Задание Г-41
Кратчайший путь на графе

Пример. Дан взвешенный орграф (рис. 4.2). Найти кратчайший путь из вершины A в B.

Решение. Применим алгоритм Е. Дейкстры [1] [10]. Пошаговый алгоритм определения кратчайшего расстояния от вершины A до B состоит в следующем. С каждой вершиной связывается метка. Метка может быть постоянной или временной. Первоначально вершине A приписывается постоянная метка 0, а всем остальным метка ∞. На первом шаге вычисляются расстояния от вершины A с постоянной меткой до всех остальных вершин. Если некоторая вершина не соединена с вершиной с постоянной меткой или дуга направлена в обратную сторону, то расстояние принимается бесконечным. Найденные расстояния являются временными метками вершин. Минимальная из временных меток берется за постоянную. На следующем шаге временные метки всех вершин (кроме тех, у которых постоянные метки) вычисляются как сумма значения последней полученной постоянной метки и расстояния от нее в случае, если это значение не больше предыдущего значения временной метки данной вершины. Таким образом, временная метка вершины может или оставаться прежней, или уменьшаться. Минимальная из временных меток всех вершин опять принимается за постоянную. Процесс продолжается до тех пор, пока вершина B не получит постоянную метку. Значение этой метки и есть кратчайшее расстояние от A до B.

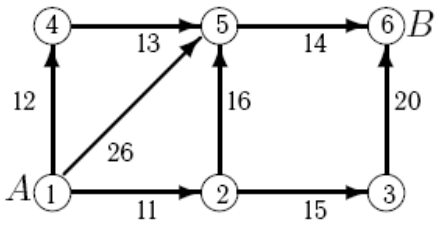

Рис. 4.2

Рассмотрим отдельные шаги решения.

1. Вершина A получает постоянную метку 0, остальные — метку ∞:

1	2	3	4	5	6
[0]	∞	∞	∞	∞	∞

2. Вычисляем расстояния от вершины 1 с постоянной меткой 0. Вершины 2, 4 и 5 меняют свои временные метки на 11, 12 и 26. Остальные имеют прежние метки — ∞. Очевидно, наименьшей меткой является 11. Она и становится постоянной:

1	2	3	4	5	6
[0]			∞	∞	∞
	[11]	∞	12	26	∞

3. Вычисляем расстояния от вершины 2 с постоянной меткой 11. Вершины 3 и 5 имеют расстояния 15 и 16 до вершины 2. Суммируя, получаем значения 26 и 27. Для вершины 5 прежнее значение, 26, было меньше нового значения 27. Следовательно, значение метки 5 не меняем; оно остается равным 26. Из трех временных меток — 12, 26 и 26 — наименьшая принадлежит вершине 4. Эта метка и становится постоянной:

1	2	3	4	5	6
[0]		∞		∞	∞
	[11]	∞	[12]	26	∞
		26			∞

.

4. Вычисляем расстояния от вершины 4 с постоянной меткой 12. Вершина 5 имеет до нее расстояние 13. Суммируя $(13+12)$, получаем значение 25 временной метки вершины 5 вместо прежнего значения 26. Из двух временных меток вершин 3 и 5 наименьшая принадлежит вершине 5. Эта метка и становится постоянной:

1	2	3	4	5	6
[0]		∞			∞
	[11]	∞	[12]		∞
		26			∞
		26		[25]	∞

.

5. На следующем этапе, вычисляя расстояния от вершины 5 с постоянной меткой 25, приходим к конечной вершине B. Но ее метка $(25+14=39)$ не становится постоянной, так как она не является минимальной. Расстояние от вершины 5 до вершины 3 принято ∞ (они не соединены). Прежнее значение временной метки вершины 3 меньше ∞. Поэтому метка вершины 3 не меняется. Метка

вершины 3 со значением 26, меньшим 39, становится постоянной. На следующем этапе ищем расстояния от нее:

1	2	3	4	5	6
[0]					∞
	[11]		[12]		∞
					∞
				[25]	∞
		[26]			39

Задание Г-42

Поток в сети

Сетью называют взвешенный орграф с двумя выделенными вершинами: истоком и стоком. Исток имеет нулевую полустепень захода, а сток — нулевую полустепень исхода. Вес дуги означает ее пропускную способность. Поток — еще одно число, приписанное дуге. Поток дуги не больше ее пропускной способности и может меняться. Поток выходит из истока и без потерь, в том же объеме заходит в сток. Условие равновесия (по объему входа и выхода) выполняется и для каждой вершины сети.

Задача о наибольшем потоке в сети — не единственная, но, вероятно, основная задача для потоков в сети. Очевидна возможность практического применения этой задачи для решения транспортных проблем (пробки на дорогах можно условно связывать с насыщением сети или отдельной ее дуги), проблем транспортировки нефтепродуктов или электроэнергии.

Пример. Задана пропускная способность дуг транспортной сети (рис. 4.4) с началом в вершине 1 и концом в вершине 8. Используя алгоритм *Форда–Фалкерсона*, найти максимальный поток по сети.

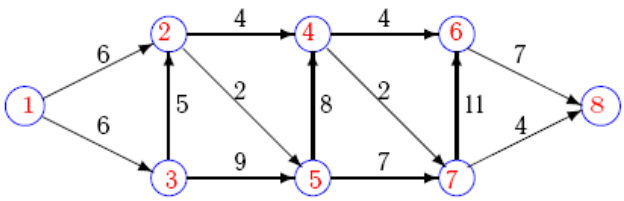

Рис. 4.4

Решение. Алгоритм состоит из двух частей — насыщения потока и его перераспределения. Поток называется насыщенным, если любая цепь из истока в сток содержит насыщенную дугу. В первой части алгоритма разыскиваются цепи из истока в сток и все дуги цепи насыщаются одинаковым возможно большим потоком, определяемым пропускной способностью наиболее «тонкой» дуги или наименьшей разностью между пропускной способностью и потоком в дуге. Различные цепи могут иметь общие дуги. Полученный поток согласован с условием сохранения в узлах (вершинах). Поток, входящий в вершину, равен потоку, выходящему из нее. Поток в сети проходит по *цепям* из истока в сток, т.е. недопустим многократный проход по отдельной дуге. Первая часть задачи считается решенной, если нет ненасыщенных цепей из истока в сток. Первая часть задачи не имеет единственного решения.

Во второй части перераспределение потока выполняется исходя из условия достижения общего по сети максимума потока. Для этого в основании графа (т.е. в графе, в котором снята ориентация дуг) разыскиваются маршруты из истока в сток, состоящие из ребер, соответствующих ненасыщенным дугам, направленным вперед, и непустым дугам, направленным назад. Потоки в дугах прямого направления увеличиваются на величину, на которую уменьшаются потоки в обратных дугах выбранного маршрута. При этом, очевидно, нельзя превышать пропускную способность дуг, направленных вперед, и допускать отрицательных потоков в обратных дугах. В некоторых случаях при удачном выборе цепей в первой части алгоритма перераспределения потока не требуется.

1. **Насыщение потока**. Рассмотрим путь 1–2–4–6–8. Пропустим через этот путь поток, равный 4. При этом дуги [2, 4] и [4, 6] будут насыщенными. Аналогично, путь 1–3–5–7–8 насытим потоком 4. Распределение потока отметим на графе (рис. 4.5). В числителе ставим пропускную способность, в знаменателе — поток. Числитель всегда больше знаменателя, а знаменатель может быть и нулем.

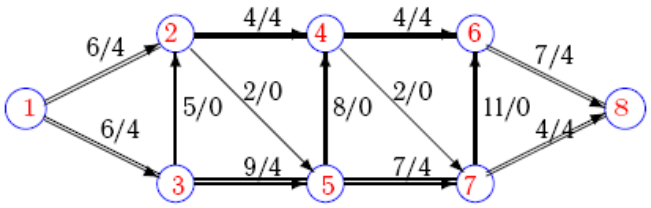

Рис. 4.5

Заметим, что из 1 в 8 есть еще ненасыщенный путь, 1–3–2–5–4–7–6–8,поток в котором можно увеличить на 2. При этом насытятся дуги [1, 3], [2, 5] и [4, 7] (рис. 4.6).

