ОГЛАВЛЕНИЕ

ОБХОД ГРАФОВ В ШИРИНУ И ГЛУБИНУ

Описание алгоритма

Пусть нам необходимо обойти граф G (V, E), который представлен списком смежности $Г$. Обход графа подразумевает некоторое систематическое перечисление его вершин. Для этого используются следующие вспомогательные структуры данных:

1) ***Структура данных T*** является своего рода вспомогательным буфером, в который временно помещаются обойденные вершины (это необходимо для обхода смежных с ними вершин). Данная структура может являться стеком (в случае поиска в глубину) или очередью (в случае поиска в ширину).

Стек – это структура данных, в которой первый помещенный в нее элемент извлекается последним. *Очередь* – это структура данных, в которой первый помещенный в нее элемент извлекается первым.

2) ***Массив X***, длина которого равна числу вершин, содержит данные о том, была ли отмечена (пройдена) вершина. Каждый элемент массива соответствует одной вершине графа и может принимать два значения:

1 – вершина отмечена (пройдена);

0 – вершина не отмечена.

Рассмотрим, как осуществляется обход поэтапно:

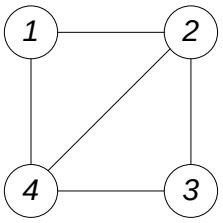
1) Обнуление массива X. До начала обхода все вершины являются неотмеченными.
2) Выбирается некоторая произвольная вершина v, с которой будет начинаться обход.
3) Вершина v помещается в структуру T и отмечается в массиве X как пройденная ($X[v] := 1$).
4) Из структуры T извлекается вершина (обозначим её u). Эта вершина является пройденной. Таким образом, именно на этом этапе мы выделяем обойденные вершины.
5) По списку смежности $Г$ поочередно выбираются все вершины смежные с u (обозначим вершину смежную с u – w) и если они не были раннее отмечены (то есть, если $X[w] = 0$), то они помещаются в структуру T и отмечаются.

Если в структуре T находятся какие-либо вершины, то осуществляется переход к пункту 4. Если же нет, то обход графа G (V, E) закончен.

Пример

Обход графа в глубину:

Рассмотрим на примере обход графа $G(V, E)$.



Список смежности (Γ)

1 – 2, 4
2 – 1, 3, 4
3 – 2, 4
4 – 1, 2, 3

1) Обнулим массив X.

X:	1	2	3	4
	0	0	0	0

2) Начнем обход с первой вершины ($v = 1$).
3) Поместим вершину ($v = 1$) в структуру T и отметим ее в массиве X как пройденную.

T:	1			

X:	1	2	3	4
	1	0	0	0

4) Извлекаем из структуры T вершину 1 ($u = 1$), т.е. вершина 1 пройдена.

T:				

5) Затем помещаем вершины смежные с первой ($u = 1$) и еще не отмеченные в массиве X в структуру T и отмечаем их в массиве X.

T:	2	4		

X:	1	2	3	4
	1	1	0	1

6) Извлекаем из структуры T вершину 4 ($u = 4$), т.е. вершина 4 пройдена.

T:	2			

7) Затем помещаем вершины смежные с четвертой ($u = 4$) и еще не отмеченные в массиве X (2 и 3) в структуру T и отмечаем их в массиве X.

T:	2	3		

X:	1	2	3	4
	1	1	1	1

8) Извлекаем из структуры T вершину 3 ($u = 3$), т.е. вершина 3 пройдена.

T:	2			

9) Все вершины смежные с третьей уже отмечены в массиве X, поэтому ничего не помещаем в структуру T.

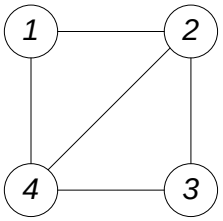
10) Извлекаем из структуры *T* вершину 2 (*u* = 2), т.е. вершина 2 пройдена.

T:

11) Все вершины смежные со второй уже отмечены в массиве *X*, поэтому ничего не помещаем в структуру *T*.
12) Структура *T* пуста. Обход вершин графа закончен: 1 – 4 – 3 – 2.

Обход графа в ширину:

Рассмотрим на примере обход графа *G* (*V*, *E*).



Список смежности (*Г*)

1 – 2, 4
2 – 1, 3, 4
3 – 2, 4
4 – 1, 2, 3

1) Обнулим массив *X*.

X:

1	2	3	4
0	0	0	0

2) Начнем обход с первой вершины (*v=1*).
3) Поместим 1 вершину (*v* = 1*)* в структуру *T* и отметим ее в массиве *X* как пройденную.

T:

1			

X:

1	2	3	4
1	0	0	0

4) Извлекаем из структуры *T* вершину 1 (*u* = 1), т.е. вершина 1 пройдена.

T:

5) Затем помещаем вершины смежные с первой (*u=1*) и еще не отмеченные в массиве *X* в структуру *T* и отмечаем их в массиве *X*.

T:

2	4		

X:

1	2	3	4
1	1	0	1

6) Извлекаем из структуры *T* вершину 2 (*u* = 2), т.е. вершина 2 пройдена.

T:

4			

7) Затем помещаем вершины смежные со второй (*u* = 2) и еще не отмеченные в массиве *X* (3) в структуру *T* и отмечаем их в массиве *X*.

T:

4	3		

X:

1	2	3	4
1	1	1	1

8) Извлекаем из структуры *T* вершину 4 (*u* = 4), т.е. вершина 4 пройдена.

T:

3			

9) Все вершины смежные с четвертой уже отмечены в массиве *X*, поэтому ничего не помещаем в структуру *T*.
10) Извлекаем из структуры *T* вершину 3 (*u* = 3), т.е. вершина 3 пройдена.

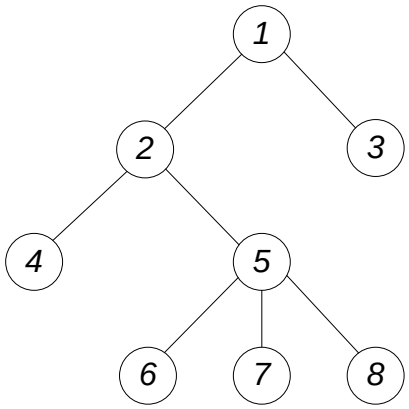
T:

11) Все вершины смежные с третьей уже отмечены в массиве *X*, поэтому ничего не помещаем в структуру *T*.
12) Структура *T* пуста. Обход вершин графа закончен: 1 – 2 – 4 – 3.

На практике обход в глубину можно использовать в частности для получения ориентированного дерева из неориентированного. Рассмотрим это на примере.

Пример получения ориентированного дерева

Исходное дерево представлено на рисунке.



Список смежности (*Г*)

1 – 2, 3
2 – 1, 4, 5
3 – 1
4 – 2
5 – 2, 6, 7, 8
6 – 5
7 – 5
8 – 5

Наша цель – посредством обхода исходного дерева в глубину, получить ориентированное дерево с корнем *v* и списком смежности *Г'*.

1) Обнулим массив *X*.

X:

1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0

2) Пусть корнем будет вершина 5 (*v* = 5). Поместим вершину 5 в структуру *T* и пометим ее как пройденную в массиве *X*.

T:

5							

X:

1	2	3	4	5	6	7	8
0	0	0	0	1	0	0	0

3) Затем извлекаем из структуры *T* вершину (*u* = 5) и помещаем смежные с ней вершины (ранее не пройденные) в *T*.

T:

2	6	7	8				

X:

1	2	3	4	5	6	7	8
0	1	0	0	1	1	1	1

4) Начинаем формировать список смежности ориентированного дерева (*Г'*). Для этого берем вершину *u* и выписываем из списка смежности исходного графа (*Г*) вершины, совпадающие с вершинами, находящимися в структуре *T* (это будут вершины смежные с *u*), то есть

Г': 5 – 2, 6, 7, 8

5) Затем извлекаем из структуры *T* вершину 8 (*u* = 8). Дополняем список смежности *Г'* восьмой вершиной. У нее нет смежных вершин.

T:

2	6	7					

Г': 5 – 2, 6, 7, 8; 8 – ;

6) Затем извлекаем из структуры *T* вершину 7 (*u* = 7). Дополняем список смежности *Г'*.

T:

2	6						

Г': 5 – 2, 6, 7, 8; 8 – ; 7 – ;

7) Затем извлекаем из структуры *T* вершину 6 (*u* = 6). Дополняем список смежности *Г'*.

T:

2							

Г': 5 – 2, 6, 7, 8; 8 – ; 7 – ; 6 –

8) Затем извлекаем из структуры *T* вершину 2 (*u* = 2). Дополняем список смежности *Г'*.

T:

1	4						

X:

1	2	3	4	5	6	7	8
1	1	0	1	1	1	1	1

Г': 5 – 2, 6, 7, 8; 8 – ; 7 – ; 6 – ; 2 – 1, 4;

9) Извлекаем из структуры *T* вершину 4 (*u* = 4). Дополняем список смежности *Г'*.

T:

1							

Г': 5 – 2, 6, 7, 8; 8 – ; 7 – ; 6 – ; 2 – 1, 4; 4 – ;

10) Извлекаем из структуры *T* вершину 1 (*u* = 1). Дополняем список смежности *Г'*. Помещаем смежные с ней вершины (ранее не пройденные) в *T*.

T:

3							

X:

1	2	3	4	5	6	7	8
1	1	1	1	1	1	1	1

Г': 5 – 2, 6, 7, 8; 8 – ; 7 – ; 6 – ; 2 – 1, 4; 4 – ; 1 – 3;

11) Извлекаем из структуры *T* вершину 3 (*u* = 3). Дополняем список смежности *Г'*.

T: | | | | | | | | |

Г': 5 – 2, 6, 7, 8; 8 – ; 7 – ; 6 – ; 2 – 1, 4; 4 – ; 1 – 3; 3 –

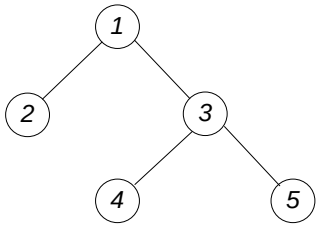
12) Структура *T* пуста. Обход графа закончен. Получили следующий список смежности ориентированного дерева (*Г'*):

1 – 3
2 – 1, 4
3 –
4 –
5 – 2, 6, 7,8
6 –
7 –
8 –

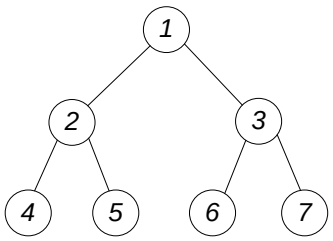
Задания

Задание 1. Построить из дерева ориентированное методом обхода в глубину из первой вершины.

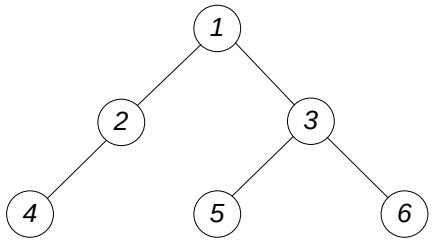
1.



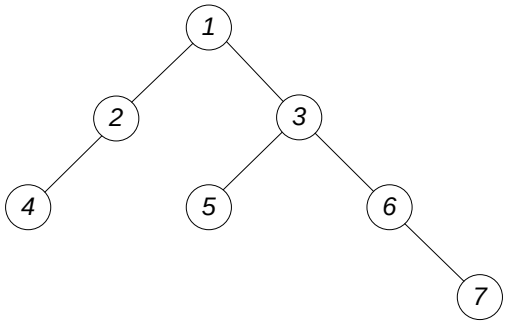
2.



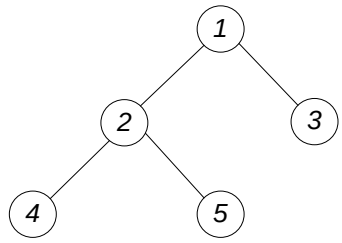
3.



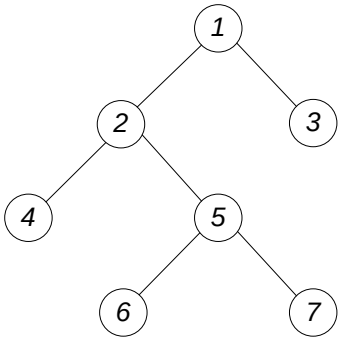
4.



5.

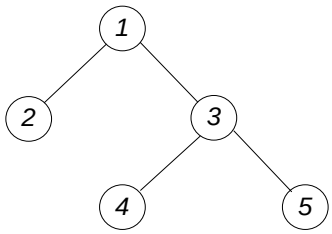


6.

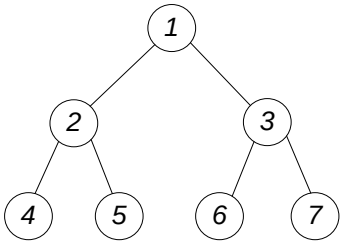


Задание 2. Построить из дерева ориентированное методом обхода в глубину из третьей вершины.

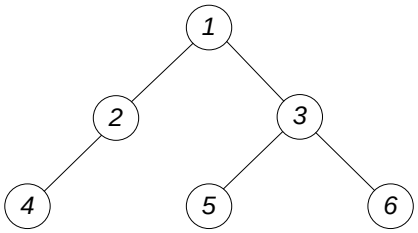
7.



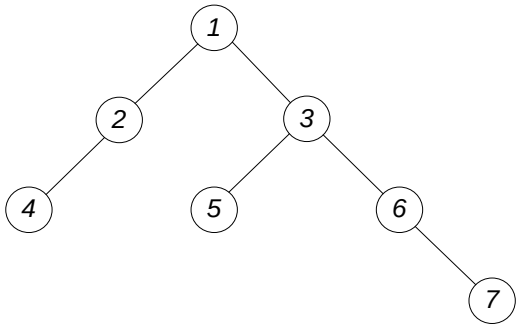
8.



