v_2\},$$

$S_2 = R(v_2) \cap Q(v_2) = \{v_2\}$ - единственная вершина, входящая во вторую сильную компоненту.

Далее ведем поиск на множестве $V - S_1 - S_2 = \{v_3, v_4\}$. В результате получаем:

$$R(v_3) = \{v_3\} \cup \{v_4\} = \{v_3, v_4\}, \quad Q(v_3) = \{v_3\} \cup \{v_4\} = \{v_3, v_4\},$$

$S_3 = R(v_3) \cap Q(v_3) = \{v_3, v_4\}$ - вершины третьей и последней сильной компоненты графа.

2.5 Матрицы достижимостей

Определим *матрицу (прямых) достижимостей* $\mathbf{R} = [r_{ij}]$ как квадратную $(0,1)$ -матрицу порядка $|G|$, в которой $r_{ij}=1$, если вершина v_j достижима из v_i , и $r_{ij}=0$ - в противном случае. Поскольку каждая вершина достижима из самой себя, диагональные элементы в $\mathbf{R}$ равны 1.

Матрица достижимостей может быть получена следующим способом. Отыскиваем множества $R(v_i)$ для всех вершин графа и полагаем $r_{ij}=1$, если $v_j \in R(v_i)$, и $r_{ij}=0$ в противном случае. Для примера из предыдущего раздела имеем:

$$\left. \begin{aligned} R(v_1) &= \{v_1, v_2, v_3, v_4, v_5, v_6\}, \\ R(v_2) &= \{v_2, v_3, v_4\}, \\ R(v_3) &= \{v_3, v_4\}, \\ R(v_4) &= \{v_3, v_4\}, \\ R(v_5) &= \{v_1, v_2, v_3, v_4, v_5, v_6\}, \\ R(v_6) &= \{v_1, v_2, v_3, v_4, v_5, v_6\}, \end{aligned} \right\} \Rightarrow \mathbf{R} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Матрицу обратных достижимостей $\mathbf{Q}=[q_{ij}]$ определим как квадратную $(0,1)$ -матрицу порядка $|G|$, в которой $q_{ij}=1$, если вершина v_i достижима из v_j , и $q_{ij}=0$ - в противном случае. Диагональные элементы в $\mathbf{Q}$, как и в $\mathbf{R}$, равны 1.

Матрица $\mathbf{Q}$ может быть получена тем же способом, что и матрица $\mathbf{R}$. Ищем множества $Q(v_i)$ для всех вершин графа и полагаем $q_{ij}=1$, если $v_j \in Q(v_i)$, и $q_{ij}=0$ - в противном случае:

$$\left. \begin{aligned} Q(v_1) &= \{v_1, v_5, v_6\}, \\ Q(v_2) &= \{v_1, v_2, v_5, v_6\}, \\ Q(v_3) &= \{v_1, v_2, v_3, v_4, v_5, v_6\}, \\ Q(v_4) &= \{v_1, v_2, v_3, v_4, v_5, v_6\}, \\ Q(v_5) &= \{v_1, v_5, v_6\}, \\ Q(v_6) &= \{v_1, v_5, v_6\}, \end{aligned} \right\} \Rightarrow \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

Однако из определений ясно, что столбец v_i в матрице $\mathbf{Q}$ совпадает со строкой v_i в матрице $\mathbf{R}$, поэтому $\mathbf{Q} = \mathbf{R}^T$.

Процедура отыскания сильных компонент может быть сделана более наглядной и эффективной, если использовать введенные выше матрицы и операцию их поэлементного умножения $\otimes$. Матрица $\mathbf{R} \otimes \mathbf{Q}$ концентрирует в себе информацию обо всех сильных компонентах графа:

$$\mathbf{R} \otimes \mathbf{Q} = \begin{matrix} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix} \sim \begin{matrix} & v_1 & v_5 & v_6 & v_2 & v_3 & v_4 \\ \begin{matrix} v_1 \\ v_5 \\ v_6 \\ v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix}$$

Действительно, строка v_i содержит единицы в тех столбцах v_j , для которых выполняется условие взаимной достижимости с v_i и которые наряду с v_i входят в состав одной и той же сильной компоненты. Ясно, что строки (и столбцы) матрицы, соответствующие вершинам одной и той же сильной компоненты, должны быть идентичны. Поэтому путем перестановки строк и столбцов матрицу $\mathbf{R} \otimes \mathbf{Q}$ можно привести к блочно-диагональному виду, когда каждая диагональная подматрица соответствует сильной компоненте и состоит из одних единиц, а все остальные элементы – нули.

2.6 Получение матрицы достижимостей

Естественными количественными характеристиками достижимости являются длины путей, связывающих вершины. При этом матрицу смежности $\mathbf{A}$ можно трактовать как матрицу достижимости при длине путей, равной единице, а матрицу $\mathbf{A}^2 = \mathbf{A} \times \mathbf{A} = [a_{ij}^{(2)}]$, где $a_{ij}^{(2)} = \bigvee_{k=1}^n a_{ik} a_{kj}$, как матрицу достижимости при длине путей, равной двум. Соответственно, матрицы $\mathbf{A}^3$, $\mathbf{A}^4$, $\mathbf{A}^5$ и т.д. можно рассматривать как матрицы достижимости при длине путей, равной (строго) трем, четырем, пяти и т.д.

Определим *ограниченную достижимость* как достижи-

мость на путях длины не более чем q . Соответствующая матрица называется матрицей ограниченных достижимостей $\mathbf{R}_q$. Очевидно, что $\mathbf{R}_1 = \mathbf{A} + \mathbf{I}$, где "+" – булевское сложение, $\mathbf{I}$ – единичная матрица, аналогично $\mathbf{R}_2 = \mathbf{R}_1 + \mathbf{A}^2 = \mathbf{I} + \mathbf{A} + \mathbf{A}^2$, и вообще $\mathbf{R}_q = \mathbf{I} + \mathbf{A} + \mathbf{A}^2 + \mathbf{A}^3 + \dots + \mathbf{A}^q = (\mathbf{I} + \mathbf{A})^q = \mathbf{R}_1^q$. Вместо последовательного вычисления $\mathbf{R}_2, \mathbf{R}_3, \dots, \mathbf{R}_q$ экономичнее получать $\mathbf{R}_2, \mathbf{R}_4, \mathbf{R}_8, \mathbf{R}_{16}, \dots$

$$\mathbf{R}_1^{2^l} = \mathbf{R}_1^{2^{(l-1)}} \mathbf{R}_1^{2^{(l-1)}}, \text{ где } l = 1, 2, 3, \dots$$

Возведение в квадрат повторяется, пока $2^l \leq q$. Если q не является степенью двойки, потребуется ряд дополнительных умножений. Например, матрица $\mathbf{R}_{13}$ может быть получена так: $\mathbf{R}_{13} = \mathbf{R}_8 \times \mathbf{R}_4 \times \mathbf{R}_1 = \mathbf{R}_1^8 \times \mathbf{R}_1^4 \times \mathbf{R}_1$. В том случае, когда требуется отыскать матрицу достижимости, возведение в квадрат следует выполнять, пока матрица не перестанет изменяться.

Более эффективное решение задачи основано на использовании транзитивного замыкания. Граф называется *транзитивным*, если при наличии дуг вида (v_i, v_k) и (v_k, v_j) в нем обязательно есть и дуга вида (v_i, v_j) . Пример транзитивного графа дан на рис. 2.5. *Транзитивным замыканием* графа G называют транзитивный граф G_{tc} , полученный путем добавления к G минимального количества дуг, необходимого для

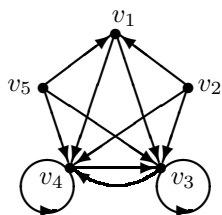


Рис. 2.5

обеспечения транзитивности. Из определения следует, что если в транзитивном графе имеется цепь, связывающая вершину v с вершиной w , то существует также и дуга (v, w) . Поэтому, если G_{tc} является графом транзитивного замыкания графа G , то его матрица смежности практически совпадает с матрицей достижимости для графа G , отличаясь только элементами главной диагонали, соответствующими одновершинным сильным компонентам G^6 . Следовательно, для нахождения $\mathbf{R}$ можно использовать алгоритмы отыскания транзитивного замыкания. Рассмотрим один из таких алгоритмов.

⁶В матрице смежности графа G_{tc} эти элементы равны нулю.

2.7 Алгоритм Уоршолла

Пусть G_0 орграф с множеством вершин $v_1, v_2, \dots, v_n$. Алгоритм Уоршолла позволяет получить последовательность графов G_i , $i=1, 2, \dots, n$, таких, что G_{i-1} является подграфом G_i , причем G_n это граф транзитивного замыкания G_0 . Граф G_i получается из G_{i-1} после обработки вершины v_i в соответствии со схемой, представленной на рис. 2.6.

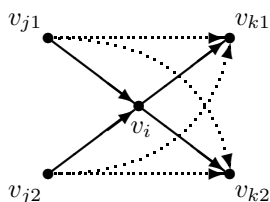


Рис. 2.6

Пусть, например, в графе G_{i-1} существуют дуги (v_{j1}, v_i) и (v_{j2}, v_i) , входящие в вершину v_i , и дуги (v_i, v_{k1}) и (v_i, v_{k2}) , выходящие из нее. Тогда при обработке вершины v_i каждая пара дуг (одна входящая, другая исходящая) "порождает" третью дугу, которая соединяет начало первой дуги пары с концом второй

дуги. В представленный на рисунке граф добавляются четыре дуги (v_{j1}, v_{k1}) , (v_{j1}, v_{k2}) , (v_{j2}, v_{k1}) и (v_{j2}, v_{k2}) (изображены пунктиром). Граф, полученный в результате подобной обработки вершины v_i , обозначается G_i .

Дадим формальное описание алгоритма. Пусть A – матрица смежности графа, $A^* = [a_{ji}]$ – матрица смежности транзитивного замыкания. Тогда запись алгоритма выглядит так:

```

begin                                { — У О Р Ш О Л Л — }
     $A^* := A$                         { Инициализация  $A^*$  }
    for  $i := 1$  to  $n$  do                { Перебор столбцов в  $A^*$  }
        for  $j := 1$  to  $n$  do            { Перебор строк в  $A^*$  }
            if  $a_{ji}^* = 1$  then          { Если есть дуга  $(v_j, v_i)$ , то }
                for  $k := 1$  to  $n$  do    { "прибавить" в  $A^*$  }
                     $a_{jk}^* := a_{jk}^* \vee a_{ik}^*$  { строку  $i$  к строке  $j$  }
end                                    { — У О Р Ш О Л Л — }

```

Отличительной особенностью алгоритма является порядок обработки элементов A^* по столбцам. Это естественное следствие того, что обработка новой вершины не начинается пока не переберутся все дуги, входящие в очередную вершину, причем порядок этого перебора может быть произвольным.

Как следует из приведенного описания, алгоритм обеспечивает получение транзитивного замыкания всего за один проход (просмотр) вершин графа. Существенным достоинством является и то, что результат формируется путем преобразования элементов матрицы $\mathbf{A}^*$ без использования вспомогательных данных, т.е. алгоритм "работает на месте".

В качестве примера найдем транзитивное замыкание для графа, показанного на рис. 2.7, с матрицей смежности $\mathbf{A}$ и матрицей достижимости $\mathbf{R}$ ⁷. Применяя алгоритм Уоршолла,

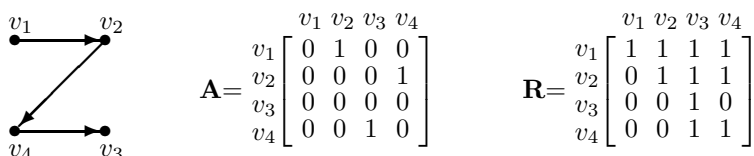


Рис. 2.7

получим следующий ряд матриц, иллюстрирующий процесс формирования матрицы транзитивного замыкания $\mathbf{A}^*$:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{\mathbf{A}(G_1)} \Rightarrow \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{\mathbf{A}(G_2)} \Rightarrow \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{\mathbf{A}(G_3)} \Rightarrow \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}_{\mathbf{A}(G_4)}$$

Соответствующие им графы изображены на рис. 2.8.

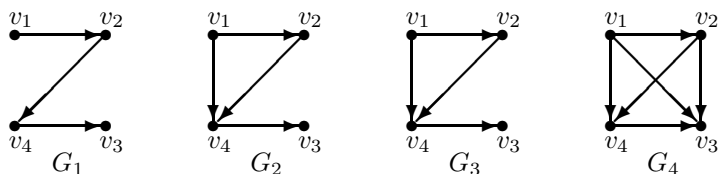


Рис. 2.8

Последний из этих графов G_4 является графом транзитивного замыкания для графа на рис. 2.7. Теперь, чтобы получить матрицу достижимости, достаточно расставить недостающие единицы на главной диагонали в матрице $\mathbf{A}^*(G) = \mathbf{A}(G_4)$.

⁷ Матрица $\mathbf{R}$ легко получается непосредственно по графу.

2.8 База графа

Базой называют множество вершин, из которого достижимы все вершины графа, причем в базе нет подмножеств, обладающих таким же свойством достижимости.

Формально базу графа $G(V, \Gamma)$ можно определить как множество $B \subset V$, такое, что $R(B) = V$, а $\forall (B' \subset B) \quad R(B') \neq V$.

Итак, база удовлетворяет двум условиям:

- каждая вершина графа достижима, по крайней мере, из одной вершины базы;
- в базе нет вершин, достижимых из других вершин базы.

Поэтому можно утверждать, что в ациклическом графе существует только одна база. Она состоит из всех вершин с полустепенью захода, равной 0, и легко отыскивается. Например, в графе на рис. 2.6 база состоит из двух вершин v_{j1} и v_{j2} , а в графе на рис. 2.7 – из одной вершины v_1 .

Из вышеприведенных условий следует также, что в базе не может быть двух вершин, принадлежащих одной и той же сильной компоненте. Поэтому, если граф содержит циклы, и, следовательно, хотя бы одну сильную компоненту, причем эта компонента образует вершину конденсации, у которой $\deg v^* = 0$, то в графе есть несколько баз. Это следствие того, что в базу можно включить любую (но только одну!) вершину такой компоненты. Например, в графе G на рис. 2.2 базу конденсации образует сильная компонента $\{v_5, v_6, v_7, v_8\}$, значит имеется четыре одновершинных базы: $\{v_5\}$, $\{v_6\}$, $\{v_7\}$ и $\{v_8\}$. В общем же случае число баз равно произведению чисел вершин в сильных компонентах, образующих базу конденсации.

С учетом изложенного поиск баз в орграфе можно проводить в следующем порядке:

1. Найти все сильные компоненты графа.
2. Построить его конденсацию.
3. Найти базу конденсации.
4. Из каждой сильной компоненты, образующей вершину базы конденсации, взять по одной вершине.

3 Деревья

Деревья являются простым, но очень важным видом графов. С их помощью легко описывается структура самых различных объектов: организаций и учреждений, книг и документов, математических формул, химических соединений, компьютерных файловых систем, программ и многое другое.

Считается, что первым использовал понятие дерева Кирхгофф в 1847г. при исследовании электрических цепей. Спустя десятилетие химик Кэли ввел термин "дерево" при изучении структуры углеводородных соединений и получил первые важные результаты в этом разделе теории графов.

3.1 Основные понятия

Неориентированные деревья. На рис. 3.1 даны три изображения одного и того же неориентированного дерева. Наиболее соответствует названию вариант *в*, но обычно при изо-

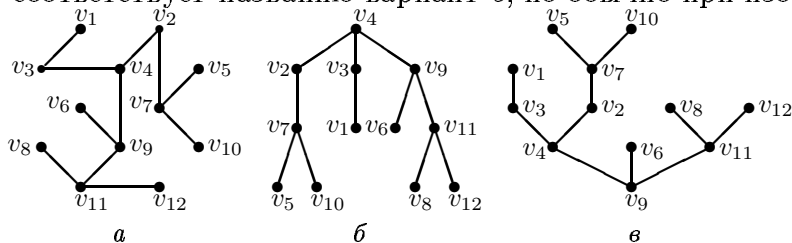


Рис. 3.1

бражении деревьев используют варианты *а* и *б*. Из рисунка становятся понятными термины *корень* и *лист*, употребляемые при рассмотрении подобных графов. Листом (*висячей вершиной*) называют вершину, степень которой равна 1, если она не рассматривается как корень. В качестве корня в неориентированном дереве можно принять любую вершину.

Существует несколько эквивалентных определений неориентированного дерева, каждое из которых отражает различные свойства последнего. Приведем некоторые из них.

- 1) *Дерево* – это связный граф без циклов.
- 2) *Дерево* – это граф, в котором любая пара вершин связана единственной цепью. $\triangleright$ Действительно, наличие двух и более цепей, соединяющих некоторую пару вершин, означает присутствие циклов, образованных несовпадающими частями таких цепей. $\triangleleft$
- 3) *Дерево* – это связный граф, имеющий n вершин и $n-1$ ребер. $\triangleright$ Непосредственной проверкой устанавливаем справедливость этого утверждения при $n=1, 2, 3$. Допустим, что оно верно для дерева с $n-1$ вершинами, и рассмотрим случай, когда дерево T имеет n вершин. Если в нем удалить любое ребро, то в силу утверждения 2, получим два дерева с $n_1 \leq (n-1)$ и $n_2 \leq (n-1)$ вершинами. По индуктивному предположению число ребер в них n_1-1 и n_2-1 соответственно. Поэтому T содержит $[(n_1-1)+(n_2-1)]+1=n_1+n_2-1=n-1$ ребер. $\triangleleft$

Естественным расширением понятия дерева является *лес* – несвязный граф, все компоненты которого – деревья.

Любому связному графу можно поставить в соответствие некоторое дерево, называемое *остовным деревом* или *остовом* этого графа. Остов графа – это его минимальный связный остовный подграф. В общем случае граф может иметь несколько остовов. В качестве примера на рис. 3.2а представлен неориентированный граф и все его остовы (б, в, г).

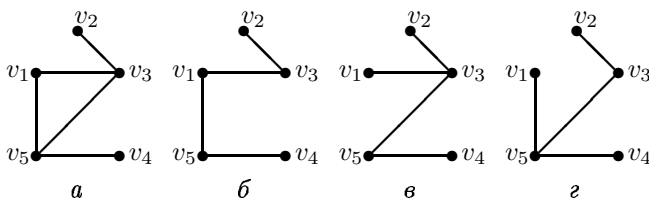


Рис. 3.2

Аналогично, любому несвязному графу можно поставить в соответствие *остовный лес* – максимальный остовный подграф, не содержащий циклов.

Ориентированные деревья. *Ориентированное дерево это граф, в котором полустепень захода у одной вершины, называемой корнем, равна нулю, а у остальных – единице.*

На рис. 3.3а,б,в изображены три ориентированных дерева.

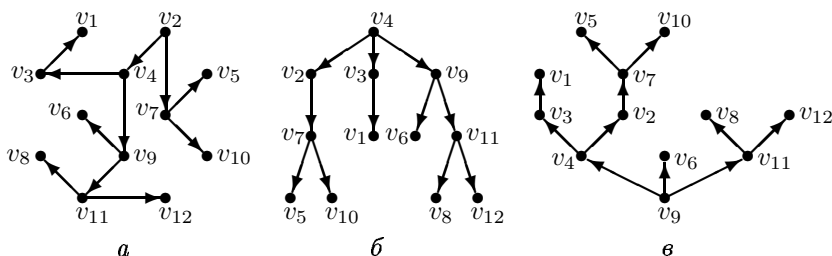


Рис. 3.3

И хотя по конфигурации они идентичны (если не учитывать ориентацию дуг) соответствующим изображениям а,б,в одного и того же дерева на рис. 3.1, это совершенно различные деревья уже потому, что у них разные корни v_2, v_4, v_9 . Неко-

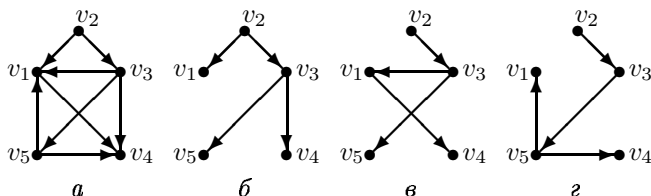


Рис. 3.4

торые связные орграфы также имеют остовы. На рис. 3.4а дан пример такого графа и приведены три его остова (б,в,г). Графы, изображенные на рис. 3.5, остова не имеют. В графе

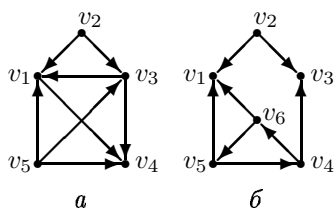


Рис. 3.5

G_1 есть две вершины с нулевой полустепенью захода (v_2, v_5), что находится в явном противоречии с определением ордерова. В графе G_2 такая вершина одна (v_2), но зато есть две вершины, у которых полустепень захода больше единицы (v_1, v_3), а это также противоречит определению ордерова.

3.2 Описание деревьев

Матричное представление. Дерево, как и любой другой граф, можно описать с помощью матриц. В качестве примера ниже приведены матрицы смежности **A** и инцидентности **B** для ордерова, изображенного на рис. 3.4,*г* :

$$\mathbf{A} = \begin{array}{c} \begin{array}{ccccc} & v_1 & v_2 & v_3 & v_4 & v_5 \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \end{array} \quad \mathbf{B} = \begin{array}{c} \begin{array}{cccc} & a_1 & a_2 & a_3 & a_4 \\ \begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{array} & \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 \end{bmatrix} \end{array} \end{array}.$$

Отметим особенности таких матриц в случае деревьев.

Отношение числа ребер дерева к числу его вершин, равно $(n-1)/n$, минимальное для связного графа, поэтому матрица смежности дерева весьма разрежена (соотношение единиц и нулей в ней равно $(n-1):(n^2-n+1) \approx 1/n$ для ориентированного дерева и $2(n-1):(n^2-2n+2) \approx 2/n$ для неориентированного).

Матрица инцидентности дерева имеет размер $n \times (n-1)$, т.е. близка к квадратной, а фактически таковой и является, если принять во внимание ее избыточность по строкам. Действительно, удалив любую строку, получаем "эквивалентную" квадратную матрицу, по-прежнему полностью характеризующую граф. Кроме того, *матрица инцидентности любого дерева путем переупорядочения строк и столбцов может быть приведена к нижней трапецевидной матрице, когда одна из единиц столбца i находится в строке i , а другая – в одной из нижележащих строк.* В связи с важностью данного утверждения рассмотрим соответствующий алгоритм.

Пусть T – заданное дерево. Выбираем любую висячую вершину v и присваиваем ей (и инцидентному ребру) номер 1. Затем рассматриваем дерево $T_1 = T - v$ и выполняем с ним ту же операцию. Теперь уже другая висячая вершина u получает вместе с инцидентным ей ребром номер 2. На следующем шаге имеем дерево $T_2 = T_1 - u$ и т.д. Одновременно с обработкой вершин и ребер формируем столбцы матрицы инцидентности.

Выбранной вершине соответствует единица на "главной диагонали". Вторая (нижележащая) единица в столбце появляется, когда соответствующая вершина дерева T , ставшая висячей, выбирается на очередном шаге. Процесс завершается за $n-1$ шагов, причем на последнем шаге дерево T_{n-1} состоит из единственного ребра, концы которого получают (произвольно) номера $n-1$ и n . В качестве примера на рис. 3.6 даны три

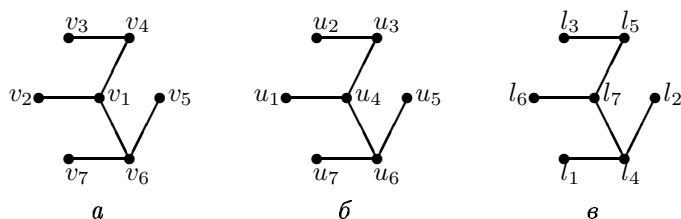


Рис. 3.6

варианта нумерации одного и того же графа: a – начальная, а две другие ($б, в$) получены описанным выше способом. Соответствующие матрицы инцидентности для a и $б$ приведены ниже. Фактически, для получения $B_б$ в матрице B_a выпол-

$$B_a = \begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{matrix} \quad B_б = \begin{matrix} & e_1 & e_4 & e_2 & e_5 & e_6 & e_3 \\ \begin{matrix} u_1(v_2) \\ u_2(v_3) \\ u_3(v_4) \\ u_4(v_5) \\ u_5(v_7) \\ u_6(v_6) \\ u_7(v_1) \end{matrix} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

нено переупорядочение строк по схеме: $1 \rightarrow 7, 2 \rightarrow 1, 3 \rightarrow 2, 4 \rightarrow 3, 5 \rightarrow 4, 7 \rightarrow 5$ и переупорядочение столбцов по схеме: $2 \rightarrow 3, 3 \rightarrow 6, 4 \rightarrow 2, 5 \rightarrow 4, 6 \rightarrow 5$. Теперь рассмотрим квадратную матрицу, которая получается, если удалить последнюю строчку матрицы $B_б$. Очевидно, что определитель этой матрицы равен 1. Седьмая строка $B_б$ соответствует первой строке B_a , поэтому определитель матрицы, полученной при вычеркивании первой строки в B_a , (с учетом числа перестановок строк и столбцов) равен -1 .

Ясно, что вышеописанная процедура позволяет, заранее выбрав некоторую вершину, получить такой вариант нумерации, когда эта вершина имеет максимальный номер, а матрица инцидентности графа (после удаления последней строки, соответствующей выбранной вершине) оказывается треугольной с единицами на главной диагонали.

Применительно к матрице инцидентности дерева это означает, что после удаления в ней любой строки получается квадратная матрица, определитель которой по абсолютной величине равен 1. Иными словами, *любой минор порядка $n-1$ матрицы инцидентности дерева равен $+1$ или -1 .*

Код Прюфера. Каждое дерево может быть однозначно описано числовым кодом, содержащим $n-2$ элемента и называемым кодом Прюфера.

Алгоритм получения кода состоит в следующем. Пусть вершины дерева помечены (пронумерованы) числами от 1 до n . Отыскиваем вершину степени 1 с наименьшим номером и включаем в код номер смежной с ней вершины, после чего удаляем найденную вершину (вместе с ребром). С получившимся подграфом выполняем ту же операцию, повторяя ее пока не останутся только два ребра (не останется внутренних вершин). Процесс формирования кода представлен на рис. 3.7.

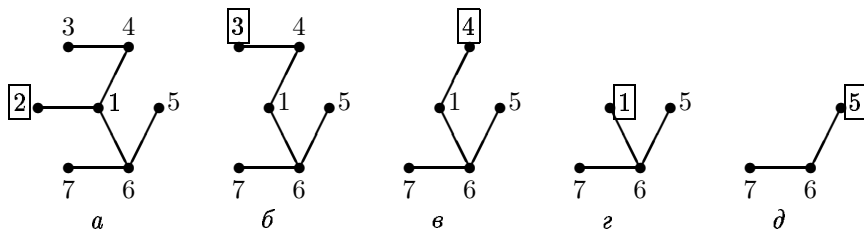


Рис. 3.7

В заданном графе (рис. 3.7а) среди вершин единичной степени минимальный номер имеет вершина с меткой 2. Смежная с ней вершина имеет метку 1. Следовательно, первая цифра кода Прюфера равна 1. Удаляя вершину с меткой 2, получим граф, изображенный на рис. 3.7,б. В этом графе среди вершин единичной степени минимальная метка равна 3, поэтому

вторая цифра кода 4. Выполнив еще три итерации, которым соответствуют графы, изображенные на рис. 3.7,в, рис. 3.7,г и рис. 3.7,д, получим дерево, состоящее из единственного ребра $\{7; 6\}$. Процесс завершен.

Результаты всех проделанных шагов могут быть представлены в виде табл. 3.1, последняя строчка в которой содержит искомый код Прюфера: 14166.

Таблица 3.1

Итерация	1	2	3	4	5
Граф на рис. 3.7	<i>a</i>	<i>б</i>	<i>в</i>	<i>г</i>	<i>д</i>
Минимальный номер вершины	2	3	4	1	5
Удаляемое ребро	$\{1;2\}$	$\{4;3\}$	$\{1;4\}$	$\{6;1\}$	$\{6;5\}$
Элемент кода	1	4	1	6	6

Восстановление дерева по коду выполняется также достаточно просто. Назовем *антикодом* упорядоченную по возрастанию последовательность номеров вершин, не вошедших в код Прюфера. Для рассмотренного примера это будет 2357.

Дерево строится путем последовательного добавления ребер. Очередное добавляемое ребро, начиная с первого, образуется парой вершин, номера которых стоят первыми в строке кода и в строке антикода. После этого использованные элементы строк вычеркиваются. Если номер, вычеркнутый из строки кода не содержится среди оставшихся в ней элементов, его следует добавить к строке антикода, не нарушая ее упорядоченность. Описанные действия повторяются с "остатками" строк кода и антикода, пока не будут вычеркнуты все элементы первой из них. При этом строка антикода будет содержать два элемента, образующих последнее ребро, которое следует добавить к формируемому дереву. В результате получаем список из $n-1$ ребер, соответствующий дереву, заданному кодом Прюфера.

В качестве примера выполним восстановление дерева по коду 14166. Соответствующий антикод, как уже было сказано выше, равен 2357. Поэтому первое ребро дерева будет $\{1; 2\}$. Вычеркивая 1 и 2, получаем 4166 в строке кода и 357 в строке

антикода. На следующей итерации вычеркиваем пару $\{4; 3\}$ и включаем в строку антикода 4 и т.д. Последовательность итераций может быть представлена табл. 3.2.

Таблица 3.2

Итерация	1	2	3	4	5	6
Строка кода	χ	4	χ	$\emptyset$	$\emptyset$	
Строка антикода	2	3	5	7		
		3	5	7		
			4	5	7	
				χ	5	7
					5	7
						6
						7
Добавляемое ребро	$\{1; 2\}$	$\{4; 3\}$	$\{1; 4\}$	$\{6; 1\}$	$\{6; 5\}$	$\{6; 7\}$

Анализируя список ребер, убеждаемся, что получено исходное дерево. Отметим, что порядок следования ребер тот же, что и в предыдущей таблице.