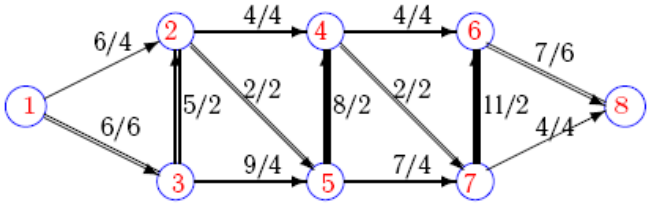

Рис. 4.6

Из 1 в 8 больше нет ненасыщенных путей. По дуге [1,3] двигаться нельзя (она уже насыщена), а движение по дуге [1,2] заканчивается в вершине 2, так как обе выходящие из нее дуги насыщены.

2. **Перераспределение потока.** Найдем последовательность вершин от 1 к 8, такую, что дуги, соединяющие соседние вершины, направленные из 1 в 8, не насыщены, а дуги, направленные в обратном направлении, не пусты. Имеем единственную последовательность: 1–2–3–5–7–6–8. Перераспределяем поток. Поток в дугах прямого направления увеличиваем на 1, а поток в дугах обратного направления уменьшаем на 1. Процесс продолжаем до тех пор, пока одна из прямых дуг не будет насыщена или какая-нибудь обратная дуга не будет пуста.

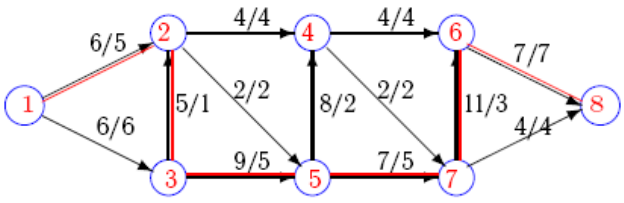

Рис. 4.7

Задание Г-43

Топологическая сортировка в сети

Пусть в сети в общем случае имеется несколько истоков и стоков. Стоки и истоки занимают в сети крайние положения. Все остальные вершины могут быть дальше или ближе к ним. Расположение вершины в сети определяется ее *уровнем*. Определим это понятие. Вершины уровня 0 — это истоки; они образуют множество N_0. Если N_i — множество вершин уровня $i \leqslant k$, то N_{k+1} — множество вершин уровня $k+1$ состоящее из тех и только тех вершин, предшественники которых принадлежат любому из множеств $N_0,\ldots,N_k$, причем среди этих множеств нет пустых [3].

Порядковой функцией сети называют отображение, сопоставляющее каждой вершине сети ее уровень.

Пример. Отсортировать топологически сеть на рис. 4.9.

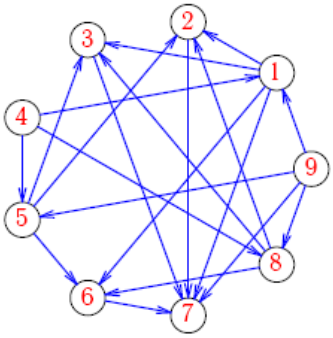

Рис. 4.9

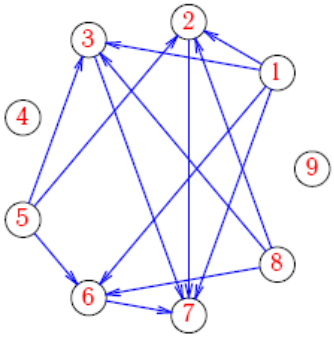

Рис. 4.10

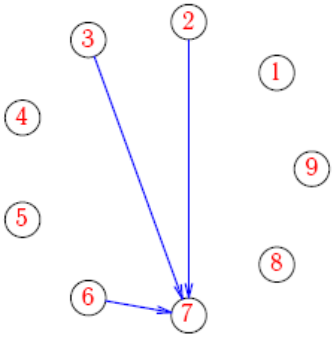

Рис. 4.11

Решение. Находим вершины (4 и 9), полустепень захода которых равна 0. Удаляем эти вершины и дуги, выходящие из них [3]. Удаленные вершины образуют первый уровень упорядоченной сети. Получаем новую сеть (рис. 4.10). В новой сети в вершины 1, 5 и 8 не входят дуги. Удаляем эти вершины (они образуют второй уровень) и получаем сеть, изображенную на рис. 4.11. Очевидно, третий уровень состоит из вершин 2, 3 и 6, а последний (сток сети) — из единственной вершины 7. Изображаем ту же сеть по уровням (рис. 4.12).

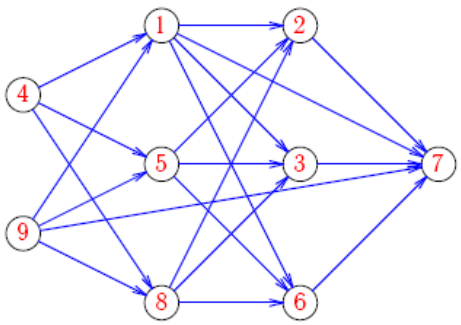

Рис. 4.12

Задание Г-44
Паросочетание в двудольном графе

Граф называется *двудольным*, если существует такое разбиение множества его вершин на две части, что концы каждого ребра принадлежат разным частям (долям).

Паросочетанием графа называется граф, ребра которого являются подмножеством ребер графа, а вершины имеют степень 1.

Паросочетание, не являющееся подмножеством другого паросочетания, называется *максимальным*.

Паросочетание, содержащее наибольшее число ребер, называется *наибольшим*. Наибольшее паросочетание является и максимальным.

Паросочетание, порядок которого равен порядку графа, называется *совершенным*. Совершенное паросочетание включает в себя все вершины двудольного графа. Очевидно, совершенное паросочетание является наибольшим и максимальным. Максимальное паросочетание может быть и не наибольшим. В одном графе могут иметься различные максимальные паросочетания одного порядка.

Число совершенных паросочетаний в двудольном графе, имеющем одинаковое число вершин в долях, можно определить, вычислив перманент [1] его матрицы смежности.

Матрица смежности двудольного графа на рис. 4.13 имеет вид

$$\begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}.$$

Перманент этой матрицы равен 4. Все четыре покрытия графа изображены на рис. 4.14–4.17.

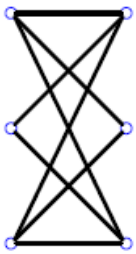

Рис. 4.13

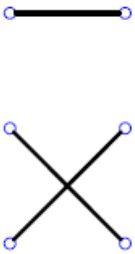

Рис. 4.14

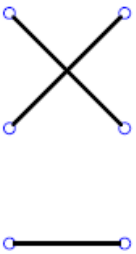

Рис. 4.15

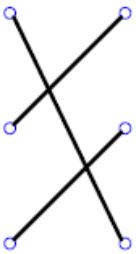

Рис. 4.16

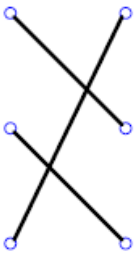

Рис. 4.17

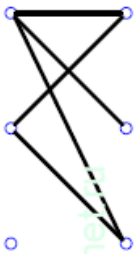

Рис. 4.18

[1]Перманент матрицы (per) равен сумме всех произведений элементов матрицы по одному из каждой строки в разных столбцах. Перманент [26] квадратной матрицы $a_{ij}(i,j=1,...,n)$ определяется, подобно определителю, рекуррентным образом. При $n=1$ имеем $\mathrm{per}(A_1)=a_{11}$. Для матрицы A_{n+1} размером $n+1$ получим разложение по первой строке: $\mathrm{per}(A_{n+1})=\sum_{k=1}^{n} a_{1k}\mathrm{per}(A_k)$, где $\mathrm{per}(A_k)$ — перманент матрицы размером n, полученной из A_{n+1} вычеркиванием первой строки и k-го столбца. Таким образом, перманент отличается от определителя только тем, что все его слагаемые берутся со знаком плюс. Например, $\mathrm{per}\begin{vmatrix}1 & 2\\ 3 & 4\end{vmatrix}=1\cdot 4+3\cdot 2$.

Если граф не имеет совершенного паросочетания, то перманент его матрицы равен нулю. Например, матрица графа с одной изолированной вершиной на рис. 4.18 имеет вид

$$\begin{vmatrix}1 & 1 & 1\\ 1 & 0 & 1\\ 0 & 0 & 0\end{vmatrix}.$$

Перманент этой матрицы равен 0 [2].

Пример. В заданном двудольном графе (рис. 4.21) найти наибольшее паросочетание.