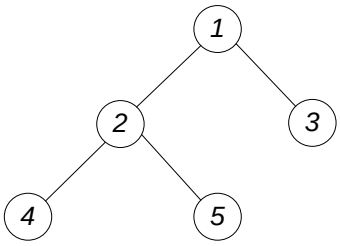
9.



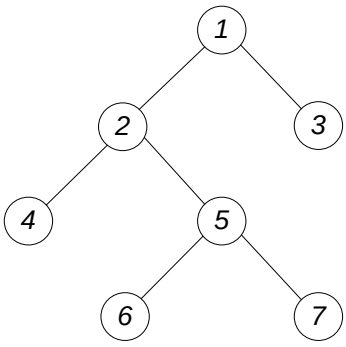
10.



11.

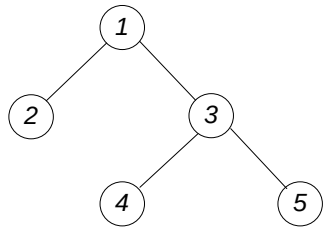


12.

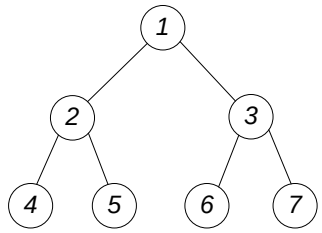


Задание 3. Построить из дерева ориентированное методом обхода в глубину из второй вершины.

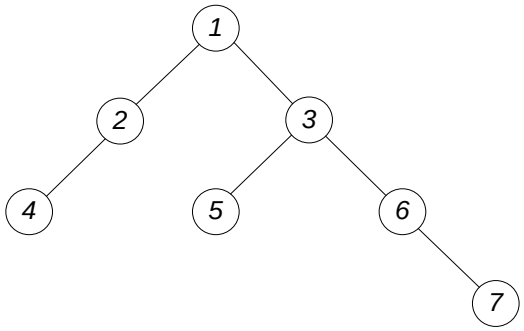
13.



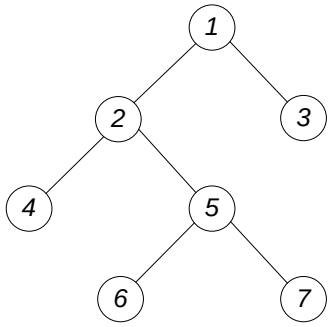
14.



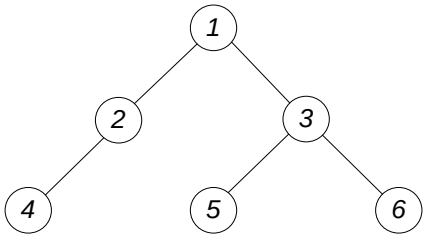
15.



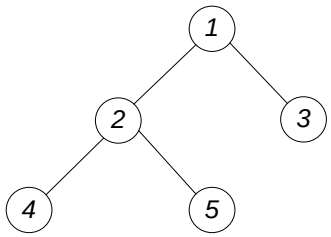
16.



17.

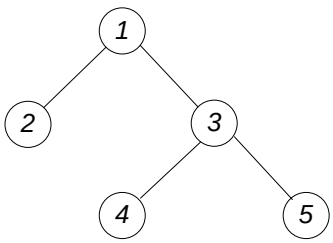


18.

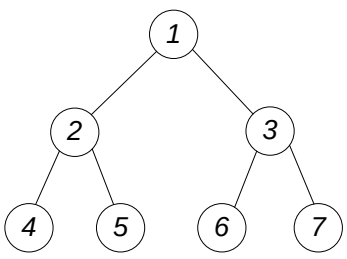


Задание 4. Построить из дерева ориентированное методом обхода в глубину из четвертой вершины.

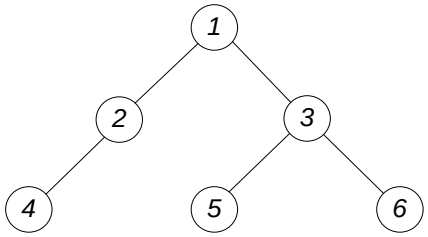
19.



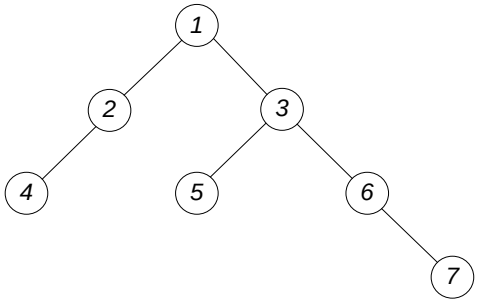
20.



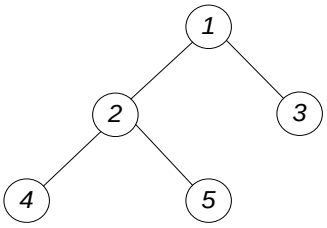
21.



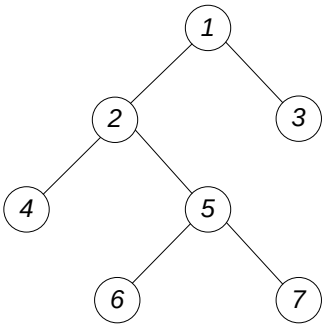
22.



23.

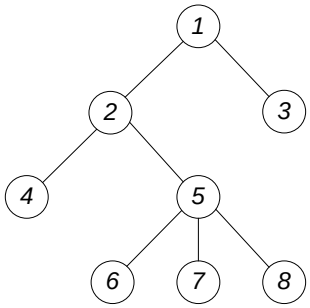


24.

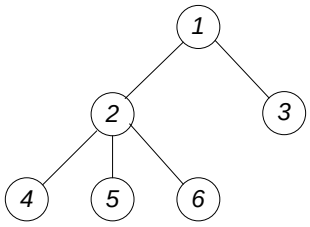


Задание 5. Построить из дерева ориентированное методом обхода в глубину из первой вершины.

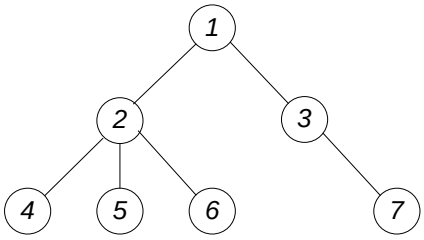
25.



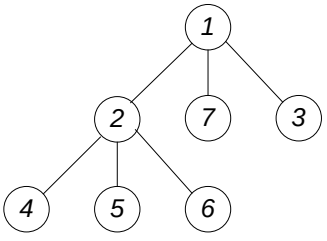
26.



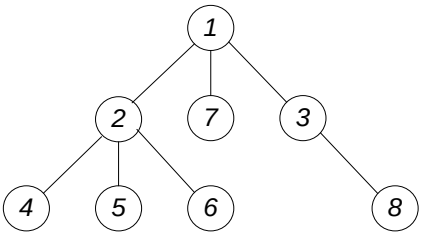
27.



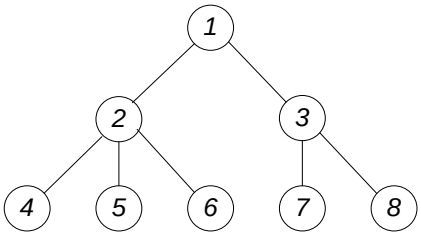
28.



29.



30.



АЛГОРИТМ ТЭРРИ

Описание алгоритма

Алгоритм Тэрри – алгоритм поиска маршрута в связном графе G (V, E), соединяющий заданные вершины v и w. Исходя из вершины v и осуществляя последовательный переход от каждой достигнутой вершины к смежной ей вершине, всегда можно найти маршрут в связном графе G (V, E), соединяющий заданные вершины v и w. Переход от вершины к вершине согласно алгоритму осуществляется по следующим правилам:

1. при проходе ребра необходимо всякий раз отмечать направление, в котором оно было пройдено;
2. исходя из некоторой вершины v' нужно всегда следовать по тому ребру, которое не было пройдено или было пройдено в противоположном направлении;
3. для всякой вершины v', отличной от v, необходимо отмечать пометкой «х» первое заходящее ребро, если вершина v' встречается первый раз;
4. исходя из вершины v', отличной от v, по первому заходящему в v' ребру нужно идти лишь тогда, когда нет других возможностей.

Пример

Рассмотрим алгоритм Тэрри на примере поиска маршрута между вершинами 1 и 4 графа G (V, E) (рис. 1). Если это не противоречит правилам алгоритма, то переходить от одной вершины будем к смежной ей с меньшим номером.

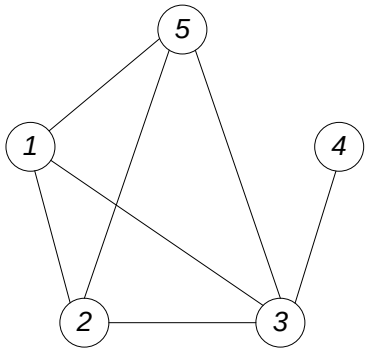


Рисунок 1

1) Переходим из вершины 1 в вершину 2. отмечаем направление прохода ребра (рис.2).

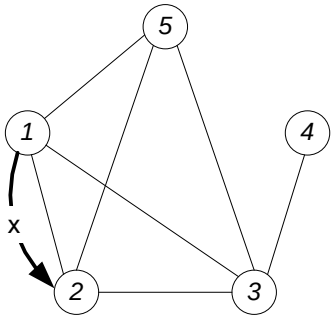


Рисунок 2

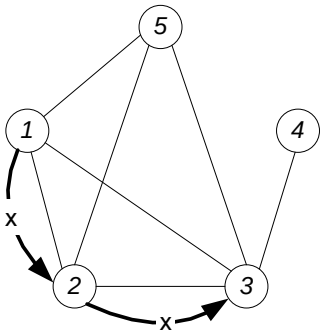


Рисунок 3

2) Из вершины 2 можем перейти к вершинам 3 и 5, а также к вершине 1, но лишь в том случае, если нет других вариантов. Переходим к вершине 3. Отмечаем направление перехода (рис. 3).

3) Из третьей вершины переходим в первую, так как она имеет наименьший номер среди всех возможных вариантов перехода (рис. 4).

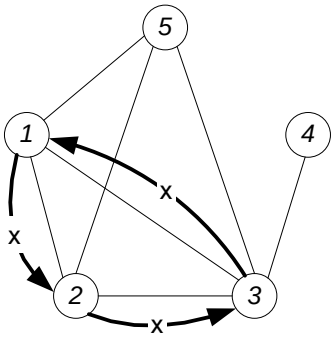


Рисунок 4

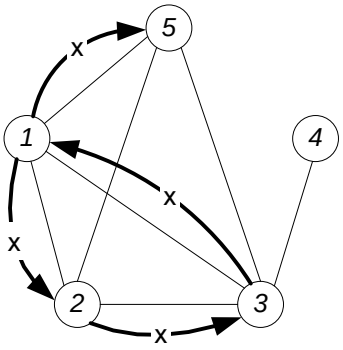


Рисунок 5

4) Первая вершина смежна с вершинами 2, 3 и 5. В вершину 2 мы не можем осуществить переход, так как это направление уже отмечено. В вершину 3 осуществлять переход не будем, так как ребро 1-3 уже отмечено как заходящее в вершину 1 и существует альтернатива перехода к вершине 5. Следовательно, переходи к вершине 5. Отмечаем направление перехода (рис. 5).

5) Из вершины 5 переходим к вершине 2, так как она имеет наименьший номер среди всех возможных вариантов перехода (рис. 6).

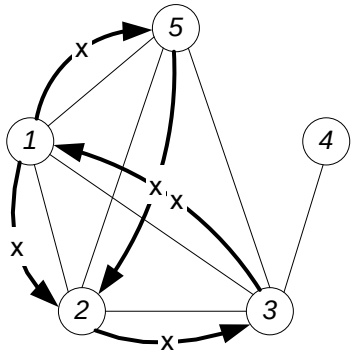


Рисунок 6

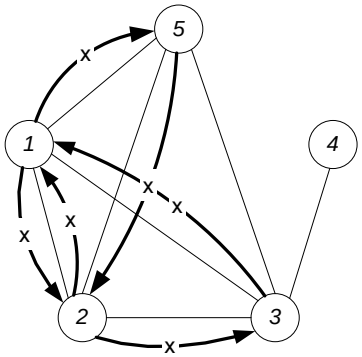


Рисунок 7

6) Из вершины 2 переходим к вершине 1, так как ребро, связывающее эти вершины, отмечено как заходящее в вершину 2, а не отмеченных ребер при рассматриваемой вершине нет. Отмечаем направление перехода (рис. 7).

7) Из вершины 1 можно перейти только к вершине 3 (рис. 8).

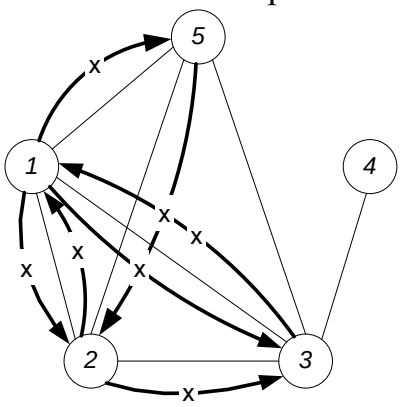


Рисунок 8

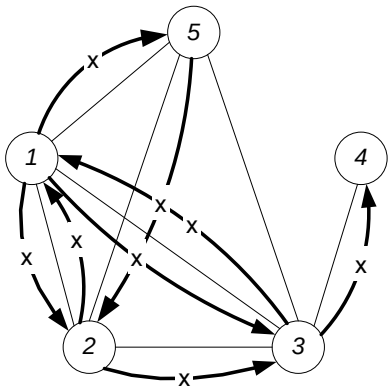


Рисунок 9

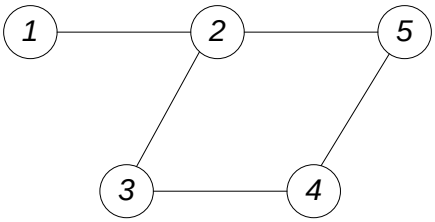
8) Из вершины 3 осуществляем переход к вершине 4, так как она имеет наименьший номер среди всех альтернативных вариантов (рис. 9). Поиск маршрута окончен.