3.3 Задачи с деревьями

3.3.1 Перечисление остовных деревьев

Пусть G является (n, m) -графом с матрицей инцидентности $\mathbf{B}$ и требуется перечислить все его остовы. Очевидно, что множество остовов это некоторая часть множества остовных подграфов графа G , имеющих $n-1$ ребер. Матрицы инцидентности всех этих подграфов легко могут быть получены из матрицы $\mathbf{B}$. Действительно, любая матрица, составленная из $n-1$ различных столбцов матрицы $\mathbf{B}$, описывает некоторый $(n, n-1)$ -граф, являющийся по отношению к G остовным подграфом. Чтобы выяснить, остов этот подграф или нет, воспользуемся следующим критерием.

Если $(n, n-1)$ -граф является деревом, то модуль любого минора $n-1$ порядка матрицы инцидентности соответствующего орграфа равен 1 и равен 0, если граф не является деревом.

Под *соответствующим* орграфом для некоторого неориентированного графа будем понимать орграф, полученный путем придания произвольной ориентации ребрам этого графа (достаточно в каждом столбце матрицы инцидентности заменить одну из единиц на -1). Первая часть приведенного утверждения, касающаяся дерева, фактически была доказана в разделе 3.2. Докажем вторую часть.

▷ Пусть $(n, n-1)$ -граф не является деревом и, значит, несвязен. Тогда его матрица инцидентности состоит из нескольких независимых блоков, представляющих собой матрицы инцидентности компонент связности. То же самое относится и к матрице инцидентности соответствующего орграфа. Кроме того, строки этой матрицы линейно зависимы, причем, поскольку граф несвязный, такая зависимость существует в каждой группе строк, описывающей ту или иную компоненту связности. Удалив некоторую строку матрицы, получим квадратную подматрицу порядка $n-1$, определитель которой равен 0. Действительно, удаление одной строки может устранить линейную зависимость только для строк, относящихся лишь к одной компоненте связности. Строки, определяющие остальные компоненты остаются по-прежнему линейно зависимыми. Поэтому любой минор $n-1$ порядка в рассматриваемом случае равен 0, что и требовалось доказать. ◁

В соответствии с изложенным, процесс получения (перечисления) всех остовов графа $G(n, m)$, представленного матрицей инцидентности, можно представить следующим образом. Последовательно формируются сочетания из m по $n-1$ столбцов заданной матрицы, образуя матрицы инцидентности $(n-1)$ -реберных остовных подграфов графа G . Для каждой такой матрицы после обработки всех ее столбцов (одной из двух единиц столбца приписывается знак " $-$ ") подсчитывается значение произвольно выбранного минора $n-1$ порядка. Если получается $+1$ или -1 , то соответствующий остовный подграф является остовом графа $G(n, m)$ и его матрица запоминается, в противном случае подграф не остов и его матрица не учитывается. Процесс завершается после перебора всех C_m^{n-1} сочетаний столбцов матрицы инцидентности.

Вместо матрицы инцидентности можно сразу взять матрицу $\overline{B}_{-i}$ размера $(n-1) \times m$, представляющую собой матрицу инцидентности соответствующего орграфа с удаленной i -й строкой, и вычислять все ее миноры порядка $n-1$.

В качестве примера получим все остовы графа, представленного на рис. 3.8 и имеющего матрицу инцидентности B .

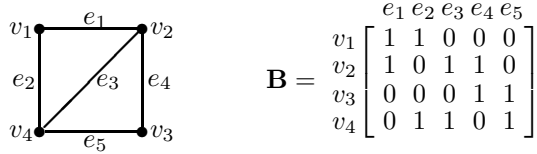


Рис. 3.8

Ниже приведена матрица $\overline{B}_{-1}$ и все ее миноры третьего порядка (их число равно $C_5^3=10$). Миноры M_1 и M_{10} равны 0.

$$\overline{B}_{-1} = \begin{matrix} & \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 \end{matrix} \\ \begin{matrix} v_2 \\ v_3 \\ v_4 \end{matrix} & \begin{bmatrix} -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & -1 & 0 & -1 \end{bmatrix} \end{matrix}$$

$$\begin{matrix} \begin{matrix} e_1 & e_2 & e_3 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & -1 & -1 \end{matrix} & =0, & \begin{matrix} e_1 & e_2 & e_4 \\ -1 & 0 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{matrix} & =1, \\ M_1 & & M_2 \end{matrix}$$

$$\begin{matrix} \begin{matrix} e_1 & e_2 & e_5 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & -1 \end{matrix} & =-1, & \begin{matrix} e_1 & e_3 & e_4 \\ -1 & 1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{matrix} & =1, & \begin{matrix} e_1 & e_3 & e_5 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & -1 \end{matrix} & =-1, & \begin{matrix} e_1 & e_4 & e_5 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{matrix} & =-1, \\ M_3 & & M_4 & & M_5 & & M_6 \end{matrix}$$

$$\begin{matrix} \begin{matrix} e_2 & e_3 & e_4 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \\ -1 & -1 & 0 \end{matrix} & =1, & \begin{matrix} e_2 & e_3 & e_5 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -1 \end{matrix} & =-1, & \begin{matrix} e_2 & e_4 & e_5 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & -1 \end{matrix} & =-1, & \begin{matrix} e_3 & e_4 & e_5 \\ 1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & -1 \end{matrix} & =0. \\ M_7 & & M_8 & & M_9 & & M_{10} \end{matrix}$$

Значит, остовные подграфы, образованные ребрами $\{e_1, e_2, e_3\}$ и $\{e_3, e_4, e_5\}$, остовами не являются, тогда как подграфы, определяемые минорами $M_2 - M_9$, остовы. Это наглядно подтверждает и рис. 3.9, где представлены все трехреберные остовные подграфы рассматриваемого графа и значения соответствующ-

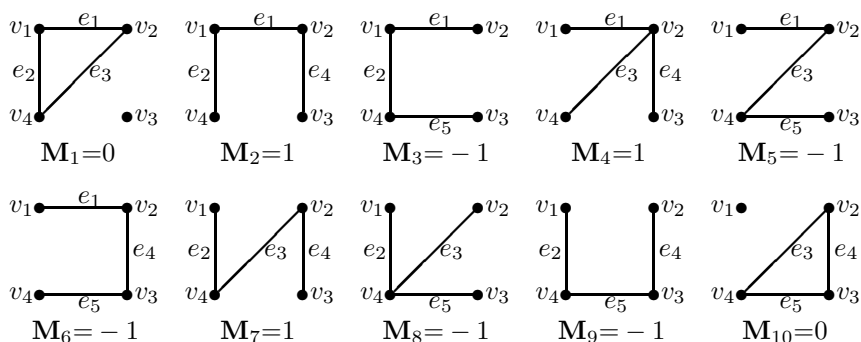


Рис. 3.9

щих им миноров третьего порядка матрицы $\overline{B}_{-1}$. Только восемь из них являются остовами, и для них абсолютная величина минора равна 1.

3.3.2 Пересчет остовных деревьев

Задача определения числа остовных деревьев графа естественным образом решается в процессе перечисления остовов. Однако, если требуется знать только количество остовов, а получать сами остовы нет необходимости, изложенный выше подход не рационален.

Эффективное решение рассматриваемой задачи можно получить на основе *теоремы Бине-Коши* для определителя произведения двух матриц, которая формулируется так.

Теорема 3.1 Если P матрица порядка $n \times t$, а Q матрица порядка $t \times n$, где $n \leq t$, то определитель матрицы PQ равен сумме произведений каждого минора порядка n матрицы P на соответствующий минор матрицы Q .

Соответствующим для минора порядка n матрицы P , образованного ее столбцами $i_1, i_2, \dots, i_n$, является минор того же порядка матрицы Q , образованный строками $i_1, i_2, \dots, i_n$ этой матрицы.

Пусть, например, $\mathbf{P} = \begin{bmatrix} 1 & 3 & -1 \\ -2 & 0 & 1 \end{bmatrix}$ и $\mathbf{Q} = \begin{bmatrix} 2 & -1 \\ 1 & 1 \\ -1 & 0 \end{bmatrix}$, тогда

в соответствии с теоремой Бине-Коши имеем:

$$|\mathbf{PQ}| = \begin{vmatrix} 1 & 3 \\ -2 & 0 \end{vmatrix} \cdot \begin{vmatrix} 2 & -1 \\ 1 & 1 \end{vmatrix} + \begin{vmatrix} 1 & -1 \\ -2 & 1 \end{vmatrix} \cdot \begin{vmatrix} 2 & -1 \\ -1 & 0 \end{vmatrix} + \begin{vmatrix} 3 & -1 \\ 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} -1 & 1 \\ -1 & 0 \end{vmatrix} = 22.$$

Перемножив матрицы непосредственно, получим:

$$\mathbf{PQ} = \begin{bmatrix} 1 & 3 & -1 \\ -2 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -1 \\ 1 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 6 & 2 \\ -5 & 2 \end{bmatrix} \text{ и, значит, } |\mathbf{PQ}| = 22.$$

Если $\mathbf{Q} = \mathbf{P}^T$, то в силу равенства соответствующих миноров матриц-сомножителей имеем $|\mathbf{PP}^T| = \mathbf{M}_1^2 + \mathbf{M}_2^2 + \dots + \mathbf{M}_{C_m}^2$, где справа от знака равенства стоит сумма квадратов всех миноров порядка n матрицы $\mathbf{P}$. Это позволяет утверждать, что *определитель квадрата матрицы порядка $n \times t$, где $n \leq t$, равен сумме квадратов всех ее миноров порядка n* . Следовательно, в том случае, когда любой минор порядка n в матрице принимает одно из трех значений: -1 , 0 или 1 , определитель квадрата матрицы оказывается равным числу ненулевых миноров. Из этого, в свою очередь, следует, что вместо перебора миноров матрицы $\overline{\mathbf{B}}_{-i}$ для получения числа остовов графа достаточно подсчитать определитель $|\overline{\mathbf{B}}_{-i} \times \overline{\mathbf{B}}_{-i}^T|$.

Например, для графа, изображенного на рис. 3.8, имеем

$$|\overline{\mathbf{B}}_{-1} \times \overline{\mathbf{B}}_{-1}^T| = \left| \begin{bmatrix} -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & -1 & -1 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \right| = \begin{vmatrix} 3 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 3 \end{vmatrix} = 8,$$

что совпадает с количеством, найденным ранее.

Поскольку определитель $|\overline{\mathbf{B}}_{-i} \times \overline{\mathbf{B}}_{-i}^T|$ равен алгебраическому дополнению элемента $k_{i,i}$ матрицы Кирхгофа, то в силу свойства последней⁸, решение задачи о числе остовов в окончательном виде может быть сформулировано в виде следую-

⁸Равенство алгебраических дополнений элементов матрицы.

щей теоремы, известной как *матричная теорема о деревьях*, или *теорема Кирхгофа*.

Теорема 3.2 *Число остовных деревьев связного графа порядка $n \geq 2$ равно алгебраическому дополнению любого элемента матрицы Кирхгофа.*

Воспользуемся теоремой для определения числа остовов полного графа K_n . В этом случае матрица Кирхгофа (порядка n) и алгебраическое дополнение любого ее диагонального элемента (порядка $n-1$) выглядят так:

$$K(K_n) = \begin{bmatrix} n-1 & -1 & \dots & -1 \\ -1 & n-1 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & n-1 \end{bmatrix}, \quad A_{i,i} = \begin{vmatrix} n-1 & -1 & \dots & -1 \\ -1 & n-1 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & n-1 \end{vmatrix}.$$

Вычтя первую строчку определителя из всех нижележащих и прибавив к первому столбцу остальные $n-2$, получим

$$A_{i,i} = \begin{vmatrix} n-1 & -1 & \dots & -1 \\ -n & n & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -n & 0 & \dots & n \end{vmatrix} = \begin{vmatrix} 1 & -1 & \dots & -1 \\ 0 & n & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & n \end{vmatrix} = n^{n-2}.$$

Таким образом число остовов полного графа K_n равно n^{n-2} , а поскольку остовы полного помеченного графа составляют все множество помеченных деревьев с n вершинами, то справедлива следующая теорема Кели

Теорема 3.3 *Число различных помеченных деревьев с n вершинами равно n^{n-2} .*

Этот результат может быть легко получен на основе кода Прюфера. Действительно, поскольку любое дерево, имеющее n вершин, помеченных числами от 1 до n , однозначно описывается $(n-2)$ -разрядным кодом, каждый разряд которого может принимать некоторое, вполне определенное значение из множества $\{1, 2, \dots, n\}$, и наоборот, каждому $(n-2)$ -разрядному коду, составленному на основе чисел из указанного множества, соответствует конкретное дерево, то общее число деревьев должно быть равно n^{n-2} .

3.4 Задача о кратчайшем остове графа

Пусть $G(V, E)$ – связный неориентированный граф, каждому ребру которого приписан некоторый вес $c_{i,j} > 0$. Требуется среди всех остовов графа найти по крайней мере один, с минимальной суммой весов ребер. Хотя, в реальной системе, которая описывается графом, вес ребра может соответствовать стоимости, трудоемкости, пропускной способности, вероятности некоторого события и т.п., чаще используют термин "длина ребра", а сама задача трактуется как задача отыскания остова с наименьшей суммой длин составляющих его ребер, или кратко, остова наименьшей длины (кратчайшего остова). Это одна из наиболее простых классических оптимизационных задач теории графов, которая считается полностью решенной. В настоящем разделе рассмотрим два наиболее известных алгоритма отыскания кратчайшего остова.

3.4.1 Алгоритм Краскала

Описание алгоритма. В основу алгоритма положена простая и интуитивно ясная идея: строить остов из наиболее "коротких" ребер, не допуская при этом образования циклов.

При выполнении алгоритма формируется последовательность графов $G_1 \rightarrow G_2 \rightarrow \dots \rightarrow G_i \rightarrow \dots \rightarrow G_{n-1}$, где

G_1 – остовный подграф, содержащий одно (самое короткое) ребро графа;

G_2 – остовный подграф, содержащий два (самых коротких) ребра графа;

G_i – лес, содержащий i самых коротких ребер, не образующих циклов;

G_{n-1} – искомым кратчайший остов.

Пусть граф представлен списком ребер предварительно упорядоченных по возрастанию веса.

Процесс начинается с самого короткого ребра $e_{кр}$, которое обязательно должно войти в кратчайший остов. Предположив обратное, мы сразу приходим к противоречию. Действительно,

пусть T – кратчайший остов, не содержащий $e_{кр}$. Ясно, что в графе $T + e_{кр}$ есть цикл. Удалив любое принадлежащее циклу ребро, отличное от $e_{кр}$, получаем остов, более короткий чем T и содержащий $e_{кр}$. Следовательно первое ребро списка безусловно включается в остов, образуя начальный фрагмент будущего остова.

Поскольку в графе нет параллельных ребер, присоединение к остову второго ребра не может привести к появлению цикла. Поэтому второе по длине ребро также должно входить в остов. Оно может либо "примкнуть" к уже существующему фрагменту либо даст начало новому (пока однореберному) фрагменту.

Для остальных ребер необходим анализ на предмет включения или невключения в остов. При этом возможны следующие четыре варианта:

- а) концы ребра не принадлежат ни одному фрагменту. Ребро должно быть включено в остов, образуя при этом новый фрагмент, которому следует присвоить очередной номер;
- б) только один из концов ребра принадлежит некоторому фрагменту. Ребро включается в остов, присоединясь к этому фрагменту;
- в) концы ребра принадлежат разным фрагментам. При включении ребра в остов происходит объединение этих фрагментов. "Новый" фрагмент получает меньший из номеров объединяемых фрагментов;
- г) концы ребра принадлежат одному фрагменту. Ребро нельзя включать в остов, так как это приводит к появлению цикла.

Построение остова завершится после присоединения к нему $n-1$ ребер. К этому моменту все фрагменты объединятся в один. Если же этого не происходит, хотя все m ребер графа просмотрены, а остов содержит менее чем $n-1$ ребер, то это значит, что граф несвязный.

Рассмотрим пример. Пусть требуется найти кратчайший остов графа, изображенного на рис. 3.10 с матрицей весов C .

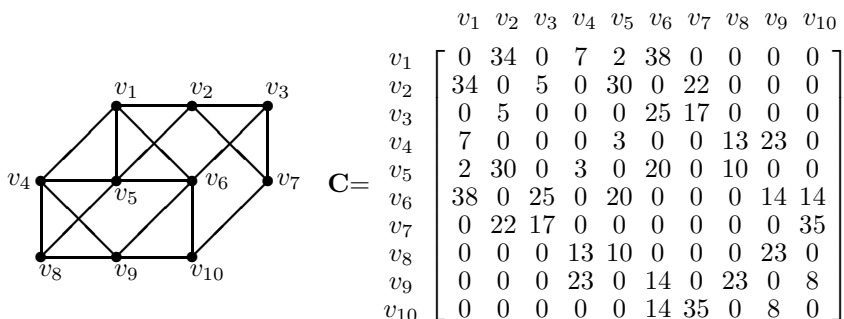


Рис. 3.10

На основе матрицы весов создаем упорядоченный список ребер в виде табл. 3.3, где столбцы e, a, b, l_{ab} содержат соответственно номера ребер, метки концов ребер и длины ребер. Два последних столбца заполняются в процессе построения кратчайшего остова. В столбце T_i отражается процесс создания, изменения и объединения фрагментов остова, а в столбце VT_i фиксируются множества вершин, принадлежащих различным фрагментам остова.

В соответствии с алгоритмом формируем первый фрагмент T_1 из ребра $e_1 = \{v_1, v_5\}$ (см. столбец T_i). Множество вершин, относящихся к T_1 , фиксируем в первой строке столбца VT_i . Ребро $e_2 = \{v_4, v_5\}$ имеет общий элемент с множеством VT_1 . Поэтому присоединим его к фрагменту T_1 , корректируя множество VT_1 , которое теперь содержит v_1, v_4, v_5 . Концы третьего ребра не принадлежат множеству VT_1 , поэтому создаем второй фрагмент T_2 на основе третьего ребра. Для него $VT_2 = \{v_2, v_3\}$. Четвертое ребро пропускаем, так как обе его вершины являются элементами VT_1 , а пятое ребро дает начало третьему фрагменту, и т.д. Процесс построения остова завершается на пятнадцатом ребре, которое объединяет два оставшиеся фрагмента. На рис. 3.11 представлены шесть из пятнадцати этапов построения остова. Число в верхнем левом углу каждой "картинки" соответствует табличному номеру анализируемого ребра (номеру итерации).

Таблица 3.3

e	a	b	l_{ab}	T_i	VT_i
1	v_1	v_5	2	$T_1 := \{v_1, v_5\}$	$\{v_1, v_5\}$
2	v_4	v_5	3	$T_1 := T_1 + \{v_4, v_5\}$	$\{v_1, v_4, v_5\}$
3	v_2	v_3	5	$T_2 := \{v_2, v_3\}$	$\{v_2, v_3\}$
4	v_1	v_4	7	—	—
5	v_9	v_{10}	8	$T_3 := \{v_9, v_{10}\}$	$\{v_9, v_{10}\}$
6	v_5	v_8	10	$T_1 := T_1 + \{v_5, v_8\}$	$\{v_1, v_4, v_5, v_8\}$
7	v_4	v_8	13	—	—
8	v_6	v_{10}	14	$T_3 := T_3 + \{v_6, v_{10}\}$	$\{v_6, v_9, v_{10}\}$
9	v_6	v_9	14	—	—
10	v_3	v_7	17	$T_2 := T_2 + \{v_3, v_7\}$	$\{v_2, v_3, v_7\}$
11	v_5	v_6	20	$T_1 := T_1 + T_3; \mathbb{K}_3$	$\{v_1, v_4, v_5, v_6, v_8, v_9, v_{10}\}$
12	v_2	v_7	22	—	—
13	v_8	v_9	23	—	—
14	v_4	v_9	23	—	—
15	v_6	v_3	25	$T_1 := T_1 + T_2; \mathbb{K}_2$	$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$
...	...	...	...	...	...

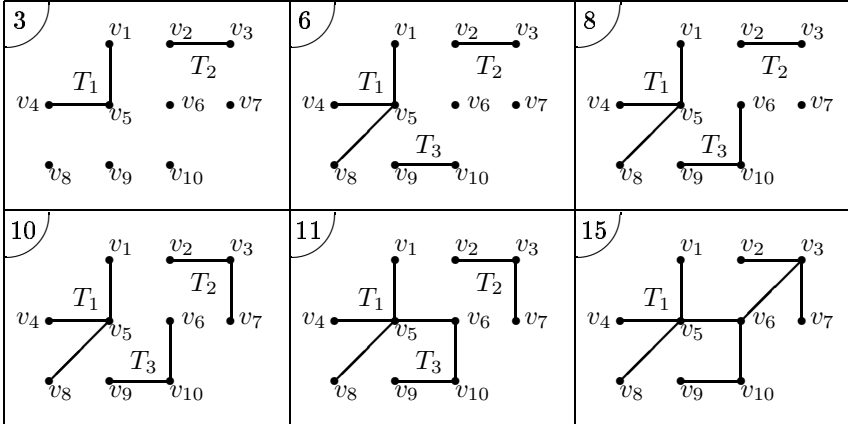


Рис. 3.11

Обоснование алгоритма. Пусть T_K – остовное дерево, полученное с помощью алгоритма Краскала, а T_S – кратчайший остов. Составим упорядоченный список ребер графа, отобразив вхождение или невхождение ребра в T_K и T_S знаком $+$ или $-$, как показано на рис. 3.12. Предположим, что у ребер с e_1 по e_{i-1} включительно знаки для T_K и T_S иден-

T_K	+	+	+	-	+	...	+	...	-	...
	e_1	e_2	e_3	e_4	e_5	...	e_i	...	e_k	...
T_S	+	+	+	-	+	...	-	...	+	...

Рис. 3.12

тичны, а первое несовпадение имеет место для e_i . Характер несовпадения может быть $\pm e_i$ или $\mp e_i$ ⁹. Допустим, что сочетание знаков имеет вид $\mp e_i$. Это значит, что ребро e_i входит в T_S и не входит в T_K . Поскольку в соответствии с алгоритмом Краскала очередное ребро пропускается, если оно образует цикл с предшествующими ребрами, среди ребер T_S его также быть не может. Таким образом, единственным возможным сочетанием знаков (при первом несовпадении !) может быть $\pm e_i$, т.е. ребро e_i присутствует в T_K и отсутствует в T_S . Пусть концами этого ребра являются вершины a и b . Поскольку T_S является деревом в нем должна существовать единственная цепь, соединяющая a и b . Среди звеньев этой цепи имеется, по крайней мере, одно ребро, не входящее в T_K (в противном случае в T_K есть цикл, что невозможно). Обозначим это ребро e_k . В последовательности упорядоченных ребер e_k может находиться только "справа" от e_i , а это значит, что длина e_i не больше, чем длина e_k . Перестроим T_S , удалив из него e_k и включив e_i . В силу порядка следования ребер эта перестройка не может удлинить остов, а более короткий чем T_S остов получить невозможно. Это значит, что e_i и e_k имеют одинаковую длину. В результате перестройки вместо T_S получен остов T'_S , также являющийся кратчай-

⁹Верхний знак соответствует вхождению (или невхождению) ребра e_i в T_K , а нижний – в T_S .

шим, в котором больше ребер, совпадающих с ребрами T_K . Повторяя описанную процедуру для T_K и T'_S , затем для T_K и T''_S и т.д., получаем последовательность кратчайших остовов, сходящуюся к T_K .

Формальное описание алгоритма. Введем обозначения; $\dots, n$ – метки вершины графа;

i – счетчик ребер графа;

k – счетчик ребер остова;

E – список ребер графа, упорядоченных по длине. Каждая строка списка состоит из пары $e_{i,1}$ и $e_{i,2}$, являющихся метками концов i -го ребра;

t – номер фрагмента остова;

S – список принадлежности вершин графа фрагментам остова. Значение s_j равно номеру фрагмента, которому на текущей итерации принадлежит вершина j . Первоначально все элементы списка равны нулю. По завершении работы алгоритма все элементы S равны единице. Для рассмотренного выше примера динамика изменения списка отражена в табл. 3.4.

Таблица 3.4

Итерация (ребро)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	Фрагменты остова
0	0	0	0	0	0	0	0	0	0	0	—
1	1	0	0	0	1	0	0	0	0	0	$T_1 := \{1; 5\}$
2	1	0	0	1	1	0	0	0	0	0	$T_1 := T_1 + \{4; 5\}$
3	1	2	2	1	1	0	0	0	0	0	$T_2 := \{2; 3\}$
5	1	2	2	1	1	0	0	0	3	3	$T_3 := \{9; 10\}$
6	1	2	2	1	1	0	0	1	3	3	$T_1 := T_1 + \{5; 8\}$
8	1	2	2	1	1	3	0	1	3	3	$T_3 := T_3 + \{6; 10\}$
10	1	2	2	1	1	3	2	1	1	1	$T_2 := T_2 + \{3; 7\}$
11	1	2	2	1	1	1	2	1	1	1	$T_1 := T_1 + T_3; T_3$
15	1	1	1	1	1	1	1	1	1	1	$T_1 := T_1 + T_2; T_2$

На итерациях 4, 7, 9, 12–14 состояние списка не меняется, так как соответствующие ребра не могут быть включены в осто́в.

С использованием введенных обозначений формальное описание алгоритма принимает следующий вид:

```

begin                                     { — К Р А С К А Л — }
   $i := 1$ ;  $t := 1$ ;  $k := 1$ ; { Формировать первый фрагмент остова }
   $s_{e_{1,1}} := t$ ;  $s_{e_{1,2}} := t$ ; { на основе кратчайшего ребра графа }
  output( $i$ ); { и напечатать номер ребра. }
  while ( $k < (n-1)$ ) & ( $i < m$ ) do
    {
       $i := i+1$ ; { Перейти к следующему ребру. }
      if ( $s_{e_{i,1}} = 0$ ) & ( $s_{e_{i,2}} = 0$ ) { Если ребро не имеет общих }
        { вершин ни с одним из фрагментов, }
        then {
           $k := k+1$ ; { включить ребро в остов, }
           $t := t+1$ ; { создав }
           $s_{e_{i,2}} := t$ ;  $s_{e_{i,1}} := t$ ; { новый фрагмент, }
          output( $i$ ); { и напечатать номер ребра. }
        }
      else
        if ( $s_{e_{i,1}} = 0$ ) & ( $s_{e_{i,2}} \neq 0$ ) { Если ребро имеет общую }
          { вершину с одним из фрагментов, то }
          then {
             $k := k+1$ ; { включить ребро в остов, }
             $s_{s_{i,1}} := s_{e_{i,2}}$ ; { присоединив его к фрагменту, }
            output( $i$ ); { и напечатать номер ребра. }
          }
        else
          if ( $s_{e_{i,1}} \neq 0$ ) & ( $s_{s_{i,2}} = 0$ ) { Если ребро имеет общую }
            { вершину с одним из фрагментов, то }
            then {
               $k := k+1$ ; { включить ребро в остов, }
               $s_{e_{i,2}} := s_{e_{i,1}}$ ; { присоединив его к фрагменту, }
              output( $i$ ); { и напечатать номер ребра. }
            }
          else
            if ( $s_{e_{i,1}} \neq 0$ ) & ( $s_{e_{i,2}} \neq 0$ ) & ( $s_{e_{i,1}} \neq s_{e_{i,2}}$ ) { Если концы ребра }
              { принадлежат разным фрагментам, }
              then {
                 $k := k+1$ ; { включить ребро в остов, }
                for  $j := 1$  to  $n$  do { объединив фрагменты }
                  {
                    if  $s_j = \max(s_{e_{i,1}}, s_{e_{i,2}})$  { под меньшим }
                      then  $s_j := \min(s_{e_{i,1}}, s_{e_{i,2}})$ ; { номером, }
                    output( $i$ ); { и напечатать номер ребра. }
                  }
                else { Концы ребра принадлежат одному фрагменту }
                  { Не включать ребро в остов. }
                }
              }
    }
  if  $k = (n-1)$ 
  then output('Получено кратчайшее остовное дерево.')
  else output('Получен кратчайший остовный лес.')
end.                                     { — К Р А С К А Л — }

```

Оценим трудоемкость приведенного варианта реализации алгоритма Краскала применительно к (m, n) -графу.

В худшем случае (самое длинное ребро принадлежит кратчайшему остову) внешний цикл перебора ребер графа должен повториться $m-1$ раз. Кроме того, в тех случаях, когда включение ребра приводит к объединению двух фрагментов будущего остова, должна быть выполнена процедура объединения множеств их вершин. Число повторений соответствующего цикла равно n . Учитывая все это, можно утверждать, что трудоемкость нашего алгоритма не хуже чем mn . Это весьма завышенная оценка, поскольку далеко не всегда ребро с максимальным весом входит в остов, и совсем не обязательно, что включение очередного ребра должно приводить к слиянию некоторой пары фрагментов.

В заключение отметим, что построение алгоритма предполагает упорядоченность списка ребер, а это, в свою очередь, требует еще порядка $m \log_2 m$ операций. При достаточно большом количестве ребер именно эта составляющая процесса получения кратчайшего остова оказывается основной.

3.4.2 Алгоритм Прима

Описание алгоритма. В отличие от алгоритма Краскала, алгоритм Прима обеспечивает построение остова графа $G(V, E)$, заданного матрицей весов $C=[c_{i,j}]$, путем последовательного наращивания единственного древовидного фрагмента $T(VT, ET)$, который вначале вообще не имеет ребер и состоит из одной произвольно выбранной вершины графа. На каждой итерации к фрагменту добавляется самое короткое ребро из числа ребер, у которых одна из концевых вершин v_i уже включена в строящийся остов, а другая v_j еще нет, т.е. $ET_k := ET_{k-1} + \{v_i, v_j\}$ и $VT_k := VT_{k-1} + v_j$, где $v_i \in VT_{k-1}$, $v_j \notin VT_{k-1}$, а $c_{i,j}$ минимально. Процесс заканчивается после присоединения к T $n-1$ ребер. Таким образом, при выполнении алгоритма формируется последовательность деревьев: $T_0 \rightarrow T_1 \rightarrow \dots \rightarrow T_k \rightarrow \dots \rightarrow T_{n-1}$, где $T_0 = \langle \{v\}, \emptyset \rangle$, а T_{n-1} - кратчайший остов. Если список ребер упорядочен по возрастанию

веса, то включению в остов на очередной итерации подлежит первое от начала списка ребро $\{v_i, v_j\}$, отвечающее указанному выше условию.