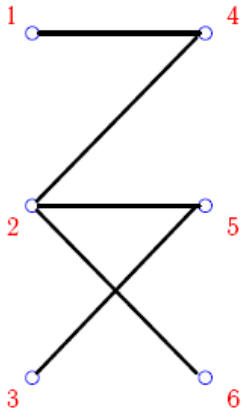

Рис. 4.21

Применим алгоритм Форда-Фалкерсона.

Образуем из двудольного графа сеть, добавляя исток 7 и сток 8 и ориентируя *ребра* из истока в сток. Пропускную способность всех полученных *дуг* положим равной 1 (рис. 4.22). Максимальный поток по этой сети соответствует искомому паросочетанию графа. Заметим, что решение будет не единственным.

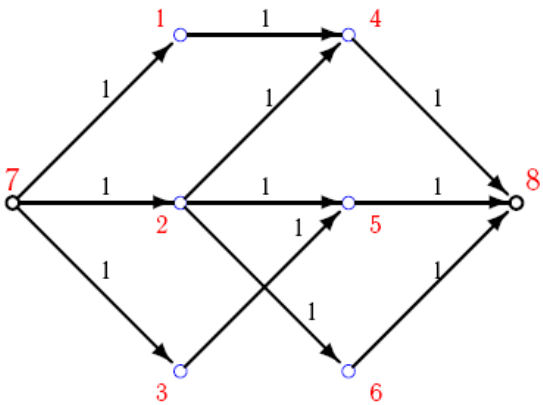

Рис. 4.22

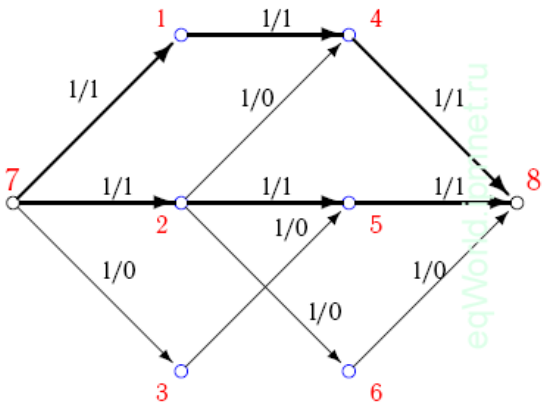

Рис. 4.23

Находим все возможные маршруты из истока в сток и насыщаем их. Таких маршрутов два: 7–1–4–8 и 7–2–5–8 (рис. 4.23). Маршрутов из 7 в 8, по которым можно было бы пропустить дополнительный поток, больше нет [1]. Перераспределяем полученный поток. Находим пути из истока в сток, содержащие ненасыщенные прямые дуги и непустые обратные. В данной задаче это путь 7–3–5–2–6–8. Уменьшаем поток в обратных дугах и увеличиваем на эту же величину поток в прямых дугах (рис. 4.24). В покрытие исходного двудольного графа войдут ребра с потоком, равным 1 (рис. 4.25).

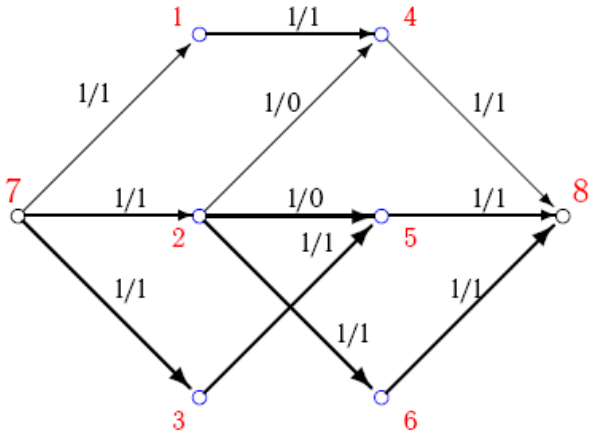

Рис. 4.24

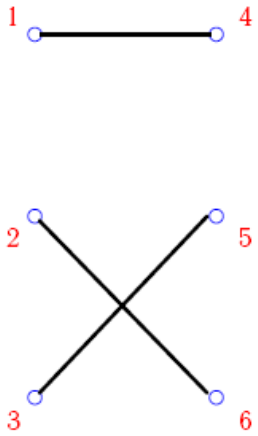

Рис. 4.25

Матрица смежности исходного двудольного графа имеет вид

$$\begin{vmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{vmatrix}.$$

Перманент этой матрицы равен 1. Следовательно, нами найдено единственное совершенное покрытие.

Задание Г-45
Задача о назначениях (венгерский алгоритм)

Известно, что совершенное паросочетание в двудольном графе может быть не единственным. Поэтому, если граф взвешенный, т.е. каждое ребро наделено весом, то естественно поставить задачу о поиске паросочетания с наибольшим или наименьшим весом. Очевидна возможность практического применения решения этой задачи. Рассмотрим, например, задачу о назначении группы n сотрудников по n рабочим местам. Каждый из сотрудников может работать на любом из этих мест, но оплата труда везде разная. Обозначим через a_{ij} оплату труда сотрудника i на рабочем месте j. Матрица коэффициентов a_{ij} в общем случае несимметрична. Оптимальное распределение сотрудников соответствует минимальным расходам на производство. Представляя сотрудников вершинами одной доли графа, а рабочие места — вершинами другой доли, получим полный двудольный взвешенный граф. Совершенное паросочетание наименьшего веса является решением задачи.

Решение. Применим для решения венгерский алгоритм [2]. Алгоритм основан на теории чередующихся цепей Дж. Петерсена. Преобразуем матрицу A. Вычтем из каждой строки минимальный элемент. Получим

$$A' = \begin{vmatrix} 0 & 6 & 0 & 2 \\ 0 & 5 & 3 & 5 \\ 16 & 0 & 4 & 0 \\ 0 & 5 & 9 & 3 \end{vmatrix}.$$

Изобразим двудольный граф, в котором ребра соответствуют нулевым элементам матрицы A' (рис. 4.27).

Пример. Оплата труда работника i на рабочем месте j определяется коэффициентом a_{ij}:

$$A = \begin{vmatrix} 1 & 7 & 1 & 3 \\ 1 & 6 & 4 & 6 \\ 17 & 1 & 5 & 1 \\ 1 & 6 & 10 & 4 \end{vmatrix}.$$

Найти одно из оптимальных назначений и суммарные затраты на производство.

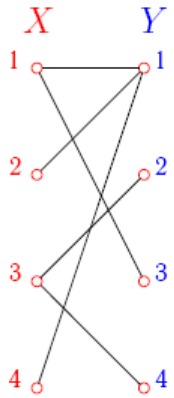

Рис. 4.27

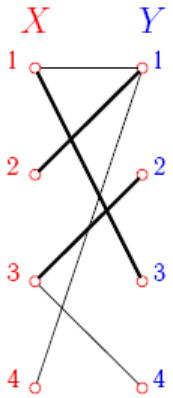

Рис. 4.28

Рассмотрим в этом графе какое-либо максимальное паросочетание. Выделим соответствующие ребра (рис. 4.28).

Чередующейся [1] цепи из X в Y нет. Следовательно, это паросочетание является наибольшим. Выделим множества вершин, не входящие в паросочетание. Это $X_m = \{x_4\}$, $Y_m = \{y_4\}$. Составим множества вершин, которые входят в цепи [2], соединяющие X_m и X:

$$X' = \{x_2, x_4\},\ Y' = \{y_1\}.$$

Преобразуем матрицу A'. Найдем минимальный элемент в строках с номерами элементов множества X' и столбцах с номерами элементов множества $Y \backslash Y'$, т.е в матрице

$$\begin{vmatrix} . & . & . & . \\ . & 5 & 3 & 5 \\ . & . & . & . \\ . & 5 & 9 & 3 \end{vmatrix}.$$

Минимальный элемент равен 3. Вычтем 3 из строк 2 и 4 и добавим 3 к столбцу 1. Получим

$$A'' = \begin{vmatrix} 3 & 6 & 0 & 2 \\ 0 & 2 & 0 & 2 \\ 19 & 0 & 4 & 0 \\ 0 & 2 & 6 & 0 \end{vmatrix}.$$

[1] Чередующаяся цепь состоит из нечетного числа чередующихся тонких (из X в Y) и толстых (в обратном направлении) ребер, начиная и кончая тонким ребром. Если бы такая цепь существовала, то число ребер в паросочетании можно было бы увеличить, поменяв толстые и тонкие ребра в этой цепи.