Маршрут из вершины 1 в вершину 4 в соответствии с алгоритмом Тэрри имеет вид:

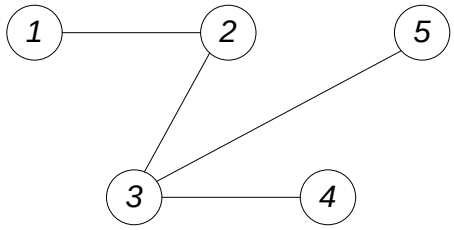
1 – 2 – 3 – 1 – 5 – 2 – 1 – 3 – 4

Задания: Определить путь обхода графов.

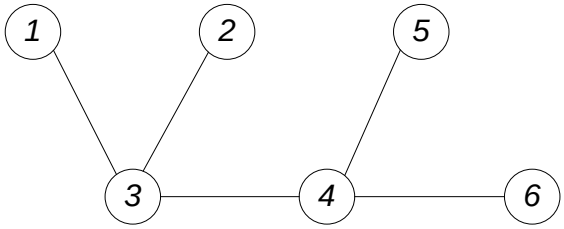
1. обход в ширину из точки 3
2. обход в ширину из точки 1
3. обход в ширину из точки 5
4. обход в глубину из точки 1
5. обход в глубину из точки 2



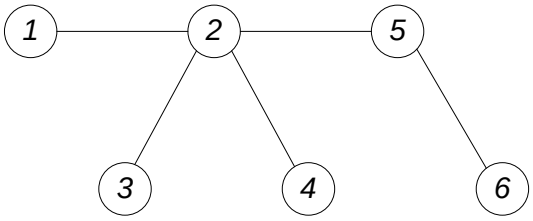
6. обход в ширину из точки 2
7. обход в ширину из точки 4
8. обход в глубину из точки 1
9. обход в глубину из точки 3
10. обход в глубину из точки 5



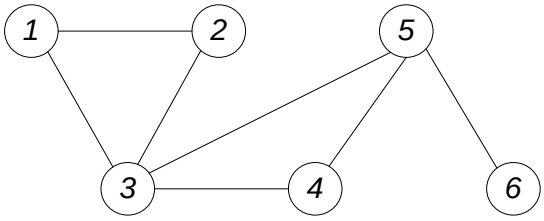
11. обход в ширину из точки 1
12. обход в ширину из точки 4
13. обход в ширину из точки 5
14. обход в глубину из точки 2
15. обход в глубину из точки 6



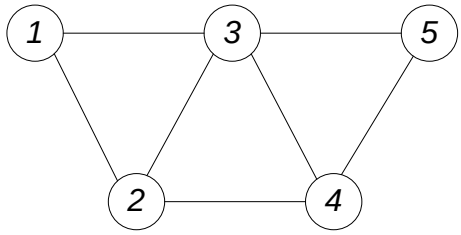
16. обход в ширину из точки 2
17. обход в ширину из точки 6
18. обход в глубину из точки 3
19. обход в глубину из точки 5
20. обход в глубину из точки 1



21. обход в ширину из точки 1
22. обход в ширину из точки 3
23. обход в ширину из точки 4
24. обход в глубину из точки 6
25. обход в глубину из точки 2



26. обход в ширину из точки 2
27. обход в ширину из точки 5
28. обход в глубину из точки 1
29. обход в глубину из точки 3
30. обход в глубину из точки 4



МАТРОИДЫ. ЖАДНЫЕ АЛГОРИТМЫ. АЛГОРИТМ КРАСКАЛА.

Описание алгоритма

Матроидом $M = < E, \varepsilon >$ называется конечное множество E, число элементов которого равно n ($|E| = n$), и семейство его подмножеств ε, ко-

торое содержится в множестве всех подмножеств множества E ($\varepsilon \subset 2^E$) такое, что выполняются следующие три аксиомы:

M_1: пустое множество принадлежит подмножеству ε ($\varnothing \in \varepsilon$)□;

M_2: если множество A принадлежит подмножеству ε, то и множество B, которое содержится в множестве A, тоже принадлежит подмножеству ε.

($A \in \varepsilon \,\&\, B \subset A \Rightarrow B \in \varepsilon$);

M_3: если множества A и B принадлежат подмножеству ε и количество элементов множества A на один элемент меньше чем в множестве B, то существует элемент принадлежащий разности множеств B и A такой, что объединение его с множеством A будет принадлежать подмножеству ε.

($A,B \in \varepsilon \;\&\; |B| = |A| + 1 \Rightarrow \exists\, e \in B \backslash A \;\; A \cup \{e\} \in \varepsilon$)

Элементы множества ε□ называются независимыми, а остальные подмножества E (2^E□ \□ ε) – зависимыми. Пусть множество X (произвольное множество) содержится в множестве E. Максимальным независимым подмножеством множества X называется множество Y, такое что если множество Y, принадлежащие независимому множеству ε, содержится в множестве X и Z, принадлежащем независимому множеству ε, содержится в множестве X, то множество Z содержится в множестве Y ($Y \subset X \;\&\; Y \in \varepsilon \;\&\; Z \subset X$□ $\Rightarrow Z \subset Y$). Множество максимальных независимых подмножеств множества X обозначим $\underline{X}$.

Рассмотрим следующее утверждение, справедливое для любого X:

M_4: максимальные независимые подмножества данного множества равномощны ($Y \in \underline{X} \;\&\; Z \in \underline{X} \Rightarrow |Y| = |Z|$).

Пусть $M = < E, \varepsilon >$ и выполнены аксиомы M_1 и M_2 . Тогда аксиома M_3 и утверждение M_4 эквивалентны, то есть M_1, M_2, M_4-эквивалентная система аксиом матроида.

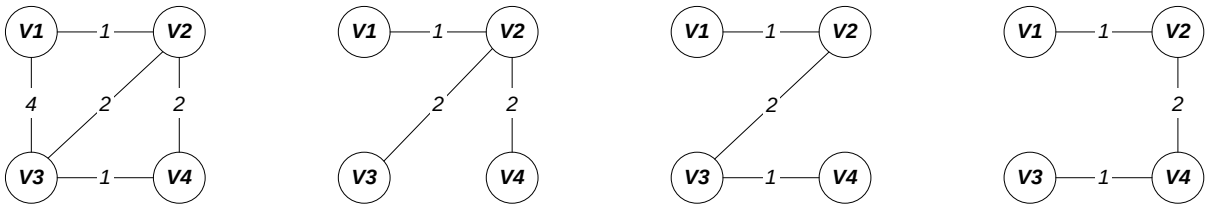
Пусть имеются конечное множество E, весовая функция ω и семейство $\varepsilon \subset 2^E$. Необходимо выбрать в указанном семействе ε подмножество X наибольшего веса. Для решения этой задачи будем считать, что множество E упорядочено в порядке убывания весов элементов. В начале множество X пусто. Затем проверяем каждый элемент множества E, начиная с первого, если объединение его с множеством X принадлежит указанному семейству ε, то добавляем его в множество X, если не принадлежит семейству ε, то рассматриваем следующий элемент, и так до n-ого.

Алгоритм такого типа называется ***жадным***. Очевидно, что жадный алгоритм является очень эффективным, он за наименьшее количество ша-

гов находит искомое подмножество. Жадные алгоритмы и их свойства исследованы сравнительно недавно, но их значение в практике программирования чрезвычайно велико. Если удается свести конкретную экстремальную задачу к такой постановке, где множество допустимых вариантов (из которых необходимо выбрать наилучший) является матроидом, то в большинстве случаев следует сразу применять жадный алгоритм, поскольку он достаточно эффективен в практическом смысле. Если же наоборот, оказывается, что множество допустимых вариантов не образует матроида, то это «плохой признак». Скорее всего, данная задача окажется труднорешаемой. В этом случае целесообразно тщательно исследовать задачу для предварительного получения теоретических оценок сложности, чтобы избежать бесплодных попыток «изобрести» эффективный алгоритм там, где это на самом деле невозможно.

Другими словами, если $M = < E, \varepsilon >$ - матроид, то для любой весовой функции ω жадный алгоритм находит независимое множество X с наибольшим весом; если же $M = < E, \varepsilon >$ не является матроидом, то существует такая функция ω, что множество X, найденное жадным алгоритмом, не будет максимальным.

Пусть G (V,E) – граф. Остовной подграф G (V,E) – это подграф, содержащий все вершины. Остовной подграф являющийся деревом, называется ***остовом***. Несвязный граф не имеет остова, связный граф может иметь много остовов. Если задать длины рёбер, то можно поставить задачу нахождения кратчайшего остова. Существует множество различных способов найти какой-то остов графа. Множество кратчайших путей из заданной вершины ко всем остальным тоже образует остов. Однако этот остов может не быть кратчайшим. На рисунке показаны диаграммы графа, дерево кратчайших путей из вершины один с суммарным весом 5 и два кратчайших остова этого графа.



Одним из способов нахождения кратчайшего остова в связном графе является алгоритм Краскала. Он является частным случаем жадного алгоритма. Заметим, что множество подмножеств множества рёбер, не содержащих циклов, образует матроид.

Алгоритм Краскала

Алгоритм Краскала основан на выборе ребер графа, составляющих его наикратчайший остов. Изначально известен список ребер графа с их длинами. Выбор происходит по двум критериям:

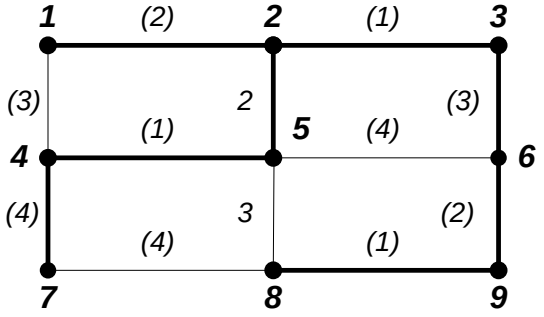
1. так как остов должен быть кратчайшим, выбор начинается с ребер минимальной длины;
2. так как остов – это дерево (структура, не содержащая циклов), каждое выбранное ребро проверяется: не составляет ли оно цикла с ранее выбранными ребрами.

Цикл проходит $n = v - 1$ раз, где v – количество вершин графа.

Таким образом, при совершении нужного количества вышеуказанных действий мы будем иметь множество ребер, составляющих наикратчайший остов графа. Рассмотрим этот алгоритм на примере.

Пример

Задан граф. Необходимо найти его минимальный остов при помощи алгоритма Краскала.



Решение:

Представим список ребер исходного графа с указанием длины каждого из них. Затем отсортируем ребра в порядке возрастания их длин.

Исходный список	**Отсортированный список**
(1,2) - 2	(2,3) - 1
(1,4) - 3	(4,5) - 1
(2,3) - 1	(8,9) - 1
(2,5) - 2	(1,2) - 2
(3,6) - 3	(2,5) - 2
(4,5) - 1	(6,9) - 2
(4,7) - 4	(1,4) - 3
(5,6) - 4	(3,6) - 3
(5,8) - 3	(5,8) - 3
(6,9) - 2	(4,7) - 4
(7,8) - 4	(5,6) - 4
(8,9) - 1	(7,8) - 4

Для решения задачи с помощью алгоритма Краскала составим таблицу, отражающую шаги выполнения алгоритма для рассматриваемого примера.

Шаг	Массив ребер	Компонента связности

В втором столбце формируем массив ребер по принципу:

Шаг 1. Помещаем в первый столбец первое ребро отсортированного списка. Компонента связности данного ребра будет соответствовать вершинам, которые данное ребро связывает.

Шаг 2…Шаг *N*. Рассматриваем *n*-ое ребро отсортированного списка и:

1) добавляем ребро в массив, если оно удовлетворяет условию ацикличности;
2) пропускаем ребро, в противном случае.

И так далее до конца списка ребер.

Проверка ацикличности формируемого массива производится при помощи третьего столбца таблицы, в который помещаются компоненты связности «леса», разрастающегося во втором столбце. Компоненты связности формируются следующим образом:

1) если вершины добавляемого ребра не входят ни в одну из уже существующих компонент связности, то формируется новая компонента связности из двух вершин.

Шаг	Массив ребер	Компонента связности
1	(2,3)	2,3
2	(2,3); (4,5)	2,3 | 4,5
3	(2,3); (4,5); (8,9)	2,3 | 4,5 | 8,9

2) если одна из вершин добавляемого ребра входит в одну из имеющихся компонент связности, то ребро заносится в массив, соответствующий компоненте связности, а вторая вершина добавляемого ребра присоединяется к имеющейся компоненте.

Шаг	Массив ребер	Компонента связности
4	(2,3); (4,5); (8,9); (1,2)	2, 3, 1 | 4,5 | 8,9

3) если одна вершина рассматриваемого ребра входит в одну из имеющихся компонент связности, а вторая – в другую, то компоненты связности и массивы ребер объединяются.

Шаг	Массив ребер	Компонента связности
5	(2,3); (4,5); (8,9); (1,2); (2,5)	2, 3, 1, 4, 5 | 8,9

4) если обе вершины рассматриваемого ребра принадлежат одной компоненте связности, то ребро исключается из рассмотрения и в таблицу изменения не вносятся. В рассматриваемом примере такими ребрами являются (1,4), (5,8), (5,6) и (7,8).

Продолжим решение задачи при помощи алгоритма Краскала, учитывая перечисленные условия.