Построение остова для графа, изображенного на рис. 3.10, иллюстрирует табл. 3.5, аналогичная табл. 3.3.

Таблица 3.5

e	a	b	l_{ab}	T_i	VT_i	i
				$T_0 = \{\{v_1\}, \emptyset\}$	$\{v_1\}$	
1	v_1	v_5	2	$T_1 := T_0 + \{v_1, v_5\}$	$\{v_1, v_5\}$	1
2	v_4	v_5	3	$T_2 := T_1 + \{v_4, v_5\}$	$\{v_1, v_4, v_5\}$	2
3	v_2	v_3	5	$T_8 := T_7 + \{v_2, v_3\}$	$\{v_1, v_2, v_3, v_4, v_5, v_6, v_8, v_9, v_{10}\}$	8
4	v_1	v_4	7	—	—	
5	v_9	v_{10}	8	$T_6 := T_5 + \{v_9, v_{10}\}$	$\{v_1, v_4, v_5, v_6, v_8, v_9, v_{10}\}$	6
6	v_5	v_8	10	$T_3 := T_2 + \{v_5, v_8\}$	$\{v_1, v_4, v_5, v_8\}$	3
7	v_4	v_8	13	—	—	
8	v_6	v_{10}	14	$T_5 := T_4 + \{v_6, v_{10}\}$	$\{v_1, v_4, v_5, v_6, v_8, v_{10}\}$	5
9	v_6	v_9	14	—	—	
10	v_3	v_7	17	$T_9 := T_8 + \{v_3, v_7\}$	$\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$	9
11	v_5	v_6	20	$T_4 := T_3 + \{v_5, v_6\}$	$\{v_1, v_4, v_5, v_6, v_8\}$	4
12	v_2	v_7	22	—	—	
13	v_8	v_9	23	—	—	
14	v_4	v_9	23	—	—	
15	v_6	v_3	25	$T_7 := T_6 + \{v_3, v_6\};$	$\{v_1, v_3, v_4, v_5, v_6, v_8, v_{10}\}$	7
...	...	...	...	...	...	

Здесь содержимое столбца T_i показывает, как растет единственный древовидный фрагмент формируемого остова, столбец VT_i отражает рост множества вершин, принадлежащих этому фрагменту, а номер итерации, на которой ребро включается в остов, указан в столбце i .

Пусть начальный фрагмент состоит из вершины v_1 . Анализ первого ребра списка показывает, что оно отвечает условию включения в остов. Полученное дерево имеет две вершины v_1 и v_5 . Проверяя второе ребро, убеждаемся, что его также следует включить в остов. Новый фрагмент содержит

уже три вершины $VT_2 = \{v_1, v_5, v_4\}$. Третье ребро пока пропускаем, так как ни v_2 ни v_3 не принадлежат VT_2 . То же самое можно сказать и о пятом ребре. Оба конца четвертого ребра, наоборот, принадлежат VT_2 , значит его включение приведет к появлению цикла, что недопустимо. И только шестое ребро отвечает условию на включение. Таким образом после третьей итерации получаем дерево T_3 , состоящее из трех ребер, т.е. $ET_3 = \{\{v_1, v_5\}, \{v_4, v_5\}, \{v_5, v_8\}\}$. На четвертой итерации к дереву добавляется одиннадцатое ребро, на пятой – восьмое, и т.д. Построение остова завершается на девятой итерации присоединением десятого ребра. На рис. 3.13 представлены

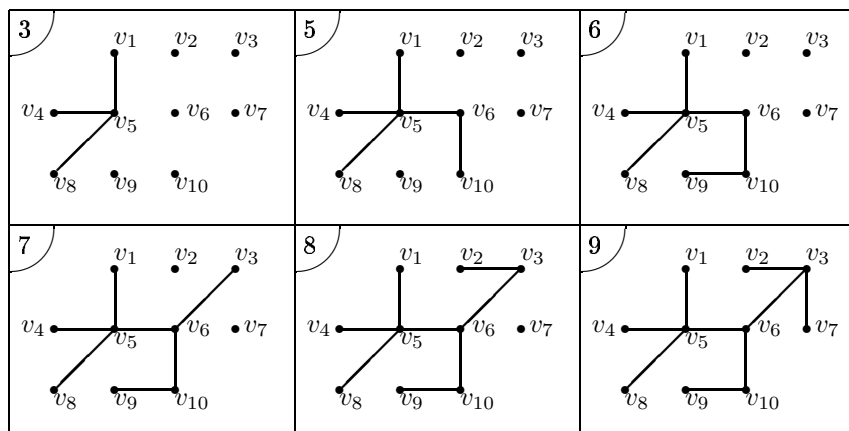


Рис. 3.13

отдельные этапы описанного процесса. Число в верхнем углу каждого прямоугольника соответствует итерации, на которой получен изображенный фрагмент.

Формализация алгоритма. Введем обозначения:

$1, 2, \dots, n$ – метки вершины графа;

i – счетчик ребер графа;

k – счетчик вершин графа и счетчик итераций;

E – список ребер графа, упорядоченных по длине. Каждая строка списка состоит из пары $e_{i,1}$ и $e_{i,2}$, являющихся метками концов i -го ребра;

S – список принадлежности вершин графа формируемому фрагменту остова. Если к текущей итерации вершина j уже включена в остов, то $s_j=1$, иначе $s_j=0$. Первоначально все элементы списка равны нулю, кроме элемента, соответствующего начальной вершине. По завершении работы алгоритма все элементы S равны единице.

Для рассмотренного выше примера динамика изменения списка отражена в табл. 3.6.

Таблица 3.6

Итерация (ребро)	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	Фрагменты остова
0	1	0	0	0	0	0	0	0	0	0	$T_0 = \langle \{v_1\}, \emptyset \rangle$
1	1	0	0	0	1	0	0	0	0	0	$T_1 := T_0 + \{1; 5\}$
2	1	0	0	1	1	0	0	0	0	0	$T_2 := T_1 + \{4; 5\}$
3	1	0	0	1	1	0	0	1	0	0	$T_3 := T_2 + \{5; 8\}$
4	1	0	0	1	1	1	0	1	0	0	$T_4 := T_3 + \{5; 6\}$
5	1	0	0	1	1	1	0	1	1	0	$T_5 := T_4 + \{6; 9\}$
6	1	0	0	1	1	1	0	1	1	1	$T_6 := T_5 + \{3; 6\}$
7	1	0	1	1	1	1	0	1	1	1	$T_7 := T_6 + \{3; 2\}$
8	1	1	1	1	1	1	0	1	1	1	$T_8 := T_7 + \{2; 3\}$
9	1	1	1	1	1	1	1	1	1	1	$T_9 := T_8 + \{3; 7\}$

С использованием введенных обозначений формальное описание алгоритма принимает следующий вид.

```

begin          { — П Р И М — }
  for  $k := 1$  to  $n$  do  $s_k := 0$ ;          { Инициализация списка  $S$  }
   $s_1 := 1$ ;          { Начать построение остова с вершины  $v_1$  }
  for  $k := 1$  to  $n-1$  do          { Цикл формирования остова }
    {
       $i := 1$ ;
      while  $s_{e_{i,1}} = s_{e_{i,2}}$  do  $i := i+1$ ; { Поиск включаемого ребра. }
       $s_{e_{i,1}} := 1$ ;  $s_{e_{i,2}} := 1$ ; { Ребро найдено, включить его в остов }
      output( $i$ );          { и напечатать номер. }
    }
end.          { — П Р И М — }

```

Оценивая, как и ранее, по максимуму трудоемкость этого варианта алгоритма Прима, убеждаемся, что ее величина пропорциональна n^2 . Таким образом, получаем несколько лучший показатель, чем для аналогичной реализации алгоритма

Краскала, особенно при большом числе ребер в графе. Однако, необходимость предварительной сортировки ребер может свести это преимущество практически к нулю.

Рассмотрим более эффективный вариант реализации алгоритма, основанный на технике пометок вершин, не требующий предварительной сортировки ребер и использующий матрицу весов C , в которой нулевые значения заменены на ∞ . Дадим эту технику на примере для графа, изображенного на рис. 3.10. Процесс получения остова отображен в табл. 3.7.

Таблица 3.7

Итера- ции	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	p
1		34	∞	7	2	38	∞	∞	∞	∞	v_1
2		30	∞	3		20	∞	10	∞	∞	v_5
3		30	∞			20	∞	10	23	∞	v_4
4		30	∞			20	∞		23	∞	v_8
5		30	25				∞		14	14	v_6
6		30	25				∞			8	v_9
7		30	25				35				v_{10}
8		5					17				v_3
9							17				v_2
<i>pred</i>		$\cancel{v_1}$ $\cancel{v_5}$ v_3	$\cancel{v_1}$ v_6	$\cancel{v_1}$ v_5	v_1	$\cancel{v_1}$ v_5	$\cancel{v_1}$ $\cancel{v_{10}}$ v_3	$\cancel{v_1}$ v_5	$\cancel{v_1}$ $\cancel{v_4}$ v_6	$\cancel{v_1}$ $\cancel{v_6}$ v_9	

Пусть T_0 состоит из единственной вершины v_1 . Присваиваем всем остальным вершинам временные пометки $l(v_i) = c_{1,i}$. Говорят, что вершины получили свои пометки из v_1 . Фиксируем это в последней строке таблицы – строке "вершин-предшественниц", обозначенной как *pred*. Затем отыскиваем минимальную временную пометку. В нашем примере это пометка вершины v_5 , равная 2. Далее считаем найденную пометку постоянной, а вершину v_5 постоянно помеченной и включаем в формируемый остов вместе с ребром $\{v_5, v_1\}$. Отметим, что вес включаемого ребра равен значению найденной пометки. На следующей итерации пересчитываем временные пометки по формуле $l(v_i) = \min[l(v_i), c(p, v_i)]$, где p – послед-

няя постоянно помеченная вершина (в данный момент $p=v_5$). При этом, если значение пометки $l(v_i)$ меняется (уменьшается), корректируем и значение в последней строке столбца, заменяя его текущим значением p . Поскольку изменяются $l(v_2)$, $l(v_4)$, $l(v_6)$ и $l(v_8)$, следует скорректировать содержимое 2, 4, 6 и 8 столбцов последней строки, заменив его на v_5 . Вновь отыскиваем минимальную временную пометку. Теперь это $l(v_4)=3$, которая становится постоянной. Соответственно последней постоянно помеченной вершиной p становится v_4 , а в остов включается ребро $\{v_4, v_5\}$.

На третьей итерации после пересчета временных пометок, причем изменяется только пометка $l(v_9)$, минимальной оказывается пометка вершины v_8 , и в остов включается ребро $\{v_8, v_5\}$. Именно v_5 , но не v_4 , как можно было бы предположить по аналогии с предыдущими итерациями! Вторая вершина пары берется из последней строки восьмого столбца.

На четвертой итерации (значения всех пометок сохраняются) минимальной оказывается пометка $l(v_8)=20$, на пятой (изменяются $l(v_3)$ и $l(v_9)$) минимальна $l(v_9)=14$ и т.д. Процесс завершается на девятой итерации после того, как все вершины получают постоянные пометки (см. табл 3.6).

В итоге получаем список ребер, образующих кратчайший остов: $\{v_2, v_3\}, \{v_3, v_6\}, \{v_4, v_5\}, \{v_5, v_1\}, \{v_6, v_5\}, \{v_7, v_3\}, \{v_8, v_5\}, \{v_9, v_6\}, \{v_{10}, v_9\}$. Первая вершина пары берется из заголовка столбца, а вторая из последней строки того же столбца. Результат полностью соответствует остову на рис. 3.13(9). Длина найденного остова равна сумме всех постоянных пометок.

Отметим, что строка $pred$ может быть сформирована и после того, как все вершины получают постоянные пометки. Достаточно в каждом столбце найти самую верхнюю строчку с минимальной пометкой. Значение $pred_i$ содержится в столбце p этой строки.

4 Пути и маршруты в графах

Существует большое разнообразие задач, связанных с путями и маршрутами в графе, начиная от стандартных задач на существование, пересчет и перечисление и кончая задачами поиска путей, отвечающих определенным требованиям. Такими требованиями могут быть максимальность или минимальность длины, пропускной способности или надежности пути, требования к множеству вершин, принадлежащих или не принадлежащих пути, и т.п. Кроме того, сами графы могут иметь определенные свойства, например, быть или не быть ориентированными, циклическими, взвешенными и т.д. Наконец, один и тот же граф может быть описан по-разному.

4.1 Существование путей

Эта задача эквивалентна задаче на достижимость в орграфе, решение которой рассмотрено в разд. 2. Действительно, матрица достижимости полностью определяет наличие или отсутствие пути для любой упорядоченной пары вершин. Поэтому ограничимся примером. Для графа, изображенного на рис. 4.1, матрица достижимости, полученная одним из ранее описанных способов, выглядит так:

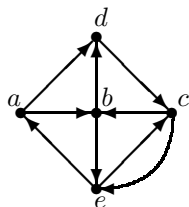


Рис. 4.1

$$\mathbf{R} = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{matrix}.$$

Простая проверка убеждает, в том, что для любой пары вершин в графе имеется хотя бы один путь. Именно это и следует из приведенной матрицы, поскольку все ее элементы равны 1.

4.2 Пересчет маршрутов и путей

Пусть $\Gamma(v_i)$ есть множество вершин, являющихся концами дуг, выходящих из v_i , а $\Gamma^{-1}(v_j)$ – множество вершин, являющихся началами дуг, входящих в v_j . Тогда их пересечение $\Gamma(v_i) \cap \Gamma^{-1}(v_j)$ представляет собой множество промежуточных вершин, принадлежащих цепям длиной в две дуги между v_i и v_j , а $|\Gamma(v_i) \cap \Gamma^{-1}(v_j)|$ – число таких цепей. Поскольку единичные элементы i -строки матрицы смежности соответствуют элементам множества $\Gamma(v_i)$, а единичные элементы j -столбца этой матрицы – элементам множества $\Gamma^{-1}(v_j)$, то $|\Gamma(v_i) \cap \Gamma^{-1}(v_j)| = \sum_{k=1}^n a_{ik} a_{kj}$. То же самое следует из анализа схем на рис. 4.2, где представлены все возможные комбинации

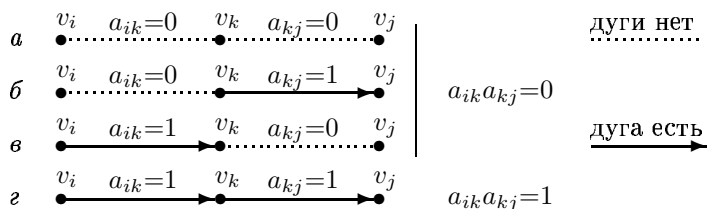


Рис 4.2

существования дуг, образующих цепь длины 2 из v_i в v_j с промежуточной вершиной v_k , а также значения соответствующих элементов a_{ik} и a_{kj} матрицы смежности. Если имеют место случаи *а*, *б* или *в*, цепь отсутствует и соответствующее слагаемое суммы $\sum_{k=1}^n a_{ik} a_{kj}$ равно нулю. В случае *г* цепь существует, а произведение $a_{ik} a_{kj} = 1$ и, следовательно, дает свой вклад в указанную сумму.

Таким образом, задача определения числа цепей (маршрутов) длины 2 между всеми парами вершин графа может быть решена простым возведением в квадрат его матрицы смежности. Аналогично, количество маршрутов длины 3 может быть получено из матрицы $\mathbf{A}^3$ и вообще количество маршрутов $a_{ij}^{(l)}$ длины l между вершинами v_i и v_j равно соответствующему элементу матрицы $\mathbf{A}^l$. Наконец, количество маршрутов дли-

ны не более p между всеми парами вершин определяется как сумма

$$\mathbf{A} + \mathbf{A}^2 + \dots + \mathbf{A}^p = \sum_{l=1}^p \mathbf{A}^l.$$

В качестве примера приведем решение задачи пересчета маршрутов графа, изображенного на рис. 4.1 при $l=1, 2, 3, 4$. Информацию о путях в одну дугу несет матрица смежности $\mathbf{A}$, остальные три матрицы получаем простым умножением:

$$\mathbf{A} = \begin{array}{c} \begin{array}{ccccc} & a & b & c & d & e \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \end{array}, \quad \mathbf{A}^2 = \begin{array}{c} \begin{array}{ccccc} & a & b & c & d & e \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 2 & 0 & 0 \\ 1 & 0 & 1 & 1 & \underline{1} \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 2 & \underline{0} & 1 & 1 \end{bmatrix} \end{array} \end{array},$$

$$\mathbf{A}^3 = \begin{array}{c} \begin{array}{ccccc} & a & b & c & d & e \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{bmatrix} 1 & 1 & 2 & 0 & 1 \\ 0 & 3 & 0 & 1 & 2 \\ 1 & 2 & 2 & 1 & \underline{1} \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & \underline{2} & 2 & 2 \end{bmatrix} \end{array} \end{array}, \quad \mathbf{A}^4 = \begin{array}{c} \begin{array}{ccccc} & a & b & c & d & e \\ \begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} & \begin{bmatrix} 1 & 3 & 1 & 2 & 3 \\ 2 & 0 & 3 & 3 & 3 \\ 1 & 3 & 2 & 3 & \underline{4} \\ 1 & 2 & 2 & 1 & 1 \\ 2 & 3 & \underline{4} & 1 & 2 \end{bmatrix} \end{array} \end{array}.$$

Непосредственной проверкой по изображению графа убеждаемся, что, например, между c и e есть

1 маршрут (путь) длины 2: $c \rightarrow b \rightarrow e$;

1 маршрут длины 3: $c \rightarrow e \rightarrow c \rightarrow e$;

4 маршрута (цепи) длины 4: $c \rightarrow b \rightarrow e \rightarrow c \rightarrow e$, $c \rightarrow b \rightarrow d \rightarrow c \rightarrow e$,
 $c \rightarrow e \rightarrow a \rightarrow b \rightarrow e$, $c \rightarrow e \rightarrow c \rightarrow b \rightarrow e$;

а между e и c есть

2 маршрута длины 3: $e \rightarrow c \rightarrow e \rightarrow c$, $e \rightarrow a \rightarrow d \rightarrow c$ (путь);

4 маршрута длины 4: $e \rightarrow c \rightarrow b \rightarrow e \rightarrow c$, $e \rightarrow a \rightarrow b \rightarrow d \rightarrow c$ (путь),
 $e \rightarrow a \rightarrow b \rightarrow e \rightarrow c$ (цепь) $e \rightarrow c \rightarrow b \rightarrow d \rightarrow c$ (цепь).

Это полностью соответствует значениям элементов $a_{ce}^{(l)}$ и $a_{ec}^{(l)}$ (выделены шрифтом и подчеркнуты) матриц $\mathbf{A}^2$, $\mathbf{A}^3$, $\mathbf{A}^4$. К сожалению, определяется число маршрутов, связывающих пару вершин, а не число путей, которые представляют наибольший интерес. Количество путей можно определить, если найти

все маршруты и отбросить те из них, где есть повторяющиеся фрагменты, т.е. необходимо решать задачу перечисления маршрутов. Прежде чем перейти к этой задаче, отметим два факта, касающихся задачи пересчета.

1. Отличные от нуля диагональные элементы матриц указывают на присутствие в графе соответствующего количества циклов длины l . Например, в матрице A^3 диагональный элемент $a_{bb}^{(3)}=3$. Это значит, что в графе есть три цикла длины 3, включающие вершину b . Непосредственно по рисунку находим: $b \rightarrow d \rightarrow c \rightarrow b$, $b \rightarrow e \rightarrow c \rightarrow b$ и $b \rightarrow e \rightarrow a \rightarrow b$.

2. Поскольку длина пути в графе не может превышать $n-1$, не имеет смысла возводить A в более высокую степень.

4.3 Перечисление маршрутов и путей

Матричный подход, использованный в предыдущих разделах, легко адаптируется к задаче о перечислении маршрутов.

Пусть $P^{(l)}$ – матрица всех маршрутов с l промежуточными вершинами (длины $l+1$), элементы которой определяются как $p_{ij}^{(l)} = \sum_{k=1}^q v_{k1} v_{k2} \dots v_{kl}$, где v_{k_l} это l -я по порядку промежуточная вершина маршрута, связывающего v_i и v_j , а q – количество таких маршрутов. Если маршрутов длины $l+1$ между v_i и v_j нет, то $p_{ij}^{(l)}=0$. Слагаемые $v_{k1} v_{k2} \dots v_{kl}$ представляют собой "произведения" вершин, лежащих на некотором маршруте между v_i и v_j . Их можно рассматривать как размещения без повторений или с повторениями из n элементов по l в зависимости от того, является маршрут путем или нет. Например, для графа на рис. 4.1 $p_{cd}^{(3)}=bea+eab+ecb$ и, значит, из v_i в v_j есть три маршрута: $c \rightarrow b \rightarrow e \rightarrow a \rightarrow d$, $c \rightarrow e \rightarrow a \rightarrow b \rightarrow d$ и $c \rightarrow e \rightarrow c \rightarrow b \rightarrow d$, два из которых являются путями, так как не содержат повторяющихся вершин. В наличии указанных маршрутов легко убедиться непосредственно по графу, а их количество равно значению элемента $a_{cd}^{(4)}$ в матрице A^4 . Кроме $P^{(l)}$, используем вспомогательную матрицу P' , в которой элемент p'_{ij} равен v_j , если в графе есть дуга $v_i v_j$, и 0, если

дуга отсутствует. Поскольку j -столбец $\mathbf{P}^{(l-1)}$ отражает все маршруты длины l , с началом в v_k ($k=\overline{1, n}$) и концом в v_j , а i -строка $\mathbf{P}'$ – концы всех дуг вида $v_i v_k$ ($k=\overline{1, n}$), то сумма $\sum_{k=1}^n p'_{ik} p_{kj}^{(l-1)}$ отражает все маршруты из v_i в v_j длины $l+1$. Следовательно, матрица маршрутов $\mathbf{P}^{(l)}$ может быть получена как $\mathbf{P}' \times \mathbf{P}^{(l-1)}$, при этом *смножители в произведениях не переупорядочиваются и не используется степенная запись повторяющихся сомножителей, чтобы не потерять информацию о порядке следования вершин*. Таким образом может быть сформирован ряд матриц $\mathbf{P}^{(1)}, \mathbf{P}^{(2)}, \mathbf{P}^{(3)}, \dots, \mathbf{P}^{(q-1)}$, в совокупности определяющих все маршруты длины $\leq q$ для любой пары вершин.

В качестве примера выполним расчеты для графа, представленного на рис. 4.1, используя соотношения:

$$\mathbf{P}^{(1)} = \mathbf{P}' \times \mathbf{P}^{(0)}; \quad \mathbf{P}^{(2)} = \mathbf{P}' \times \mathbf{P}^{(1)}; \quad \mathbf{P}^{(3)} = \mathbf{P}' \times \mathbf{P}^{(2)}.$$

За матрицу $\mathbf{P}^{(0)}$ (пути без промежуточных вершин), необходимую для получения $\mathbf{P}^{(1)}$, примем матрицу смежности.

$$\mathbf{P}' = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & b & 0 & d & 0 \\ 0 & 0 & 0 & d & e \\ 0 & b & 0 & 0 & e \\ 0 & 0 & c & 0 & 0 \\ a & 0 & c & 0 & 0 \end{bmatrix} \end{matrix}, \quad \mathbf{P}^{(0)} = \mathbf{A} = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{matrix},$$

$$\mathbf{P}^{(1)} = \mathbf{P}' \times \mathbf{P}^{(0)} = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} 0 & 0 & d & b & b \\ e & 0 & d+e & 0 & 0 \\ e & 0 & e & b & b \\ 0 & c & 0 & 0 & c \\ 0 & a+c & 0 & a & c \end{bmatrix} \end{matrix},$$

$$\mathbf{P}^{(2)} = \mathbf{P}' \times \mathbf{P}^{(1)} = \begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} be & dc & bd+be & 0 & dc \\ 0 & ea+dc+ec & 0 & ea & dc+ec \\ be & ea+ec & bd+be & ea & ec \\ ce & 0 & ce & cb & cb \\ ce & 0 & ad+ce & ab+cb & ab+cb \end{bmatrix} \end{matrix}.$$

Наконец для $\mathbf{P}^{(3)} = \mathbf{P}' \times \mathbf{P}^{(2)}$ матрица принимает вид

$$\begin{matrix} & \begin{matrix} a & b & c & d & e \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \end{matrix} & \begin{bmatrix} dce & bea+bdc+bec & dce & bea+dc b & dcb+bdc+bec \\ dce+ece & 0 & ead+dce+ece & eab+dc b+ec b & eab+dc b+ec b \\ ece & bea+bdc+bec & ead+ece & bea+eab+ec b & eab+ec b+ \\ & & & & +bdc+bec \\ cbe & cea+cec & cbd+cbe & cea & cec \\ abe+cbe & cea+adc+cec & abd+cbd+ \\ & & +abe+cbe & cea & adc+cec \end{bmatrix} \end{matrix}$$

Как видно из примера, уже при небольших значениях l матрицы могут быть весьма громоздкими. Однако, если цель поиска – пути, объем вычислений можно существенно уменьшить, отбрасывая произведения с повторяющимися вершинами. Очевидно также, что в этом случае $l \leq (n-2)$, так как путь в графе не может содержать более чем $n-2$ промежуточных вершин¹⁰. Кроме того, с увеличением длины количество путей в общем случае должно сокращаться.

4.4 Задачи о кратчайших путях

Большинство задач этого вида можно рассматривать как расширение, обобщение или модификацию следующей задачи.

В графе G с матрицей весов $\mathbf{C}=[c_{i,j}]$ требуется найти путь между заданной парой вершин s и t , у которого сумма весов составляющих его дуг (ребер) минимальна. Здесь

¹⁰В случае замкнутого пути (контура) $l=n-1$.

s и t являются соответственно начальной и конечной вершинами. Такое различие в обозначениях концевых вершин пути, необходимо для орграфов, так как в этом случае, в отличие от простых графов, кратчайшие (s, t) - и (t, s) -пути различны.

Понятно, что любой участок кратчайшего пути сам является кратчайшим путем между парой его концевых вершин. Значит, если найден кратчайший (s, t) -путь, промежуточные вершины которого образуют множество V' , то фактически решена задача отыскания кратчайших путей из s во все вершины V' . Поэтому представляется естественным следующее обобщение сформулированной выше задачи.

Найти кратчайшие пути между заданной начальной вершиной s и всеми остальными вершинами графа.

Наконец, в самой широкой постановке задача выглядит так.

Найти кратчайшие пути между всеми вершинами графа.

Нетрудно видеть, что определяющим в приведенных формулировках является количество пар концевых вершин, для которых следует найти кратчайший связующий путь: в первой задаче это одна пара, во второй $(n-1)$ пар, а в третьей — C_n^2 (для простого графа) или $2C_n^2$ (для ориентрованного).

Еще одна разновидность задачи возникает, когда для некоторой пары вершин кроме кратчайшего требуется найти еще $k-1$ путей, длины которых составляют неубывающую числовую последовательность $L_{Kp_1} \leq L_{Kp_2} \leq \dots \leq L_{Kp_k}$. Эта задача известна как *задача поиска k кратчайших путей между двумя заданными вершинами*.

В общем случае элементы матрицы C могут быть разных знаков. Наличие отрицательных значений среди элементов $c_{i,j}$ усложняет решение задачи поиска кратчайших путей. Задача становится еще более сложной, если в графе присутствуют циклы с отрицательным суммарным весом. Наоборот, если выполняется так называемое "правило треугольника": $c_{i,j} \leq c_{i,k} + c_{k,j}$, то задача становится существенно проще и фактически может рассматриваться как задача поиска кратчайшего пути в графе с ребрами единичной длины.

4.4.1 Графы с дугами единичной длины

Рассмотрим случай, когда веса всех дуг (ребер) графа одинаковы. Не нарушая общности, можно считать, что для любой дуги (ребра) $c_{i,j}=1$. Эффективное решение задачи получается с помощью простой процедуры, в результате которой каждая обработанная вершина получает пометку, равную расстоянию от s . Процесс начинается с того, что все $v \in \Gamma(s)$ получают пометку, равную единице. На второй итерации все $v \in \Gamma^2(s)$ помечаются двойкой, затем все $v \in \Gamma^3(s)$ помечаются тройкой, и вообще на i итерации для всех $v \in \Gamma^i(s)$ имеем $l(v)=i$. Процесс завершается, когда помеченной оказывается вершина t . Описанная процедура расстановки пометок называется *поиском в ширину*. Рассмотрим реализацию такого вида поиска на примере. Пусть требуется найти кратчайший путь между вершинами v_5 и v_7 в графе, представленном на рис. 4.3.

