Поток в насыщенной сети можно посчитать по потоку, выходящему из истока 1 или входящему в сток 8. Очевидно, эти числа должны быть равны. Кроме того, для проверки решения следует проверить условие сохранения потока по узлам. Для каждого узла суммарный входящий поток должен быть равен выходящему. В рассматриваемом примере поток равен 11. Распределение потока по дугам при одном и том же суммарном минимальном потоке в сети не единственное.

Maple-программа для определения максимального потока в сети приведена на с. 123.

4.3. Топологическая сортировка 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,...,N_k, причем среди этих множеств нет пустых [3].

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

Задача. Найти порядковую функцию сети (рис. 4.8).

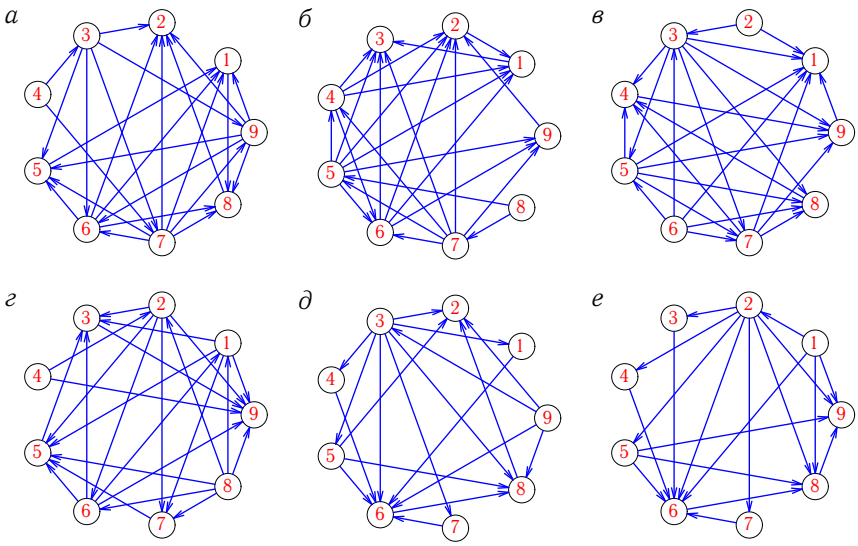

Рис. 4.8

ж *з* *и*

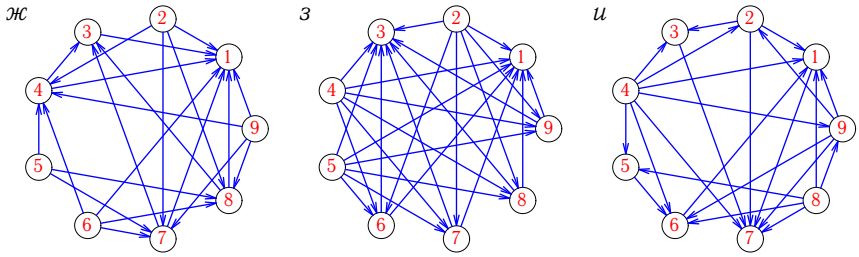

Рис. 4.8 (продолжение)

Ответы

№	Порядковая функция
а	4, 3, 7, 9, 6, 5, 1, 8, 2
б	8, 7, 5, 4, 6, 9, 2, 1, 3
в	(2, 6), 3, 5, 7, 8, 4, 9, 1
г	(4, 8), 2, 6, 1, 7, 5, 3, 9
д	9, 3, (1, 7, 4, 5), 6, 8, 2
е	1, 2, (7, 3, 4, 5), 6, 8, 9
ж	(5, 6, 2, 9), (4, 7, 8), 3, 1
з	(2, 5, 4), (8, 6, 7, 9), (3, 1)
и	(4, 8), (5, 9), (6, 2), (1, 3), 7

Пример. Отсортировать топологически сеть на рис. 4.9.

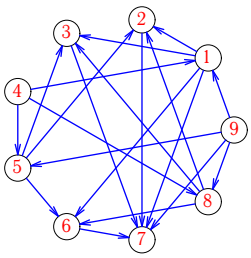

Рис. 4.9

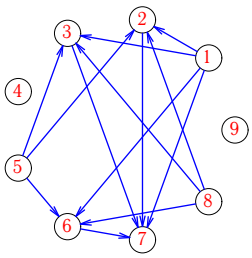

Рис. 4.10

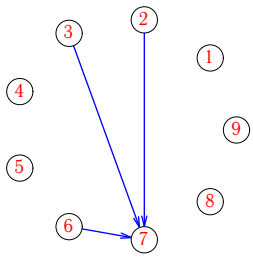

Рис. 4.11

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

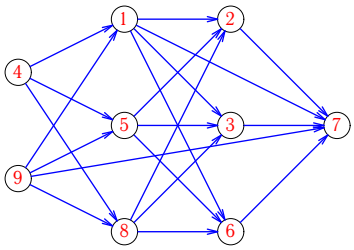

Рис. 4.12

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).