Шаг	Массив ребер	Компонента связности
6	(2,3); (4,5); (8,9); (1,2); (2,5); (6,9)	2, 3, 1, 4, 5 | 8, 9, 6
7	(2,3); (4,5); (8,9); (1,2); (2,5); (6,9) [(1,4)]	2, 3, 1, 4, 5 | 8, 9, 6
8	(2,3); (4,5); (8,9); (1,2); (2,5); (6,9); (3,6)	2, 3, 1, 4, 5, 8, 9, 6
9	(2,3); (4,5); (8,9); (1,2); (2,5); (6,9); (3,6) [(5,8)]	2, 3, 1, 4, 5, 8, 9, 6
10	(2,3); (4,5); (8,9); (1,2); (2,5); (6,9); (3,6); (4,7)	2, 3, 1, 4, 5, 8, 9, 6, 7
11	(2,3); (4,5); (8,9); (1,2); (2,5); (6,9); (3,6); (4,7) [(5,6)]	2, 3, 1, 4, 5, 8, 9, 6, 7
12	(2,3); (4,5); (8,9); (1,2); (2,5); (6,9); (3,6); (4,7) [(7,8)]	2, 3, 1, 4, 5, 8, 9, 6, 7

Примечание: в квадратных скобках указано рассматриваемое на данном шаге ребро, но не включаемое в массив ребер в силу выполнения условия 4.

В результате рассмотрения всех ребер отсортированного списка получим строку таблицы, определяющую минимальный остов графа (шаг 12).

Таким образом, минимальным остовом исходного графа является остов, состоящий из ребер (2,3); (1,2); (2,5); (4,5); (8,9); (6,9); (3,6); (4,7).

Задания: Найти минимальный остов графа, заданного списком ребер, при помощи алгоритма Краскала.

1. (1,2) – 3; (1,3) – 6; (1,4) – 2; (1,5) – 5; (2,3) – 4; (2,4) – 3; (2,5) – 1; (3,4) – 2; (3,5) – 5; (4,5) – 2;

2. (1,2) – 6; (1,6) – 5; (1,8) – 5; (2,3) – 4; (3,4) – 1; (3,8) – 3; (4,5) – 1; (4,7) – 2; (5,6) – 2; (6,7) – 4; (7,8) – 1;

3. (1,2) – 5; (1,3) – 3; (1,6) – 4; (2,3) – 6; (2,6) – 1; (3,4) – 2; (3,5) – 5; (4,5) – 1; (4,6) – 1; (5,6) – 4;

4. (1,2) – 1; (1,3) – 7; (1,6) – 5; (2,3) – 4; (2,6) – 2; (3,4) – 3; (3,5) – 6; (3,6) – 1; (4,5) – 2; (4,6) – 5; (5,6) – 3;

5. (1,2) – 1; (1,8) – 2; (2,3) – 2; (2,4) – 7; (2,8) – 3; (3,4) – 5; (4,5) – 5; (4,6) – 6; (5,6) – 3; (6,7) – 4; (6,8) – 1; (7,8) – 1;

6. (1,2) – 4; (1,3) – 5; (1,4) – 2; (1,5) – 3; (2,3) – 1; (2,4) – 2; (2,5) – 3; (3,4) – 5; (3,5) – 6; (4,5) – 7;

7. (1,2) – 4; (1,6) – 1; (1,8) – 6; (2,3) – 5; (3,4) – 2; (3,8) – 7; (4,5) – 2; (4,7) – 3; (5,6) – 3; (6,7) – 5; (7,8) – 5;

8. (1,2) – 1; (1,3) – 7; (1,6) – 5; (2,3) – 4; (2,6) – 2; (3,4) – 3; (3,5) – 6; (4,5) – 2; (4,6) – 5; (5,6) – 3;

9. (1,2) – 5; (1,3) – 3; (1,6) – 4; (2,3) – 6; (2,6) – 1; (3,4) – 2; (3,5) – 5; (3,6) – 2; (4,5) – 1; (4,6) – 1; (5,6) – 4;

10. (1,2) – 5; (1,8) – 1; (2,3) – 1; (2,4) – 3; (2,8) – 2; (3,4) – 1; (4,5) – 4; (4,6) – 5; (5,6) – 4; (6,7) – 6; (6,8) – 2; (7,8) – 2;

11. (1,2) – 5; (1,3) – 3; (1,4) – 7; (1,5) – 2; (2,3) – 2; (2,4) – 1; (2,5) – 2; (3,4) – 4; (3,5) – 5; (4,5) – 2;

12. (1,2) – 5; (1,6) – 2; (1,8) – 5; (2,3) – 3; (3,4) – 1; (3,8) – 2; (4,5) – 7; (4,7) – 2; (5,6) – 2; (6,7) – 4; (7,8) – 1;

13. (1,2) – 2; (1,3) – 2; (1,6) – 4; (2,3) – 5; (2,6) – 1; (3,4) – 2; (3,5) – 5; (4,5) – 7; (4,6) – 1; (5,6) – 3;

14. (1,2) – 3; (1,3) – 1; (1,6) – 4; (2,3) – 7; (2,6) – 2; (3,4) – 5; (3,5) – 4; (3,6) – 1; (4,5) – 3; (4,6) – 2; (5,6) – 1;

15. (1,2) – 3; (1,8) – 2; (2,3) – 3; (2,4) – 1; (2,8) – 5; (3,4) – 2; (4,5) – 4; (4,6) – 4; (5,6) – 1; (6,7) – 7; (6,8) – 1; (7,8) – 5;

16. (1,2) – 2; (1,3) – 4; (1,4) – 1; (1,5) – 1; (2,3) – 3; (2,4) – 5; (2,5) – 2; (3,4) – 3; (3,5) – 4; (4,5) – 6;

17. (1,2) – 2; (1,6) – 3; (1,8) – 4; (2,3) – 4; (3,4) – 5; (3,8) – 6; (4,5) – 1; (4,7) – 2; (5,6) – 1; (6,7) – 3; (7,8) – 1;

18. (1,2) – 3; (1,3) – 6; (1,6) – 3; (2,3) – 2; (2,6) – 5; (3,4) – 1; (3,5) – 4; (4,5) – 1; (4,6) – 1; (5,6) – 2;

19. (1,2) – 2; (1,3) – 2; (1,6) – 4; (2,3) – 5; (2,6) – 1; (3,4) – 2; (3,5) – 5; (3,6) – 3; (4,5) – 7; (4,6) – 1; (5,6) – 3;

20. (1,2) – 2; (1,8) – 1; (2,3) – 7; (2,4) – 2; (2,8) – 2; (3,4) – 1; (4,5) – 4; (4,6) – 5; (5,6) – 3; (6,7) – 5; (6,8) – 3; (7,8) – 4;

21. (1,2) – 7; (1,3) – 1; (1,4) – 3; (1,5) – 5; (2,3) – 3; (2,4) – 2; (2,5) – 2; (3,4) – 4; (3,5) – 4; (4,5) – 1;

22. (1,2) – 7; (1,6) – 3; (1,8) – 4; (2,3) – 1; (3,4) – 2; (3,8) – 1; (4,5) – 3; (4,7) – 2; (5,6) – 5; (6,7) – 4; (7,8) – 2;

23. (1,2) – 3; (1,3) – 1; (1,6) – 4; (2,3) – 7; (2,6) – 2; (3,4) – 5; (3,5) – 4; (4,5) – 3; (4,6) – 2; (5,6) – 1;

24. (1,2) – 2; (1,3) – 7; (1,6) – 4; (2,3) – 1; (2,6) – 3; (3,4) – 5; (3,5) – 6; (3,6) – 5; (4,5) – 4; (4,6) – 7; (5,6) – 2;

25. (1,2) – 2; (1,8) – 3; (2,3) – 4; (2,4) – 7; (2,8) – 5; (3,4) – 7; (4,5) – 4; (4,6) – 6; (5,6) – 2; (6,7) – 1; (6,8) – 5; (7,8) – 5;

26. (1,2) – 1; (1,3) – 2; (1,4) – 4; (1,5) – 5; (2,3) – 2; (2,4) – 3; (2,5) – 1; (3,4) – 4; (3,5) – 6; (4,5) – 7;

27. (1,2) – 1; (1,6) – 2; (1,8) – 6; (2,3) – 2; (3,4) – 3; (3,8) – 7; (4,5) – 4; (4,7) – 1; (5,6) – 5; (6,7) – 4; (7,8) – 7;

28. (1,2) – 2; (1,3) – 7; (1,6) – 4; (2,3) – 1; (2,6) – 3; (3,4) – 5; (3,5) – 6; (4,5) – 4; (4,6) – 7; (5,6) – 2;

29. (1,2) – 3; (1,3) – 1; (1,6) – 4; (2,3) – 7; (2,6) – 2; (3,4) – 5; (3,5) – 4; (3,6) – 5; (4,5) – 3; (4,6) – 2; (5,6) – 1;

30. (1,2) – 3; (1,8) – 2; (2,3) – 3; (2,4) – 1; (2,8) – 5; (3,4) – 2; (4,5) – 4; (4,6) – 4; (5,6) – 1; (6,7) – 7; (6,8) – 5; (7,8) – 5.

Сетевые алгоритмы. Выбор кратчайшего пути

Задача о нахождении кратчайшего пути имеет столько практических применений и интерпретаций, что важность её не нуждается в обсуждении. Ниже рассматриваются классические алгоритмы, которые должен знать каждый программист.

В дальнейшем мы используем понятие сети, оно нуждается в некотором уточнении, так как нет единого подхода в его понимании. Под *сетью* подразумевается просто связный ориентированный граф D (V, E), в котором, возможно, выделены вход и выход. Более узкое толкование термина «сеть» предполагает существование одного источника и одного стока, об этом, по мере необходимости мы будем говорить отдельно. Нужно отметить, что существуют и другие толкования термина. Отметим также, что в

ориентированном графе D (V, E) как и в неориентированном G (V, E) используется название «вершина», а не узел.

Нахождение кратчайшего пути в сети без контуров

Описание алгоритма

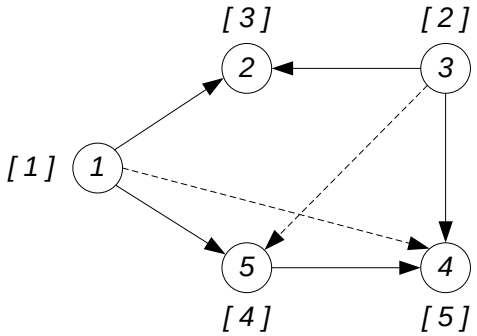
Этот алгоритм используется для решения задачи о нахождении кратчайшего пути в сети без контуров. Пусть задана сеть из $n + 1$ вершины, то есть ориентированный граф. В нем выделены две вершины – вход (нулевая вершина) и выход (вершина с номером n). Для каждой дуги заданы числа, называемые *длинами* дуг. *Длиной пути (контура)* называется сумма длин входящих в него дуг (если длины дуг не заданы, то длина пути (контура) определяется как число входящих в него дуг).

Задача заключается в поиске кратчайшего пути (пути минимальной длины) от входа до выхода сети. Будем предполагать, что в любую вершину сети можно попасть из входа, и из любой вершины можно попасть в выход (вершины, не удовлетворяющие этому требованию, можно удалить). Известно что, для существования кратчайшего пути необходимо и достаточно отсутствия в сети контуров отрицательной длины.

Предположим, что в сети нет контуров. Тогда всегда можно пронумеровать вершины таким образом, что для любой дуги (i, j) имеет место $j > i$. Такая нумерация называется правильной. В сети без контуров всегда существует правильная нумерация.

Если изначально в задаче дается неправильная (произвольная) нумерация вершин, то прежде чем использовать метод потенциалов нужно определить правильную нумерацию, используя алгоритм топологической сортировки и алгоритм Уоршалла.

Алгоритм Уоршалла заключается в вычислении транзитивного замыкания. Для того чтобы понять принцип алгоритма Уоршела рассмотрим пример. Пусть задан граф с произвольной нумерацией вершин. Составим для этого графа две матрицы смежности.



Матрица смежности для первоначального графа

	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	0	0
3	0	1	0	1	0
4	0	0	0	0	0
5	0	0	0	1	0

Матрица смежности с транзитивным замыканием

	1	2	3	4	5
1	0	1	0	1	1
2	0	0	0	0	0
3	0	1	0	1	1
4	0	0	0	0	0
5	0	0	0	1	0

Номер строки в матрице соответствует номеру исходящей вершины, а номер столбца – входящей. Единица на пересечении *i*-ой строки и *j*-го столбца ставится в том случае, если есть путь, выходящий из *i*-ой вершины и входящий в *j*-ую. Во все остальные ячейки матрицы заносятся нули.

Вторую матрицу с транзитивным замыканием можно построить с помощью исходной матрицы. Для этого сравниваем каждую строку с каждым столбцом, если в обоих есть хотя бы одна «1», то на их пересечении в ячейку заносится «1». При этом учитывается, что диагональ матрицы (из левого верхнего угла в правый нижний угол) всегда заполняется нулями, и что номер вершины, из которой идет дуга, должен быть меньше номера вершины, в которую она входит.

С помощью полученной матрице смежности с транзитивным замыканием можно достроить на графе дополнительные дуги: добавленные в матрицу единицы обозначают новые дуги в графе (на рисунке они обозначены пунктиром).

Алгоритм топологической сортировки – это алгоритм дополнения частичного порядка на конечном множестве. За основу берется матрица смежности, полученная алгоритмом Уоршалла. Рассматриваем в ней столбцы. Выбираем столбец, содержащий только нулевые элементы, и условно его вычеркиваем вместе со строкой этого же номера. Номер этого столбца записываем, причем вариантов выкидывания может быть несколько из-за того, что может оказаться несколько таких столбцов. Остается матрица меньшая предыдущей на столбец и строку, с ней проделывается туже операцию, и так до тех пока не останется один элемент, номер которого также записываем.

Каждый раз, записывая номер вычеркнутого столбца, получаем последовательность номеров вершин, которая идет не по порядку. Под ней выписываем номера по порядку.

1 – 3 – 2 – 5 – 4
1 – 2 – 3 – 4 – 5

Это означает, что номеру, произвольно поставленному у вершины, соответствует другой номер, полученный в результате топологической сортировки. Чтобы теперь получить правильную нумерацию, необходимо заменить первоначальные номера вершин на правильные (на рисунке правильная нумерация дается в квадратных скобках).

Замечание: алгоритм Уоршела и топологическая сортировка выполнены верно, если после проставлении правильной нумерации на графе, номер вершины, из которой идет дуга, меньше номера вершины, в которую она входит.