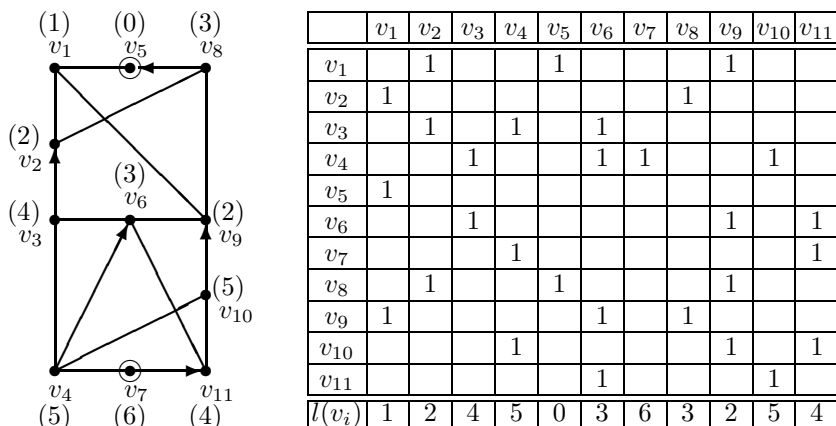


Рис. 4.3

Для расчетов используем матрицу смежности графа, дополнив ее строкой для пометок вершин. Сразу пометим вершину v_5 нулевым значением. На первой итерации, просматривая пятую строку матрицы, находим единицу в первом столбце. Следовательно, пометка первой вершины $l(v_1)=1$. На второй итерации просматриваем первую строку матрицы. Здесь

единицы содержатся во втором, пятом и девятом столбцах. Пятая вершина уже помечена, поэтому помечаем только v_2 и v_9 , записывая во вспомогательной строке двойки в соответствующих столбцах. На третьей итерации анализируем вторую и девятую строки матрицы. В результате пометку 3 получают v_8 и v_6 . Продолжая процесс, в итоге получаем совокупность пометок, как на рис. 4.3. Поскольку v_7 имеет пометку 6, длина искомого кратчайшего пути равна 6. Остается найти сам путь, т.е. множество составляющих его дуг (ребер), или что тоже самое, последовательность проходимых вершин.

Поиск начинаем с v_7 . Будем переходить от вершины к вершине, так чтобы пометка каждый раз уменьшалась на единицу, пока не достигнем вершины v_5 , причем обработку выполняем по столбцы матрицы. В седьмом столбце имеется только одна единица в четвертой строке. Значит предпоследней вершиной пути является v_4 с пометкой 5. В четвертом столбце единицы находятся в третьей, седьмой и десятой строках. Соответствующие вершины имеют пометки $l(v_3)=4$, $l(v_7)=6$, $l(v_{10})=5$. Из них подходящей является только v_3 . Это и есть третья от конца вершина пути. Содержимое третьего столбца указывает на вершины v_4 и v_6 . По значению пометки ($l(v_4)=5$, $l(v_6)=3$) подходит только v_6 . Действуя аналогичным образом, находим еще две промежуточные вершины пути v_9 и v_1 . Окончательно получаем, что кратчайший путь из v_5 в v_7 проходит через v_1, v_9, v_6, v_3, v_4 .

Алгоритм Ли. В качестве примера применения поиска в ширину рассмотрим задачу трассировки печатных плат, являющихся составной частью любой современной электронной аппаратуры. Печатная плата – это тонкая (1,5–2мм) жесткая пластина из электроизоляционного материала, на которой устанавливаются интегральные схемы, транзисторы, диоды, конденсаторы и т.п. Электрическое соединение всех этих элементов осуществляется с помощью проводников, представляющих собой тончайшие узкие металлические полоски, нанесенные по определенной технологии на поверхность платы. Среди прочих оптимизационных задач при проектировании печатных плат важное место занимает задача отыскания

варианта размещения проводников, отвечающего ряду требований (отсутствие пересечений, ограничение максимальной длины проводников, минимальность их суммарной длины и т.п.), известная как задача трассировки. Многие практические алгоритмы решения этой задачи в своей основе имеют классический *алгоритм Ли*, названный так по имени автора и позволяющий найти кратчайший путь между двумя ячейкам прямоугольной координатной сетки.

Пусть фрагмент некоторой печатной платы представлен сеткой, изображенной на рис. 4.4. Отдельные ячейки (на рисунке заштрихованы) заняты уже протрассированными проводниками и поэтому не могут использоваться. Необходимо

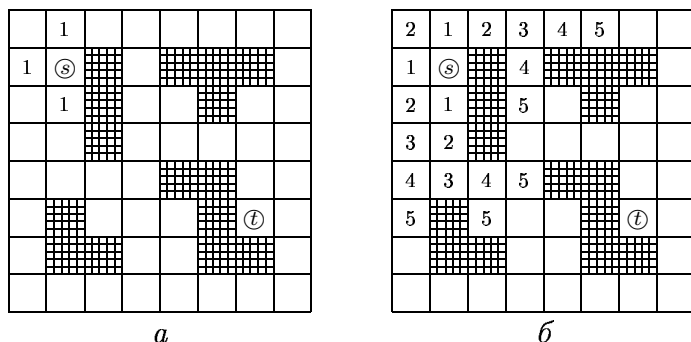


Рис. 4.4

соединить ячейки s и t кратчайшей ломаной линией, задающей трассу будущего проводника и состоящей из горизонтальных и вертикальных отрезков, проходящих по свободным ячейкам сетки. Вначале все свободные ячейки пусты. Процесс начинается с того, что в свободные ячейки, смежные с s , записываются единицы, как показано на рис. 4.4а. Далее в пустые ячейки, примыкающие к ячейкам с единицами, пишутся двойки. На третьей итерации оцифрованная область сетки обрамляется тройками, затем – четверками, и т.д. После пятой итерации получаем разметку как на рис. 4.4б. Смена разметки от итерации к итерации напоминает распространение волны, с той лишь разницей, что высота волнового фронта по мере уда-

ления от начальной точки s не убывает, а растет¹¹. Процесс завершается, когда фронт достигает ячейки t . Соответствующая разметка приведена на рис. 4.5а. Конфигурация

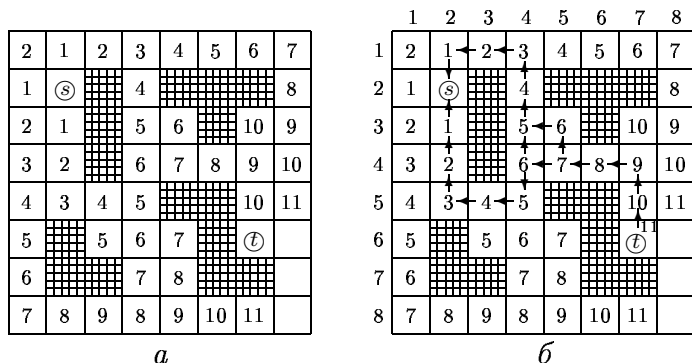


Рис. 4.5

самой трассы для проводника выявляется при движении по ячейкам сетки, если, начиная с ячейки t , переходить к соседней с меньшим значением пометки, пока не будет достигнута ячейка s . Эта операция отображена на рис. 4.5б, из которого получаем следующие варианты кратчайших путей из s в t :

$$\textcircled{s}-(1;2)-(1;3)-(1;4)-(2;4)-(3;4)-\frac{(3;5)}{(4;4)}-(4;5)-(4;6)-(4;7)-(5;7)-\textcircled{t};$$

$$\textcircled{s}-(3;2)-(4;2)-(5;2)-(5;3)-(5;4)-(4;4)-(4;5)-(4;6)-(4;7)-(5;7)-\textcircled{t},$$

где в круглых скобках указаны координаты ячеек в соответствии с нумерацией строк и столбцов сетки на рис. 4.5б. Практически вариант, использующий клетку (3;6), не целесообразен, так как содержит больше углов, и значит, менее надежен.

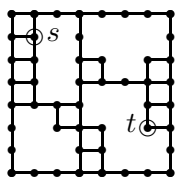


Рис. 4.6

В качестве дополнительной иллюстрации к рассмотренному примеру на рис. 4.6 изображен граф, соответствующий печатной плате: каждая свободная клетка представлена вершиной, а факт соседства клеток (по горизонтали и вертикали) отображен ребром.

¹¹Поэтому алгоритм называют также *волновым*.

Пусть Fr_1 – множество клеток (вершин графа), образующих фронт волны на текущей итерации; Fr_2 – фронт волны на следующей итерации. Тогда формализованное описание алгоритма имеет вид:

```

begin    { – В О Л Н О В О Й  А Л Г О Р И Т М – }
  for  $v \in V$  do  $l(v) := \infty$ ; {Пометить все вершины как свободные}
   $l(s) := 0$ ;    $Fr_1 := \text{adj } s$ ; {Сформировать начальный фронт и}
  for  $v \in Fr_1$  do  $l(v) := 1$ ; {пометить его вершины единицами.}
  while  $l(t) = \infty$  do {Пока фронт волны не достиг вершины  $t$ }
    {
       $Fr_2 := \emptyset$ ;
      for  $v \in Fr_1$  do    {для каждой вершины текущего фронта}
        {
          for  $w \in \text{adj } v$  do    {проверить все соседние вершины}
            {
              if  $l(w) = \infty$     {и если найдется свободная}
                then
                  {
                     $l(w) := l(v) + 1$ ;    {включить ее во вновь}
                     $Fr_2 := Fr_2 \cup \{w\}$ ; {формируемый фронт.}
                  }
                end if
              end for
            }
           $Fr_1 := Fr_2$ 
        }
      end for
    }
  end.    { – В О Л Н О В О Й  А Л Г О Р И Т М – }

```

4.4.2 Графы со взвешенными дугами (ребрами)

Кратчайшие пути и в этом случае можно находить с помощью волнового алгоритма. Пусть r – наибольший общий делитель весов ребер (дуг) графа G . Преобразуем граф, заменив каждое из ребер $\{v_i, v_j\}$ простой цепью, состоящей из $c_{i,j}/r$ ребер единичной длины. Ясно, что в полученном при этом графе G' полностью сохраняются метрические соотношения, характеризующие расстояния между вершинами графа G . Вместе с тем G' – это граф с ребрами (дугами) единичной длины, для которого задача решается по описанной выше схеме. Очевидный недостаток такого подхода – высокая трудоемкость из-за весьма большой размерности графа G' . Действительно число вершин (и ребер) в последнем больше, чем в G на величину, равную $\frac{1}{r} \sum_{\{v_i, v_j\} \in E} c_{i,j} - m$. Поэтому используют

более эффективные алгоритмы решения задачи. Два из них – алгоритм Дийкстры и алгоритм Флойда рассмотрены ниже.

Алгоритм Дийкстры. Позволяет найти кратчайший путь между некоторой вершиной s и остальными вершинами при условии, что в графе нет дуг (ребер) с отрицательным весом. Рассмотрим основную идею алгоритма на примере графа на рис. 4.7, с матрицей¹² C , взяв в качестве s вершину v_5 .

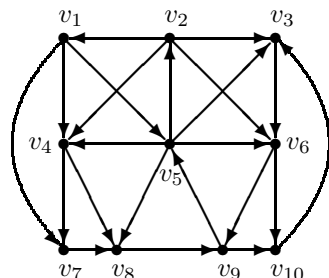
$$C = \begin{matrix} & \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} \end{matrix} \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \\ v_{10} \end{matrix} & \begin{bmatrix} & & & 3 & 19 & 11 & & & & \\ 10 & & 7 & 19 & & 22 & & & & \\ & & & & & 13 & & & & \\ & & & & & & 5 & 3 & & \\ & 5 & 14 & 20 & & 29 & & 23 & & \\ & & & & & & & & 2 & 8 \\ & & & & & & & 4 & & \\ & & & & & & & & 8 & \\ & & & & 7 & & & & & 6 \\ & & 9 & & & & & & & \end{bmatrix} \end{matrix}$$


Рис. 4.7

Выделим все вершины $v_i \in \Gamma(v_5)$ и припишем им числовые метки $l(v_i)$, равные весам дуг (v_5, v_i) , как показано на рис. 4.8 (веса проставлены около дуг, а метки – около вершин и заключены в круглые скобки).

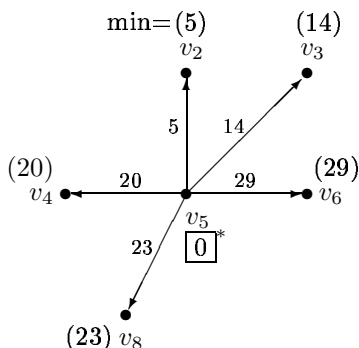


Рис. 4.8

Очевидно, что кратчайший путь до v_2 состоит из единственной дуги (v_5, v_2) , а его длина $l(v_2)=5$. Зафиксируем эту вершину и будем считать ее метку постоянной. Таким образом для v_2 задача решена. Относительно остальных четырех вершин можно лишь утверждать, что длины кратчайших путей до них не превосходят значений $l(v_i)$. Поэтому соответствующие метки пока считаем временными. В дальнейшем, чтобы различать постоянные и временные метки, постоянные будем заключать в рамку.

В дальнейшем, чтобы различать постоянные и временные метки, постоянные будем заключать в рамку.

¹²Пустые клетки означают, что соответствующих дуг в графе нет. Можно считать также, что дуги есть и каждая имеет вес ∞ .

Выделим все вершины $v_i \in \Gamma(v_2) = \{v_1, v_3, v_4, v_6\}$ и рассмотрим дуги (v_2, v_i) (изображены на рис. 4.9 тонкими линиями). Поскольку $l(v_2) + c(v_2, v_3) < l(v_3)$, "обходной" путь $v_5 \rightarrow v_2 \rightarrow v_3$

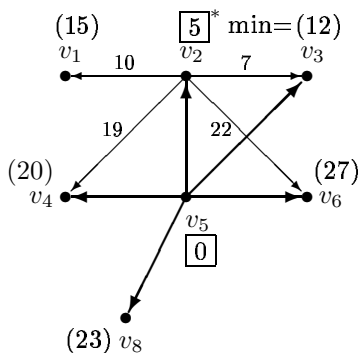


Рис. 4.9

короче, чем "прямой" $v_2 \rightarrow v_3$, поэтому за новое значение метки $l(v_3)$ примем сумму $l(v_2) + c(v_2, v_3) = 12$. Аналогичной проверкой устанавливаем, что прямой путь $v_5 \rightarrow v_4$ короче, чем обходной $v_5 \rightarrow v_2 \rightarrow v_4$, следовательно, значение метки $l(v_4)$ сохраняется прежним. Для вершины v_6 , как и для v_3 , метка корректируется и принимает значение 27. Кроме вершин v_3, v_4 и v_6 , достижимых из v_5 не-

посредственно, есть еще одна вершина v_1 , единственный путь к которой лежит через постоянно помеченную v_2 . Поэтому в качестве значения метки для нее естественно взять сумму

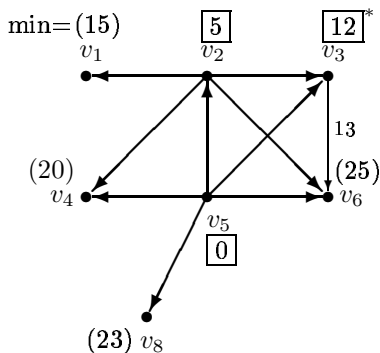


Рис. 4.10

$l(v_2) + c(v_2, v_1)$. Будем говорить, что теперь метки $l(v_1), l(v_3)$ и $l(v_6)$ получены из вершины v_2 . Из пяти временных меток минимальной является $l(v_3) = 12$, которая и определяет длину кратчайшего пути до v_3 . Так как метка получена из v_2 , путь имеет вид $v_5 \rightarrow v_2 \rightarrow v_3$. Таким образом, задача решена и для v_3 , которая становится постоянно помеченной. Этот факт отражен на рис. 4.10.

Кроме уже рассмотренных дуг, изображена и единственная дуга, выходящая из v_3 . Наличие этой дуги требует пересчета значения метки $l(v_6)$. Так как $l(v_3) + c(v_3, v_6) < 27$, за новое значение $l(v_6)$ примем сумму $l(v_3) + c(v_3, v_6) = 25$. Минималь-

ной среди временных меток является $l(v_1)=15$. Это значит, что выявлен кратчайший путь $v_5 \rightarrow v_2 \rightarrow v_1$, а метка вершины v_1 становится постоянной. Последующие действия выполняются аналогично. Для v_1 , как последней постоянно помеченной, находим множество вершин $\Gamma(v_1)=\{v_4, v_5, v_7\}$ и анализируем их метки, принимая во

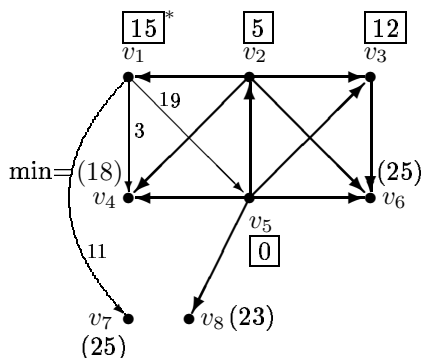


Рис. 4.11

внимание длины дуг, связывающих v_1 с этими вершинами. Так как $l(v_1)+c(v_1, v_4) < l(v_4)$, метка вершины v_4 корректируется, становясь равной 18. У вершины v_5 метка постоянная и поэтому не меняется. Наконец, v_7 , до этого не рассматривавшаяся, получает начальную метку, равную $l(v_1)+c(v_1, v_7)=25$. Граф, отражающий ситуацию, представлен на рис. 4.11. Здесь ми-

нимимальной временной меткой оказывается $d(v_4)=18$. Теперь вершина v_4 становится постоянно помеченной. Так как свою метку она получила из v_1 , кратчайший путь к ней состоит из кратчайшего пути до v_1 и дуги (v_1, v_4) и выглядит так $v_5 \rightarrow v_2 \rightarrow v_1 \rightarrow v_4$, а его длина равна 18.

Итерационный процесс продолжается до тех пор, пока конечная вершина искомого кратчайшего пути не станет постоянно помеченной. Если же требуется найти кратчайшие пути до всех вершин, итерации выполняются, пока все вершины не получат постоянные метки.

Обработку данных в соответствии с алгоритмом целесообразно оформлять в виде таблицы, строки которой соответствуют итерациям, а столбцы – вершинам графа и отражают процесс изменения их временных меток вплоть до момента, когда временная метка становится постоянной. Два дополнительных столбца таблицы нужны, чтобы зафиксировать вершину, которая становится постоянно помеченной на очередной итерации, и длину кратчайшего пути до этой вершины.

В табл. 4.2 содержатся результаты расчетов для разобранного выше примера. Покажем, как она заполняется.

Таблица 4.2

Итерации	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	p	$l(p)$
1	∞	∞	∞	∞	0	∞	∞	∞	∞	∞	v_5	0
2	∞	5	14	20		29	∞	23	∞	∞	v_2	5
3	15		12	20		27	∞	23	∞	∞	v_3	12
4	15			20		25	∞	23	∞	∞	v_1	15
5				18		25	25	23	∞	∞	v_4	18
6						25	23	21	∞	∞	v_8	21
7						25	23		29	∞	v_7	23
8						25			29	∞	v_6	25
9									27	33	v_9	27
10										33	v_{10}	33
<i>pred</i>	v_2	v_5	v_2	v_1	v_5	v_3	v_4	v_4	v_6	v_6		

На первой итерации полагаем временные метки для всех вершин, кроме начальной, равными ∞ , а для начальной – 0. Последней постоянно помеченной считаем вершину v_5 , а ее метку равной 0.

На второй итерации, используя матрицу C , находим все вершины множества $\Gamma(v_5)$ и пересчитываем их временные метки по формуле $l(v_i) = \min\{l(v_i), l(v_p) + c(v_p, v_i)\}$. Среди временных меток ищем минимальную. Это метка вершины v_2 , теперь ее считаем постоянно помеченной.

На третьей итерации находим вершины множества $\Gamma(v_2)$ и пересчитываем их временные метки. Среди всех временных вновь ищем минимальную метку. На этой итерации постоянно помеченной становится вершина v_3 , и т.д.

Процесс заканчивается на десятой итерации, когда постоянную метку получает v_{10} . Результаты – длины кратчайших путей – содержатся в двух последних столбцах таблицы.

Чтобы находить сами пути, сформируем еще одну строчку таблицы (*pred*), в которой для каждой вершины графа указана ее предшественница на кратчайшем пути. С этой целью в каждом столбце отыскиваем самую нижнюю строку с предпоследним значением метки. Например, в первом столб-

це это вторая строка, в восьмом – пятая. Искомая вершина-предшественница содержится в столбце p найденной строки. Для v_1 это вершина v_2 , а для v_8 – вершина v_4 . Теперь для определения самого пути достаточно записать цепочку вершин-предшественниц, начиная с конечной вершины. Так, для v_{10} имеем: $pred(v_{10})=v_6$, $pred(v_6)=v_3$, $pred(v_3)=v_2$ и $pred(v_2)=v_5$. Это значит, что кратчайший путь между v_5 и v_{10} имеет вид $v_5 \rightarrow v_2 \rightarrow v_3 \rightarrow v_6 \rightarrow v_{10}$.

Формализованная запись алгоритма Дийкстры, в которой p – последняя постоянно помеченная вершина;
 l – массив числовых меток вершин (длин путей);
 $pred$ – массив предшествующих вершин¹³;
 $post$ – массив признака постоянной метки вершины;
 v^* – вершина с минимальной временной меткой;
 может быть представлена в следующем виде:

```

begin      { – А Л Г О Р И Т М   Д И Й К С Т Р Ы – }
   $p := s$       {  $s$  – последняя постоянно помеченная вершина }
  for  $v \in V$  do      { Инициализировать : }
    {
       $l(v) := \infty$       { массив числовых меток вершин, }
       $pred(v) := v$       { массив предшествующих вершин, }
       $post(v) := 0$       { массив признака постоянной метки. }
    }
   $d(p) := 0$ 
   $post(p) := 1$ 
  while  $p \neq t$  do    { Пока  $t$  не станет постоянно помеченной, }
    {
      for  $v \in \Gamma(p)$  do { для всех вершин множества  $\Gamma(p)$ , }
        {
          if  $l(p) + c_{pv} < l(v)$  { метки которых  $> l(p) + c_{pv}$ , }
          then { выполнить коррекцию }
            {
               $l(v) := l(p) + c_{pv}$  { самой метки }
               $pred(v) := p$  { и предшествующей вершины }
            }
           $Найми\ v^*$  { Найти минимальную временную метку }
           $p := v^*$  { и далее считать соответствующую }
           $post(p) := 1$  { вершину постоянно помеченной }
        }
    }
end.      { – А Л Г О Р И Т М   Д И Й К С Т Р Ы – }

```

¹³Формируется параллельно с массивом d . Значение $pred(v_i)$ корректируется при каждом изменении $l(v_i)$, принимая значение, равное p .

Алгоритм Флойда. Обеспечивает решение задачи отыскания кратчайших путей для всех $n(n-1)$ упорядоченных пар вершин оргграфа и всех $n(n-1)/2$ неупорядоченных пар вершин простого графа даже при наличии дуг (ребер) с отрицательным весом. Единственным условием является отсутствие в графе контуров (циклов) с суммарным отрицательным весом. Более того, алгоритм позволяет выявить такие контуры.

Пронумеруем вершины графа числами $1, 2, \dots, n$ и обозначим через $d_{ij}^{(m)}$ длину кратчайшего пути из вершины i в вершину j , который может содержать в качестве промежуточных только первые m вершин. Если между i и j нет пути такого типа, полагаем $d_{ij}^{(m)} = \infty$. Естественно также считать, что при отсутствии в графе контуров отрицательного веса все $d_{ii}^{(m)} = 0$ для любого m . Теперь элемент матрицы $C = [c_{ij}]$ можно рассматривать как длину кратчайшего пути из i в j без промежуточных вершин, и обозначать $d_{ij}^{(0)}$, а длину кратчайшего пути между i и j (без каких-либо ограничений на промежуточные вершины) обозначать как $d_{ij}^{(n)}$.

Пусть $D^{(m)}$ квадратная матрица $[d_{ij}^{(m)}]$ порядка n . При $m=0$ она совпадает с матрицей C и записывается как $D^{(0)}$. Начиная с $D^{(0)}$, с помощью алгоритма Флойда формируется ряд матриц $D^{(1)}, D^{(2)}, \dots, D^{(n)}$, последняя из которых является искомой матрицей длин кратчайших путей между всеми парами вершин. Алгоритм Флойда, как и алгоритм Уоршолла, "работает на месте", т.е. не требуется хранить все матрицы $D^{(m)}$. Каждая последующая матрица в этом ряду получается из предыдущей с помощью простого соотношения:

$$d_{ij}^{(m)} = \min\{d_{ij}^{(m-1)}, d_{im}^{(m-1)} + d_{mj}^{(m-1)}\},$$

в основе которого лежит следующее, практически очевидное положение. Длина кратчайшего пути из вершины i в вершину j , использующего в качестве промежуточных только первые m вершин, не может быть больше длины кратчайшего пути из i в j , использующего в качестве промежуточных только первые $(m-1)$ вершин, и не более суммы длин двух кратчай-

ших путей из i в m и из m в j , в которых промежуточными могут быть лишь первые $(m-1)$ вершин. Расчетное соотношение существенно упрощается, когда m является одной из конечных вершин (начальной или конечной) рассматриваемого пути. Тогда имеем:

$$d_{im}^{(m)} = d_{im}^{(m-1)}, \quad d_{mj}^{(m)} = d_{mj}^{(m-1)}.$$

Это означает, что m -я строка и m -й столбец не изменяются при переходе от матрицы $\mathbf{D}^{(m-1)}$ к матрице $\mathbf{D}^{(m)}$ и поэтому могут не пересчитываться. Если заранее известно, что в графе нет контуров (циклов) отрицательного веса, то и диагональные элементы матриц также можно не пересчитывать. Отмеченные факты, позволяют утверждать, что трудоемкость вычисления любой из матриц $\mathbf{D}^{(m)}$ пропорциональна $n^2 - 3n + 2$. Учитывая, что для получения результата следует выполнить n итераций, в целом трудоемкость алгоритма можно оценить как $O(n^3)$.

В качестве примера найдем матрицу длин кратчайших путей между всеми парами вершин для графа на рис. 4.12, имеющего матрицу весов $\mathbf{C}$. Принимая $\mathbf{D}^{(0)} = \mathbf{C}$, с помощью алгоритма Флойда получаем следующий ряд матриц:

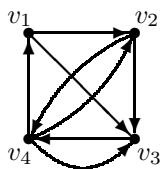


Рис. 4.12

$$\mathbf{D}^{(0)} = \mathbf{C} = \begin{bmatrix} 0 & 5 & 10 & \infty \\ \infty & 0 & 3 & 13 \\ \infty & \infty & 0 & 8 \\ 7 & 15 & 20 & 0 \end{bmatrix}, \quad \mathbf{D}^{(1)} = \begin{bmatrix} 0 & 5 & \underline{10} & \underline{\infty} \\ \infty & 0 & 3 & 13 \\ \infty & \infty & 0 & 8 \\ 7 & 12 & \underline{17} & 0 \end{bmatrix},$$

$$\mathbf{D}^{(2)} = \begin{bmatrix} 0 & 5 & 8 & \underline{18} \\ \infty & 0 & 3 & \underline{13} \\ \infty & \infty & 0 & 8 \\ 7 & 12 & 15 & 0 \end{bmatrix}, \quad \mathbf{D}^{(3)} = \begin{bmatrix} 0 & \underline{5} & 8 & 16 \\ \underline{\infty} & 0 & \underline{3} & 11 \\ \underline{\infty} & \underline{\infty} & 0 & 8 \\ 7 & 12 & 15 & 0 \end{bmatrix}, \quad \mathbf{D}^{(4)} = \begin{bmatrix} 0 & 5 & 8 & 16 \\ 18 & 0 & 3 & 11 \\ 15 & 20 & 0 & 8 \\ 7 & 12 & 15 & 0 \end{bmatrix}.$$

В каждой матрице выделены m -й столбец и m -я строка, не подлежащие пересчету на соответствующей итерации. Поскольку веса всех дуг не отрицательны, диагональные эле-

менты матриц постоянны и равны нулю. Кроме того, следует иметь в виду, что

если $d_{im}^{(m-1)} = \infty$, то $d_{ij}^{(m)} = d_{ij}^{(m-1)}$ для всех j и

если $d_{mj}^{(m-1)} = \infty$, то $d_{ij}^{(m)} = d_{ij}^{(m-1)}$ для всех i .

Иными словами, если i элемент выделенного столбца равен ∞ , то все элементы i строки на текущей итерации не изменяются. Аналогичное утверждение справедливо и для выделенной строки. Учет этих соотношений позволяет сократить объем вычислений на начальных итерациях, особенно для разреженных графов. Проиллюстрируем сказанное на примере. Второй и третий элементы первого столбца, а также четвертый элемент первой строки матрицы $\mathbf{D}^{(0)}$ равны ∞ . Поэтому, кроме первого столбца и первой строки, при переходе к $\mathbf{D}^{(1)}$ сохраняют свое значение элементы второй и третьей строк, а также элементы четвертого столбца. Пересчету подлежат лишь $d_{4,2}^{(0)}=15$ и $d_{4,3}^{(0)}=20$ ¹⁴. Чтобы получить, например, $d_{4,2}^{(1)}$, достаточно сложить $d_{1,2}^{(0)}=5$ и $d_{1,4}^{(0)}=7$, которые находятся в первой строке и первом столбце на позициях, соответствующих пересчитываемому элементу, сравнить найденную сумму с $d_{4,2}^{(0)}$ и выбрать наименьшее из двух чисел. Получаем $d_{4,2}^{(1)}=12$. Точно так же решается задача и для $d_{4,3}^{(1)}=17$. Нетрудно убедиться, что на второй итерации следует вычислять только $d_{1,3}^{(2)}$, $d_{1,4}^{(2)}$ и $d_{4,3}^{(2)}$, а все остальные элементы матрица $\mathbf{D}^{(2)}$ "наследует" от $\mathbf{D}^{(1)}$. Действительно, вторая строка и второй столбец не изменяются потому, что $m=2$, а первый столбец и третья строка – в силу приведенных выше соотношений. Все три названных элемента получают новые значения, равные соответственно 8, 18 и 15. На третьей итерации считаются $d_{1,4}^{(3)}$ и $d_{2,4}^{(3)}$, а на четвертой – $d_{1,2}^{(4)}$, $d_{1,3}^{(4)}$, $d_{2,3}^{(4)}$, $d_{2,1}^{(4)}$, $d_{3,1}^{(4)}$, $d_{3,2}^{(4)}$, причем из этих шести три первые сохраняют прежние значения.

Если граф имеет дуги с отрицательным весом, целесообразно наряду с остальными пересчитывать и диагональные

¹⁴Элементы, подлежащие пересчету, в матрицах подчеркнуты.

элементы матриц $\mathbf{D}^{(m)}$. Появление на некоторой итерации отрицательных значений на главной диагонали матрицы показывает, что в графе есть контур (цикл) с отрицательным весом. В этом случае дальнейшие вычисления теряют смысл, так как в подобных ситуациях алгоритм "не работает".