[2] От X к Y можно двигаться по тонким ребрам, в обратном направлении — по толстым.

Соответствующий граф (см. рис. 4.28) изменился. Исчезло ребро (1, 1), и добавились ребра (2, 3) и (4, 4) (рис. 4.29). В этом графе есть ребро (4, 4), не входящее в выделенное паросочетание и не инцидентное его вершинам, но соединяющее X и Y. Паросочетание стало совершенным (рис. 4.30), множества X_m и Y_m пусты, следовательно, задача решена. Таким образом, выбирая элементы исходной матрицы A с номерами ребер полученного паросочетания, найдем наименьшие затраты: $\Sigma = a_{13} + a_{21} + a_{32} + a_{44} = 1 + 1 + 1 + 4 = 7$.

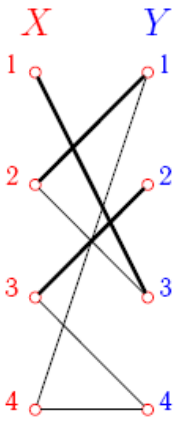

Рис. 4.29

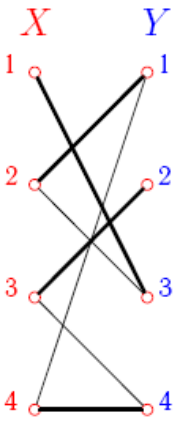

Рис. 4.30

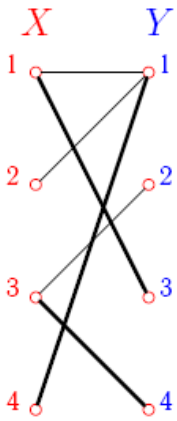

Рис. 4.31

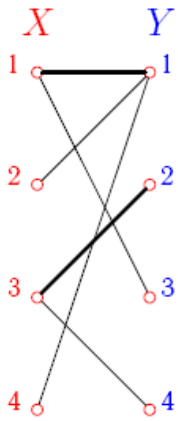

Рис. 4.32

З а м е ч а н и е 1. Паросочетание на рис. 4.28 не единственное. Имеется еще одно наибольшее (3 ребра) паросочетание (рис. 4.31). Множества вершин, не входящие в паросочетание, — это $X_m = \{x_2\}$ и $Y_m = \{y_2\}$. При этом множества X' и Y' не изменятся:

$$X' = \{x_2, x_4\}, \quad Y' = \{y_1\}.$$

З а м е ч а н и е 2. Паросочетание на рис. 4.28 сразу оказалось наибольшим. Однако можно было бы выбрать и не наибольшее, но максимальное паросочетание (рис. 4.32). В этом случае, обратив чередующуюся цепь x_4, y_1, x_1, y_3 (толстые ребра меняются на тонкие, и наоборот), получим наибольшее покрытие (рис. 4.31).

Задание Г-46
Остов наименьшего веса

Остовом графа G называют граф, не содержащий циклов и состоящий из ребер графа G и всех его вершин. Таким образом, остов графа является деревом. Число ребер остова равно рангу графа ($\nu^* = n - k$). Число остовов графа можно вычислить по матрице Кирхгофа.

Пример. Дан взвешенный граф (рис. 4.34).

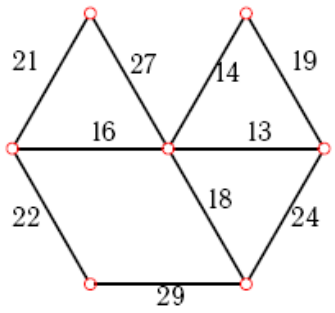

Рис. 4.34

Найти число остовов графа и остов графа минимального веса (экстремальное дерево).

Решение. *Число остовов графа.* Число остовов графа можно вычислить по его матричным характеристикам. Известна следующая теорема.

Теорема 18 (Кирхгофа [1]). *Число остовов графа равно алгебраическому дополнению любого элемента матрицы Кирхгофа.*

Элементы матрицы Кирхгофа B графа G определяются следующим образом. Если вершины i и j графа G смежны, то $b_{ij} = -1$, а если вершины i и j не смежны, то $b_{ij} = 0$. Диагональные элементы матрицы Кирхгофа B равны степеням вершин: $b_{ii} = \deg(i)$. Очевидно свойство: сумма элементов в каждой строке и каждом столбце матрицы Кирхгофа равна 0. Известно также, что алгебраические дополнения всех элементов матрицы Кирхгофа равны, а ранг матрицы Кирхгофа неографа порядка n можно вычислить по формуле $\operatorname{rank} B = n - k$, где k — число компонент графа.

Пронумеруем вершины графа (рис. 4.35).

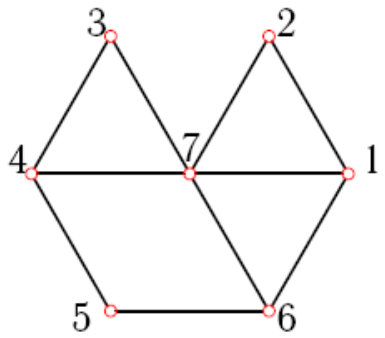

Рис. 4.35

В соответствии с определением запишем матрицу Кирхгофа:

$$B = \begin{bmatrix} 3 & -1 & 0 & 0 & 0 & -1 & -1 \\ -1 & 2 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 2 & -1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 3 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ -1 & 0 & 0 & 0 & -1 & 3 & -1 \\ -1 & -1 & -1 & -1 & 0 & -1 & 5 \end{bmatrix}.$$

Рассмотрим минор элемента $b_{1,1}$:

$$m_{1,1} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & -1 \\ 0 & 2 & -1 & 0 & 0 & -1 \\ 0 & -1 & 3 & -1 & 0 & -1 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 3 & -1 \\ -1 & -1 & -1 & 0 & -1 & 5 \end{bmatrix}.$$

Вычислим определитель: $\det m_{1,1} = 79$. Таким образом, граф имеет 79 остовов, среди которых надо выбрать экстремальное дерево, т.е дерево, обладающее некоторыми эстремальными свойствами, в данным случае — минимальным весом.

Заметим, что матрицу Кирхгофа можно найти, используя формулу

$$B = II^T - 2A, \tag{4.1}$$

где I и A — матрицы инцидентности и смежности графа. Матрицу инцидентности запишем для следующей последовательности номеров ребер: $(3,7)$, $(6,7)$, $(1,2)$, $(6,1)$, $(5,6)$, $(3,4)$, $(2,7)$, $(4,7)$, $(1,7)$, $(4,5)$. Строки матрицы инцидентности соответствуют вершинам, столбцы — ребрам. В неографе матрица инцидентности I состоит из нулей и единиц, а матрица смежности A симметричная:

$$I = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Формула (4.1) будет проще, если рассмотреть некоторую *ориентацию графа*. Под ориентацией графа понимают орграф, получающийся заменой каждого ребра дугой произвольного направления. Рассмотрим, например, вариант, приведенный на рис. 4.36.

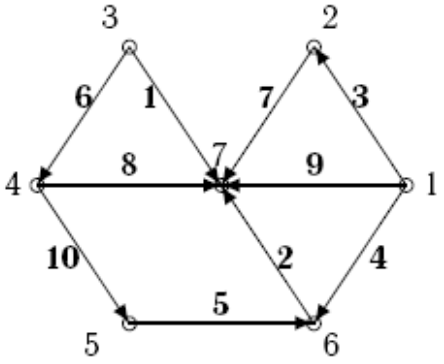

Рис. 4.36

Номера дуг на рисунке указаны полужирным шрифтом в соответствии с номерами ребер исходного графа. Матрица инцидентности I_1 ориентации графа имеет вид

$$I_1 = \begin{bmatrix} 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

Сумма элементов в любом столбце этой матрицы всегда равна 0. Матрица Кирхгофа может быть вычислена по формуле

$$B = I_1 I_1^T. \tag{4.2}$$

Задание Г-46_1
Определение остова наименьшего веса с помощью алгоритма ближайшего соседа

Алгоритм ближайшего соседа построения остова минимального веса. Алгоритм Дж. Краскала требует на каждом шаге проверки на цикличность и предварительной сортировки ребер по весам, что затруднительно для графов с большим числом ребер. Несколько проще следующий алгоритм [13].