Maple-программа для топологической сортировки сети приведена на с. 127.

4.4. 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).

Теорема 16 (Кенига [1]). *Для двудольности графа необходимо и достаточно, чтобы он не содержал циклов нечетной длины.*

Если любые две вершины двудольного графа, входящие в разные доли, смежны, то граф называется *полным двудольным*. Обозначение для полного двудольного графа с n и m вершинами в долях — $K_{n,m}$.

Так как по определению в двудольном графе вершины одной доли никогда не соединяются ребрами, для избежания ввода ненужной информации матрицу смежности для двудольного графа можно записать в сокращенной форме: элемент a_{ij} матрицы смежности двудольного графа равен 1, если i-я вершина одной доли соединена с j-й вершиной другой доли; в противном случае $a_{ij} = 0$. Если нумерация вершин графа общая, а не по долям, то j-ю вершину другой доли в определении надо заменить на $(j + n)$-ю вершину графа $K_{n,m}$. Очевидно, матрица двудольного графа в общем случае имеет размер $n \times m$, т.е. не является квадратной.

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

[1] König D.

```
> pr1:=(x)->eweight(x,H)-potok1[x]; # Процедура 1
> pr2:=(x)->potok1[x]:              # Процедура 2
> if notupik1 then # Перераспределяем поток
>   m1:=min(op(map(pr1,out2)));
>   m2:=min(op(map(pr2,in2)));
>   ptk:=min(m1,m2);
>   for i in in2 do
>    potok1[i]:=potok1[i]-ptk;
>   od:
>   for i in out2 do
>    potok1[i]:=potok1[i]+ptk; od:
> fi;
> od: # while
> edg2:=incident(v2,H,'In'):# Дуги, входящие в сток
> Поток=map(`+`,op(eweight([op(edg2)],H)));
> satur1:=[]:
> for x in edges(H) do
>   if pr1(x)=0 then satur1:=[op(satur1),x];fi;
> od;
> satur1;  # Насыщенные дуги
```

$$\text{Поток} = 11$$

$$[e4,\ e5,\ e6,\ e7,\ e10,\ e12,\ e8]$$

5.20. Топологическая сортировка сети

Вводятся данные сети с рис. 4.9 (см. с. 65). В алгоритме используется удобная функция `delete`, позволяющая удалять список вершин и соответствующие дуги. Каждый уровень организуется в виде списка для того, чтобы использовать результаты в операторе `Linear` топологически упорядоченной сети. После формирования списка вершин одного уровня эти вершины удаляются из графа. Работа цикла `while NV<>0 do` продолжается до тех пор, пока множество вершин не станет пустым. Счетчик числа вершин — `NV`. Копия графа `H` выводится на экран по полученным уровням.

Если граф содержит цикл, то на очередном этапе поиска вершины с нулевой полустепенью захода список уровня окажется пустым и произойдет досрочный выход из цикла с помощью оператора `break`. К графу на рис. 4.9 достаточно добавить, например, дугу `[3,4]`, для того чтобы образовался контур и топологическая сортировка оказалась невозможной.

Программа 25

```
> restart: with(networks):
> new(G):V:=$1..9: addvertex([V],G):
> E:=[[1,2],[1,3],[1,7],[2,7],[3,7],[4,1],[4,5],
> [4,8],[5,2],[5,3],[5,6],[6,7],[8,6],[8,3],[8,2],
> [9,1],[9,5],[9,7],[9,8]]:# Дуги
> addedge(E,G):draw(G):H:=duplicate(G):
> S0:=[]:
> while NV<>0 do
>   S1:=[]:
>   for v in vertices(G) do
>     if indegree(v,G)=0 then
>       S1:=[op(S1),v];
>     fi; # Множество вершин одного уровня
>   od;
>   if nops(S1)=0 then print("Сеть содержит контур");
>     break
>   fi;
>   S0:=[op(S0),S1]; # Добавляем уровень
>   delete(S1,G):    # Удаляем вершины
>   NV:=nops(vertices(G)) # Число вершин
> od:
> S0;
```

$$[[4, 9], [1, 5, 8], [2, 3, 6], [7]]$$

```
> draw(Linear(op(S0)),H);
```