Решая задачу, наряду с длиной каждого кратчайшего пути, необходимо определить и последовательность входящих в него вершин. С этой целью одновременно с матрицей $\mathbf{D}^{(m)}$ формируется матрица $\mathbf{P}^{(m)}$, которую условно можно назвать "матрицей вершин-предшественниц". Элемент $p_{i,j}^{(m)}$ этой матрицы указывает предпоследнюю вершину кратчайшего (i, j) -пути, который может содержать в качестве промежуточных только первые m вершин графа. Вначале для всех i и j полагаем $p_{i,j}^{(0)} = i$, если $c_{ij} \neq \infty$, и $p_{i,j}^{(0)} = 0$ – в противном случае. Далее на каждой итерации матрица $\mathbf{P}^{(m)}$ пересчитывается одновременно с матрицей $\mathbf{D}^{(m)}$ с помощью соотношения:

$$p_{i,j}^{(m)} = \begin{cases} p_{m,j}^{(m-1)}, & \text{если } d_{i,j}^{(m-1)} > d_{i,m}^{(m-1)} + d_{m,j}^{(m-1)}, \\ p_{i,j}^{(m-1)}, & \text{в противном случае.} \end{cases}$$

Таким образом, элемент матрицы $\mathbf{P}$ корректируется при каждом изменении соответствующего элемента матрицы $\mathbf{D}$. Для нашего примера имеем следующий ряд матриц:

$$\begin{array}{c} \mathbf{P}^{(0)} \qquad \mathbf{P}^{(1)} \qquad \mathbf{P}^{(2)} \qquad \mathbf{P}^{(3)} \qquad \mathbf{P}^{(4)} \\ \left[\begin{array}{cccc} 0 & 1 & 1 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 3 & 3 \\ 4 & \underline{4} & \underline{4} & 0 \end{array} \right], \left[\begin{array}{cccc} 0 & 1 & \underline{1} & \underline{0} \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 \\ 4 & 1 & \underline{1} & 0 \end{array} \right], \left[\begin{array}{cccc} 0 & 1 & 2 & \underline{2} \\ 0 & 0 & 2 & \underline{2} \\ 0 & 0 & 0 & 3 \\ 4 & 1 & 2 & 0 \end{array} \right], \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ \underline{0} & 0 & 2 & 3 \\ \underline{0} & \underline{0} & 0 & 3 \\ 4 & 1 & 2 & 0 \end{array} \right], \left[\begin{array}{cccc} 0 & 1 & 2 & 3 \\ 4 & 0 & 2 & 3 \\ 4 & 1 & 0 & 3 \\ 4 & 1 & 2 & 0 \end{array} \right]. \end{array}$$

Подчеркнуты элементы, которые на очередной итерации получают новые значения. Например, как было показано ранее, на первой итерации в матрице $\mathbf{D}$ изменились элементы $d_{4,2}$ и $d_{4,3}$. Поэтому в соответствии с приведенным выше соотношением элементы $p_{4,2}$ и $p_{4,3}$ матрицы $\mathbf{P}$ получают новые значения, равные $p_{1,2}$ и $p_{1,3}$ соответственно. И вообще, на m -й итерации изменяющийся элемент матрицы получает зна-

чение, равное элементу, находящемуся в том же столбце на m -й позиции. Последняя из полученных матриц $P^{(4)}$ позволяет сформировать кратчайший путь для любой пары вершин. Пусть, например, это 1 и 4, причем 1 – начальная, а 4 – конечная вершины. Элемент $p_{1,4}^{(4)}=3$, т.е. предпоследней вершиной пути является вершина 3. В свою очередь, ей предшествует вершина 2, так как $p_{1,3}^{(4)}=2$. Наконец, $p_{1,2}^{(4)}=1$. Следовательно, (1,2,3,4) – кратчайший (1;4)-путь. Как следует из матрицы $D^{(4)}$, длина этого пути равна $d_{1,4}^{(4)}=16$. Пусть теперь 4 – начальная вершина пути, а 1 – конечная. Так как $p_{4,1}^{(4)}=4$, т.е. предпоследняя вершина совпадает с начальной, кратчайший (4;1)-путь состоит из дуги, связывающей 4 и 1.

С использованием введенных ранее обозначений формализованное описание алгоритма Флойда можно представить так:

```

begin    { – А Л Г О Р И Т М   Ф Л О Й Д А – }
  for i := 1 to n do                                { Формирование }
    {
      for j := 1 to n do                            { начальных значений }
        {
          {  $d_{i,j} := c_{i,j}$ ; }                      { элементов }
          if  $c_{ij} = \infty$  then  $p_{ij} = 0$  else  $p_{ij} = i$ ; { матриц }
        }
         $d_{ii} := 0$ ; { D и P }
      }
    for m := 1 to n do { Задание числа промежуточных вершин }
      {
        for i := 1 to n do { Перебор строк матрицы D }
          {
            if  $(i \neq m) \& (d_{i,m} \neq \infty)$ 
            then
              for j := 1 to n do { Перебор столбцов матрицы D }
                {
                  if  $(j \neq m) \& (d_{m,j} \neq \infty)$ 
                  then
                    if  $d_{ij} > d_{im} + d_{mj}$  { Если путь через }
                    then { вершину m короче, корректировать }
                      {
                         $d_{ij} := d_{im} + d_{mj}$ ; { длину пути и }
                         $p_{ij} := p_{mj}$ ; { предпоследнюю вершину. }
                      }
                  }
                }
              }
            for i := 1 to n do
              {
                if  $d_{ii} < 0$  then Прекратить процесс;
                { В графе обнаружен цикл отрицательного веса. }
              }
          }
        }
      }
    end.    { – А Л Г О Р И Т М   Ф Л О Й Д А – }

```

4.4.3 Ациклические орграфы

Кратчайший путь. Задача отыскания кратчайшего пути в ациклическом орграфе вполне может быть решена с помощью алгоритма Дийкстры. Однако существует более экономичный способ, использующий специфику таких графов.

Пусть в ациклическом орграфе с матрицей весов $[c_{i,j}]$ вершины пронумерованы так, что для любой дуги $(v_i v_j)$ выполняется отношение $i < j$, т.е. дуга всегда направлена от вершины с меньшим индексом (номером) к вершине с большим индексом (номером). Подобная нумерация всегда может быть выполнена на основе порядковой функции графа. Не нарушая общности, будем считать, что имеется только одна вершина s с нулевой полустепенью захода и одна вершина t с нулевой полустепенью исхода, называемые соответственно истоком и стоком. Требуется найти кратчайший путь из s в t . Как и в случае алгоритма Дийкстры фактически отыскиваются кратчайшие пути до всех вершин графа.

Длина кратчайшего пути $l(v_j)$ до вершины v_j легко находится, если известны длины кратчайших путей до всех вершин $v_i \in \Gamma^-(v_j)$, т.е. вершин, являющихся началами дуг, входящих в вершину v_j . Очевидно, что

$$l(v_j) = \min_{v_i \in \Gamma^-(v_j)} [l(v_i) + c_{i,j}], \quad (*)$$

причем $|\Gamma^-(v_j)| < j$, так как при используемой системе нумерации вершин число входящих дуг для любой вершины всегда меньше, чем ее индекс (номер).

Полагая $l(s) = l(v_1) = 0$, на первой итерации получаем, что $l(v_2) = l(v_1) + c_{1,2} = c_{1,2}$, так как вторая вершина может иметь лишь одну входящую дугу. На второй итерации получаем $l(v_3) = \min_{v_i \in \Gamma^-(v_3)} [l(v_i) + c_{i,3}]$ (не более двух входящих дуг). Затем $l(v_4) = \min_{v_i \in \Gamma^-(v_4)} [l(v_i) + c_{i,4}]$ (не более трех входящих дуг) и т.д., пока не будет получена длина кратчайшего пути для s .

Рассмотрим пример. Пусть требуется найти длину кратчайшего пути между вершинами v_1 (исток) и v_{13} (сток) в

ациклическом графе, изображенном на рис. 4.13. Нетрудно убедиться, что нумерация вершин графа является "правильной" в указанном выше смысле.

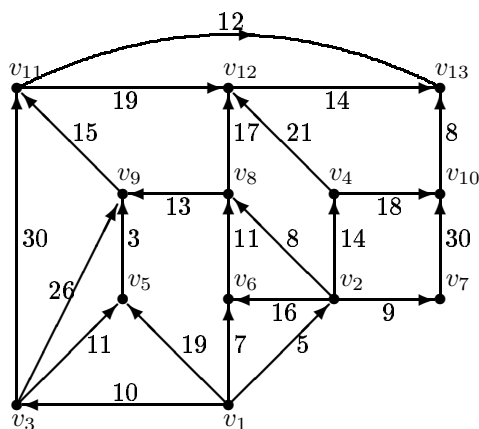


Рис. 4.13

Процесс вычислений можно выполнить непосредственно по рисунку, приняв $l(v_1)=0$ и далее переходя от вершины к вершине. При этом совсем не обязательно соблюдать очередность обработки вершин строго в соответствии с их нумерацией. Для данного графа вполне допустим, например, и такой порядок $v_1, v_3, v_5, v_2, v_7, v_4, v_{10}, v_6, v_8, v_9, v_{11}, v_{12}, v_{13}$. Важно только, чтобы к моменту обработки очередной вершины все ее "предшественницы" были обработаны. Тогда в любом случае результат (в смысле значений пометок вершин) будет таким, как показано на рис. 4.14, причем пометка v_{13} равна 44.

Для получения самой трассы кратчайшего пути следует выполнить процедуру "обратного хода". Сперва отыскивается предшественница для v_{13} , лежащая на искомом кратчайшем пути. Это одна из трех вершин v_{10}, v_{11}, v_{12} , а именно та, для которой выполняется соотношение

$$l(v_{13}) - l(v_i) = c_{i,13}, \text{ где } v_i \in \{v_{10}, v_{11}, v_{12}\}.$$

Такой вершиной является v_{12} . Для нее также ищется вершина-предшественница, лежащая на кратчайшем пути. Из трех возможных (v_4, v_8, v_{11}) таковой оказывается v_8 . В свою очередь,

этой вершине предшествует v_2 , а той – v_1 . Записывая найденную последовательность в обратном порядке, получаем собственно сам кратчайший путь $v_1-v_2-v_8-v_{12}-v_{13}$. На рис. 4.14 входящие в него дуги изображены толстыми линиями.

Процесс обратного хода можно упростить, если учесть, что вершина v_{13} получила свою пометку из v_{12} , а та, в свою очередь, из v_8 , и т.д. Поэтому на этапе прямого хода целесообразно наряду с присвоением пометки очередной вершине дополнительно отмечать и ту вершину, из которой эта пометка получена. Тогда на этапе обратного хода остается только пройти от стока к истоку, двигаясь лишь по этим дополнительно отмеченным вершинам.

Задачу можно решать и не прибегая к изображению графа. Техника получения пометок вершин непосредственно по матрице весов показана на рис. 4.14. Как и ранее, вершины должны быть пронумерованы так, чтобы для любой дуги $(v_i v_j)$ выполнялось отношение $i < j$. Отметим, что матрица в этом случае оказывается верхней треугольной. Незаполненные позиции означают, что соответствующих дуг в графе нет, или дуги есть, но их вес равен ∞ . Результаты вычислений целесообразно размещать на главной диагонали, т.е. считать, что $c_{ii}=l(v_i)$. На рис. 4.14 эти элементы выделены шрифтом и заключены в рамку. Слева приведена запись выражения (*) для соответствующей вершины. Например, в девятом столбце матрицы есть три "не пустых" значения $c_{3,9}=26$, $c_{5,9}=3$ и $c_{8,9}=13$. Это значит, что существуют три дуги, входящие в v_9 , начальные вершины которых имеют пометки $l(v_3)=10$, $l(v_5)=19$ и $l(v_8)=13$ ¹⁵. В этом случае выражение (*) выглядит так $\min[26+10; 3+19; 13+13]=21$, т.е. пометка v_9 равна 21. Очевидно, что вершиной-предшественницей является v_5 , поэтому дополнительно помечаем (звездочкой) элемент $c_{5,9}$. Подобным образом обрабатываются все столбцы матрицы, начиная со второго. На этапе обратного хода порядок просмотра столбцов меняется на обратный. Отыскивается помеченный звездочкой элемент последнего столбца. Строка, в которой он

¹⁵ См. соответствующие диагональные элементы $c_{3,3}$, $c_{5,5}$ и $c_{8,8}$.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
v_1	0	5*	10*		19*	7*							
v_2		5		14*		16	9*	8*					
v_3			10		11				26		30		
v_4	$\min[14+5]=$	19								18*		21	
v_5	$\min[19; 11+10]=$	19							3*				
v_6		$\min[7; 16+5]=$	7				11						
v_7			$\min[9+5]=$	14					30				
v_8			$\min[8+5; 11+7]=$	13	13							17*	
v_9		$\min[26+10; 3+19; 13+13]=$	22							15*			
v_{10}			$\min[18+19; 30+14]=$	37									8
v_{11}				$\min[30+10; 15+22]=$	37	19	12						
v_{12}				$\min[21+19; 17+13; 19+37]=$	30	14*							
v_{13}					$\min[8+37; 12+37; 14+30]=$	44							

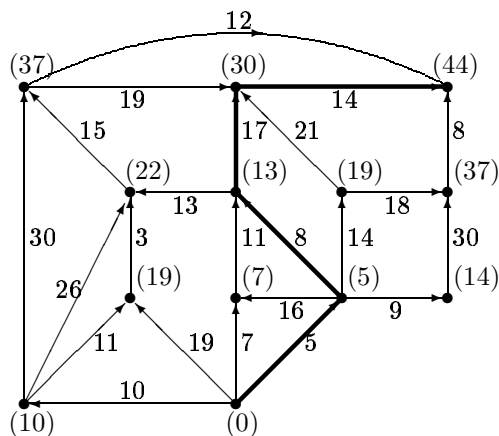


Рис. 4.14

находится, определяет предпоследнюю вершину кратчайшего пути. В столбце, соответствующем этой вершине, вновь ищется помеченный элемент – получаем третью (от конца) вершину кратчайшего пути и т.д., пока помеченным не окажется элемент в первой строке матрицы.

В формализованной записи (с использованием ранее введенных обозначений) рассмотренный алгоритм выглядит так:

```

begin      { — Алгоритм поиска кратчайшего — }
            { — пути в ациклическом орграфе — }
 $l(v_1) := 0;$ 
for  $j := 2$  to  $n$  do
    {
        Найти  $v_i^* \in \Gamma^-(v_j)$ , для которой  $l(v_i^*) + c_{i,j}$  минимально;
         $l(v_j) := l(v_i^*) + c_{i,j};$ 
         $pred(v_j) := v_i^*;$ 
    }
output  $\{l(v_n)\};$ 
 $v := v_n;$ 
while  $v \neq v_1$  do
    {
         $v \rightarrow$  стек;
         $v := pred(v);$ 
    }
output  $\{v_1, \text{содержимое стека}\};$ 
            { — Алгоритм поиска кратчайшего — }
end.      { — пути в ациклическом орграфе — }

```

Очевидно, что трудоемкость первого этапа равна $O(m)$, поскольку необходимо просмотреть все ребра, а трудоемкость второго — $O(n)$, так как, в худшем случае, кратчайший путь может включать все вершины графа.

Критический путь. Ациклические орграфы являются наглядным средством отображения взаимосвязей и порядка следования этапов сложных долговременных проектов, таких как строительство крупных сооружений, проведение широкомаштабных научно-исследовательских работ и т.п. Пусть каждая вершина графа отображает определенный этап проекта, дуга, идущая от вершины v_i к вершине v_j , определяет отношение предшествования соответствующих этапов, а ее вес временную задержку между началом v_i и началом v_j . При такой интерпретации граф на рис. 4.13 вполне может рассматриваться как сетевой график некоторого проекта. При этом естественно возникает задача об определении наиболее ранних сроков его выполнения. Ясно, что минимально возможное время реализации проекта равно времени прохождения по самому длинному пути между начальной и конечной вершинами. Этот путь называют *критическим*. Метод отыскания

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}
v_1	0	5*	10*		19	7							
v_2		5		14*		16*	9*	8					
v_3			10		11*				26		30		
v_4				19						18		21	
v_5					21				3				
v_6						21		11*					
v_7							14			30*			
v_8								32	13*			17	
v_9									45		15*		
v_{10}										44			8
v_{11}											60	19*	12
v_{12}												79	14*
v_{13}													93

Основное
расчетное
соотношение

$$l(v_j) = \max_{v_i \in \Gamma^-(v_j)} [l(v_i) + c_{i,j}], \quad j=2, 13$$

Критический путь

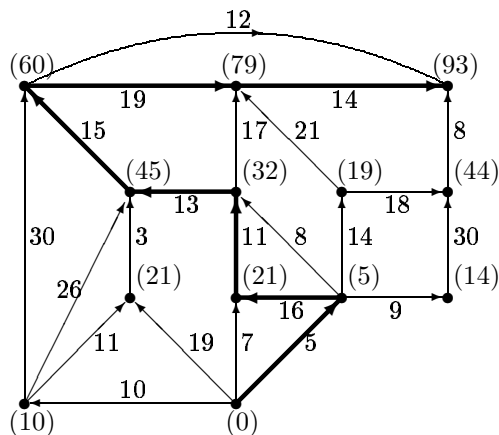
$$v_1 - v_2 - v_6 - v_8 - v_9 - v_{11} - v_{12} - v_{13} \quad l(v_{13}) = 93$$


Рис. 4.15

такого пути тот же, что и в задаче о кратчайшем пути, только в формуле (*) операция взятия минимума заменяется на операцию взятия максимума. На рис. 4.15 приведено решение задачи о критическом пути для рассматриваемого графа.

5 Циклы

5.1 Фундаментальные циклы и разрезы

Будем представлять циклы (n, m) -графа в виде m -разрядных двоичных кодов, в которых разряд j равен 1, если ребро e_j принадлежит циклу, и 0, если ребро циклу не принадлежит. Пусть, например, множество ребер некоторого графа $E = \{e_1, e_2, \dots, e_{10}\}$, и в графе есть цикл $e_1 - e_2 - e_6 - e_9 - e_8 - e_4$, тогда соответствующий код запишется как $C = 1101010110$.

В общем случае в графе может присутствовать некоторое множество циклов. Так, в графе, изображенном на рис. 5.1,

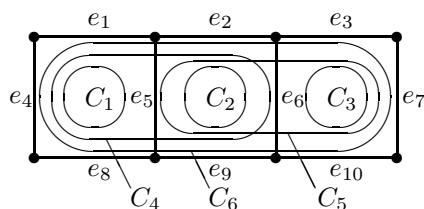


Рис. 5.1

существует шесть циклов $C_1 - C_6$. Однако они не независимы, поскольку любой из этих циклов может быть представлен как суперпозиция других (с помощью операции поразрядного сложения соответствующих кодов по модулю два). В частности, цикл $C_6 = C_1 \oplus C_2 \oplus C_3$, цикл $C_4 = C_1 \oplus C_2$ и цикл $C_5 = C_2 \oplus C_3$. В этих примерах циклы C_1, C_2 и C_3 выступают как "полное" множество независимых циклов, так как на их основе можно получить любой из остальных циклов графа. Среди таких множеств независимых циклов выделяют *множества фундаментальных циклов*.

Множеством фундаментальных циклов графа относительно остова T этого графа принято называть множество циклов, каждый из которых получается при добавлении к T некоторого ребра графа, не принадлежащего T .

Из определения следует, что мощность множества фундаментальных циклов не зависит от выбранного остова и все-

гда равна $\nu = m - n + 1$. Это *цикломатическое число* – одна из важнейших характеристик графа. Ясно также, что все циклы такого множества независимы, поскольку каждый из них содержит ребро, принадлежащее только этому циклу.

В качестве примера для графа на рис. 5.2 представлены все фундаментальные циклы относительно остова, образованного ребрами e_5, e_6, e_7, e_8, e_9 (выделены толстыми линиями). На рис. 5.3 для того же графа показано множество независи-

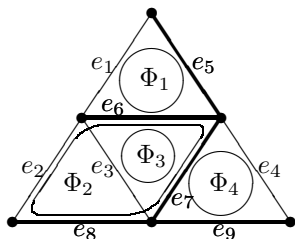


Рис. 5.2

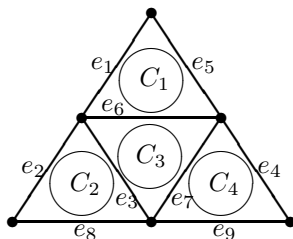


Рис.5.3

мых циклов, которое не является множеством фундаментальных циклов, так как нет соответствующего остова¹⁶.

Описание фундаментальных циклов удобно выполнять в виде матрицы $\Phi = [\phi_{ij}]$, строки которой соответствуют циклам, а столбцы – ребрам, причем ϕ_{ij} равно 1, если e_j принадлежит циклу Φ_i , или 0, если не принадлежит. Каноническая форма записи матрицы получается, если предварительно пронумеровать сначала все неостовные ребра, а затем остовные, как на рис 5.2. В этом случае матрица выглядит так:

$$\Phi = \begin{matrix} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \\ \begin{matrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \Phi_4 \end{matrix} & \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \end{array} \right], \end{matrix}$$

или сокращенно $\Phi = (\mathbf{I}_\nu | \Phi')$, где $\mathbf{I}_\nu$ - единичная матрица порядка ν , а Φ' - матрица размера $\nu \times (n-1)$, определяющая,

¹⁶Цикл C_3 не может быть получен путем суперпозиции C_1, C_2, C_4 , и поэтому независим.

какие ребра остова принадлежат соответствующему циклу.

С понятием цикла связано понятие *разреза*, определяемого как *множество ребер S , удаление которых приводит к несвязному графу*. Примеры разрезов даны на рис. 5.4а, б.

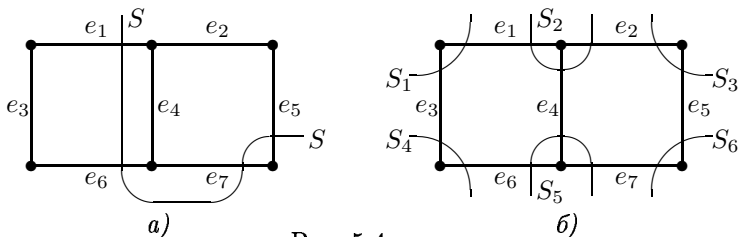


Рис. 5.4

Разрез S называется *простым* или *правильным*, если ни какое его собственное подмножество не является разрезом. Иными словами, простой разрез является минимальным разрезом и всегда разбивает граф на две компоненты. Некоторое множество простых разрезов графа показано на рис. 5.4б:

$$\begin{aligned} S_1 &= \{e_1, e_3\}, & S_2 &= \{e_1, e_2, e_4\}, & S_3 &= \{e_2, e_5\}, \\ S_4 &= \{e_3, e_6\}, & S_5 &= \{e_4, e_6, e_7\}, & S_6 &= \{e_5, e_7\}. \end{aligned}$$

Разрез $S = \{e_1, e_5, e_6, e_7\}$ на рис. 5.4а простым не является, так как состоит из двух простых $\{e_1, e_6\}$ и $\{e_5, e_7\}$.

По аналогии с циклами будем описывать разрезы m -разрядными двоичными кодами, в которых разряд j равен 1, если ребро e_j принадлежит разрезу, и 0, если не принадлежит. В этом случае описание разрезов $S_1 - S_6$ имеет вид:

$$\begin{aligned} S_1 &= 1010000, & S_2 &= 1101000, & S_3 &= 0100100, \\ S_4 &= 0010010, & S_5 &= 0001011, & S_6 &= 0000101. \end{aligned}$$

Легко заметить, что код любого из этих шести разрезов может быть получен как сумма остальных пяти по модулю два, то есть разрезы $S_1 - S_6$ зависимы. Если же ограничиться любыми пятью из них, то получим множество простых независимых разрезов, с помощью которых можно сформировать любой разрез в графе. Так, разрез S на рис. 5.4а можно получить как сумму $S = S_2 \oplus S_3 \oplus S_5$.

Среди множеств независимых разрезов выделяют *множества фундаментальных разрезов*.

Множеством фундаментальных разрезов графа относительно некоторого остова T этого графа называют множество разрезов, каждый из которых содержит только одно ребро, принадлежащее T .

Из определения следует, что мощность множества фундаментальных разрезов не зависит от выбранного остова и равна $\rho = n - 1$. Это *коцикломатическое число* графа. Такое название объясняется тем, что наряду с термином "фундаментальный разрез" используют и термин "коцикл", подчеркивая этим связь фундаментальных циклов и разрезов.

Описание фундаментальных разрезов удобно выполнять в виде матрицы $S = [s_{ij}]$, строки которой соответствуют разрезам, а столбцы – ребрам, причем s_{ij} равно 1, если e_j принадлежит разрезу S_i , или 0, если не принадлежит. Каноническая форма записи матрицы получается, если предварительно пронумеровать сначала все неостовные ребра, а затем остовные. Тогда матрица фундаментальных разрезов выглядит как на

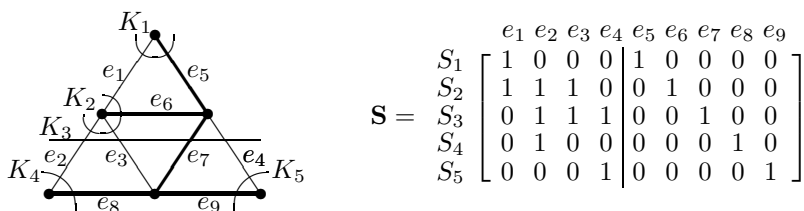


Рис. 5.5

рис. 5.5 или сокращенно $S = (S' | I_\rho)$, где I_ρ - единичная матрица порядка ρ , определяющая, какое ребро остова принадлежит разрезу, а S' - матрица размера $\rho \times \nu$.

Покажем, как связаны матрицы Φ и S . Число общих ребер любого цикла и любого разреза может быть только четным (в том числе и нулевым), и этот факт также означает, что количество единиц, стоящих на одинаковых позициях в строке i матрицы Φ и в строке j матрицы S , четно. Следовательно, $\Phi S^T \equiv 0 \pmod{2}$. С другой стороны, если принять во внимание блочную структуру Φ и S , то, по правилам умножения блочных матриц, можно записать

$$\Phi S^T = (I_\nu | \Phi') (S' | I_\rho)^T = (I_\nu | \Phi') \begin{pmatrix} S'^T \\ I_\rho^T \end{pmatrix} = I_\nu S'^T + \Phi' I_\rho = S'^T + \Phi',$$

где I_ν и I_ρ – единичные матрицы порядка ν и ρ соответственно, а операции выполняются по модулю два. Учитывая, что $\Phi S^T = 0 \pmod{2}$, имеем $S'^T + \Phi' = 0$, откуда следует, что $S'^T = \Phi'$, так как сложение ведется по модулю два.

5.2 Эйлеровы циклы

5.2.1 Определение и условия существования

Эйлеров цикл это простой цикл, содержащий все ребра связного неориентированного мультиграфа. Простую цепь, проходящую через все ребра, называют *эйлеровой цепью*. Граф, в котором есть эйлеров цикл, называется *эйлеровым*, а граф, содержащий эйлерову цепь – *полуэйлеровым*.

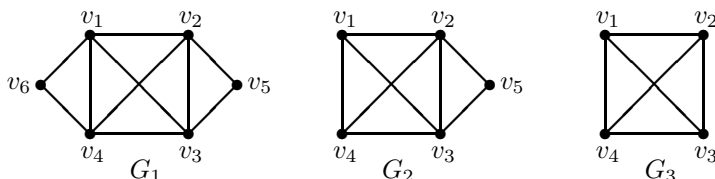


Рис. 5.6

Среди графов, представленных на рис. 5.6, граф G_1 является эйлеровым, так как в нем есть цикл, содержащий все ребра, например, $(v_1 - v_2 - v_3 - v_4 - v_1 - v_3 - v_5 - v_2 - v_4 - v_6 - v_1)$; в графе G_2 подобный цикл отсутствует, но есть эйлерова цепь, например, $(v_1 - v_2 - v_3 - v_4 - v_1 - v_3 - v_5 - v_2 - v_4)$, поэтому граф G_2 полуэйлеров; наконец, граф G_3 не эйлеров и не полуэйлеров: в нем нет соответствующего цикла или цепи.

Своим названием рассматриваемые здесь циклы обязаны Эйлеру, который установил простой признак существования таких циклов в графе, решив знаменитую в свое время задачу о Кенигсбергских мостах. Сама задача состоит в следующем. Семь мостов в тогдашнем Кенигсберге (ныне Калининград) соединяли берега реки Прегель, а также два острова на ней,

в соответствии со схемой, изображенной на рис. 5.6а. Спрашивается, можно ли, выйдя из некоторого пункта в городе, вернуться обратно, пройдя *один* раз по каждому мосту?

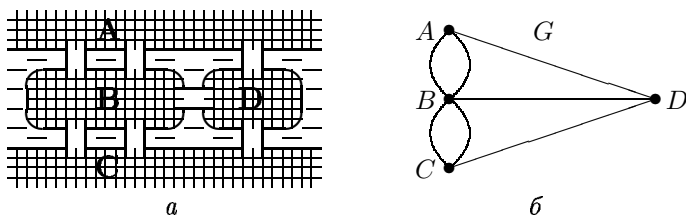


Рис. 5.7

Дадим графовое представление схемы на рис. 5.6а, отобразив четыре участка суши **A, B, C, D**, разделенные рекой, четырьмя вершинами A, B, C, D , а семь мостов, связывающих эти участки семью ребрами мультиграфа G , как показано на рис. 5.6б. При таком представлении задачу о Кёнигсбергских мостах можно сформулировать иначе: есть ли в графе G цикл, содержащий все его ребра? В данном случае ответ отрицателен в соответствии со следующей теоремой, носящей также имя Эйлера.