1. Отмечаем произвольную вершину графа, с которой начнется построение. Строим ребро наименьшего веса, инцидентное этой вершине.

2. Ищем ребро минимального веса, инцидентное одной из двух полученных вершин. В множество поиска не входит построенное ребро.

3. Продолжаем далее, разыскивая каждый раз ребро наименьшего веса, инцидентное построенным вершинам, не включая в круг поиска все ребра, их соединяющие.

В нашем примере начнем с вершины A. На рисунках 4.43–4.48 дана последовательность действий.

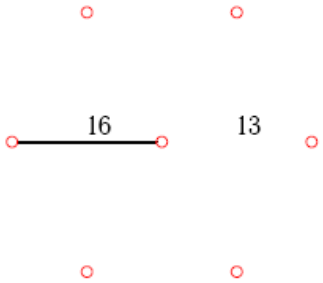

Рис. 4.43

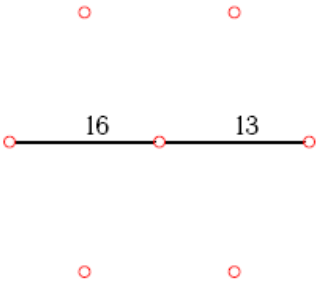

Рис. 4.44

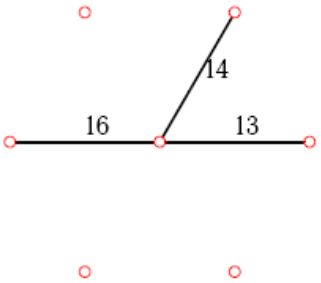

Рис. 4.45

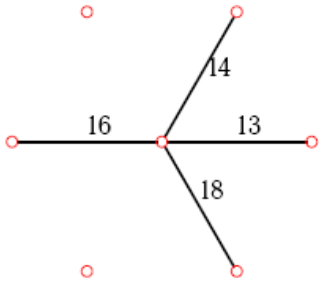

Рис. 4.46

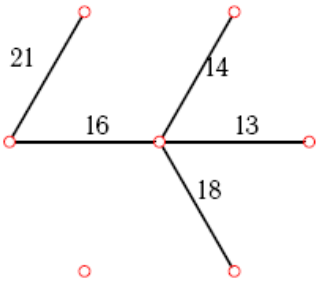

Рис. 4.47

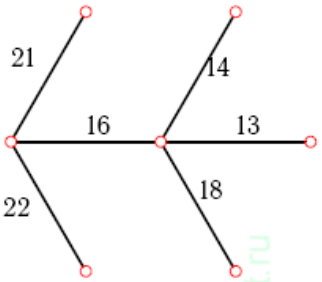

Рис. 4.48

Задание Г-46_1
Определение остова наименьшего веса с помощью алгоритма ближайшего соседа

Алгоритм ближайшего соседа построения остова минимального веса. Алгоритм Дж. Краскала требует на каждом шаге проверки на цикличность и предварительной сортировки ребер по весам, что затруднительно для графов с большим числом ребер. Несколько проще следующий алгоритм [13].

1. Отмечаем произвольную вершину графа, с которой начнется построение. Строим ребро наименьшего веса, инцидентное этой вершине.

2. Ищем ребро минимального веса, инцидентное одной из двух полученных вершин. В множество поиска не входит построенное ребро.

3. Продолжаем далее, разыскивая каждый раз ребро наименьшего веса, инцидентное построенным вершинам, не включая в круг поиска все ребра, их соединяющие.

В нашем примере начнем с вершины A. На рисунках 4.43–4.48 дана последовательность действий.

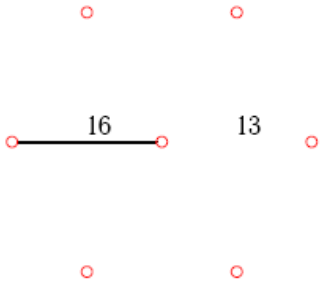

Рис. 4.43

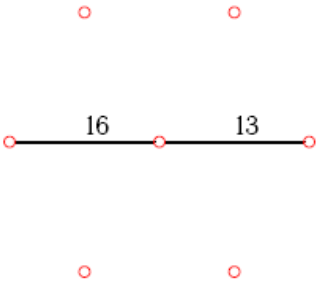

Рис. 4.44

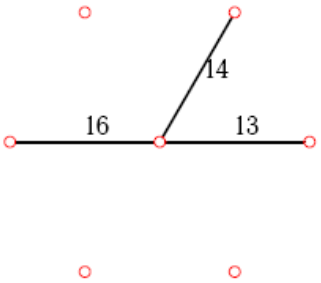

Рис. 4.45

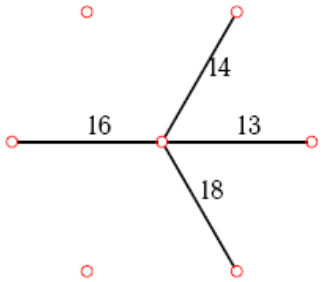

Рис. 4.46

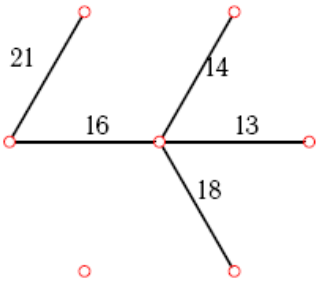

Рис. 4.47

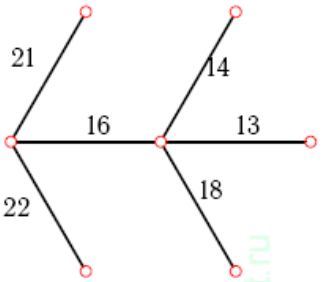

Рис. 4.48

Задание Г-46_1
Определение остова наименьшего веса с помощью алгоритма ближайшего соседа

Алгоритм ближайшего соседа построения остова минимального веса. Алгоритм Дж. Краскала требует на каждом шаге проверки на цикличность и предварительной сортировки ребер по весам, что затруднительно для графов с большим числом ребер. Несколько проще следующий алгоритм [13].

1. Отмечаем произвольную вершину графа, с которой начнется построение. Строим ребро наименьшего веса, инцидентное этой вершине.

2. Ищем ребро минимального веса, инцидентное одной из двух полученных вершин. В множество поиска не входит построенное ребро.

3. Продолжаем далее, разыскивая каждый раз ребро наименьшего веса, инцидентное построенным вершинам, не включая в круг поиска все ребра, их соединяющие.

В нашем примере начнем с вершины A. На рисунках 4.43–4.48 дана последовательность действий.

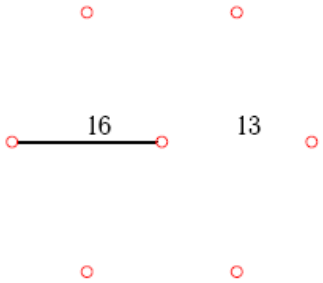

Рис. 4.43

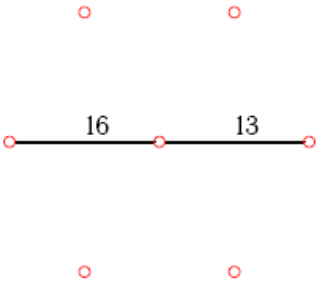

Рис. 4.44

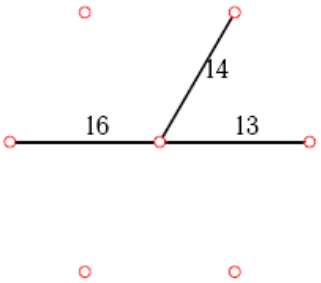

Рис. 4.45

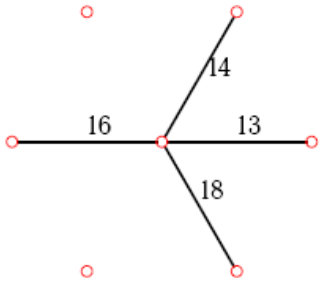

Рис. 4.46

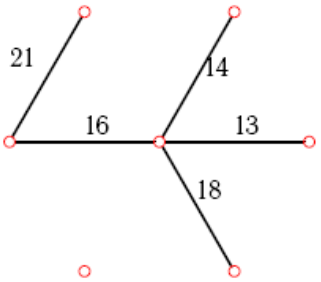

Рис. 4.47

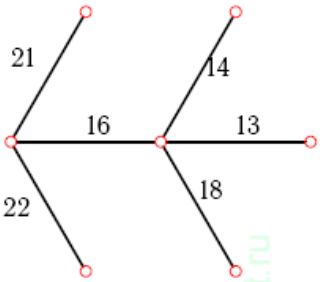

Рис. 4.48

Задание Г-47_1
Гамильтоновы циклы

Простой цикл, проходящий через все вершины графа, называется *гамильтоновым* [1]. Простая цепь, проходящая через все вершины графа, называется *гамильтоновой*.