После того, как будет выбрана правильная нумерация вершин, можно использовать метод потенциалов для нахождения кратчайшего пути на графе. Он заключается в следующих шагах:

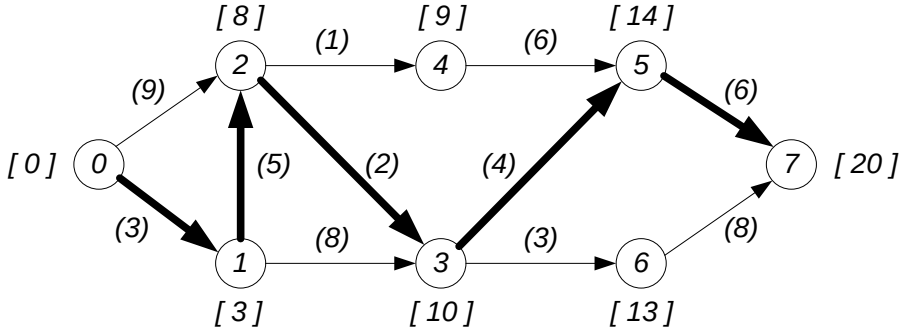
Шаг 0: помечаем нулевую вершину (вход) индексом $\lambda_0 = 0$;

Шаг k: помечаем вершину k индексом $\lambda_k = \min (\lambda_i + l_{ik})$.

Здесь используются следующие обозначения: l_{ij} – длина дуги (i, j); λ_n – потенциал вершины (индекс выхода), равный длине кратчайшего пути.

Кратчайший путь здесь ищется по принципу оптимальности Беллмана. Он формулируется так: если ищется кратчайший путь между двумя точками, то длина пути между любыми двумя точками кратчайшего пути также должна быть минимальна. Когда потенциалы установлены, кратчайший путь определяется методом обратного хода от выхода к входу, то есть кратчайшим является путь $\mu = (0; i_1; i_2; \ldots; i_{n-1}; n)$, такой, что $l_{in;1n} = \lambda_n - \lambda_{n-1}$ и т.д.

Пример: На рисунке приведен пример применения метода потенциалов для определения кратчайшего пути.



Задана сеть из 8 вершин, входом является вершина с номером 0, а выходом – вершина с номером 7. Числа у дуг равны длинам дуг. Нумерация вершин является правильной, т.к. для любой дуги (i, j) имеет место $j > i$.

Помечаем первую вершину (вход) индексом 0, т.е. потенциал этой вершины равен 0. Индексы вершин – потенциалы – на рисунке помещены в квадратные скобки.

Далее определяем потенциал для первой вершины. В первую вершину можно попасть только из нулевой, поэтому складываем потенциал этой вершины с длиной дуги между ними (то есть [0]+3=[3]). Потенциал первой вершины равен 3.

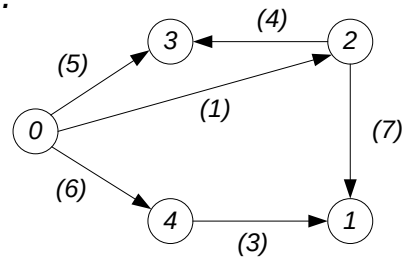
В отличие от первой во вторую вершину можно попасть либо из нулевой вершины, либо из первой. Определяем длины этих путей. В первом случае он будет равен 9 ([0]+9), а во втором – 8 (потенциал первой вершины + длина дуги между первой и второй вершиной, т.е. [3]+5). Из двух путей выбираем наименьший. Таким образом, потенциал второй вершины будет равен 8.

Определяя потенциал третьей вершины, рассматриваем пути только от ближайших вершин, то есть от первой и второй вершин. В первом случае путь равен 11, а во втором – 10. Соответственно потенциал 3-ей вершины равен 10.

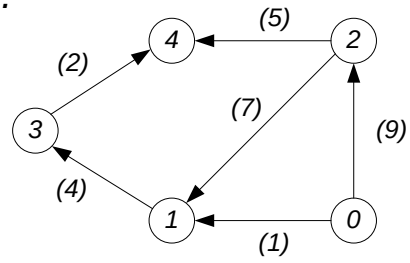
Далее аналогичным образом расставляем потенциалы всех остальных вершин в графе. После того как все потенциалы расставлены, методом обратного хода устанавливаем кратчайший путь. На рисунке он выделен жирными линиями.

Задания: Определить кратчайший путь в графе

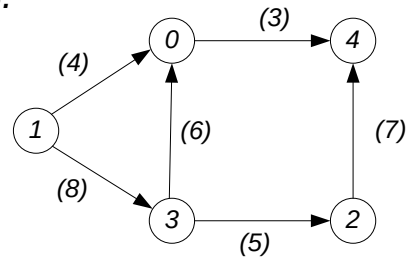
1.



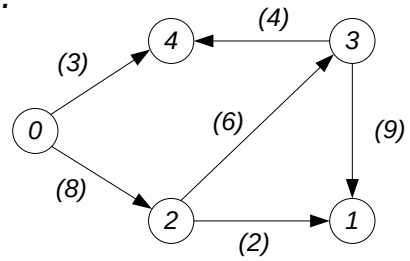
2.



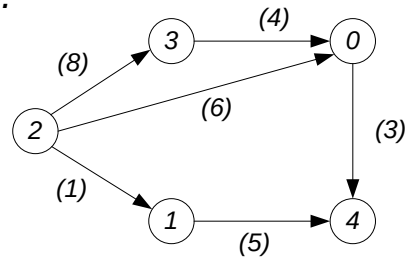
3.



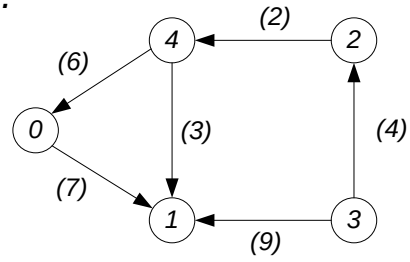
4.



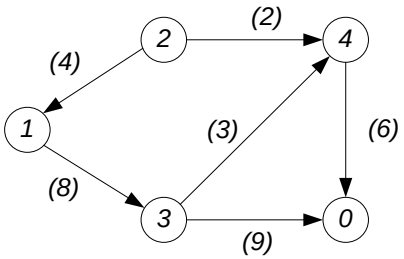
5.



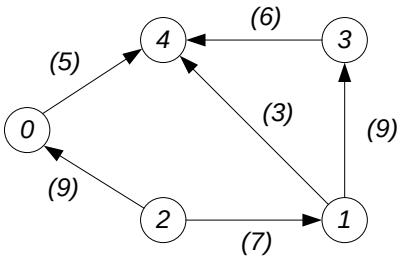
6.



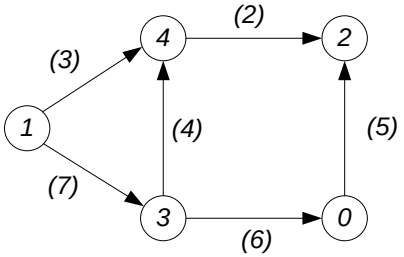
7.



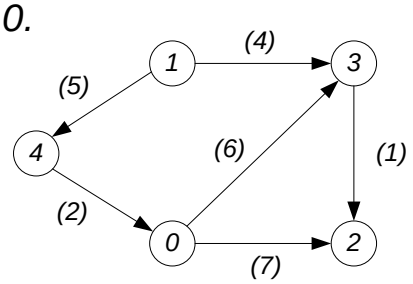
8.



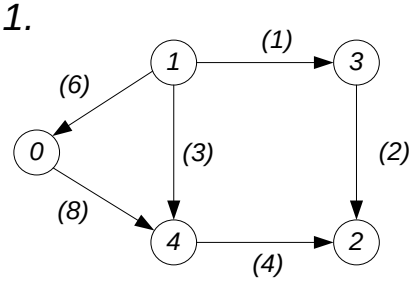
9.



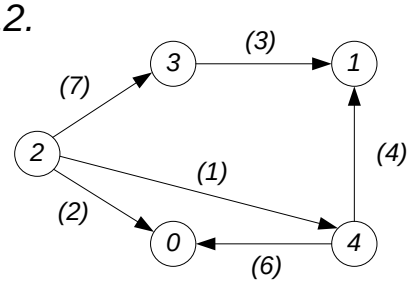
10.



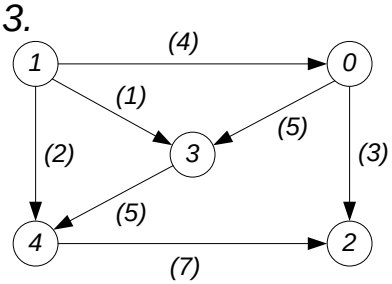
11.



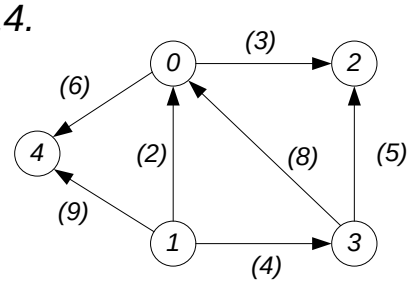
12.



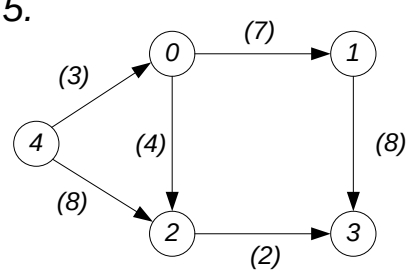
13.



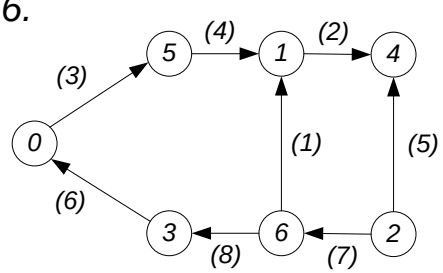
14.



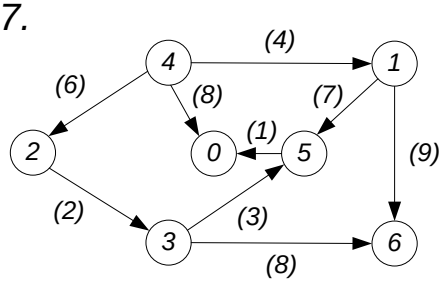
15.



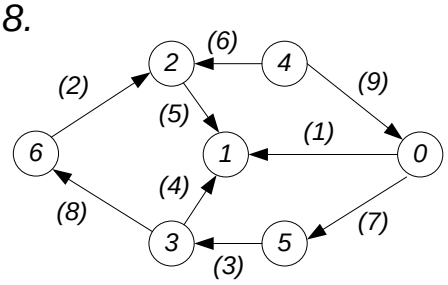
16.



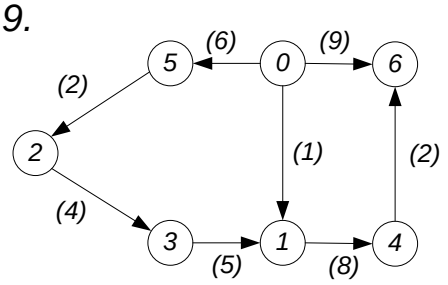
17.



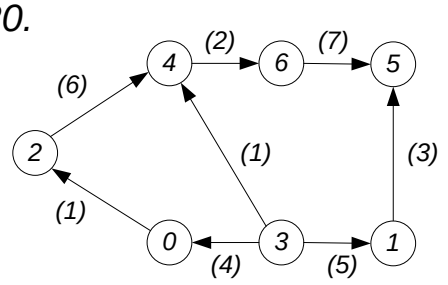
18.



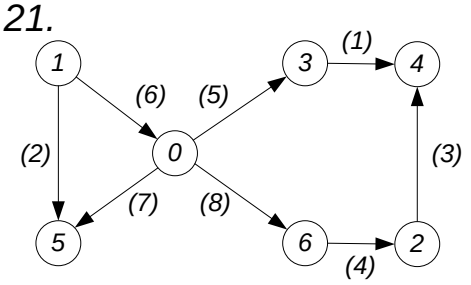
19.



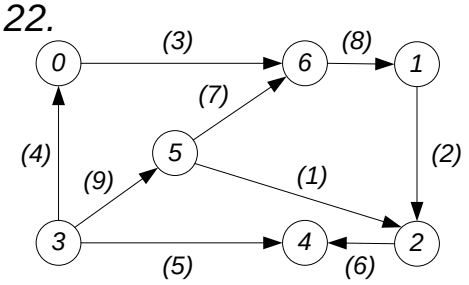
20.



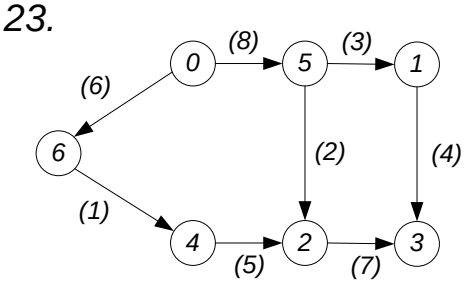
21.



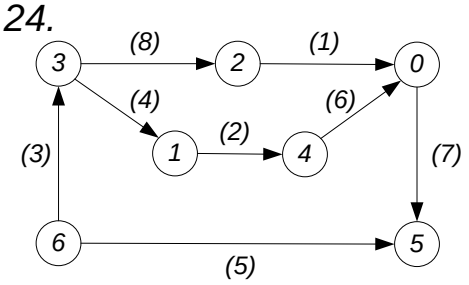
22.



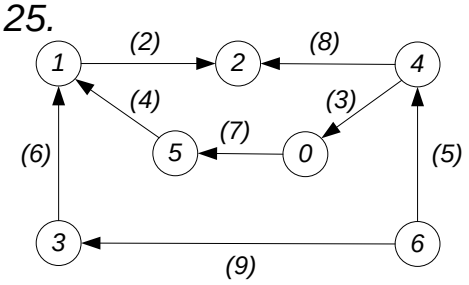
23.