Теорема 5.1 *Связный неориентированный граф содержит эйлеров цикл тогда и только тогда, когда степени всех его вершин четны.*

▷ *Необходимость.* Пусть граф содержит эйлеров цикл. Поставим в соответствие каждой вершине графа счетчик с нулевым начальным значением. Обойдем цикл, начиная с произвольного ребра и увеличивая на 2 содержимое счетчика вершины при каждом ее прохождении, что соответствует использованию двух очередных инцидентных вершине ребер. Очевидно, что по окончании обхода, когда все ребра использованы, счетчики будут содержать степени вершин, которые при вышеуказанном приращении могут иметь только четные значения.

Достаточность. Пусть степени всех вершин графа $G(V, E)$ четны. Такой граф является, по крайней мере, двусвязным и, следовательно, содержит цикл. Для построения цикла произвольно выберем одну из вершин. Пусть это будет v_1 . Перейдем по некоторому ребру в смежную вершину, пометив ребро как

использованное. Будем повторять эту операцию, каждый раз выбирая новое ребро, пока не окажемся в исходной вершине v_1 . Обозначим найденный цикл как C_1 . Если C_1 включает все ребра графа G , то это и будет требуемый эйлеров цикл. В противном случае рассмотрим граф $G_1(V, E - EC_1)$, получающийся при удалении из G всех ребер, относящихся к C_1 . Этот граф может быть как связным, так и несвязным, но любая его компонента в силу связности G имеет, хотя бы одну общую вершину с графом C_1 . Так как каждая вершина в G и C_1 имеет четную степень, то и в G_1 степени всех вершин должны быть четными. Пусть v_2 общая вершина G_1 и C_1 , не являющаяся изолированной в графе G_1 . Найдем в G_1 цикл, содержащий эту вершину, и обозначим его как C_2 . Объединим циклы C_1 и C_2 , как показано на рис. 5.8. Если цикл

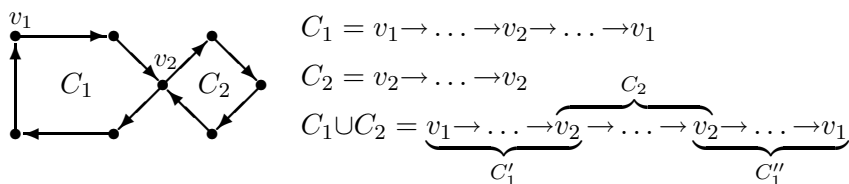


Рис. 5.8

$C_1 \cup C_2$ эйлеров, то процесс завершен. В противном случае следует перейти к рассмотрению графа $G_2(V, E - EC_1 - EC_2)$, отыскать в нем цикл C_3 , который следует объединить с C_1 и C_2 и т.д. Так как каждый раз количество неиспользованных ребер уменьшается, то описанный процесс конечен и должен закончиться построением эйлерова цикла. $\triangleleft$

Аналогичная теорема для орграфов формулируется так:

Теорема 5.2 *Связный ориентированный граф содержит эйлеров орицикл (орцепь) тогда и только тогда, когда полустепени исхода и полустепени захода всех вершин удовлетворяют условиям:*

$$\deg^+ v = \deg^- v \quad \forall v \in V \quad - \text{ в случае орицикла;}$$

$$\deg^+ v = \deg^- v \quad \forall v \neq (u \text{ или } w) \text{ и}$$

$$\deg^+ u = \deg^- u + 1, \quad \deg^- w = \deg^+ w + 1 \quad - \text{ в случае орцепи,}$$

где u - начальная, w - конечная вершины цепи.

5.2.2 Алгоритм поиска эйлерова цикла

Существует очень простая и наглядная процедура отыскания эйлерова цикла, известная как *алгоритм Флёрри*, которая заключается в следующем. Начиная с некоторой вершины, произвольным образом "идем" по смежным ребрам графа, удаляя пройденное ребро и вершину, ставшую после удаления ребра изолированной. Не проходим по ребру, если после его удаления граф становится несвязным. Нетрудно показать, что процесс завершается после обхода (удаления) всех ребер, а значит, и вершин графа, и обязательно в исходной вершине, степень которой к этому моменту становится нулевой. Отметим, что каждая вершина (в отличие от ребер, проходимых единожды), вообще говоря, может проходиться неоднократно, точнее – $\deg v_i / 2$ раз. Порядок, в котором были пройдены ребра, определяет конфигурацию найденного цикла.

Пусть, например, требуется найти эйлеров цикл в графе, представленном на рис. 5.9а. Непосредственной проверкой

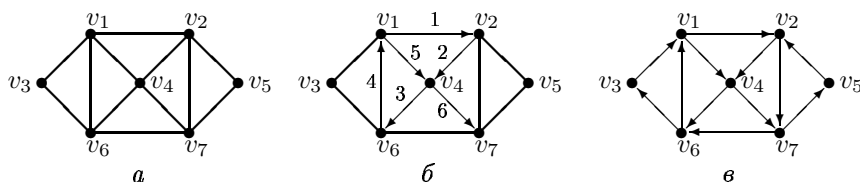


Рис. 5.9

убеждаемся, что степени всех вершин четны, т.е. граф эйлеров. В качестве начальной вершины выберем v_1 . На первой итерации перейдем в вершину v_2 и удалим ребро $\{v_1, v_2\}$ ¹⁷. На второй итерации перейдем в вершину v_4 и удалим ребро $\{v_2, v_4\}$, на третьей – в v_6 , удалив $\{v_4, v_6\}$, затем вновь попадаем в v_1 , потом еще раз в v_4 и, наконец, в v_7 . То, что получится после этого, представлено на рис. 5.9б. Вершина v_4 оказалась изолированной и может быть удалена. Ребро $\{v_7, v_6\}$ стало мостом. Поэтому допустим переход только в v_2 или в v_5 . Перейдем в вершину v_5 , удалив ребро $\{v_7, v_5\}$, как пока-

¹⁷Удаленные ребра пронумерованы и отображены тонкими линиями со стрелками, показывающими направление обхода.

зано на рис.5.9в. Теперь "остаток" графа – простая цепь v_5 , следовательно, дальнейшие переходы оределены однозначно $\rightarrow v_2 \rightarrow v_7 \rightarrow v_6 \rightarrow v_3 \rightarrow v_1$ (рис. 5.9в). Окончательно получим

$$v_1 \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_1 \rightarrow v_4 \rightarrow v_7 \rightarrow v_5 \rightarrow v_2 \rightarrow v_7 \rightarrow v_6 \rightarrow v_3 \rightarrow v_1.$$

Вышеописанная процедура поиска эйлерова цикла ориентирована, главным образом, на работу с изображением графа и идеально подходит для решения задач-головоломок типа "нарисовать граф (картинку), не отрывая карандаш от бумаги и не повторяя линий". Если же граф задан иначе, например в матричной форме, или в виде списка, то, чтобы установить, не является ли очередное выбираемое ребро мостом (не прибегая к изображению), требуются дополнительные операции. Поэтому для компьютерной реализации лучше подходит алгоритм, основанный на процедуре, использованной при доказательстве теоремы 5.1, в соответствии с которым первоначально найденный цикл последовательно расширяется за счет объединения с другими циклами, пока не будут задействованы все ребра. Формализованная запись алгоритма имеет вид:

```

begin                                { — Э Й Л Е Р — }
  select  $v \in V$ ;                      { Выбрать произвольную вершину }
   $v \rightarrow S$ ;                        { и занести в стек. }
  while  $S \neq \emptyset$  do { Пока стек не пуст, выполнять действия: }
    { if  $\Gamma(s_1) \neq \emptyset$  { если окружение вершины  $s_1$ , }
      { находящейся в верхушке стека, не пусто, }
      { то }
      then {
        select  $u \in \Gamma(s_1)$ ; { выбрать одну из ее }
         $u \rightarrow S$ ; { "соседок" и занести в стек, }
         $\Gamma(s_1) := \Gamma(s_1) - \{u\}$ ; { а затем удалить }
         $\Gamma(u) := \Gamma(u) - \{s_1\}$  { пройденное ребро; }
        { иначе }
      }
      else {
         $S \rightarrow v$ ; { получить из стека }
        output( $v$ ) { и вывести очередную }
        { вершину цикла. }
      }
    }
  end.                                { — Э Й Л Е Р — }

```

Предполагается, что граф задан списком вершин (списком смежности), а в качестве основной рабочей структуры данных, позволяющей легко зафиксировать последовательность

прохождения вершин, использован стек (S).

Рассмотрим работу алгоритма на примере. Пусть требуется найти эйлеров цикл для графа, изображенного на рис. 5.10 и заданного списками смежности. Фактически мы имеем мульт-

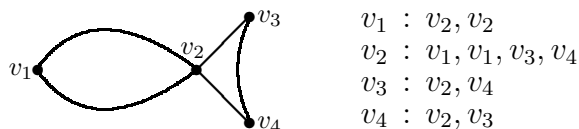


Рис. 5.10

тиграф, поэтому списки для вершин v_1 и v_2 содержат повторяющиеся элементы, отражая наличие параллельных ребер. Выберем в качестве начальной вершину v_1 и занесем ее в стек. Дальнейшие действия отражены в табл. 5.1 и табл. 5.2.

Таблица 5.1

Итерация	Стек	$S \leftarrow$	Удаляемое ребро	$S \rightarrow$	Итерация	Стек	$S \rightarrow$
1	v_1	v_2	$\{v_1, v_2\}$	—	7	$v_1 v_2 v_3 v_4 v_2$	v_2
2	$v_1 v_2$	v_1	$\{v_2, v_1\}$	—	8	$v_1 v_2 v_3 v_4$	v_4
3	$v_1 v_2 v_1$	—	—	v_1	9	$v_1 v_2 v_3$	v_3
4	$v_1 v_2$	v_3	$\{v_2, v_3\}$	—	10	$v_1 v_2$	v_2
5	$v_1 v_2 v_3$	v_4	$\{v_3, v_4\}$	—	11	v_1	v_1
6	$v_1 v_2 v_3 v_4$	v_2	$\{v_4, v_2\}$	—		$\emptyset$	—

Таблица 5.2

Итерации	1	2	3	4	5	6	7
$v_1 :$	$\cancel{v_2}, v_2$	$\cancel{v_2}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$v_2 :$	$\cancel{v_1}, v_1, v_3, v_4$	$\cancel{v_1}, v_3, v_4$	v_3, v_4	$\cancel{v_3}, v_4$	v_4	$\cancel{v_4}$	$\emptyset$
$v_3 :$	v_2, v_4	v_2, v_4	v_2, v_4	$\cancel{v_2}, v_4$	$\cancel{v_4}$	$\emptyset$	$\emptyset$
$v_4 :$	v_2, v_3	v_2, v_3	v_2, v_3	v_2, v_3	$v_2, \cancel{v_3}$	$\cancel{v_2}$	$\emptyset$

В табл. 5.1 представлены операции со стеком, а в табл. 5.1 показано, как изменяются списки вершин от итерации к итерации (столбцы) по мере удаления ребер.

Сначала в вершине стека находится v_1 . Поэтому в списке

смежности для v_1 произвольно выбираем некоторую вершину (в данном случае это может быть только v_2) и заносим ее в стек. Удаляем пройденное ребро $\{v_1, v_2\}$. Для этого из списка смежности для v_1 исключаем вершину v_2 , а из списка смежности для v_2 вершину v_1 . Теперь состояние стека соответствует второй строке табл. 5.1, а вид списка смежности – второму столбцу табл. 5.2. На второй итерации в вершине стека содержится v_2 . Теперь выбираем вершину из текущего списка смежности для v_2 (см. второй столбец табл. 5.2). Пусть это будет вершина v_1 . Занесем ее в стек и удалим ребро $\{v_2, v_1\}$. В результате новое состояние стека соответствует третьей строке табл. 5.1, а вид списка смежности – третьему столбцу табл. 5.2. На третьей итерации список смежности для вершины v_1 , находящейся в верхушке стека, оказался пустым. Выталкиваем ее из стека. Это означает, что в графе найден цикл, содержащий все ребра, инцидентные v_1 , но не являющийся эйлеровым.

На трех последующих итерациях в стек заносятся вершины v_3, v_4, v_2 , а из списков смежности удаляются все оставшиеся элементы. Таким образом, к началу седьмой итерации содержимое стека принимает вид $v_1 v_2 v_3 v_4 v_2$, причем вершине соответствует самый правый элемент v_2 . Так как все списки смежности теперь пусты, в течение последних пяти итераций (7 – 11) выполняется только выталкивание вершин из стека. Список всех вершин, выданных из стека (в порядке выдачи), имеет вид $v_1 v_2 v_4 v_3 v_2 v_1$ и, как нетрудно заметить, соответствует эйлерову циклу данного графа.

В этом графе есть еще один эйлеров цикл $v_1 v_2 v_3 v_4 v_2 v_1$. Ясно, что любые два таких цикла отличаются друг от друга только порядком обхода ребер.

5.2.3 О количестве эйлеровых графов

Задача определения числа эйлеровых графов является достаточно сложной. Рассмотрению этой, а также других задач пересчета и перечисления на графах, посвящена работа [XX]. Здесь ограничимся теоремой, характеризующей долю эйлеро-

вых графов в общем числе простых помеченных графов.

Теорема 5.3 *Почти все графы не являются эйлеровыми.*

▷ Пусть $\mathcal{G}(n)$ – множество всех простых помеченных графов порядка n , а $\mathcal{G}_\text{ч}(n)$ – множество всех графов из $\mathcal{G}(n)$, не имеющих вершин нечетной степени.

Любой граф из $\mathcal{G}_\text{ч}(n)$ можно получить, добавляя к некоторому графу из $\mathcal{G}(n-1)$ еще одну вершину и соединяя ее со всеми вершинами нечетной степени (если таковые имеются), поэтому $|\mathcal{G}_\text{ч}(n)| < |\mathcal{G}(n-1)|$. Так как множество эйлеровых графов $\mathcal{G}_\text{э}(n)$ является подмножеством $\mathcal{G}_\text{ч}(n)$, запишем $|\mathcal{G}_\text{э}(n)| < |\mathcal{G}(n-1)|$. Разделив обе части неравенства на $|\mathcal{G}(n)|$ и учитывая, что $|\mathcal{G}(n)| = 2^{C(n,2)}$ и $|\mathcal{G}(n-1)| = 2^{C(n-1,2)}$, получим $|\mathcal{G}_\text{э}(n)|/|\mathcal{G}(n)| < 2^{C(n-1,2)-C(n,2)}$. Упростив показатель степени $C(n-1,2) - C(n,2) = \frac{(n-1)!}{2!(n-3)!} - \frac{n!}{2!(n-2)!} = \frac{(n-1)(n-2)}{2} - \frac{n(n-1)}{2} = -(n-1)$, окончательно имеем, что $|\mathcal{G}_\text{э}(n)|/|\mathcal{G}(n)| < 2^{-(n-1)}$, а переходя к пределу при $n \rightarrow \infty$, $\lim_{n \rightarrow \infty} |\mathcal{G}_\text{э}(n)|/|\mathcal{G}(n)| = 0$. Это значит, что при неограниченном увеличении числа вершин, в общем числе графов доля эйлеровых становится исчезающе малой, т.е. $|\mathcal{G}_\text{э}(n)| = o(|\mathcal{G}(n)|)$. ◁

5.2.4 Задача почтальона

Пусть G связный граф, имеющий k вершин нечетной степени и заданный матрицей весов ребер. Требуется найти кратчайший замкнутый маршрут, проходящий по всем ребрам графа. В качестве приложений, где может потребоваться решение этой или сходной задачи, можно указать контроль различных коммуникационных сетей (дорожных, электрических, газовых и т.п.), когда необходимо проверять все протяженные компоненты сети. В литературе по теории графов сформулированная выше задача называется *задачей почтальона*. Последнему далеко не безразлично знание наиболее короткого замкнутого маршрута обхода всех улиц своего участка (или некоторого их подмножества). Аналогичная задача стоит и перед курьером большого учреждения, доставляющим документы от одного "источника" к различным получателям.

Покажем, как можно решить рассматриваемую задачу. Ясно, что в графе существуют $k(k-1)/2$ кратчайших простых цепей, связывающих все пары вершин нечетной степени. Выберем из них $k/2$ цепей, таких, что

- а) никакие две не имеют одинаковых конечных вершин;
- б) суммарная длина всех этих цепей минимальна.

Заметим, что выбранные цепи не могут иметь общих ребер, как на рис. 5.11, что противоречило бы пункту б. Действительно, если цепь $v_1 - \dots - v_2$ имеет общее ребро e с цепью $v_3 - \dots - v_4$, то, удалив это ребро, получим две цепи, связывающих v_1 с v_3 и v_2 с v_4 , суммарная длина которых меньше,

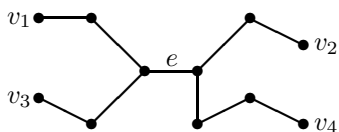


Рис. 5.11

чем у двух первых, на величину равную удвоенному весу ребра e . Продублируем ребра всех выбранных цепей в графе G . В результате получим эйлеров мультиграф G' , в котором нетрудно найти эйлеров цикл.

Этот цикл и определяет искомый маршрут, отличаясь от него тем, что вместо обхода двух параллельных ребер в графе G' дважды проходит по соответствующему ребру в графе G . Очевидно, что длина маршрута равна сумме длин ребер G' .

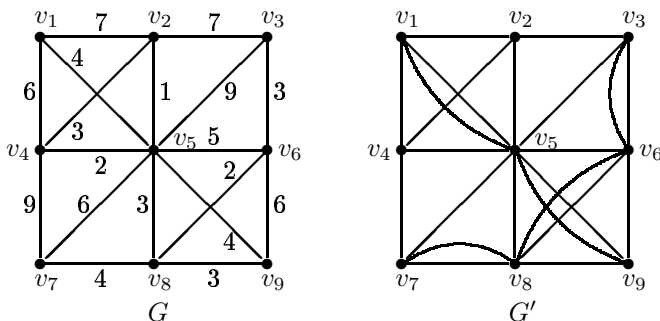


Рис. 5.12

Рассмотрим пример. Пусть требуется найти "маршрут почтальона" в графе G , изображенном на рис. 5.12 (веса ребер проставлены на рисунке). Имеются четыре вершины нечетной степени v_1, v_3, v_7 и v_9 , из которых можно составить $C_4^2=6$

различных пар. Найдем кратчайшую цепь для каждой пары. С этой целью составим матрицу весов графа и, используя алгоритм Флойда, получим матрицу $D^{(9)}$ длин кратчайших путей (цепей) между всеми вершинами¹⁸. Выделим из неё подматрицу $D_{1,3,7,9}^{(9)}$, образованную строками и столбцами с номерами 1, 3, 7 и 9. Это матрица длин всех кратчайших цепей, связывающих только вершины нечетной степени в рассматриваемом графе. Обе матрицы $D^{(9)}$ и $D_{1,3,7,9}^{(9)}$ приведены ниже:

$$D^{(9)} = \begin{matrix} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 \\ \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{matrix} & \begin{bmatrix} 0 & 5 & 12 & 6 & 4 & 9 & 10 & 7 & 8 \\ 5 & 0 & 7 & 3 & 1 & 6 & 7 & 4 & 5 \\ 12 & 7 & 0 & 10 & 8 & 3 & 9 & 5 & 8 \\ 6 & 3 & 10 & 0 & 2 & 7 & 8 & 5 & 6 \\ 4 & 1 & 8 & 2 & 0 & 5 & 6 & 3 & 4 \\ 9 & 6 & 3 & 7 & 5 & 0 & 6 & 2 & 5 \\ 10 & 7 & 9 & 8 & 6 & 6 & 0 & 4 & 7 \\ 7 & 4 & 5 & 5 & 3 & 2 & 4 & 0 & 3 \\ 8 & 5 & 8 & 6 & 4 & 5 & 7 & 3 & 0 \end{bmatrix} \end{matrix}, D_{1,3,7,9}^{(9)} = \begin{matrix} & v_1 & v_3 & v_7 & v_9 \\ \begin{matrix} v_1 \\ v_3 \\ v_7 \\ v_9 \end{matrix} & \begin{bmatrix} 0 & 12 & 10 & 8 \\ 12 & 0 & 9 & 8 \\ 10 & 9 & 0 & 7 \\ 8 & 8 & 7 & 0 \end{bmatrix} \end{matrix}.$$

Теперь, используя матрицу $D_{1,3,7,9}^{(9)}$, следует выбрать пару цепей, отвечающих приведенным ранее условиям. Имеется всего три варианта образования таких пар цепей, представленные в табл. 5.3. Поэтому задача решается путем простого перебора.

Таблица 5.3

Пара цепей	Цепь	Длина	Цепь	Длина	Общая длина
1	$v_1 - v_3$	12	$v_7 - v_9$	7	$12+7=19$
2	$v_1 - v_7$	10	$v_3 - v_9$	8	$10+8=18$
3	$v_1 - v_9$	8	$v_3 - v_7$	9	$8+9=17$

Как следует из таблицы наилучшим является вариант 3. Следовательно, дублированию подлежат ребра, образующие кратчайшие цепи из v_1 в v_9 и из v_3 в v_7 . Непосредственно по рис. 5.12

¹⁸В общем случае, кроме $D^{(n)}$, необходимо вычислять и матрицу вершин-предшественниц $P^{(n)}$ (см. разд. 4).

легко установить, что это цепи $v_1-v_5-v_9$ и $v_1-v_6-v_8-v_7$. После дублирования пяти входящих в них ребер получаем эйлеров мультиграф G' , изображенный на рис. 5.12. Используя, алгоритм Флчри, находим следующий вариант эйлерова цикла в графе G' : $v_1v_2v_3v_6v_3v_5v_2v_4v_1v_5v_4v_7v_8v_7v_5v_6v_9v_8v_6v_8v_5v_9v_5v_1$, который одновременно является и кратчайшим замкнутым маршрутом почтальона в графе G при условии, что ребра $\{v_1, v_5\}$, $\{v_5, v_9\}$, $\{v_3, v_6\}$, $\{v_6, v_8\}$, $\{v_7, v_8\}$ обходятся дважды.

В рассмотренном примере граф содержал только четыре вершины нечетной степени. Поэтому поиск оптимума потребовал анализа всего трех вариантов решения. При большем числе вершин количество вариантов быстро возрастает. Так, если $k=6$, то существует $C_6^2=15$ различных кратчайших цепей, связывающих пары вершин нечетной степени. При этом число возможных комбинаций по три цепи с различными концевыми вершинами равно 15. Если $k=8$, то число возможных комбинаций составляет 105, причем каждая включает в себя по четыре цепи. В общем же случае количество комбинаций, которые придется анализировать, определяется как $3 \cdot 5 \cdot 7 \cdot \dots (k-1)$, где k -число вершин нечетной степени. Таким образом, фактически приходится решать самостоятельную задачу, эквивалентную задаче отыскания *наибольшего паросочетания*¹⁹ *минимального веса* в полном графе с четным числом вершин.

5.3 Гамильтоновы циклы

5.3.1 Определение и условия существования

Гамильтоновым называют простой цикл, включающий все вершины связного неориентированного графа. Аналогичным образом для орграфа определяется гамильтонов контур. Граф, в котором есть гамильтонов цикл (контур), носит название гамильтонова.

¹⁹Паросочетанием называется множество ребер графа, выбранных так, что никакие два из них не являются смежными.

Слово "гамильтонов" в этих определениях является производным от имени математика У. Гамильтона, предложившего следующую игру под названием "Кругосветное путешествие". Каждой из двадцати вершин додекаэдра приписано название одного из городов мира. Требуется, переходя по ребрам додекаэдра от одного города к другому и посетив каждый из них только один раз, оказаться в исходной точке. Очевидно, что задача, сводится к отысканию в графе додекаэдра (граф G_1 на рис. 5.13) простого цикла, проходящего через все вершины.

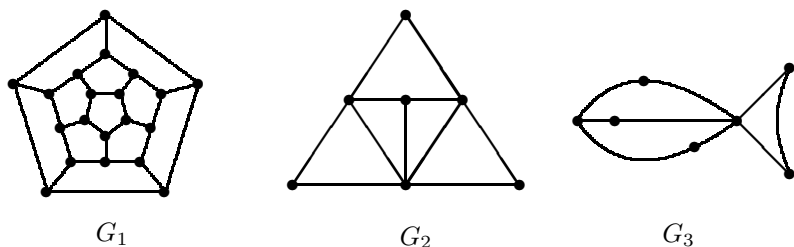


Рис. 5.13

В графе G_2 , изображенном на рис. 5.13, гамильтонов цикл отсутствует, но имеется простая цепь, включающая все вершины. Такую цепь называют *гамильтоновой цепью*, а содержащий её граф – *полугамильтоновым*. Наконец, в графе G_3 на том же рисунке не существует ни гамильтоновой цепи, ни тем более, гамильтонова цикла. В таком случае говорят, что граф не гамильтонов.

Рассмотрим следующую задачу планирования, характерную для химической промышленности. Имеется единственный комплекс аппаратуры или реактор, на котором можно производить n различных видов продукции v_i , причем в любой момент времени производится только один вид. При переходе к производству другого вида в зависимости конкретного сочетания предшествующего и последующего видов либо требуется подготовка аппаратуры (настройка, очистка, охлаждение и т.п.), либо не требуется. Продукты производятся в непрерывном цикле, так что после производства в заданных количествах всех n видов процесс повторяется. Естественно возникает вопрос о том, существует ли циклическая последова-

тельность производства n продуктов, вообще не требующая остановок для подготовки аппаратуры.

Пусть G – оргграф, вершины которого соответствуют продуктам v_i и v_j , а дуги $\{v_i, v_j\}$ отражают тот факт, что продукт v_j может производиться после v_i без остановки аппаратуры. Теперь ответ на поставленный вопрос зависит от того, существует ли в графе гамильтонов контур или не существует.

Сравнивая определения эйлера и гамильтонова циклов, нельзя не заметить их существенное сходство. В первом случае накладывается требование однократного прохождения по каждому ребру графа (через вершины можно проходить неоднократно!), а во втором – требование однократного прохождения через каждую вершину (ребра, задействованные в цикле также проходятся однократно). И поскольку для эйлеровых циклов имеется весьма простое необходимое и достаточное условие существования, естественно предположить (в силу отмеченного сходства), что и для гамильтоновых циклов есть подобный критерий. Однако такой критерий не найден. Более того, его поиск – одно из важных направлений теории графов. В настоящее время известно несколько достаточных условий существования гамильтоновых циклов. Одно из них устанавливается следующей теоремой Оре.

Теорема 5.4 *Если для любой пары $\{v, w\}$ несмежных вершин графа G порядка $n \geq 3$ выполняется неравенство $\deg v + \deg w \geq n$, то G – гамильтонов граф.*

▷ Предположим, что существует негамильтонов граф $G(V, E)$, в котором для любой пары несмежных вершин выполняется неравенство $\deg v + \deg w \geq n$. Будем добавлять к графу новые вершины, соединяя каждую из них с каждой вершиной G , до тех пор, пока получающийся при этом граф $G'(V', E')$ не станет гамильтоновым. Обозначим через k минимальное количество таких дополнительных вершин.

Пусть $v \rightarrow p \rightarrow w \rightarrow \dots \rightarrow v$ – гамильтонов цикл в G' , где $v, w \in V$, а p – одна из добавленных вершин. Заметим, что вершины v и w не являются смежными ни в G' , ни тем более, в G (в противном случае, при указанной структуре цикла вершина p не нужна, что противоречит предположению о мини-

мальности k). Более того, любая вершина $w' \in \text{adj } w$ не может следовать за вершиной $v' \in \text{adj } w$ потому, что в этом случае цикл принимает вид $v \rightarrow p \rightarrow w \rightarrow \dots \rightarrow v' \rightarrow w' \rightarrow \dots \rightarrow v$ и тогда опять вершина p оказывается лишней. Действительно, если "перевернуть" фрагмент $w \rightarrow \dots \rightarrow v'$ ($v' \rightarrow \dots \rightarrow w$), то цикл можно организовать так $v \rightarrow v' \rightarrow \dots \rightarrow w \rightarrow w' \rightarrow \dots \rightarrow v$, не используя p . Таким образом, в гамильтоновом цикле графа G' не может быть фрагментов $v' \rightarrow w'$, т.е. за каждой вершиной, смежной с v должна следовать вершина не смежная с w . Это возможно при условии, что $|\text{adj } w| \geq \deg v$, где $\text{adj } w$ – множество вершин не смежных с w . Количество таких вершин в графе G' , очевидно, равно $n+k - \deg w$. Поэтому приведенное условие можно переписать так: $n+k - \deg w \geq \deg v$, или, собирая степени в правой части, $n+k \geq \deg w + \deg v$. При построении графа G' степень каждой вершины графа G увеличилась на k , значит применительно к G неравенство принимает вид $n+k \geq \deg w + \deg v + 2k$, а с учетом исходного предположения $\deg v + \deg w \geq n$, имеем $n+k \geq n+2k$. Последнее неравенство удовлетворяется лишь при $k=0$. Поэтому $G'=G$ и, следовательно, исходный граф гамильтонов. $\triangleleft$

Из доказанной теоремы вытекает следующая:

Теорема 5.5 *Если для любой вершины v графа G порядка $n > 3$ выполняется неравенство $\deg v \geq n/2$, то G – гамильтонов граф.*

К сожалению, эффективность приведенных в обеих теоремах критериев оставляет желать лучшего. Действительно, так как сумма степеней всех вершин графа равна удвоенному числу его ребер, то очевидно, что в гамильтоновом графе, отвечающем условиям последней теоремы, должно быть не менее $n^2/4$ ребер. В частности, при $n=20$ это составляет 100 ребер. Однако граф с таким количеством вершин может быть гамильтоновым и при существенно меньшем числе ребер. Например, в графе G_1 на рис. 5.13, также имеющем двадцать вершин, всего тридцать ребер, и степени всех его вершин равны трем (а не десяти, как того требует теорема), но он гамильтонов. Еще более разительным примером является простой цикл C_{20} , в котором только двадцать ребер, а степени вершин равны двум.

Несколько утрируя суть дела, можно сказать: данные критерии, отражают тот очевидный факт, что при одном и том же числе вершин чем больше в графе ребер, тем выше вероятность того, что он гамильтонов.

Низкая эффективность подобных критериев, по-видимому, объясняется тем, что граница между гамильтоновыми и негамильтоновыми графами определяется не только величиной его связности, которая во многом определяется степенями вершин, но и другими структурными особенностями графа. Наглядным подтверждением этого может служить следующая теорема, являющаяся одним из наиболее эффективных критериев, касающихся гамильтоновых графов.