Задача нахождения гамильтоновых циклов получила свое развитие в связи с рядом практических задач. Одной из них является так называемая *задача коммивояжера*, в которой определяется кратчайший гамильтонов цикл.

В отличие от поиска эйлеровых циклов, проходящих через каждое ребро графа по одному разу, для которых еще Эйлером получено необходимое и достаточное условие существования цикла, для гамильтоновых циклов такого условия не найдено. Существуют, однако, достаточные условия существования гамильтоновых циклов.

Пример. Дан граф (рис. 4.50). Найти все гамильтоновы циклы графа.

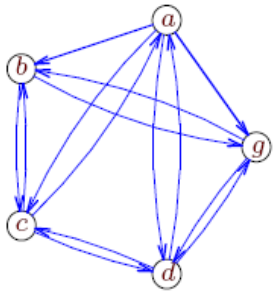

Рис. 4.50

Для заданного графа получим матрицу смежности A

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Введем модифицированную матрицу смежности B по правилу: элемент матрицы $\beta_{ij} = v_j$, если существует путь из вершины v_i в вершину v_j.

$$B = \begin{bmatrix} 0 & b & c & d & g \\ 0 & 0 & c & 0 & g \\ a & b & 0 & d & 0 \\ a & 0 & c & 0 & g \\ 0 & b & 0 & d & 0 \end{bmatrix}.$$

Далее последовательно находим вспомогательные матрицы P'_k: и P_k

$$P'_{k+1} = B \cdot P_k,\ P_1 = A,\ P_k = \Phi(P'_k).$$

Матрицы B и P_k умножаются по обычному правилу умножения матриц, а под произведением вершин понимается некоммутативная бинарная операция склеивания, заданная на множестве слов. Слово — упорядоченная последовательность вершин (букв). Произведение слова $u_1u_2, ..., u_k$ и слова $v_1v_2, ..., v_m$ есть слово $u_1u_2, ..., u_kv_1v_2, ..., v_m$.

Матрица P'_k преобразуется в матрицу P_k специальным оператором Φ. Оператор Φ, действующий на элементы p_{ij} матрицы, вычеркивает (обнуляет) те ее элементы, в которых содержится вершина v_i или v_j. Такие элементы содержат контуры, замыкающиеся на v_i или v_j. Так как на главной диагонали P_k содержится информация о циклах, для упрощения и ускорения расчетов можно обнулять главную диагональ всех матриц P_k, кроме, разумеется, последней — P_n, поскольку ее диагональ и содержит искомые циклы без начальной и конечной вершины. Эти вершины легко добавить после окончания основной итеративной процедуры.

Для заданного графа получим последовательность матриц P_k (вспомогательные матрицы P'_k не выписаны). Имеем

$$P_2 = \begin{bmatrix} 0 & c+g & b+d & c+g & b+d \\ c & 0 & 0 & c+g & 0 \\ d & a & 0 & a & a+b+d \\ c & a+c+g & a & 0 & a \\ d & 0 & b+d & 0 & 0 \end{bmatrix}.$$

Заметим, что матрица P_2 получена из P'_2 действием оператора Φ: в первом столбце и первой строке нет слов с вершиной a, во втором столбце и второй строке отсутствует вершина b и т.д. Далее

$$P_3 = \begin{bmatrix} 0 & dc+dg & gb+gd & bc+bg & cb+cd \\ cd+gd & 0 & gd & ca & ca+cd \\ 0 & ag+da+dg & 0 & ag+bg & ab+ad+da \\ 0 & ac+ag+ca & ab+gb & 0 & ab+ca+cb \\ bc+dc & da+dc & da & bc & 0 \end{bmatrix},$$

$$P_4 = \begin{bmatrix} 0 & cdg+gdc & bgd+dgb & cbg+gbc & bcd+dcb \\ gdc & 0 & gda & cag & cad+cda \\ bgd & adg+dag & 0 & abg & dab \\ gbc & cag & agb & 0 & acb+cab \\ bcd & dac+dca & dab & bca & 0 \end{bmatrix}.$$

Последняя матрица (P_5) — диагональная. Все ее элементы — нулевые, кроме $P_{5,11} = bgdc + cbgd + dgbc + gbcd$, $P_{5,22} = cadg + cdag + gdac + gdca$, $P_{5,33} = abgd + adgb + bgda + dagb$, $P_{5,44} = acbg + agbc + cabg + gbca$, $P_{5,55} = bcad + bcda + dacb + dcab$. К полученным слагаемым в элементе матрицы P_{kk} добавим в начале или в конце по вершине v_k, где $v_1 = a$, $v_2 = b$, $v_3 = c$, $v_4 = d$, $v_5 = g$, и выполним круговую перестановку так, чтобы вершина a оказалась первой. Многие гамильтоновы циклы повторяются. Множество ответов для данной задачи будет следующим:

$$abgdc,\ adgbc,\ acbgd,\ agbcd.$$

Полный граф порядка n имеет $(n-1)!$ гамильтоновых циклов.

Задание Г-48-1
Задача о коммивояжере
(алгоритм ближайшего соседа)

Это образное название устойчиво закрепилось за одной из самых интересных, практически значимых и одновременно сложных задач теории графов. Задача, берущая свое начало из работ Гамильтона, состоит в определении кратчайшего гамильтонова цикла в графе. Ее решение связано [18] с решением задачи о назначениях и с задачей об остове наименьшего веса.

Пример. Дан взвешенный орграф (рис. 4.52). Найти кратчайший гамильтонов цикл.

Решение. Способ 1. Рассмотрим алгоритм ближайшего соседа (Nva — Nearest vertex add) [1]. Начнем движение по графу из произвольной вершины, например из вершины a. Выбирая из двух возможных дуг, $a \to f$ и $a \to b$, короткую дугу, $a \to b$, длиной 1, далее до вершины c двигаемся без разветвлений: $a \to f \to d \to c$. Из c направляемся по более короткой дороге в g, затем в b и возвращаемся в a. Суммарная длина цикла равна $1 + 3 + 8 + 1 + 3 + 3 = 19$. Алгоритм ближайшего соседа не гарантирует оптимального решения. Решение зависит от выбора начальной вершины и от выбора направления движения при наличии двух и более направлений одинакового веса. Проверим другую вершину в качестве начальной.

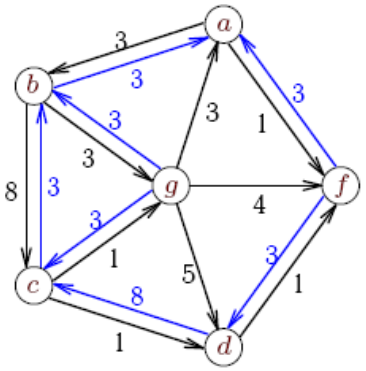

Рис. 4.52

Если начинать движение из b в сторону a по дуге весом 3, то цикл $b \to a \to f \to d \to c \to g \to b$ будет совпадать с найденным, отличаясь только началом отсчета. При выборе направления $b \to g$, также весом 3, получим цикл $b \to g \to c \to d \to f \to a \to b$, имеющий вес $3 + 3 + 1 + 1 + 3 + 3 = 14$. Этот цикл, наименьший по весу, можно принять за окончательный ответ. В действительности, суммарный вес 14 является наименьшим для этой задачи.

Известна оценка алгоритма ближайшего соседа [2] для графа, веса которого удовлетворяют неравенству треугольника (см. с. 81):

$$c_{\text{Nva}} \leqslant 0,5([\ln n] + 1)c_{\text{opt}},$$

где c_{opt} — вес оптимального маршрута. В нашем случае коэффициент в этой оценке $0,5([\ln n] + 1) = 1$.