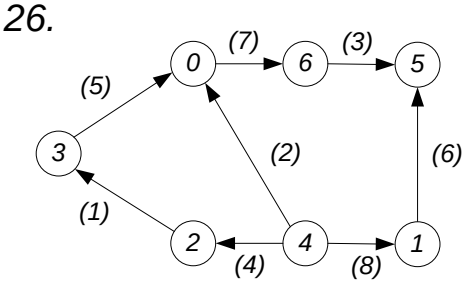
24.



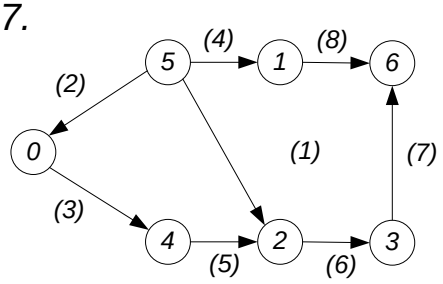
25.



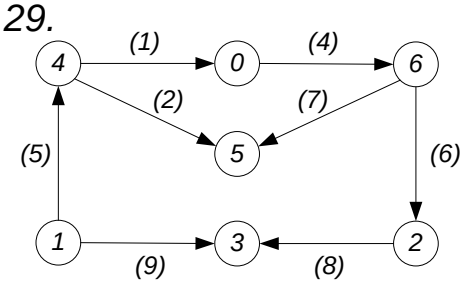
26.



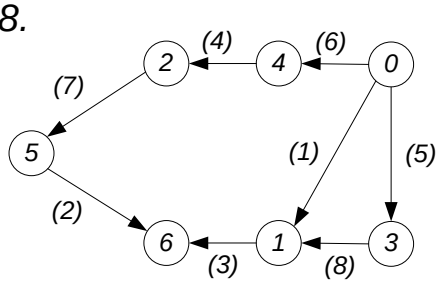
27.



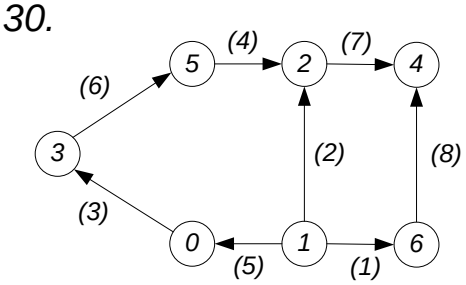
29.



28.



30.



Алгоритм Беллмана – Форда

Описание алгоритма

Предположим, что вершина 1 является узлом-входом и требуется найти длины кратчайших путей от вершины 1 до каждой другой вершины графа. Для этого алгоритма дуговые расстояния могут быть как положительными, так и отрицательными, но не должно быть циклов отрицательной длины. Предположим, что если в графе отсутствует та или иная дуга, то её вес равен бесконечно большому числу. Основная идея алгоритма Беллмана – Форда состоит в том, чтобы сначала найти длины кратчайших путей, при условии, что пути содержат не более двух дуг и т.д. Кратчайший путь при условии, что он содержит не более h дуг, будем называть кратчайшим путем в графе. Согласно идее Беллмана-Форда, Р.Прим предложил алгоритм нахождения длин кратчайших путей, ведущих из произвольной вершины графа во все остальные его вершины (в которые есть пути из вершины-входа), состоящий в присвоении вершинам некоторых потенциалов. Признаком окончания алгоритма является остановка процесса изменения потенциалов.

Алгоритм, предложенный Р. Примом, состоит в выполнении следующих операций:

1. вершине-входу присваиваем потенциал $\varphi_1 = 0$; остальным вершинам – потенциал $\varphi_i = \infty$, $i = 1 \ldots n$, где n - число вершин в рассматриваемом графе.

2. для каждой вершины i, смежной с вершиной j из массива вершин L, находим $\varphi_j + l\,(j, i)$ и проверяем неравенство:

$$\varphi_j + l\,(j, i) < \varphi_i$$

если неравенство выполняется, то

$$\varphi_i := \varphi_j + l(j, i)$$

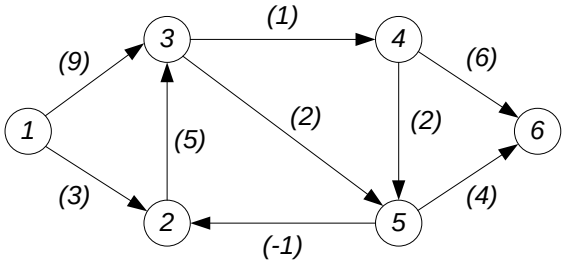
Вершину i заносим в массив L_1.

3. проверяем массив $L_1 \neq \varnothing$. Если неравенство выполняется $L := L_1$ и осуществляем переход к пункту 2.
4. конец алгоритма нахождения кратчайших путей.

Рассмотрим работу алгоритма на примере.

Пример

Допустим, имеется граф, представленный списком дуг:



(1,2) – 3 (4,6) – 6
(1,3) – 9 (3,5) – 2
(2,3) – 5 (5,2) – -1
(3,4) – 1 (5,6) – 4
(4,5) – 2 (6,0) – 0

Решение:

Результаты работы алгоритма по шагам вычислений сведем в таблицу.

№ шага вычислений	Массив L	Потенциалы вершин						Массив L_1
		1	2	3	4	5	6	
0	Ø	0	∞	∞	∞	∞	∞	1
1	1	0	3	9	∞	∞	∞	2, 3
2	2, 3	0	3	8	10	11	∞	3, 4, 5
3	3, 4, 5	0	3	8	9	10	15	4, 5, 6
4	4, 5, 6	0	3	8	9	10	14	6
5	6	0	3	8	9	10	14	Ø

На нулевом шаге производится инициализация согласно пункту 1 алгоритма. Далее на первом шаге вычислений рассматриваются вершины, смежные с вершиной-входом, потенциалы рассматриваемых вершин становятся равными длине соединяющей их дуги, если таковое существует, в противном случае – потенциал вершины остается равным ∞.

На третьем шаге рассматриваем вершины, достижимые из вершины-входа при помощи пути, состоящего из трех дуг. Таковыми являются:

1) 1 – 3 – 4 – 6: потенциал вершины 6 изменяется $\varphi_6 = 16$;
2) 1 – 2 – 3 – 5: потенциал вершины 5 изменяется $\varphi_5 = 10$;
3) 1 – 3 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 10$);

4) 1 – 3 – 4 – 5: потенциал вершины 5 не изменится ($\varphi_5 < 12$);
5) 1 – 3 – 5 – 6: потенциал вершины 6 изменится $\varphi_6 = 15$.

На четвертом шаге вычислений рассматриваем вершины, достижимые из вершины-входа за путь, состоящий из четырех ребер. Таковыми являются пути:

1) 1 – 2 – 3 – 5 – 6: потенциал вершины 6 изменится $\varphi_6 = 14$;
2) 1 – 2 – 3 – 4 – 5: потенциал вершины 5 не изменится ($\varphi_5 < 11$);
3) 1 – 2 – 3 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 9$);
4) 1 – 3 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 15$);
5) 1 – 3 – 4 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 11$).

На пятом шаге вычислений рассматриваются вершины, достижимые из вершины-входа за путь, состоящий из пяти дуг:

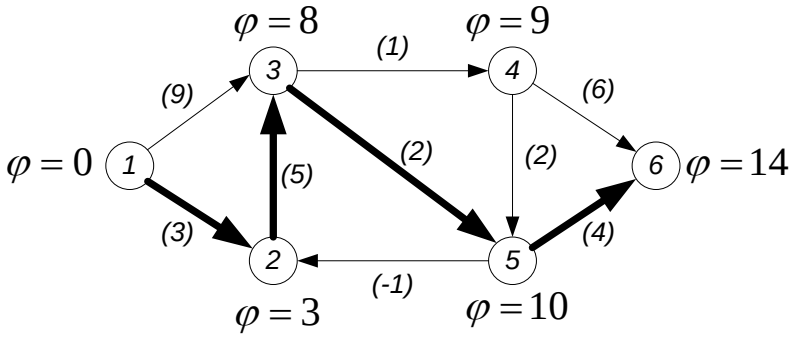
1) 1 – 3 – 5 – 2 – 3 – 5: потенциал вершины 5 не изменится ($\varphi_5 < 17$);
2) 1 – 3 – 5 – 2 – 3 – 4: потенциал вершины 4 не изменится ($\varphi_4 < 16$);
3) 1 – 3 – 4 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 14$);
4) 1 – 2 – 3 – 4 – 5 – 2: потенциал вершины 2 не изменится ($\varphi_2 < 10$);
5) 1 – 2 – 3 – 4 – 5 – 6: потенциал вершины 6 не изменится ($\varphi_6 = 14$);
6) 1 – 2 – 3 – 5 – 2 – 3: потенциал вершины 3 не изменится ($\varphi_3 < 14$).

На пятом шаге вычислений не произошло изменений ни в одном из потенциалов вершин графа, делаем вывод об окончании работы алгоритма.

Восстановим кратчайший путь из вершины 6 в вершину 1. Для этого для каждой вершины *j* определим такую вершину *i*, что будет выполняться правило:

$$\varphi_i := \varphi_j + l(j,i)$$

Следуя правилу, последовательно находим кратчайший путь:
вершина 6: $i = 5$; вершина 5: $i = 3$; вершина 3: $i = 2$; вершина 2: $i = 1$.



Кратчайшим путем является путь 1 – 2 – 3 – 5 – 6.

Задания

Дан список рёбер. Составьте по нему рисунок ориентированного графа. Найдите для этого графа наименьший путь от вершины-входа до вершины с максимальным номером.

1. (0;1) – 3, (0;2) – 9, (1;2) – 5, (2;4) – 1, (1;3) – 8, (2;3) – 2, (3;5) – 4, (4;5) – 6.

2. (0;1) – 4, (0;2) – 5, (1;2) – 8, (2;4) – 3, (1;3) – 11, (2;3) – 5, (3;5) – 3, (4;5) – 6.

3. (0;1) – 3, (0;2) – 9, (1;2) – 12, (2;4) – 1, (1;3) – 2 , (2;3) – 3, (3;5) – 10, (4;5) – 5.

4. (0;1) – 6, (0;2) – 2, (2;1) – 3, (2;4) – 6, (1;3) – 1, (2;3) – 5, (3;5) – 8, (4;5) – 7.

5. (0;1) – 6, (0;2) – 5, (1;2) – 1, (2;4) – 6, (1;3) – 7, (2;3) – 6, (3;5) – 8, (4;5) – 7.

6. (0;1) – 3, (0;2) – 2, (2;1) – 1, (2;5) – 3, (1;5) – 4, (5;4) – 8, (5;3) – 5, (3;4) – 3, (4;6) – 2, (3;6) – 4.

7. (0;1) – 10, (0;2) – 5, (2;1) – 4, (2;5) – 8, (1;5) – 3, (5;4) – 4, (5;3) – 2, (3;4) – 1, (4;6) – 5, (3;6) – 7.

8. (0;1) – 3, (0;2) – 2, (2;1) – 2, (2;5) – 12, (1;5) – 8, (5;4) – 2, (5;3) – 6, (3;4) – 1, (4;6) – 8, (3;6) – 3.

9. (0;1) – 2, (0;2) – 7, (2;1) – 1, (2;5) – 6, (1;5) – 12, (5;4) – 10, (5;3) – 5, (3;4) – 4, (4;6) – 2, (3;6) – 7.

10. (0;1) – 4, (0;2) – 2, (2;1) – 1, (2;5) – 7, (1;5) – 5, (5;4) – 4, (5;3) – 1, (3;4) – 4, (4;6) – 3, (3;6) – 7.

11. (0;2) – 2, (0;1) – 7, (2;1) – 4, (2;4) – 9, (1;3) – 3, (3;4) – 1, (4;6) – 2, (3;5) – 8, (6;5) – 4, (6;7) – 10, (5;7) – 5.

12. (0;2) – 10, (0;1) – 5, (2;1) – 1, (2;4) – 4, (1;3) – 3, (3;4) – 5, (4;6) – 3, (3;5) – 10, (6;5) – 10, (6;7) – 5, (5;7) – 1.

13. (0;2) – 4, (0;1) – 6, (2;1) – 4, (2;4) – 6, (1;3) – 3, (3;4) – 2, (4;6) – 4, (3;5) – 7, (6;5) – 3, (6;7) – 8, (5;7) – 5.

14. (0;2) – 3, (0;1) – 1, (2;1) – 4, (2;4) – 2, (1;3) – 6, (3;4) – 4, (4;6) – 3, (3;5) – 6, (6;5) – 4, (6;7) – 12, (5;7) – 7.

15. (0;2) – 8, (0;1) – 12, (2;1) – 3, (2;4) – 6, (1;3) – 5, (3;4) – 4, (4;6) – 10, (3;5) – 4, (6;5) – 6, (6;7) – 10, (5;7) – 6.

16. (0;1) – 3, (0;2) – 3, (1;4) – 3, (2;5) – 3, (1;3) – 2, (0;3) – 4, (2;3) – 2, (3;4) – 2, (3;6) – 4, (3;5) – 2, (4;6) – 3, (5;6) – 3.

17. (0;1) – 3, (0;2) – 2, (1;4) – 13, (2;5) – 13, (1;3) – 7, (0;3) – 11, (2;3) – 9, (3;4) – 5, (3;6) – 10, (3;5) – 3, (4;6) – 4, (5;6) – 7.

18. (0;1) – 4, (0;2) – 5, (1;4) – 7, (2;5) – 7, (1;3) – 5, (0;3) – 8, (2;3) – 4, (3;4) – 2, (3;6) – 7, (3;5) – 1, (4;6) – 4, (5;6) – 6.

19. (0;1) – 5, (0;2) – 5, (1;4) – 4, (2;5) – 5, (1;3) – 3, (0;3) – 10, (2;3) – 3, (3;4) – 3, (3; 6) – 10, (3;5) – 3, (4;6) – 7, (5;6) – 5.