Теорема 5.6 *Любой 4-связный²⁰ планарный²¹ граф является гамильтоновым.*

В плане рассмотрения вопроса о критериях гамильтоновости несколько "утешает" факт, декларируемый следующей теоремой, которая отражает типичный случай.

Теорема 5.7 *Почти все графы гамильтоновы.*

Это значит, что с ростом числа вершин, доля гамильтоновых графов постоянно увеличивается, стремясь в пределе к 1. Так из 34 графов, имеющих 5 вершин, только 8 гамильтоновых, тогда как среди 156 графов с 6 вершинами их уже 48.

5.3.2 Методы поиска гамильтоновых циклов

В основе этих методов лежит тот или иной способ реализации перебора различных вариантов цепей. Наиболее простым, но вместе с тем и наиболее трудоемким представляется решение, когда после выбора вершины $v_{\text{нач}}$ рассматриваются различные перестановки из множества остальных $n-1$ вершин и для каждой проверяется наличие в графе соответствующей

²⁰Связность (реберная) определяется как наименьшее количество ребер, удаление которых приводит к несвязному графу.

²¹Планарным называется граф, который может быть изображен на плоскости так, что никакие два ребра не пересекаются.

гамильтоновой цепи. Если такая цепь есть и последняя вершина цепи связана с $v_{\text{нач}}$, найден гамильтонов цикл. Очевидно, что в худшем случае при этом методе потребуется сформировать все $(n-1)!$ перестановок, причем большая часть работы оказывается проделанной впустую. На практике применяют методы частичного перебора, в основе которых лежат различные критерии, позволяющие исключить рассмотрение вариантов цепей, отсутствующих в заданном графе. Рассмотрим два таких метода, демонстрирующих разные подходы к поиску гамильтоновых циклов.

Алгебраический метод. В раз. 4.3 была введена матрица всех маршрутов с l промежуточными вершинами для всех упорядоченных пар вершин графа, обозначенная как $\mathbf{P}^{(l)}$ и определяемая с помощью рекуррентного соотношения $\mathbf{P}^{(l)} = \mathbf{P}' \times \mathbf{P}^{(l-1)}$, $l=1, 2, \dots$, где в качестве $\mathbf{P}^{(0)}$ берется матрица смежности графа, а $\mathbf{P}'$ получается из той же матрицы смежности, если в каждом ее столбце единицы заменить меткой соответствующей вершины.

Очевидно, что матрицу всех гамильтоновых цепей графа можно получить, исключив из матрицы $\mathbf{P}^{(n-2)}$ произведения с повторяющимися сомножителями (считая и конечные вершины цепей). Диагональные элементы обработанной таким образом матрицы $\mathbf{P}^{(n-2)}$ в совокупности описывают все простые циклы графа, содержащие $(n-1)$ вершин, а диагональные элементы матрицы $\mathbf{P}^{(n-1)}$ после аналогичной обработки дают все простые циклы графа, включающие n вершин, т.е. гамильтоновы циклы, причем *любой диагональный элемент отражает все циклы*. Это значит, что вместо матрицы $\mathbf{P}^{(n-1)}$ достаточно сформировать только один из ее столбцов, соответствующий произвольно выбранной вершине v . Искомый результат содержится в строке v найденного столбца. Для вычислений можно использовать соотношение $\mathbf{P}_v^{(l)} = \mathbf{P}' \times \mathbf{P}_v^{(l-1)}$, где $l=1, 2, \dots, (n-1)$, а $\mathbf{P}_v^{(l)}$ – столбец матрицы $\mathbf{P}^{(l)}$, соответствующий вершине v . В качестве $\mathbf{P}_v^{(0)}$ следует принять v -столбец матрицы смежности графа.

Работа только с одним столбцом матрицы $P_v^{(l)}$ позволяет в n раз уменьшить трудоемкость вычислений. Последняя, однако, и в этом случае остается весьма существенной (порядка n^3). Дополнительный выигрыш можно получить, если на каждом шаге после получения очередного столбца $P_v^{(l)}$ исключать из него произведения с повторяющимися сомножителями, а также произведения, в которых хоть один из элементов совпадает с концевой вершиной соответствующей цепи. Кроме того, целесообразно "обнулять" и диагональный элемент столбца, так как последний отражает лишь циклы длины $(l+1) < n$, содержащие вершину v .

В качестве примера на применение описанного метода решим задачу поиска гамильтоновых циклов в графе с матрицей смежности A , представленном на рис. 5.14. Пусть $v = a$.

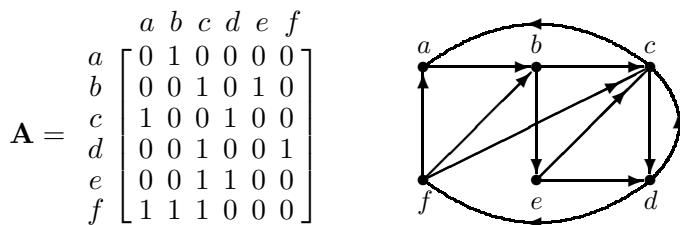


Рис. 5.14

Тогда $P_a^{(0)}$ – первый столбец матрицы смежности, а матрица P' получается из матрицы смежности, как описано ранее.

$$P' = \begin{matrix} & \begin{matrix} a & b & c & d & e & f \end{matrix} \\ \begin{matrix} a \\ b \\ c \\ d \\ e \\ f \end{matrix} & \begin{bmatrix} 0 & b & 0 & 0 & 0 & 0 \\ 0 & 0 & c & 0 & e & 0 \\ a & 0 & 0 & d & 0 & 0 \\ 0 & 0 & c & 0 & 0 & f \\ 0 & 0 & c & d & 0 & 0 \\ a & b & c & 0 & 0 & 0 \end{bmatrix} \end{matrix}, \quad P_a^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

В данном графе шесть вершин. Поэтому решение получим за пять итераций. Результаты вычислений приведены ниже. Подчеркнуты элементы, подлежащие удалению перед выполнени-

ем следующей итерации. Это диагональные элементы промежуточных столбцов и произведения с повторами вершин.

$$1) \mathbf{P}'\mathbf{P}_a^{(0)} = \begin{bmatrix} 0 \\ c \\ 0 \\ f \\ c \\ c \end{bmatrix} = \mathbf{P}_a^{(1)}; \quad 2) \mathbf{P}'\mathbf{P}_a^{(1)} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \begin{bmatrix} \underline{bc} \\ \underline{ec} \\ \underline{df} \\ \underline{fc} \\ \underline{df} \\ \underline{bc} \end{bmatrix}, \quad \mathbf{P}_a^{(2)} = \begin{bmatrix} 0 \\ \underline{ec} \\ \underline{df} \\ \underline{fc} \\ \underline{df} \\ \underline{bc} \end{bmatrix};$$

$$3) \mathbf{P}'\mathbf{P}_a^{(2)} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \begin{bmatrix} \underline{bec} \\ \underline{cdf + edf} \\ \underline{dfc} \\ \underline{cdf + fbc} \\ \underline{cdf + dfc} \\ \underline{bec + cdf} \end{bmatrix}, \quad \mathbf{P}_a^{(3)} = \begin{bmatrix} 0 \\ \underline{cdf + edf} \\ 0 \\ \underline{fbc} \\ \underline{cdf + dfc} \\ \underline{bec} \end{bmatrix};$$

$$4) \mathbf{P}'\mathbf{P}_a^{(3)} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \begin{bmatrix} 0 \\ \underline{ecd f + ed f c} \\ \underline{dfbc} \\ \underline{fbec} \\ \underline{dfbc} \\ \underline{bcd f + bed f} \end{bmatrix}, \quad \mathbf{P}_a^{(4)} = \begin{bmatrix} 0 \\ \underline{ecd f + ed f c} \\ 0 \\ \underline{fbec} \\ \underline{dfbc} \\ 0 \end{bmatrix};$$

$$5) \mathbf{P}'\mathbf{P}_a^{(4)} = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \begin{bmatrix} \underline{becdf + bedfc} \\ \underline{edfbc} \\ \underline{dfbec} \\ 0 \\ \underline{dfbec} \\ \underline{becdf + bedfc} \end{bmatrix}, \quad \mathbf{P}_a^{(5)} = \begin{bmatrix} \underline{becdf + bedfc} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Диагональный элемент последнего столбца содержит два произведения вершин. Это означает, что в рассматриваемом графе есть два гамильтоновых цикла: $a \rightarrow b \rightarrow e \rightarrow c \rightarrow d \rightarrow f \rightarrow a$ и $a \rightarrow b \rightarrow e \rightarrow d \rightarrow f \rightarrow c \rightarrow a$, что легко проверяется непосредственно по рис. 5.14.

Метод последовательного перебора. В противоположность алгебраическому методу, когда одновременно формируются все цепи, этот метод предполагает их последовательное построение. На каждом шаге "наращивается" только одна

цепь путем добавления дуги, связывающей ее последнюю вершину с одной из смежных, еще *не включенных* в цепь. Очевидно, что этот процесс конечен в силу указанного условия и конечности числа вершин графа и завершается, когда такое наращивание становится невозможным. Если к моменту останова цепь содержит все вершины графа, то есть является гамильтоновой, остается выяснить, имеется ли дуга, соединяющая конец цепи с ее началом. В противном случае имеем простую цепь, которая не может быть даже частью гамильтонова цикла.

Пусть a – начальная вершина сформированной цепи, v – конечная, а P – множество вершин, вошедших в цепь. Тогда варианты завершения описанного процесса выглядят так:

- а) $|P|=n$ и $a \in \Gamma(v)$ – найден гамильтонов цикл;
- б) $|P|=n$ и $a \notin \Gamma(v)$ – найдена гамильтонова цепь;
- в) $|P|<n$ и $\Gamma(v) \subseteq P$ – получена простая цепь длины $|P|-1$, которая не может быть продлена.

Если требуется найти лишь один гамильтонов цикл, то в случае (а) задачу можно считать решенной. Если же ищутся все циклы, а также в случаях (б) и (в), необходимо вернуться к предпоследней вершине найденной цепи $u = \text{Pred}(v)$, отметив при этом вершину v как "прозондированную" из вершины u и удалив ее из P . Теперь в качестве конечной вершины следует рассматривать вершину u и попытаться продлить цепь, выбрав одну из вершин множества $\Gamma(u)$, исключая вершины, содержащиеся в P , и вершины, прозондированные из u . Если же множество $\Gamma(u) - P$ – "прозондированные из u " оказывается пустым, следует выполнить возврат к предпоследней вершине текущей цепи, то есть к вершине $w = \text{Pred}(u)$, исключив при этом u из множества P и сняв пометку "прозондирована" со всех вершин множества $\Gamma(u) - P$. В процессе обработки текущая цепь меняется на каждом шаге, удлиняясь или укорачиваясь, причем ее промежуточные состояния соответствуют всем простым цепям графа, начинающимся в вершине a . Перебор завершается, когда прозондированными оказываются все вершины множества $\Gamma(a)$. К этому моменту должны быть выявлены все гамильтоновы циклы графа.

Покажем на примере как "работает" рассматриваемый метод. Для этого используем граф из предыдущего раздела, продублировав его на рис. 5.15 и дополнив (вместо матрицы инцидентности) списком смежности. Для простоты изложения

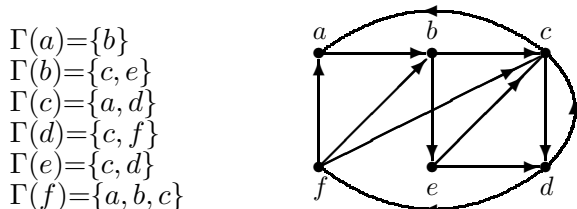


Рис. 5.15

будем рассматривать множества P и $\Gamma(v_i)$ как упорядоченные, а результаты преобразования цепи по шагам заносить в таблицу. В качестве исходной опять берем вершину a . Поэтому в начале множество $P = \{a\}$. Фиксируем это в первой строке табл. 5.4. Имеется единственная вершина, непосредственно доступная из a . Это – вершина b . Поэтому включаем

Таблица 5.4

Итерация	Цепь	Итерация	Цепь	Итерация	Цепь	Итерация	Цепь
1	a	7	a, b, c	13	a, b, e, c, d	19	a, b, e, d, f
2	a, b	8	a, b	14	a, b, e, c	20	a, b, e, d
3	a, b, c	9	a, b, e	15	a, b, e	21	a, b, e
4	a, b, c, d	10	a, b, e, c	16	a, b, e, d	22	a, b
5	a, b, c, d, f	11	a, b, e, c, d	17	a, b, e, d, f	23	a
6	a, b, c, d	12	a, b, e, c, d, f	18	a, b, e, d, f, c	24	

венно доступная из a . Это – вершина b . Поэтому включаем ее в формируемую цепь. Соответственно множество P принимает вид $\{a, b\}$ ²². Теперь пытаемся нарастить цепь от вершины b . Из нее, как показывает список смежности (и рисунок), выходят две дуги (b, c) и (b, e) , и обе могут быть использованы, так как ни одна из вершин c и e не принадле-

²²Здесь и далее P отождествляем с формируемой цепью.

жит множеству P . Выбираем первую из них c и включаем в P , получая после этого цепь a, b, c . На третьем шаге имеем $\Gamma(c) = \{a, d\}$, но первая в этом списке вершина $a \in P$, поэтому берем вторую – d , с результатом $P = \{a, b, c, d\}$. На четвертом шаге, поступая аналогично, добавляем вершину f , и цепь принимает вид a, b, c, d, f . Дальнейшее наращивание становится невозможным, так как все вершины множества $\Gamma(f)$ уже содержатся в сформированной цепи. Поэтому возвращаемся к вершине d и вновь анализируем ее список смежности $\Gamma(d) = \{c, f\}$. Здесь $c \in P$, а вершина f только что прозондирована, и значит "кандидатов" для присоединения к цепи нет. Исключаем d и возвращаемся к вершине c . С ней ситуация повторяется, поэтому после седьмой итерации цепь сокращается до двух вершин и принимает вид a, b , причем вершина c помечена как прозондированная из b . Список смежности вершины b содержит вершины c и e , из которых только e не прозондирована из b и не содержится в текущей цепи. Поэтому следует перейти к вершине e . На девятом шаге из двух подходящих вершин ($\Gamma(e) = \{c, d\}$) выбираем c , после чего цепь выглядит так a, b, e, c . Еще две итерации (10 и 11) продлевают цепь до гамильтоновой, которая имеет вид a, b, e, c, d, f , и поскольку в графе есть дуга (f, a) , получаем первый гамильтонов цикл. Далее следуют итерации, сокращающие цепь до тех пор, пока в списке смежности последней вершины не найдется подходящего продолжения. Такая ситуация наступает на шаге 15, когда цепь принимает вид a, b, e . Из двух вершин множества $\Gamma(e)$ одна еще не использована. Это вершина d . Выбрав ее, в итоге после шага 17 получаем еще один гамильтонов цикл a, b, e, d, f, c . Дальнейшие действия приводят только к сокращению цепи. Процесс завершается на шаге 23, когда в цепи остается только исходная вершина a . Таким образом, окончательно имеем два цикла.

Формализованное описание рассмотренного способа поиска гамильтоновых циклов, в котором:

P – множество вершин, включенных в формируемую цепь;

$Pred_u$ – вершина, предшествующая вершине u в цепи;

W_u – множество вершин, "прозондированных" из u ;

$v_{\text{нач}}$ – исходная вершина при поиске циклов;

v – последняя вершина текущей цепи,

можно представить в следующем виде:

```

begin      { — Г А М И Л Ь Т О Н — }
  for  $u \in V$  do      { Вначале для каждой вершины графа  $u$  }
    {  $W_u := \emptyset;$       { множество прозондированных из  $u$  пусто; }
       $Pred_u := u;$       { в качестве предшествующей принята  $u$ . }
     $v := v_{\text{нач}};$       { За последнюю вершину цепи взята исходная, }
     $P := \{v\};$           { т.е. вначале цепь состоит из одной вершины. }
    while  $P \neq \emptyset$  do      { Выполнять пока  $P$  не пусто: }
      { if  $\Gamma(v) - P - W_v \neq \emptyset$  { Есть возможность продлить цепь? }
        { да }
        then {
          { select  $u \in \Gamma(v) - P - W_v;$  { Выбрать вершину  $u;$  }
             $P := P + \{u\};$               { включить в цепь; }
             $Pred_u := v;$               { теперь  $v$  предшествует  $u;$  }
             $v := u;$                   { далее считать  $u$  последней в цепи. }
            if  $(P=V)$  and  $(v_{\text{нач}} \in \Gamma(v))$ 
              { найден гамильтонов цикл }
              then {
                while  $Pred_u \neq \emptyset$  do { Вывод списка }
                  { output( $u$ );          { вершин цепи, }
                     $u := Pred_u;$         { начиная }
                  output( $u$ );          { с конечной. }
                { нет }
              }
            {  $P := P - \{v\};$  { Исключить последнюю вершину; }
               $W_v := \emptyset;$  { "опустошить" множество вершин, }
                { прозондированных из  $v;$  }
            else {
               $u := Pred_v;$       {  $u$  – предпоследняя вершина; }
               $W_u := W_u + \{v\};$  {  $v$  прозондирована из  $u;$  }
               $v := u;$           { далее считать  $u$  последней в цепи. }
            }
          }
        { — Г А М И Л Ь Т О Н — }
      end.

```

Выявление негамильтоновости графа. В общем случае установить является граф гамильтоновым или негамильтоновым, как уже было сказано в 5.3.1, достаточно сложно. Однако в некоторых случаях негамильтоновость можно выявить, не прибегая к трудоемким алгоритмам. Рассмотрим несколько примеров, иллюстрирующих простые приемы решения этой задачи. Пусть требуется определить характер гра-

фов, представленных на рис. 5.16.

Граф G_1 имеет семь вершин. Значит соответствующий гамильтонов цикл должен состоять из семи ребер, т.е. иметь длину равную семи. Но, из рисунка видно, что граф G_1 двудольный и поэтому (в силу теоремы 1.2) может содержать

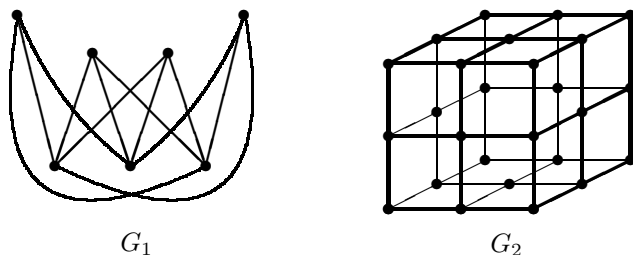


Рис. 5.16

только циклы четной длины. Отсюда следует, что граф негамильтонов. Обобщая, можно утверждать, что *любой двудольный граф с нечетным числом вершин не может быть гамильтоновым*. Таким образом, решая задачу выявления негамильтоновости для графа с нечетным числом вершин, целесообразно проверить его на двудольность, используя алгоритм, изложенный в разделе 1.3.²³

Перейдем к анализу графа G_2 , изображение которого на рис. 5.16 совмещено с трехмерным кубом так, что вершины графа совпадают с серединами граней и ребер куба, а также с его вершинами. В подобных случаях говорят, что граф *уложен* на некоторой поверхности. В данном случае это поверхность куба. Легко подсчитать, что граф G_2 имеет 26 вершин и 48 ребер. Так как число вершин четно, вышеописанный прием здесь неприемлем. Поступим иначе, но предварительно введем понятие *независимого множества вершин*, под которым следует понимать любое множество вершин, таких, что никакие две из них не смежны. Независимое множество называ-

²³Заметим, что в общем случае в отличие от представленного на рис. 5.16 изображения G_1 , двудольность графа совсем не очевидна. Так граф G_2 на рис. 5.16 также двудольный, но чтобы установить это требуются некоторые усилия.

ется *максимальным*, если оно не является собственным подмножеством некоторого независимого множества вершин. В графе G_2 есть два таких множества. Одно из них V_1 включает все вершины, находящиеся в середине ребер куба, другое V_2 состоит из вершин, совпадающих с вершинами куба и серединами его граней. Рассмотрим множество V_1 . Каждая из входящих в него вершин инцидентна четырем ребрам, из которых только два могут принадлежать гамильтонову циклу. Так как $|V_1|=12$, то общее число ребер, не входящих в цикл только по множеству V_1 , равно 24. Оставшихся 24 ребра из 48 не достаточно для организации гамильтонова цикла, который в данном графе должен бы (по числу вершин) иметь длину 26. А это значит, что граф G_2 негамильтонов.

Оригинальный способ решения задачи, использующий особенности структуры графа, предложил Ю.Н.Заваровский²⁴.

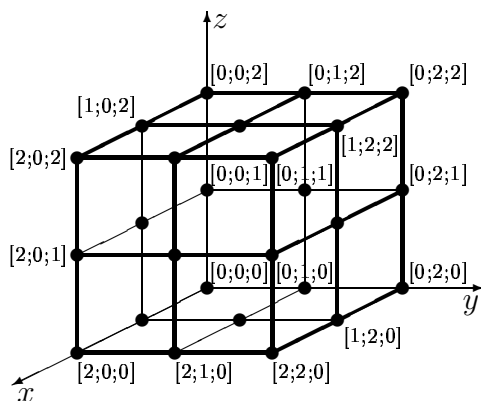


Рис. 5.17

Свяжем с кубом, на котором уложен рассматриваемый граф, декартову систему координат с единичным отрезком, соответствующим ребру графа, как показано на рис. 5.17. Тогда каждой вершине можно приписать числовую пометку, равную сумме ее координат. Ясно, что при переходе из одной вершины в другую, смежную с ней, значение пометки увеличивает-

²⁴ Доцент кафедры высшей и прикладной математики ПензГУ. Он же ознакомил автора с задачей, хотя и несколько в иной формулировке.

ся или уменьшается на единицу, становясь либо четным, либо нечетным. При таком характере чередования в гамильтоновом цикле на 26 вершин должно быть 13 вершин с четными и 13 с нечетными пометками. Однако простой подсчет показывает, что в графе 14 "четных" и 12 "нечетных" вершин, т.е. гамильтонова цикла быть не может.

Если использовать понятие двудольности, можно дать и более простое обоснование полученного результата. Поскольку граф двудольный, любой цикл в нем, в том числе и гамильтонов, должен содержать четное число вершин при строгом чередовании по их принадлежности к разным долям. Следовательно для гамильтоновости графа, необходимо чтобы доли имели равное число вершин. В рассматриваемом случае это не так, значит граф не гамильтонов.

Вышеизложенное позволяет сделать вывод о том, что *любой двудольный граф с четным числом вершин может быть гамильтоновым только при равенстве его долей.*

5.4 Задача коммивояжера

Так называется широко известная классическая задача исследования операций, которая в графовой интерпретации формулируется следующим образом. Имеется полный граф (орграф), заданный матрицей весов C . Требуется найти гамильтонов цикл с минимальной суммой, составляющих его ребер (дуг). Если трактовать вес ребра $c_{i,j} \geq 0$ как расстояние между вершинами v_i и v_j , то задача заключается в отыскании кратчайшего гамильтонова цикла.

В литературе по исследованию операций часто приводится следующая формулировка, определившая название задачи. Коммивояжер (агент по сбыту) должен посетить n городов, начиная с одного из них, и вернуться в исходный пункт, побывав в каждом только один раз. Для любой пары городов v_i и v_j известно расстояние (время или стоимость переезда, перелета) $c_{i,j}$. Каким должен быть кратчайший (наиболее дешевый) маршрут коммивояжера?

5.4.1 Применение и методы решения

На практике с этой оптимизационной задачей и различными ее модификациями постоянно сталкиваются фирмы и агентства, занимающиеся доставкой грузов. Еще одна область применения это планирование производства n различных продуктов на единственном комплекте технологического оборудования (см. раздел 5.3.1). Если $c_{i,j}$ это время на переналадку оборудования при переходе к производству продукта v_j после продукта v_i , то наилучший по времени порядок производства всех n продуктов определяется кратчайшим гамильтоновым циклом в соответствующем полном графе.

Наглядный пример применения задачи коммивояжера приведен в [10]. При производстве печатных плат необходимо сделать большое количество отверстий в различных точках платы. Автоматический сверлильный станок, выполняющий эту операцию должен наряду с собственно сверлением, выполнять позиционирование сверла (платы). Временные затраты на это действие непосредственно зависят от порядка обхода вышеуказанных точек. Минимизировать общее время выполнения подобной обработки платы можно за счет выбора оптимальной последовательности обхода точек сверления, т.е. перед нами задача коммивояжера.

Не смотря на простоту формулировки задача оказывается весьма сложной. Это одна из оптимизационных комбинаторных задач, для которых эффективные алгоритмы неизвестны. Существуют различные подходы к ее решению.

Простейший подход, основанный на переборе всех вариантов, годится лишь для графов малой размерности. Так при числе вершин $n=6$, необходимо сформировать $(n-1)! = 5! = 120$ перестановок, каждая из которых определяет некоторый гамильтонов цикл, подсчитать, используя матрицу весов, длину всех этих циклов и выбрать из них оптимальный(е).

Более продуктивным является подход, основанный на частичном переборе, когда на основе некоторых критериев отбрасываются заведомо бесперспективные варианты. Именно такой подход реализуется с помощью метода ветвей и границ.

5.4.2 Метод ветвей и границ

Начнем с изложения принципа, составляющего суть метода, после чего перейдем к деталям и примеру на применение метода. Рассмотрение будем вести применительно к ориентированным графам. Имеется множество S_0 гамильтоновых циклов различной длины. Нас интересует цикл (циклы), длина которого минимальна. Естественным первоначальным шагом является оценка нижней границы этой величины. Соответствующую процедуру рассмотрим позднее, а пока полагаем, что оценка нижней границы L_0 найдена. Верхняя граница длины минимального цикла для реализации метода не требуется, хотя на практике ее знание может быть полезным. В качестве верхней границы достаточно принять длину любого гамильтонова цикла, например, цикла $v_1, v_2, v_3, \dots, v_n$, которая легко подсчитывается по весовой матрице. Описываемый далее процесс для четырехвершинного графа можно представить с помощью бинарного дерева, изображенного на рис. 5.18.

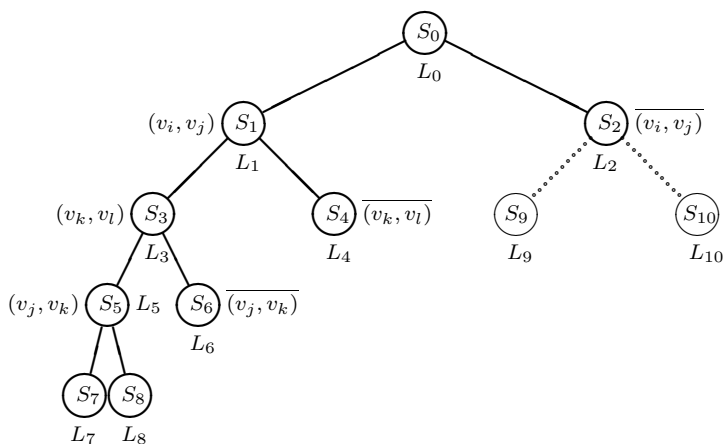


Рис. 5.18

Мысленно разобьем множество всех циклов на два непересекающихся подмножества, используя в качестве критерия вхождение в цикл некоторой дуги, например, (v_i, v_j) . Пусть S_1 подмножество гамильтоновых циклов, содержащих дугу

(v_i, v_j) , а S_2 подмножество циклов, не содержащих эту дугу. Для каждого подмножества находим нижнюю границу длины минимального цикла. Пусть это будут L_1 и L_2 соответственно, причем $L_1 < L_2$. Выбираем подмножество S_1 с меньшей нижней границей и разбиваем его на два непересекающихся подмножества S_3 и S_4 , но уже на основе другой дуги, например (v_k, v_l) . Затем находим нижние границы L_3 и L_4 для этих двух подмножеств. Предположим, что $L_3 < L_4$. Это значит, что разбиению подлежит подмножество S_3 . Используя для этой цели дугу (v_j, v_k) , получаем S_5 и S_6 с нижними границами L_5 и L_6 соответственно. Наконец, разбиение множества S_5 по дуге (v_l, v_i) дает подмножества S_7 и S_8 , из которых первое состоит из единственного гамильтонова цикла (v_i, v_j, v_k, v_l) , причем нижняя граница L_7 есть длина этого цикла. Подмножество S_8 пусто, так как в четырехвершинном графе не может быть гамильтоновых циклов, содержащих дуги (v_i, v_j) , (v_j, v_k) , (v_k, v_l) и не содержащих дугу (v_l, v_i) . В качестве нижней границы для S_8 положим ∞ . На этом процесс ветвления дерева можно считать законченным, при условии, что L_7 строго меньше нижних границ всех подмножеств, соответствующих листьям дерева, (L_2, L_4, L_6, L_8) . В разбираемом примере по отношению к S_6 и S_8 это условие выполняется автоматически ($L_6 = L_8 = \infty$), тогда как S_4 и S_2 вполне могут иметь нижние границы не превышающие L_7 . Если например $L_2 \leq L_7$, то процесс ветвления следует продолжить, как показано на рисунке пунктирными линиями. И вполне может оказаться, что оптимальное решение окажется на этом пути. Но в любом случае процесс завершается, когда подмножество с минимальной нижней границей сократится до одного цикла.

Заметим, что в полном орграфе с четырьмя вершинами любая дуга определяет два различных гамильтонова цикла, то есть в нашем случае множество S_1 состоит всего из двух циклов. Более того, в таком графе любой гамильтонов цикл однозначно определяется двумя дугами. Поэтому множество S_3 содержит один цикл, включающий дуги (v_i, v_j) и (v_k, v_l) , а это значит, что и S_4 также состоит из единственного цикла, а S_6 пусто.