Задание Г-48-2
Задача о коммивояжере
(алгоритм ближайшей вставки)

Это образное название устойчиво закрепилось за одной из самых интересных, практически значимых и одновременно сложных задач теории графов. Задача, берущая свое начало из работ Гамильтона, состоит в определении кратчайшего гамильтонова цикла в графе. Ее решение связано [18] с решением задачи о назначениях и с задачей об остове наименьшего веса.

Пример. Дан взвешенный орграф (рис. 4.52). Найти кратчайший гамильтонов цикл.

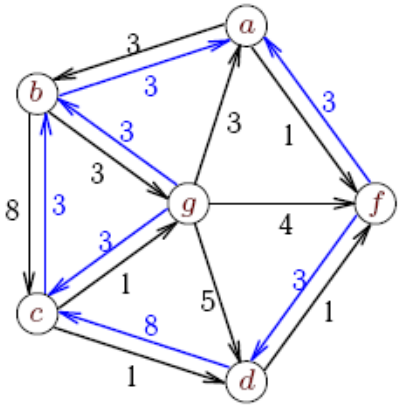

Рис. 4.52

Способ 2. Рассмотрим алгоритм ближайшей вставки (Nvi — Nearest insert) [2]. Построение цикла начнем с любой вершины, например c. Присоединим к c вершину, образующую вместе с ней цикл. Из трех возможных вариантов, $c \to d \to c$, $c \to g \to c$ и $c \to b \to c$, возьмем самый короткий, $c \to g \to c$, длиной 4 (рис. 4.53). На следующем шаге можно добавить либо вершину d, вставляя дуги $g \to d$ и $d \to c$ вместо дуги $g \to c$, либо b уже с двумя вариантами циклов: $g \to c \to b \to g$ и $g \to b \to c \to g$. Выбираем цикл наименьшей длины: $g \to c \to b \to g$ (рис. 4.54). Аналогично последовательно вставляем в цикл вершины d, f, a (рисунки 4.55–4.57). В результате получаем цикл длиной $3 + 3 + 1 + 3 + 8 + 3 = 21$.

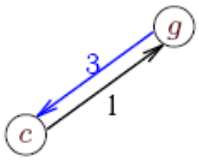

Рис. 4.53

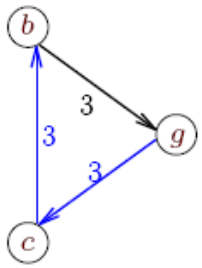

Рис. 4.54

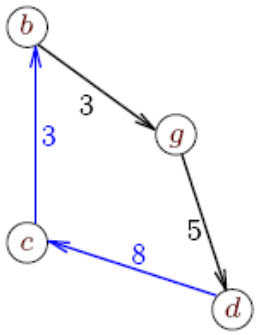

Рис. 4.55

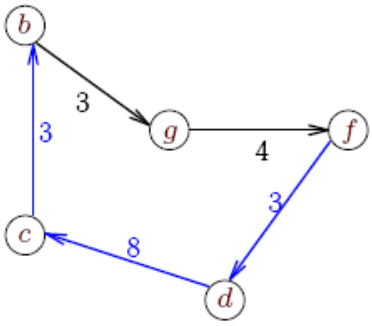

Рис. 4.56

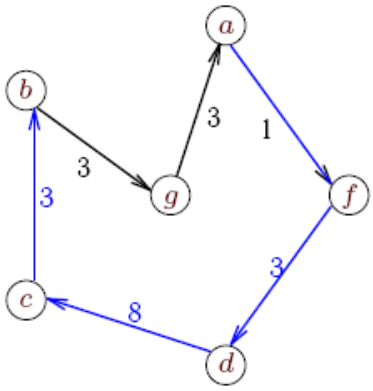

Рис. 4.57

Задание Г-48-2
Задача о коммивояжере
(алгоритм ближайшей вставки)

Это образное название устойчиво закрепилось за одной из самых интересных, практически значимых и одновременно сложных задач теории графов. Задача, берущая свое начало из работ Гамильтона, состоит в определении кратчайшего гамильтонова цикла в графе. Ее решение связано [18] с решением задачи о назначениях и с задачей об остове наименьшего веса.

Пример. Дан взвешенный орграф (рис. 4.52). Найти кратчайший гамильтонов цикл.

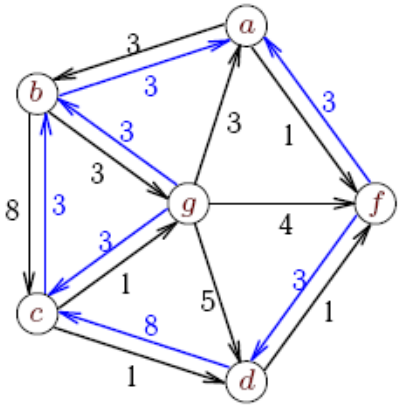

Рис. 4.52

Способ 2. Рассмотрим алгоритм ближайшей вставки (Nvi — Nearest insert) [2]. Построение цикла начнем с любой вершины, например c. Присоединим к c вершину, образующую вместе с ней цикл. Из трех возможных вариантов, $c \to d \to c$, $c \to g \to c$ и $c \to b \to c$, возьмем самый короткий, $c \to g \to c$, длиной 4 (рис. 4.53). На следующем шаге можно добавить либо вершину d, вставляя дуги $g \to d$ и $d \to c$ вместо дуги $g \to c$, либо b уже с двумя вариантами циклов: $g \to c \to b \to g$ и $g \to b \to c \to g$. Выбираем цикл наименьшей длины: $g \to c \to b \to g$ (рис. 4.54). Аналогично последовательно вставляем в цикл вершины d, f, a (рисунки 4.55–4.57). В результате получаем цикл длиной $3 + 3 + 1 + 3 + 8 + 3 = 21$.

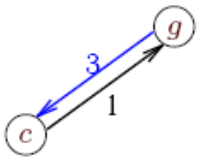

Рис. 4.53

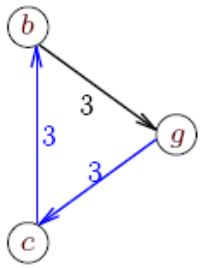

Рис. 4.54

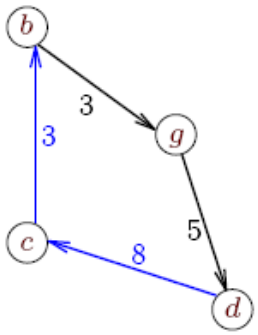

Рис. 4.55

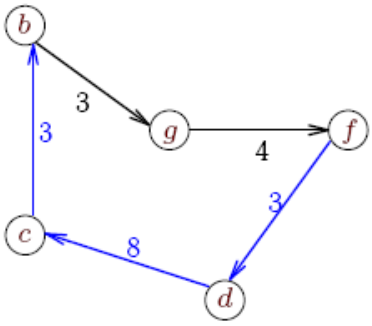

Рис. 4.56

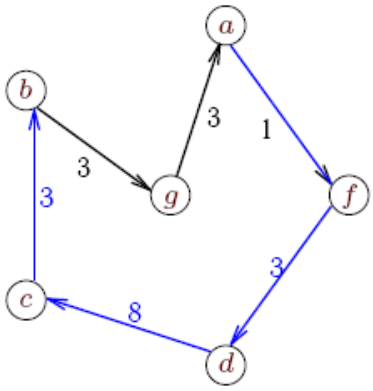

Рис. 4.57

Задание Г-48-3
Задача о коммивояжере
(муравьиный алгоритм)

Это образное название устойчиво закрепилось за одной из самых интересных, практически значимых и одновременно сложных задач теории графов. Задача, берущая свое начало из работ Гамильтона, состоит в определении кратчайшего гамильтонова цикла в графе. Ее решение связано [18] с решением задачи о назначениях и с задачей об остове наименьшего веса.

Пример. Дан взвешенный орграф (рис. 4.52). Найти кратчайший гамильтонов цикл.