20. (0;1) – 6, (0;2) – 7, (1;4) – 18, (2;5) – 19, (1;3) – 8, (0;3) – 14, (2;3) – 6, (3;4) – 8, (3;6) – 14, (3;5) – 5, (4;6) – 5, (5;6) – 9.

21. (0;1) – 2, (1;2) – 3, (0;2) – 6, (2;3) – 4, (3;4) – 2, (2;4) – 7, (4;5) – 5, (5;6) – 3, (4;6) – 9.

22. (0;1) – 3, (1;2) – 5, (0;2) – 7, (2;3) – 6, (3;4) – 5, (2;4) – 12, (4;5) – 3, (5;6) – 2, (4;6) – 4.

23. (0;1) – 5, (1;2) – 4, (0;2) – 10, (2;3) – 4, (3;4) – 5, (2;4) – 8, (4;5) – 7, (5;6) – 6, (4;6) – 14.

24. (0;1) – 2, (1;2) – 4, (0;2) – 5, (2;3) – 4, (3;4) – 5, (2;4) – 8, (4;5) – 3, (5;6) – 2, (4;6) – 4.

25. (0;1) – 3, (1;2) – 2, (0;2) – 5, (2;3) – 1, (3;4) – 1, (2;4) – 3, (4;5) – 2, (5;6) – 2, (4;6) – 3.

26. (0;1) – 10, (0;2) – 4, (1;4) – 8, (2;5) – 20, (3;1) – 5, (4;3) – 3, (2;3) – 4, (3;5) – 17, (4;6) – 7, (5;6) – 5.

27. (0;1) – 6, (0;2) – 2, (1;4) – 4, (2;5) – 15, (3;1) – 2, (4;3) – 3, (2;3) – 13, (3;5) – 2, (4;6) – 7, (5;6) – 1.

28. (0;1) – 2, (0;2) – 3, (1;4) – 1, (2;5) – 10, (3;1) – 1, (4;3) – 6, (2;3) – 4, (3;5) – 5, (4;6) – 20, (5;6) – 6.

29. (0;1) – 2, (0;2) – 3, (1;4) – 7, (2;5) – 6, (3;1) – 1, (4;3) – 2, (2;3) – 2, (3;5) – 3, (4;6) – 8, (5;6) – 10.

30. (0;1) – 3, (0;2) – 7, (1;4) – 8, (2;5) – 3, (3;1) – 1, (4;3) – 4, (2;3) – 2, (3;5) – 2, (4;6) – 7, (5;6) – 6.

Алгоритм Дейкстра

Описание алгоритма

Алгоритм Дейкстра требует, чтобы длины всех дуг были положительны. Объем вычислений в худшем случае для этого алгоритма значительно меньше, чем у алгоритма Беллмана – Форда. Основная его идея состоит в том, чтобы отыскивать кратчайшие пути в порядке возрастания длины пути. Кратчайшим среди всех кратчайших путей от вершины-входа является путь, состоящий из одной дуги, соединяющий вершину-вход с ближайшим соседним узлом, так как любой путь, состоящий из нескольких дуг, будет всегда длиннее первой дуги вследствие предположения о положительности всех дуговых длин. Следующим кратчайшим среди кратчайших путей должен быть ли путь из одной дуги к следующему ближайшему соседу вершины-входа, либо кратчайший путь из двух дуг, проходящий через вершину, выбранный на первом шаге и т.д. Алгоритм Дейкстра состоит в выполнении следующих операций:

Шаг 0. Помечаем нулевую вершину индексом $\lambda_0 = 0$;

Шаг k: Пусть уже помечено некоторое количество вершин. Обозначим Q – множество непомеченных вершин, смежных с помеченными. Для каждой вершины k принадлежащей Q вычисляем величину $\xi_k = \min (\lambda_k + l_{ki})$, где минимум берется по всем помеченным вершинам i, смежным с вершиной k. Помечаем вершину k, для которой величина ξ_k минимальна, индексом $\lambda_k = \xi_k$. Подобную процедуру повторяем до тех пор, пока не будет помечена вершина n. Длина кратчайшего пути равна λ_n, а сам кратчайший путь определяется так, как это было описано выше.

Пример: Найдем кратчайший путь из вершины 1 в вершину 7 (рис. 1).

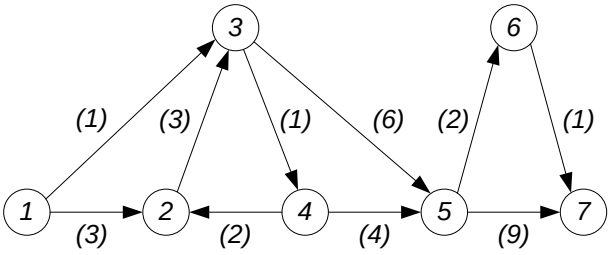


Рисунок 1

1) Пометим вершину 1 индексом $\lambda_1 = 0$.
2) Смежными с 1 являются вершины 2 и 3. $\xi_2 = 0 + 3 = 3$. $\xi_3 = 0 + 1 = 1$. Величина ξ_3 минимальна. Помечаем вершину 3 - $\lambda_3 = 1$. Помечены вершины 1 и 3.
3) Смежными с помеченными вершинами (1 и 3) являются 2, 4 и 5. $\xi_2 = 0 + 3 = 3$. $\xi_4 = 1 + 1 = 2$. $\xi_5 = 1 + 6 = 7$. Величина ξ_4 минимальна. Помечаем вершину индексом $\lambda_4 = 2$. Помечены вершины 1, 3 и 4.
4) Смежными с помеченными вершинами (1, 3 и 4) являются 2 и 5. $\xi_5 = 2 + 4 = 6$. $\xi_5 = 1 + 6 = 7$. $\xi_2 = 0 + 3 = 3$. $\xi_2 = 2 + 2 = 4$. Помечаем вершину $\lambda_2 = 3$. Помечены вершины 1, 2, 3, 4.
5) Смежной с помеченными вершинами (1, 2, 3, 4) является вершина 5. $\xi_5 = 2 + 4 = 6$. $\xi_5 = 1 + 6 = 7$. Помечаем 5 индексом $\lambda_5 = 6$. Помечены вершины 1, 2, 3, 4 и 5.
6) Смежными с помеченными вершинами (1, 2, 3, 4, 5) являются 6 и 7. $\xi_6 = 6 + 2 = 8$. $\xi_7 = 6 + 9 = 15$. Величина ξ_6 минимальна для вершины 6. Помечаем вершину $\lambda_6 = 8$.
7) Смежная с помеченными вершинами (1, 2, 3, 4, 5, 6) – вершина 7. $\xi_7 = 6 + 9 = 15$. $\xi_7 = 8 + 1 = 9$. Помечаем вершину индексом $\lambda_7 = 9$.
8) Для 6 вершины смежной является вершина 7. $\xi_7 = 8 + 1 = 9$. Величина ξ_7 минимальна. Помечаем вершину 7 индексом $\lambda_7 = 9$.

С помощью алгоритма Дейкстра мы получили длину кратчайшего пути из вершины 1 в вершину 7, которая составит 9. Кратчайшим путем является путь 1 – 3 – 4 – 5 – 6 – 7 (рис. 2).

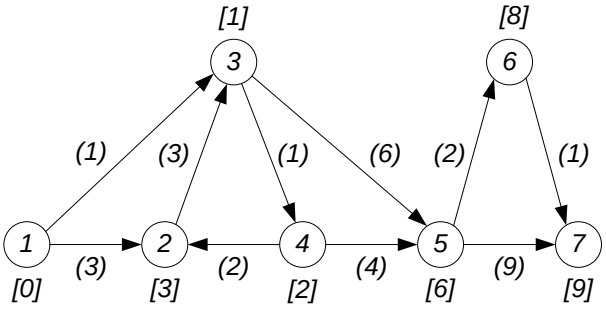


Рисунок 2

Задания

Для орграфа дана матрица весов. В позиции (i, j) записана длина дуги из вершины i в вершину j (ноль означает, что пути из i в j не существует). Если согласно матрице путь $i \to j$ существует, то путь $j \to i$ не существует. Задание: найти кратчайший путь из вершины 1 в вершину 5.

1. $\begin{pmatrix} 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

2. $\begin{pmatrix} 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$

3. $\begin{pmatrix} 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$

4. $\begin{pmatrix} 0 & 5 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$

5. $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

6. $\begin{pmatrix} 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$

7. $\begin{pmatrix} 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 2 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$

8. $\begin{pmatrix} 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$

9. $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

10. $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

11. $\begin{pmatrix} 0 & 0 & 0 & 3 & 9 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

12. $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{pmatrix}$

13. $\begin{pmatrix} 0 & 1 & 0 & 1 & 9 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

14. $\begin{pmatrix} 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

15. $\begin{pmatrix} 0 & 9 & 0 & 2 & 0 \\ 0 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

16. $\begin{pmatrix} 0 & 9 & 0 & 9 & 0 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

17. $\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix}$

18. $\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix}$

19. $\begin{pmatrix} 0 & 1 & 0 & 7 & 0 \\ 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 4 & 0 & 0 & 0 \end{pmatrix}$

20. $\begin{pmatrix} 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

21. $\begin{pmatrix} 0 & 4 & 2 & 0 & 0 \\ 0 & 0 & 0 & 4 & 7 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 24. $\begin{pmatrix} 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$ 27. $\begin{pmatrix} 0 & 0 & 0 & 1 & 9 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 30. $\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

22. $\begin{pmatrix} 0 & 1 & 2 & 0 & 3 \\ 0 & 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 25. $\begin{pmatrix} 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 5 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$ 28. $\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 3 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

23. $\begin{pmatrix} 0 & 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}$ 26. $\begin{pmatrix} 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 1 & 0 & 0 & 2 & 6 \\ 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 29. $\begin{pmatrix} 0 & 0 & 0 & 1 & 9 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 5 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Алгоритм Флойда – Уоршалла

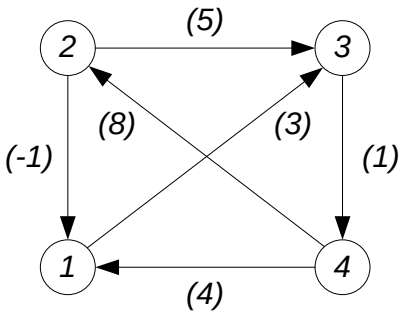
Описание алгоритма

Алгоритм Флойда находит кратчайшие пути между всеми парами вершин в (ор) графе. В отличии от алгоритма Дейкстра и алгоритма Беллмана – Форда находит кратчайшие пути сразу для всех пар вершин. Как и в алгоритме Беллмана – Форда, дуговые расстояния могут быть как положительными, так и отрицательными, но также не должно быть циклов отрицательной длины. Во всех трех алгоритмах решение находится методом итераций, но в каждом алгоритме итерируются разные величины. Если в алгоритме Беллмана – Форда итерируется число дуг в пути, в алгоритме Дейкстра – длина пути, то в алгоритме Флойда – Уоршалла итерируется множество вершин, которые допускается иметь в качестве промежуточных узлов на путях. Как и оба других алгоритма, алгоритм Флойда – Уоршалла начинает с расстояний одной дуги (то есть без промежуточных узлов), выбранных в качестве исходных оценок для длин кратчайших путей. Затем вычисляются кратчайшие пути с тем ограничением, что промежуточной вершиной может быть только вершина 1, затем с ограничением, что промежуточными вершинами могут быть только вершины 1 и 2 и т.д.

Этот алгоритм в его «чистом» варианте абсолютно не подходит для нахождения кратчайших путей вручную, так как он весьма трудоёмок (скажем, для графа 4×4 мы имеем уже 64 итерации). Поэтому следует по-

нять, на каких итерациях матрица не меняется и затем попросту исключить их из рассмотрения.

Пример



Условно процедуру нахождения кратчайшего пути в орграфе можно разбить на несколько шагов.

1) Инициализация. На данном этапе мы формируем матрицу C [1..p,1..p] и «предварительный вариант» матрицы H. Элемент матрицы C_{ij} равен длине дуги из вершины i в вершину j, а если такой дуги нет, то он равен бесконечности.

Соответствующие элементы матрицы H равны вершине, в которую проведена дуга, длина которой указана в матрице C:

$$T_{ij} = C_{ij} = \begin{pmatrix} 0 & \infty & 3 & \infty \\ -1 & 0 & 5 & \infty \\ \infty & \infty & 0 & 1 \\ 4 & 8 & \infty & 0 \end{pmatrix}; \quad H_{ij} = \begin{pmatrix} 0 & 0 & 3 & 0 \\ 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \\ 1 & 2 & 0 & 0 \end{pmatrix}$$

2) «Вычёркиваем» первый столбец и первую строку матрицы T. Затем ищем в данной строке и данном столбце ∞ и «вычёркиваем» соответственно столбец или строку, где ∞ была найдена. Видим, что остаётся два элемента: 5 и ∞.

T_{ij} =

0	∞	3	∞
−1	0	**5**	∞
∞	∞	0	1
4	8	**∞**	0

3) Проверяем оставшиеся элементы t_{ij} на неравенство $t_{ij} > t_{1j} + t_{i1}$. Если оно выполняется, то записываем на место проверяемого элемента сумму $t_{1j} + t_{i1}$ и параллельно пишем в соответствующую ячейку матрицы H номер вершины из её первого столбца, а если нет, то оставляем элемент без изменений.