Теперь обратимся к способу оценки нижней границы длины гамильтоновых циклов. Используем тот факт, что если к

элементам i строки матрицы C прибавить или отнять некоторое число α_i , то длина всех гамильтоновых циклов соответственно увеличится или уменьшится на эту величину. Действительно, элементы i строки определяют длины всех дуг, выходящих из вершины v_i , и поскольку любой цикл обязательно содержит одну (и только одну) из этих дуг его длина изменится на величину α_i . Ясно, что при этом соотношения между циклами по длине сохраняются, т.е. самый короткий таковым и останется. Точно такие же рассуждения справедливы и для j столбца матрицы весов, с той лишь разницей, что речь идет обо всех дугах, входящих в вершину v_j .

Пусть $\alpha_i = \min\{c_{i,j}\}$, $j = \overline{1, n}$ – минимальный элемент i строки матрицы C . Преобразуем матрицу путем вычитания из элементов каждой строки соответствующего минимального элемента этой строки, положив $c_{i,j} = c_{i,j} - \alpha_i$, $j = \overline{1, n}$. Такая операция носит название *редукции матрицы по строкам*. Очевидно, что после ее выполнения каждая строка должна содержать хотя бы по одному нулевому элементу.

Совершенно аналогично определяется операция *редукции матрицы по столбцам*. Находятся минимальные элементы по столбцам $\beta_j = \min\{c_{i,j}\}$, $i = \overline{1, n}$, после чего выполняется преобразование $c_{i,j} = c_{i,j} - \beta_j$, $i = \overline{1, n}$ по всем столбцам матрицы. В результате в каждом столбце появится хотя бы один нулевой элемент.

Понятно, что в результате полной редукции весовой матрицы, как по строкам так и по столбцам, длины всех гамильтоновых циклов в графе должны уменьшиться на величину $\Delta = \sum_{i=1}^n \alpha_i + \sum_{j=1}^n \beta_j$. Эта величина может рассматриваться как безусловная нижняя граница длины кратчайшего гамильтонова цикла в полном орграфе.

Покажем как определяется нижняя граница множества гамильтоновых циклов, при условии, что все они содержат некоторую дугу, например, (v_i, v_j) . Так как любая вершина в цикле может иметь только одну входящую и одну выходящую дугу, использование (v_i, v_j) фактически накладывает запрет на включение остальных дуг, выходящих из v_i и входящих

в v_j . Запретить использование этих дуг можно, исключив из весовой матрицы строку i и столбец j . Кроме того, необходимо запретить использование дуги (v_j, v_i) , которая образует цикл с (v_i, v_j) . С этой целью достаточно положить $c_{j,i} = \infty$. Редукция преобразованной таким образом матрицы позволяет получить нижнюю границу множества циклов, содержащих дугу (v_i, v_j) .

Нижняя граница для множества гамильтоновых циклов, не содержащих (v_i, v_j) , получается путем редукции весовой матрицы, в которой элемент $c_{i,j}$ полагаем равным ∞ .

Наконец, дугу, по которой происходит разбиение очередного рассматриваемого подмножества гамильтоновых циклов, (см. бинарное дерево, на рис. 5.18) целесообразно выбирать из числа тех, веса которых в редуцированной матрице оказались нулевыми. Во-первых, это наиболее короткие дуги и, во-вторых, они являются элементами естественной (возможно даже циклической) структуры в графе, имеющей минимальную длину. Однако, количество таких дуг довольно велико, не менее числа вершин в графе. Задача выбора может быть решена путем оценки для каждой из них "штрафа" (потерь) за невключение дуги в гамильтонов цикл. Пусть, например, (v_i, v_j) одна из дуг с нулевым весом, не включенная в цикл. Значит в цикле имеется орцепь, связывающая вершину v_i с v_j . Начальной дугой этой цепи должна быть одна из дуг, выходящих из v_i и отличная от (v_i, v_j) , а конечной – одна из дуг, входящих в v_j и также отличная от (v_i, v_j) . На сумму весов этих дуг в редуцированной матрице и произойдет удлинение цикла при отказе от использования дуги (v_i, v_j) . Обозначим минимальный элемент i строки (не считая $c_{i,j}$) как A_i , а минимальный элемент j столбца (также, не считая $c_{i,j}$) как B_j . Назовем штрафом за неиспользование дуги (v_i, v_j) сумму $\phi_{i,j} = A_i + B_j$. Максимальный из штрафов по множеству "нулевых" дуг редуцированной матрицы определяет ту из них, по которой целесообразно выполнять дальнейшее разбиение очередного подмножества циклов.

Разберем пример на применение описанного варианта реализации метода ветвей и границ. Пусть граф задан матрицей

весов, представленной на рис. 5.19а.²⁵ Заметим, что в силу особой роли "нулевых" дуг, в качестве значения для диагональных элементов матрицы вместо нуля используется ∞ .

	v_1	v_2	v_3	v_4	v_5	v_6			
v_1	∞	3	93	13	33	9		v_1	∞ 0 75 2 30 6 3 2
v_2	4	∞	77	42	21	16		v_2	0 ∞ 58 30 17 12 4 12
v_3	45	17	∞	36	16	28		v_3	29 1 ∞ 12 0 12 16 1
v_4	39	90	80	∞	56	7		v_4	32 83 58 ∞ 49 0 7 32
v_5	28	46	88	33	∞	25		v_5	3 21 48 0 ∞ 0 25 3
v_6	3	88	18	46	92	∞		v_6	0 85 0 35 89 ∞ 3 0
								β_j	0 0 15 8 0 0 $\Delta=81$
								B_j	0 0 1 48 2 17 0 $\phi_{6,3}=48$

а
б

Рис. 5.19

Выполнив редукцию по строкам и по столбцам, получаем матрицу, показанную на рис. 5.19б. Там же представлены столбец α_i и строка β_j , использованные при редукции. Суммируя их содержимое, находим начальную оценку нижней границы длины гамильтоновых циклов $L_0 = \Delta = 81$. Для каждой "нулевой" дуги полученной матрицы отыскиваем составляющие штрафа $\phi_{i,j} = A_i + B_j$ и записываем их в столбец A_i и строку B_j . Максимальный штраф $\phi_{6,3} = A_6 + B_3 = 48$ соответствует дуге (v_6, v_3) . Поэтому именно ее целесообразно взять в качестве основы при разбиении множества гамильтоновых циклов на два подмножества $S(v_6, v_3)$ и $S(\overline{v_6}, \overline{v_3})$, таких, что все циклы первого содержат указанную дугу, все циклы второго ее не содержат. Этим подмножествам соответствуют матрицы на рис. 5.20а,в. Первая получена путем вычеркивания шестой строки и третьего столбца в матрице на рис. 5.19б. Кроме того элемент $c_{3,6}$ принят равным ∞ , чтобы заблокировать использование дуги (v_3, v_6) , образующей цикл с (v_6, v_3) . Вторая матрица (в) получается, если в матрице на

²⁵ Не обязательно, чтобы матрица была симметричной, хотя это представляется наиболее естественным при трактовке весов дуг как расстояний. Рассматриваемый алгоритм дает решение в самом общем случае.

рис. 5.19б значение элемента $c_{6,3}$ положить равным ∞ , запрещая тем самым использование дуги (v_6, v_3) .

Нижняя граница длины циклов подмножества $S(\overline{v_6, v_3})$ очевидно равна $L_2 = L_0 + \phi_{6,3} = 81 + 48 = 129$, т.е. по сравнению с найденной нижней границей длины всех циклов возрастает на величину $\phi_{6,3}$, как "расплата" за неиспользование дуги (v_6, v_3) .

$S(v_6, v_3)$						$S(\overline{v_6, v_3})$					
v_1	v_2	v_4	v_5	v_6		v_1	v_2	v_3	v_4	v_5	v_6
∞	0	2	30	6		∞	0	75	2	30	6
0	∞	30	17	12		0	∞	58	30	17	12
29	1	12	0	$\frac{12}{\infty}$		29	1	∞	12	0	12
32	83	∞	49	0		32	83	58	∞	49	0
3	21	0	∞	0		3	21	48	0	∞	0
						0	85	∞	35	89	∞
$L_1 = L_0 + \Delta = 81$						$L_2 = L_0 + \phi_{6,3} = 129$					
a						b					

Рис. 5.20

Чтобы определить L_1 – нижнюю границу длины циклов подмножества $S(v_6, v_3)$ необходимо провести редукцию матрицы, представленной на рис. 5.20а. Любая строка и столбец этой матрицы содержат нулевое значение, следовательно редукционированная матрица идентична исходной. В этом случае $\Delta = \sum \alpha_i + \sum \beta_j = 0$ (см. рис. 5.20б). Поэтому нижняя граница для подмножества $S(v_6, v_3)$ совпадает с ранее найденной нижней границей для всего множества гамильтоновых циклов, т.е. равна 81. Сравнивая нижние границы обоих подмножеств, убеждаемся, что для $S(v_6, v_3)$ она ниже. Значит разбиению подлежит именно это подмножество. Результаты поиска дуги, по которой следует производить разбиение представлены на рис. 5.20б. Максимальное значение штрафа $\phi_{4,6} = 32$ указывает на дугу (v_4, v_6) . Поэтому разбиваем $S(v_6, v_3)$ на два подмножества одно из которых – $S(v_4, v_6)$ состоит из циклов, содержащих дугу (v_4, v_6) , циклы другого – $S(v_4, v_6)$ этой дуги не содержат. Названным подмножествам соответствуют

матрицы на рис. 5.21а,в. Первая получается, если удалить строку v_4 и столбец v_6 матрицы на рис. 5.20. Это соответствует включению в цикл дуги (v_4, v_6) , в результате чего формируется орцепь (v_4, v_6, v_3) . Чтобы не допустить образования трехвершинного цикла (v_4, v_6, v_3, v_4) следует заблокировать использование обратной дуги (v_3, v_4) . Поэтому $c_{3,4}$ полагаем равным ∞ . Вторая матрица (в) получается, если в матрице на рис. 5.20б значение элемента $c_{4,6}$ положить равным ∞ , запрещая тем самым использование дуги (v_4, v_6) .

$S(v_4, v_6)$					$S(\overline{v_4, v_6})$												
	v_1	v_2	v_4	v_5		v_1	v_2	v_4	v_5	α_i	A_i		v_1	v_2	v_4	v_5	v_6
v_1	∞	0	2	30	v_1	∞	0	2	30	0	2	v_1	∞	0	2	30	6
v_2	0	∞	30	17	v_2	0	∞	30	17	0	17	v_2	0	∞	30	17	12
v_3	29	1	$\frac{12}{\infty}$	0	v_3	29	1	∞	0	0	1	v_3	29	1	12	0	∞
v_5	3	21	0	∞	v_5	3	21	0	∞	0	3	v_4	32	83	∞	49	∞
$L_3=L_1+\Delta=81$					β_j	0	0	0	0	0		v_5	3	21	0	∞	0
					B_j	3	1	2	17		20	$L_4=L_1+\phi_{4,6}=113$					
$\mathfrak{a}$					$\mathfrak{b}$								$\mathfrak{c}$				

Рис. 5.21

Нижняя граница длины циклов подмножества $S(\overline{v_4, v_6})$, т.е. циклов, содержащих (v_6, v_3) и не содержащих (v_4, v_6) , равна сумме нижней границы подмножества $S(v_6, v_3)$ и штрафа $\phi_{4,6}$ за неиспользование (v_4, v_6) : $L_4=L_1+\phi_{4,6}=81+32=113$.

Для определения нижней границы подмножества $S(v_4, v_6)$, в котором все циклы содержат (v_4, v_6) и (v_6, v_3) , редуцируем матрицу, представленную на рис. 5.21а. И здесь редуцированная матрица идентична исходной (см. рис. 5.21б). Поэтому нижняя граница L_3 подмножества $S(v_4, v_6)$ совпадает с ранее найденной нижней границей для $S(v_6, v_3)$ и равна 81. Это меньше, чем нижняя граница L_4 для $S(\overline{v_4, v_6})$ и нижняя граница L_2 для $S(\overline{v_6, v_3})$. Значит разбиению подлежит $S(v_4, v_6)$. Результаты поиска дуги, по которой следует производить разбиение представлены на рис. 5.21б. Максимальное значение штрафа $\phi_{2,1}=20$ указывает на дугу (v_2, v_1) . Разбива-

ем $S(v_4, v_6)$ на два подмножества $S(v_2, v_1)$ и $S(\overline{v_2, v_1})$. Первое состоит из циклов, содержащих дугу (v_2, v_1) , циклы второго этой дуги не содержат. Указанным подмножествам соответствуют матрицы на рис. 5.22а, в. Матрица *a* получается, если удалить строку v_2 и столбец v_1 из матрицы на рис. 5.21б, а элемент $c_{1,2}$ положить равным ∞ , чтобы исключить обратную дугу (v_1, v_2) . Матрица *в* получается, если в матрице на рис. 5.21б элемент $c_{2,1}$ положить равным ∞ , запрещая тем самым использование дуги (v_2, v_1) .

$S(v_2, v_1)$				$S(\overline{v_2, v_1})$			
	v_2	v_4	v_5		v_2	v_4	v_5
v_1	$\frac{0}{\infty}$	2	30	v_1	∞	0	28
v_3	1	∞	0	v_3	0	∞	0
v_5	21	0	∞	v_5	20	0	∞
$L_5=L_3+\Delta=84$				$L_6=L_3+\phi_{2,1}=101$			
	v_2	v_4	v_5	α_i	A_i		v_1
β_j	1	0	0	3			v_1
B_j	20	0	28		28		v_2
<i>a</i>				<i>б</i>			
	v_2	v_4	v_5		v_1	v_2	v_4
v_1	∞	0	2	v_1	∞	0	30
v_3	∞	∞	30	v_3	29	1	∞
v_5	21	0	∞	v_5	3	21	0
<i>в</i>				<i>г</i>			

Рис. 5.22

Нижняя граница L_6 подмножества $S(\overline{v_2, v_1})$ находится как сумма нижней границы L_3 для $S(v_4, v_6)$ и штрафа $\phi_{2,1}$ за неиспользование дуги (v_2, v_1) : $L_6 = 81 + 20 = 101$.

Чтобы найти нижнюю границу L_5 подмножества $S(v_2, v_1)$ следует выполнить редукцию матрицы *a*. Результат этой операции представлен на рис. 5.22б. Параметр $\Delta = \sum \alpha_i + \sum \beta_j$ в данном случае равен 3. Поэтому нижняя граница для $S(v_2, v_1)$ имеет значение $L_5 = L_3 + \Delta = 81 + 3 = 84$.

Так как L_5 меньше, чем нижние границы L_6 , L_4 и L_2 подмножеств $S(\overline{v_2, v_1})$, $S(v_4, v_6)$ и $S(\overline{v_6, v_3})$ соответственно, то очередному разбиению подлежит $S(v_2, v_1)$.

Используя редуцированную матрицу рис. 5.22б, находим, что максимальный суммарный штраф $\phi_{1,4} = A_1 + B_4 = 28 + 0 = 28$ соответствует дуге (v_1, v_4) . Разбиение множества $S(v_2, v_1)$ по этой дуге порождает еще два подмножества $S(v_1, v_4)$ и $S(\overline{v_1, v_4})$.

Соответствующие матрицы представлены на рис. 5.23а,в.

Первая матрица (а) получена из матрицы рис. 5.22б путем удаления строки v_1 и столбца v_4 . Кроме того, поскольку дуга (v_1, v_4) совместно с ранее выбранными дугами образует орцепь $(v_2, v_1, v_4, v_6, v_3)$ необходимо запретить использование дуги (v_3, v_2) , замыкающей эту цепь. Полагаем $c_{3,2}=\infty$.

Вторая матрица (в) получается, если в матрице рис. 5.22б положить $c_{1,4}=\infty$, чтобы запретить использование дуги (v_1, v_4) . Нижние границы для подмножеств $S(v_1, v_4)$ и $S(\overline{v_1, v_4})$ соответственно равны $L_7=104$ и $L_8=112$, причем для отыскания L_7 выполнена редукция матрицы а (см. рис. 5.23б).

$S(v_1, v_4)$			$S(\overline{v_1, v_4})$		
v_3	v_5	A	v_1	v_4	v_5
∞	0	∞	∞	∞	28
20	∞	∞	0	∞	0
β_7	0	0	20	∞	∞
B	∞	∞	∞	∞	∞
$L_7=L_5+\Delta=104$			$L_8=L_5+\phi_{1,4}=112$		
а			в		

Рис. 5.23

Содержимое матрицы на рис. 5.23б указывает на единственную возможность получения гамильтонова цикла на основе сформированной к данному моменту цепи $(v_2, v_1, v_4, v_6, v_3)$. Это использование "нулевых" дуг (v_3, v_5) и (v_5, v_2) . В результате имеем гамильтонов цикл $(v_2, v_1, v_4, v_6, v_3, v_5, v_2)$, длина которого равна L_7 , т.е. 104. На фоне значений весов дуг, содержащихся в исходной матрице (см. рис. 5.19а), полученный результат представляется очень хорошим, однако утверждать, что это кратчайший цикл, пока нельзя. Действительно, нижняя граница L_6 , для множества $S(\overline{v_2, v_1})$ ²⁶ имеет значение 101 (см. рис. 5.22в), а это меньше, чем длина найденного гамильтонова цикла. Поэтому вполне возможно, что среди циклов этого

²⁶Множество гамильтоновых циклов, содержащих дуги (v_4, v_6) и (v_6, v_3) , но не содержащих дугу (v_2, v_1) .

множества есть цикл(ы) меньшей или равной длины. Следовательно необходимо продолжить процесс разбиения, начиная с множества $S(\overline{v_2, v_1})$, как показано на рис. 5.24. Результаты

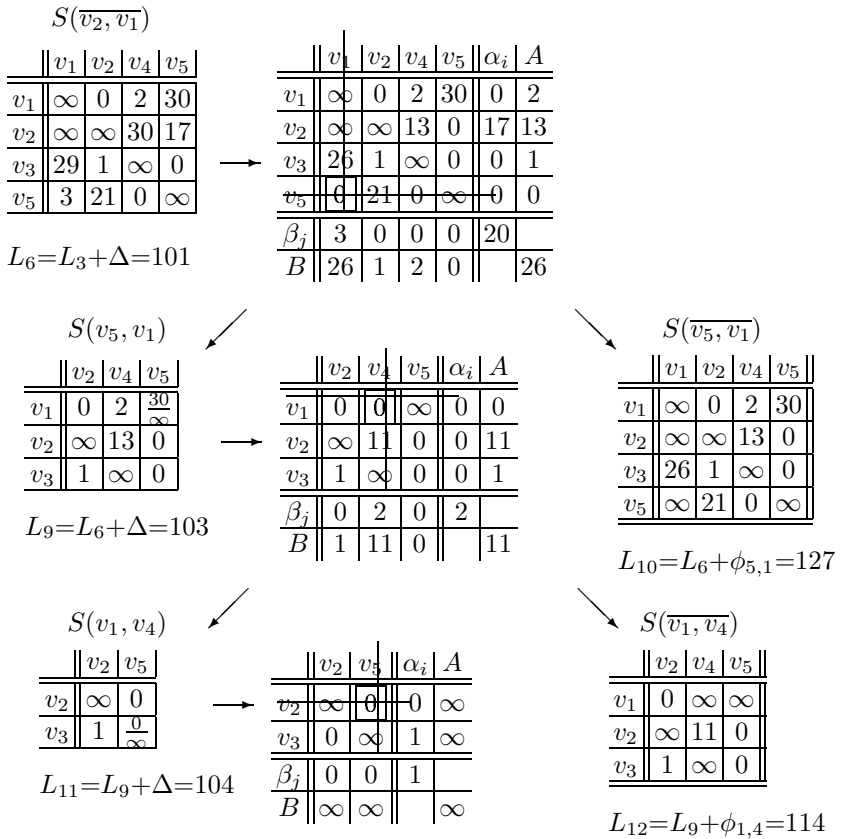


Рис. 5.24

проделанных операций свидетельствуют о наличии множества гамильтоновых циклов $S(v_1, v_4)$ ²⁷ с нижней границей $L_{11}=104$, включающих дуги (v_6, v_3) , (v_4, v_6) , (v_5, v_1) , (v_1, v_4) , которые в совокупности образуют орцепь $(v_5, v_1, v_4, v_6, v_3)$. Поскольку ре-

²⁷В данном случае это множество состоит лишь из одного цикла.

дуцированная матрица для $S(v_1, v_4)$ содержит две "нулевых" дуги (v_2, v_5) и (v_3, v_2) , замыкающих эту цепь в гамильтонов цикл $(v_2, v_5, v_1, v_4, v_6, v_3, v_2)$, длина последнего равна $L_{11}=104$.

Бинарное дерево, отражающее все выполнявшиеся разбиения множества гамильтоновых циклов рассматриваемого графа, изображено на рис. 5.25. Две концевые вершины дерева,

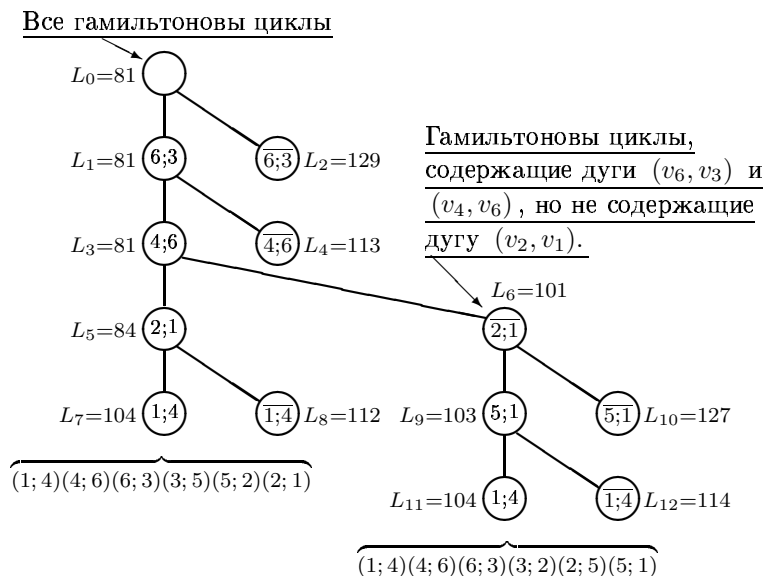


Рис. 5.25

имеющие наименьшее значение нижней границы, определяют оптимальное решение задачи. В данном графе существуют два кратчайших гамильтоновых цикла, длина которых равна 104.

Список литературы

- [1] Харари Ф. Теория графов. – М.: Мир, 1973.–300с.
- [2] Харари Ф., Палмер Э. Перечисление графов. – М.:Мир, 1977.–324 с.
- [3] Уилсон Р. Введение в теорию графов. – М.: Мир, 1977. – 207 с.
- [4] Кристофидес Н. Теория графов. Алгоритмический подход. – М.: Мир, 1978. – 432 с.
- [5] Оре О. Теория графов. – М.: Наука, Главная редакция физико-математической литературы, 1980. – 336 с.
- [6] Рейнгольд Э., Нивергельт Ю., Део Н. Комбинаторные алгоритмы. Теория и практика. – М.: Мир, 1980. – 476 с.
- [7] Свами М., Тхуласираман К. Графы, сети и алгоритмы. – М.: Мир, 1984. – 454 с.
- [8] Майника Э. Алгоритмы оптимизации на сетях и графах. – М.: Мир, 1985. – 323 с.
- [9] Евстигнеев В.А. Применение теории графов в программировании. – М.: Наука, 1985. – 352 с.
- [10] Горбатов В.А. Основы дискретной математики: Учеб. пособие для студентов вузов. – М.: Высш. шк., 1986. – 311 с.
- [11] Зыков А.А. Основы теории графов. – М.: Наука, Гл. ред. физ.-мат. лит., 1987. – 384 с.
- [12] Лекции по теории графов / В.А. Емеличев, О.И. Мельников, В.И. Сарванов, Р.И. Тышкевич – М.: Наука, 1990. – 384 с.
- [13] Романовский И.В. Дискретный анализ. Учеб. пособие. – СПб.: Невский диалект, 2000. – 240 с.
- [14] Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. – М.: Издательский дом "Вильямс", 2001. – 384 с.
- [15] Новиков Ф.А. Дискретная математика для программистов. – СПб. : Питер, 2001. – 301 с.
- [16] Иванов Б.Н. Дискретная математика. Алгоритмы и программы: Учеб. пособие. – М.: Лаборатория Базовых Знаний, 2002. – 288 с.

Указатель обозначений

G	граф
G^*	граф конденсации
$\tilde{G}$	граф транзитивного замыкания
O_n	пустой граф
C_n	цикл
P_n	цепь
K_n	полный граф
K_{n_1, n_2}	полный двудольный граф
T_n	дерево
T	турнир
V	множество вершин
E	множество ребер
A	множество дуг
v, u, w	вершины
$\left. \begin{matrix} e \\ \{v, u\} \end{matrix} \right\}$	ребро
$\left. \begin{matrix} a \\ (v, u) \end{matrix} \right\}$	дуга
n	число вершин
m	число ребер (дуг)
A	матрица смежности
B	матрица инцидентности
C	матрица весов ребер (дуг)
	матрица достижимости
R	прямой
Q	обратной
D	матрица расстояний
K	матрица Кирхгофа
S	матрица разрезов
Φ	матрица циклов
$\deg v$	степень вершины v
$\deg^+ v$	полустепень исхода v
$\deg^- v$	полустепень захода v
$l(v)$	метка вершины v
$e(v)$	эксцентриситет вершины v
$r(G)$	радиус графа G
$d(G)$	диаметр графа G
$\text{adj } v$	окружение вершины v
$d(v, u)$	расстояние между v и u
ν	цикломатическое число (ранг)
ρ	коцикломатическое число (ранг)
$\Gamma^+(v)$	концы дуг, выходящих из v

Предметный указатель

- абстрактный граф, 7
- алгоритм Демукрона, 26
- алгоритм Дейкстры, 77
- алгоритм Краскала, 52
- алгоритм Ли, 73, 74
- алгоритм Прима, 59
- алгоритм Уоршолла, 36
- алгоритм Флёри, 100
- алгоритм Флойда, 82
- ациклический орграф, 24

- база графа, 38

- вершинно-порожденный подграф, 9
- волновой алгоритм, 75, 76

- гамильтонов граф, 107
- гамильтонов контур, 107
- гамильтонов цикл, 107
- гамильтонова цепь, 108
- граф конденсации, 29

- диаметр графа, 19

- задача коммивояжера, 121
- задача почтальона, 104

- изоморфизм, 5
- инвариант, 6

- код Прюфера, 44
- коцикл, 96
- коцикломатическое число, 96
- критический путь, 91

- лес, 40

- маршрут, 27
- матрица (прямых) достижимостей, 33
- матрица инцидентности орграфа, 22

- матрица смежности орграфа, 21
- матрицу обратных достижимостей, 33
- матричная теорема о деревьях, 51
- метод ветвей и границ, 122
- множество фундаментальных разрезов, 95
- множество фундаментальных циклов, 93

- надграф, 8
- неориентированное дерево, 39
- неориентированный граф, 3
- непомеченный граф, 7

- ограниченная достижимость, 34
- односторонне связный граф, 28
- односторонняя компонента, 29
- ориентированная простая цепь, 27
- ориентированная цепь, 27
- ориентированное дерево, 41
- ориентированный граф, 20
- ориентированный маршрут, 27
- остов, 40
- остовное дерево, 40
- остовный лес, 40
- остовный подграф, 8

- подграф, 8
- поиск в ширину, 72
- полупуть, 28
- полустепень захода, 21
- полустепень исхода, 21
- полуэйлеров граф, 97
- порядковая функция, 25
- порядок графа, 4
- простой разрез, 95
- путь, 27

- радиус графа, 19
- разрез, 95

реберно-порожденный подграф, 9

сильная компонента, 29

сильно связный граф, 28

слабая компонента, 29

слабо связный граф, 28

список вершин, 18

список ребер, 18

степенная последовательность, 4

степень вершины, 4

суграф, 9

теорема Бине-Коши, 49

теорема Кели, 51

теорема Кирхгофа, 51

теорема Кэнига, 13

топологическая сортировка, 24

транзитивное замыкание графа,
35

транзитивный граф, 35

турнир, 23

формула Пойя, 8

центр графа, 19

центральная вершина, 19

цикломатическое число, 94

частичный граф, 9

эйлеров граф, 97

эйлеров цикл, 97

эйлерова цепь, 97

эксцентриситет, 19

Содержание

1	Основные понятия теории графов	3
1.1	Определение графа	3
1.2	Подграфы	8
1.3	Виды графов	10
1.4	Матрицы графов	15
1.5	Диаметр, радиус и центр графа	19
1.6	Ориентированные графы	20
1.7	Маршруты, цепи и простые цепи	27
2	Связность в орграфах	28
2.1	Основные понятия	28
2.2	Компоненты связности	29
2.3	Конденсация орграфа	29
2.4	Отыскание сильных компонент	30
2.5	Матрицы достижимостей	33
2.6	Получение матрицы достижимостей	34
2.7	Алгоритм Уоршолла	36
2.8	База графа	38
3	Деревья	39
3.1	Основные понятия	39
3.2	Описание деревьев	42
3.3	Задачи с деревьями	46
3.3.1	Перечисление остовных деревьев	46
3.3.2	Пересчет остовных деревьев	49
3.4	Задача о кратчайшем остове графа	52
3.4.1	Алгоритм Краскала	52
3.4.2	Алгоритм Прима	59
4	Пути и маршруты в графах	65

4.1	Существование путей	65
4.2	Пересчет маршрутов и путей	66
4.3	Перечисление маршрутов и путей	68
4.4	Задачи о кратчайших путях	70
4.4.1	Графы с дугами единичной длины	72
4.4.2	Графы со взвешенными дугами (ребрами)	76
4.4.3	Ациклические оргграфы	87
5	Циклы	93
5.1	Фундаментальные циклы и разрезы	93
5.2	Эйлеровы циклы	97
5.2.1	Определение и условия существования	97
5.2.2	Алгоритм поиска эйлерова цикла	100
5.2.3	О количестве эйлеровых графов	103
5.2.4	Задача почтальона	104
5.3	Гамильтоновы циклы	107
5.3.1	Определение и условия существования	107
5.3.2	Методы поиска гамильтоновых циклов	111
5.4	Задача коммивояжера	121
5.4.1	Применение и методы решения	122
5.4.2	Метод ветвей и границ	123