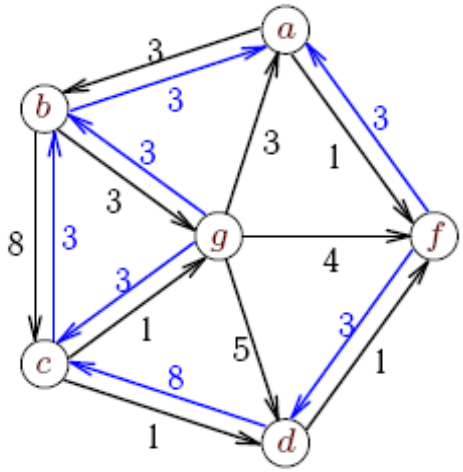

Рис. 4.52

Способ 3. Одним из методов искусственного интеллекта является муравьиный алгоритм Марко Дориго [1]. Основная идея алгоритма подсмотрена в природе и имитирует движение колонии муравьев. По форме этот алгоритм похож на жадный Nva (способ 1) и в некоторой степени является его обобщением. Если в алгоритме ближайшего соседа выбор дальнейшего пути производится, исходя из минимального расстояния до очередной вершины, то здесь выбором управляет случайная функция, направляющая движение от текущего положения с большей вероятностью в вершину j, в которой наибольшее значение некоторой функции $P_{ij,k}$ (где i — номер вершины, в которой производится выбор, k — номер муравья, движущегося по дугам графа). Как и в Nva, во время движения создается список пройденных вершин, что позволяет избежать преждевременного зацикливания. Приведем вид функции, управляющей переходом из данной вершины i в вершину j:

$$P_{ij,k} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum\limits_{m}\tau_{im}^{\alpha}\eta_{im}^{\beta}}, \qquad (4.4)$$

где τ_{ij} — количество феромона (pheromon), оставленного муравьями на дуге $[i, j]$; η_{ij} — величина, обратная весу (длине) дуги $[i, j]$; α, β — эмпирические коэффициенты. Функция $P_{ij,k}$ подсказывает муравью номер вершины j, в которую он должен направиться. В знаменателе (4.4) стоит нормирующий коэффициент, такой, что $0 \leqslant$ $\leqslant P_{ij,k} \leqslant 1$. Индекс m в сумме пробегает по всем непройденным вершинам, смежным с i. В реальности муравей оставляет след (феромон) во время прохождения пути, и чем чаще он возвращается в исходную точку (а это возможно, если он выбирает оптимальные пути), тем четче след. В математической же модели функция τ_{ij} увеличивается только по завершении маршрута на величину, обратно пропорциональную длине маршрута. При $\alpha = 0$ алгоритм совпадает с Nva — муравей руководствуется только длиной пути. При $\beta =$ $= 0$ основой для выбора пути является только опыт (количество феромона, или «глубина следа») предыдущих муравьев-исследователей. Важно отметить еще одно отличие от алгоритма Nva. Выбор пути производится не по максимуму функции $P_{ij,k}$, а случайным образом, но на случай, конечно, влияет значение $P_{ij,k}$. Поясним это на примере. Пусть муравей k подошел к некоторой вершине 8 и обнаружил, что перед ним 7 возможных путей к семи вершинам (на уже пройденные он внимания не обращает). Куда идти? Муравей доверяется случаю. Он «пускает рулетку» (рис. 4.58). В какой

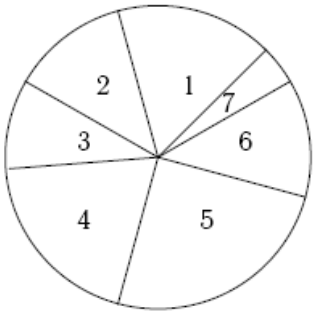

Рис. 4.58

сектор «шарик» закатится, туда и идти. Однако рулетка, размеченная функцией $P_{8j,k}$, $j = 1, ..., 7$, имеет неравные сектора. Чем ближе вершина и чем глубже туда след, тем больше сектор. Таким образом, муравей использует и опыт предшественников (τ_{ij}), и здравый смысл (η_{ij}), и случайный фактор, т.е. все как в жизни.

Для того чтобы не пропустить оптимальное решение, в муравьином алгоритме предусмотрено «испарение» следа. Это достигается введением коэффициента p в итеративной формуле $\tau_{ij} = (1 - p)\tau_{ij}$, применяющейся после каждого цикла обхода графа.

В алгоритме действует целая колония муравьев. Математически это означает, что в каждом цикле обхода движение производится из разных вершин независимым образом.

Задание Г-48-4
Задача о коммивояжере
(алгоритм отжига)

Это образное название устойчиво закрепилось за одной из самых интересных, практически значимых и одновременно сложных задач теории графов. Задача, берущая свое начало из работ Гамильтона, состоит в определении кратчайшего гамильтонова цикла в графе. Ее решение связано [18] с решением задачи о назначениях и с задачей об остове наименьшего веса.

Пример. Дан взвешенный орграф (рис. 4.52). Найти кратчайший гамильтонов цикл.

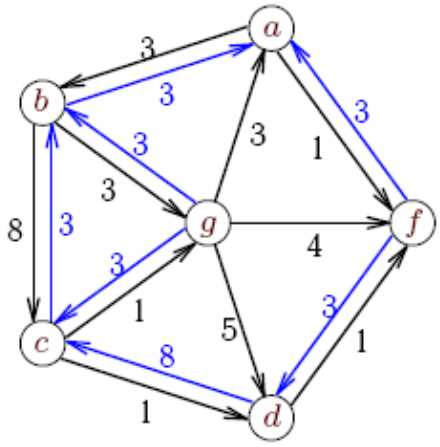

Рис. 4.52

Способ 4. Алгоритм отжига. Этот алгоритм, как и муравьиный алгоритм, относится к вероятностным методам решения. Ключевым моментом в таких подходах является случайный выбор одного из нескольких возможных решений вместо анализа каждого. Это позволяет сократить время счета, а время счета в некоторых задачах на графах имеет принципиальное значение. Простой перебор для задач, сложность которых (время счета) растет по показательному или факториальному закону в зависимости от порядка графа, может даже для небольших графов оказаться недопустимо длительным. Оценка этой характеристики в каждом приложении своя. Например, для мобильного робота [1], выполняющего маневр в поиске оптимального решения, бортовой компьютер должен решать задачу коммивояжера за доли секунды. Прямой же перебор вариантов при пяти-десяти пунктах назначения возможен только за несколько минут, что в данном случае никак не-приемлемо.

При оптимизации проектируемых в лабораторных условиях тепловых, газовых, электрических или транспортных сетей, как правило, с сотней или более вершин (узлов), проект может затянуться даже на многие годы. Именно поэтому актуальны вероятностные подходы, дающие, может быть, не самые точные, но вполне допустимые решения. Рассмотрим метод отжига, обычно изучаемый в курсах теории искусственного интеллекта.

Образное название лишь отчасти передает содержание и связывается с процессом образования структуры металла при нагревании и дальнейшем охлаждении. Известно, что только контролируемое охлаждение приводит к желаемой структуре металла. При большей температуре степень свободы частиц (в нашем случае — количество вариантов решения) больше, при меньшей — меньше. Если дать возможность алгоритму случайно выбирать решение, оптимальное на каждом шаге (жадный алгоритм), то можно пропустить ход, не лучший локально, но дающий в результате более оптимальное решение.

В методе отжига очередной порядок следования по маршруту между городами выбирается случайно, небольшим изменением предыдущего решения, предположительно оптимального. Самый простой вариант изменения — перестановка двух случайно выбранных городов в маршруте следования. Если полученный маршрут лучше всех существовавших ранее, то этот маршрут берется за очередной [1]. Если маршрут хуже, то самый простой вариант — это не брать его (зачем ухудшать решение?). Однако так решение может закатиться в один из локальных минимумов, которыми изобилуют подобные задачи (рис. 4.59). Именно

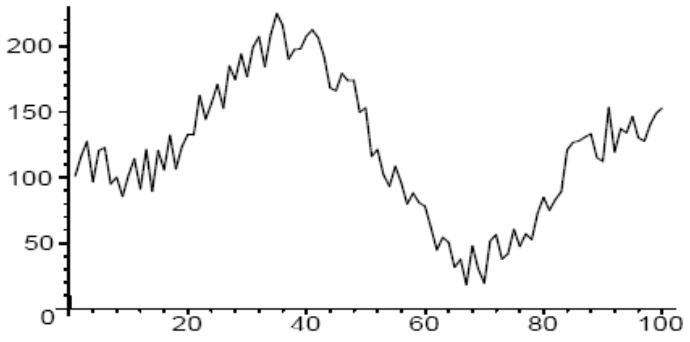

Рис. 4.59

поэтому плохому решению надо дать шанс. Шанс этот (или, правильнее, вероятность) рассчитывается по формуле

$$P = \exp(-\Delta L/T),$$