T_{ij} =

0	∞	(3)	∞
(−1)	0	**2**	∞
∞	∞	0	1
(4)	8	**7**	0

$$H_{ij} = \begin{pmatrix} 0 & 0 & 3 & 0 \\ (1) & 0 & \boxed{1} & 0 \\ 0 & 0 & 0 & 4 \\ (1) & 2 & \boxed{1} & 0 \end{pmatrix}$$

4) Повторяем шаги 1-3 для второго столбца и второй строки, третьего столбца и третьей строки, и т. д. до p-го столбца и p-ой строки:

i	*Исходные данные на i-том шаге*	*Результат на i-том шаге*
2	$T_{ij} = \begin{pmatrix} 0 & \infty & 3 & \infty \\ -1 & 0 & 2 & \infty \\ \infty & \infty & 0 & 1 \\ 4 & 8 & 7 & 0 \end{pmatrix}$ $H_{ij} = \begin{pmatrix} 0 & 0 & 3 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 4 \\ 1 & 2 & 1 & 0 \end{pmatrix}$	
3	$T_{ij} = \begin{pmatrix} 0 & \infty & 3 & \infty \\ -1 & 0 & 2 & \infty \\ \infty & \infty & 0 & 1 \\ 4 & 8 & 7 & 0 \end{pmatrix} \rightarrow$ $H_{ij} = \begin{pmatrix} 0 & 0 & 3 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 4 \\ 1 & 2 & 1 & 0 \end{pmatrix} \rightarrow$	$T_{ij} = \begin{pmatrix} 0 & \infty & 3 & 4 \\ -1 & 0 & 2 & 3 \\ \infty & \infty & 0 & 1 \\ 4 & 8 & 7 & 0 \end{pmatrix}$ $H_{ij} = \begin{pmatrix} 0 & 0 & 3 & 3 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 4 \\ 1 & 2 & 1 & 0 \end{pmatrix}$
4	$T_{ij} = \begin{pmatrix} 0 & \infty & 3 & 4 \\ -1 & 0 & 2 & 3 \\ \infty & \infty & 0 & 1 \\ 4 & 8 & 7 & 0 \end{pmatrix} \rightarrow$ $H_{ij} = \begin{pmatrix} 0 & 0 & 3 & 3 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 4 \\ 1 & 2 & 1 & 0 \end{pmatrix} \rightarrow$	$T_{ij} = \begin{pmatrix} 0 & 12 & 3 & 4 \\ -1 & 0 & 2 & 3 \\ 5 & 9 & 0 & 1 \\ 4 & 8 & 7 & 0 \end{pmatrix}$ $H_{ij} = \begin{pmatrix} 0 & 3 & 3 & 3 \\ 1 & 0 & 1 & 1 \\ 4 & 4 & 0 & 4 \\ 1 & 2 & 1 & 0 \end{pmatrix}$

5) Таким образом, на выходе мы получаем, как и указано в алгоритме, две матрицы:

$$T_{ij} = \begin{pmatrix} 0 & 12 & 3 & 4 \\ -1 & 0 & 2 & 3 \\ 5 & 9 & 0 & 1 \\ 4 & 8 & 7 & 0 \end{pmatrix} \quad \text{и} \quad H_{ij} = \begin{pmatrix} 0 & 3 & 3 & 3 \\ 1 & 0 & 1 & 1 \\ 4 & 4 & 0 & 4 \\ 1 & 2 & 1 & 0 \end{pmatrix}.$$

Каждый элемент t_{ij} матрицы T содержит длину кратчайшего пути из вершины i в вершину j, а каждый элемент h_{ij} матрицы H содержит номер первой вершины, проходимой нами при следовании по этому пути.

Задания

Для орграфа дана матрица весов. В позиции (*i, j*) записана длина дуги из вершины *i* в вершину *j* (∞ означает, что пути из *i* в *j* не существует). Задание: нарисовать изображение графа, составить матрицы кратчайших расстояний и маршрутов.

1. $\begin{pmatrix} 0 & 9 & \infty & 3 & \infty \\ \infty & 0 & 3 & \infty & 1 \\ \infty & \infty & 0 & \infty & 2 \\ \infty & \infty & 4 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

2. $\begin{pmatrix} 0 & \infty & \infty & 6 & 1 \\ \infty & 0 & \infty & \infty & 3 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 2 & 8 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

3. $\begin{pmatrix} 0 & 1 & \infty & 1 & \infty \\ \infty & 0 & 7 & \infty & \infty \\ \infty & \infty & 0 & \infty & 3 \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

4. $\begin{pmatrix} 0 & 9 & \infty & 8 & \infty \\ \infty & 0 & 1 & \infty & 4 \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 5 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

5. $\begin{pmatrix} 0 & \infty & \infty & 1 & 9 \\ \infty & 0 & \infty & \infty & 3 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 3 & 8 & 0 & 2 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

6. $\begin{pmatrix} 0 & 5 & \infty & \infty & \infty \\ \infty & 0 & \infty & 5 & \infty \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 5 & 0 & \infty \\ 3 & \infty & \infty & \infty & 0 \end{pmatrix}$

7. $\begin{pmatrix} 0 & 3 & \infty & \infty & \infty \\ \infty & 0 & 2 & 6 & \infty \\ \infty & \infty & 0 & 5 & \infty \\ \infty & \infty & \infty & 0 & 1 \\ \infty & 1 & \infty & \infty & 0 \end{pmatrix}$

8. $\begin{pmatrix} 0 & \infty & 1 & 2 & \infty \\ \infty & 0 & \infty & \infty & 8 \\ \infty & \infty & 0 & \infty & 1 \\ \infty & 5 & 9 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

9. $\begin{pmatrix} 0 & 5 & \infty & \infty & \infty \\ \infty & 0 & 1 & 4 & \infty \\ \infty & \infty & 0 & \infty & 2 \\ \infty & \infty & 2 & 0 & \infty \\ 3 & \infty & \infty & \infty & 0 \end{pmatrix}$

10. $\begin{pmatrix} 0 & 1 & \infty & \infty & 1 \\ \infty & 0 & 3 & 4 & \infty \\ \infty & \infty & 0 & 5 & \infty \\ \infty & \infty & \infty & 0 & 1 \\ \infty & 1 & \infty & \infty & 0 \end{pmatrix}$

11. $\begin{pmatrix} 0 & \infty & \infty & 1 & 9 \\ \infty & 0 & \infty & \infty & 4 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 5 & 5 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

12. $\begin{pmatrix} 0 & 2 & \infty & 3 & \infty \\ \infty & 0 & \infty & 4 & \infty \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 3 & 0 & \infty \\ 3 & \infty & \infty & \infty & 0 \end{pmatrix}$

13. $\begin{pmatrix} 0 & 3 & \infty & \infty & \infty \\ \infty & 0 & 4 & \infty & \infty \\ 2 & \infty & 0 & 2 & 4 \\ 3 & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

14. $\begin{pmatrix} 0 & 9 & \infty & \infty & \infty \\ \infty & 0 & \infty & 7 & 2 \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 2 & 0 & \infty \\ 8 & \infty & \infty & \infty & 0 \end{pmatrix}$

15. $\begin{pmatrix} 0 & \infty & 2 & 5 & \infty \\ \infty & 0 & \infty & 4 & \infty \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 2 & 0 & \infty \\ 6 & \infty & \infty & \infty & 0 \end{pmatrix}$

16. $\begin{pmatrix} 0 & 1 & 3 & \infty & \infty \\ \infty & 0 & \infty & 4 & 1 \\ \infty & \infty & 0 & 2 & \infty \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

21. $\begin{pmatrix} 0 & \infty & 1 & 2 & \infty \\ \infty & 0 & \infty & \infty & 5 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & 3 & \infty & 0 & \infty \\ 3 & \infty & \infty & \infty & 0 \end{pmatrix}$

26. $\begin{pmatrix} 0 & 1 & \infty & 7 & \infty \\ \infty & 0 & 2 & 3 & \infty \\ \infty & \infty & 0 & 5 & \infty \\ \infty & \infty & \infty & 0 & 1 \\ \infty & 4 & \infty & \infty & 0 \end{pmatrix}$

17. $\begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ \infty & 0 & 3 & 1 & \infty \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

22. $\begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty \\ \infty & \infty & 0 & 5 & 2 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 3 & 0 \end{pmatrix}$

27. $\begin{pmatrix} 0 & 1 & \infty & 1 & 9 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 3 \\ \infty & \infty & 2 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

18. $\begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ \infty & 0 & 9 & \infty & \infty \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

23. $\begin{pmatrix} 0 & 1 & \infty & 2 & \infty \\ \infty & 0 & 3 & \infty & 1 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & 1 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

28. $\begin{pmatrix} 0 & 1 & 2 & \infty & 3 \\ \infty & 0 & \infty & 4 & 3 \\ \infty & \infty & 0 & 3 & \infty \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

19. $\begin{pmatrix} 0 & \infty & \infty & 2 & \infty \\ \infty & 0 & \infty & \infty & 5 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & 7 & \infty & 0 & \infty \\ 3 & \infty & \infty & \infty & 0 \end{pmatrix}$

24. $\begin{pmatrix} 0 & 1 & \infty & \infty & \infty \\ \infty & 0 & 3 & \infty & 3 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

29. $\begin{pmatrix} 0 & 1 & \infty & 2 & \infty \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 4 \\ \infty & 3 & \infty & 0 & 5 \\ 3 & \infty & \infty & \infty & 0 \end{pmatrix}$

20. $\begin{pmatrix} 0 & \infty & \infty & 2 & 9 \\ \infty & 0 & \infty & 6 & \infty \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 5 & 0 & 2 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

25. $\begin{pmatrix} 0 & \infty & \infty & 3 & \infty \\ \infty & 0 & \infty & 4 & \infty \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 2 & 0 & 2 \\ 3 & \infty & \infty & \infty & 0 \end{pmatrix}$

30. $\begin{pmatrix} 0 & 4 & 2 & \infty & \infty \\ \infty & 0 & \infty & 4 & 7 \\ \infty & \infty & 0 & 3 & \infty \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$

Алгоритм Форда – Фалкерсона

Описание алгоритма

В этом разделе под сетью мы понимаем связный орграф D (V, E), у которого есть одна вершина s со степенью захода 0 (источник) и одна вершина t со степенью исхода 0 (сток). Дуги сети нагружены неотрицатель-

ными вещественными числами, то есть, задана функция c: $E \to R$. Если $e \in E$ - дуга, то $c(e)$ - пропускная способность дуги.

Разрез сети - это разбиение множества вершин V на два непересекающихся подмножества S и T таких, что $s \in S$, $t \in T$. Обозначим через P^+ множество всех дуг от S к T (то есть начало дуги в S, а конец дуги в T), через P^- - множество всех дуг от T к S. Под разрезом также понимают и $P = P^+ \cup P^-$; под ориентированным разрезом (орразрезом) – множество дуг P^+. Сумма пропускных способностей всех дуг разреза (орразреза) называется пропускной способностью и обозначается соответственно $C(P)$, $C(P^+)$.

Функция $\varphi : E \to R$ является потоком в сети, если выполнены условия:

1) для любой дуги $e \in E$ - $0 \le \varphi(e) \le c(e)$;
2) для любой вершины $u \in V \setminus \{s, t\}$ - сумма потоков по дугам входящим в u равна сумме потоков по дугам исходящим из u ($\varphi(e)$ - поток по дуге e).

Величиной потока назовем сумму потоков по дугам исходящим из S, обозначим её через $W(\varphi)$. Нетрудно показать, что сумма потоков по дугам, входящим в t равна $W(\varphi)$. Если $P = P^+ \cup P^-$ - разрез, то справедливо равенство:

$$W(\varphi) = \Phi(P^+) - \Phi(P^-)$$

где $\Phi(P^+)$, $\Phi(P^-)$ – суммы потоков по дугам, входящим в P^+, P^- соответственно.

Поток максимальный, если его величина принимает наибольшее возможное значение, эта величина называется ***пропускной способностью сети***. Орразрез называется минимальным, если минимальна его пропускная способность среди всех его орразрезов. Из равенства, приведенного выше, следует, что $W(\varphi) \le \Phi(P^+) \le C(P^+)$, то есть величина любого потока не превосходит пропускной способности любого орразреза. Следовательно, пропускная способность сети не превосходит пропускной способности минимального орразреза.

ТЕОРЕМА ФОРДА – ФАЛКЕРСОНА: пропускная способность сети равна пропускной способности минимального орразреза.

Поскольку доказательство теоремы носит конструктивный характер и используется при построении алгоритма, то кратко изложим его суть.

Соединим s с вершиной u при помощи цепи $< s, u >$, не учитывая направления дуг. Цепь $< s, u >$ называется ***аугментальной***, если дуги e^+, направленные против движения, имеют положительный поток; иными словами все величины $c(e^+) - \varphi(e^+)$, $\varphi(e^-)$ строго положительны. Пусть t достижима из s при помощи такой цепи, тогда величину потока можно

увеличить. Обозначим через K минимальное среди всех значений $c(e^+) - \varphi(e^+)$, $\varphi(e^-)$; $K > 0$.

Перестроим поток следующим образом: $\varphi(e^+)$ заменим на $\varphi(e^+) + K$, а $\varphi(e^-)$ на $\varphi(e^-) - K$. Условия (1) и (2) останутся выполненными, а величина потока при этом увеличится на K. Повторяем процедуру до тех пор, пока t перестанет быть достижимой. Пусть S - множество всех вершин, достижимых из s, а $T = V \setminus S$ содержит t. Следовательно, это разрез, при этом дуги, входящие в P^+ насыщены, а дуги из P^- имеют нулевой поток (иначе S можно было бы расширить).

Таким образом, $W(\varphi) = \Phi(P^+) - \Phi(P^-) = C(P^+) - 0 = C(P^+)$, то есть поток максимальный, а P^+ - минимальный орразрез.

Пример: Найти максимальный поток φ для сети T (рис. 1).

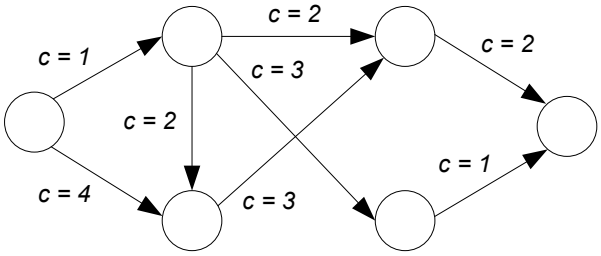


Рисунок 1

Решение:

1) Перенумеруем поток произвольным образом. Вершина-исток должна иметь номер 1, а вершина-сток – максимальное значение среди всех вершин сети. Зададим начальный поток в сети $\varphi = 0$ и присвоим вершине-истоку нулевую метку (рис. 2).

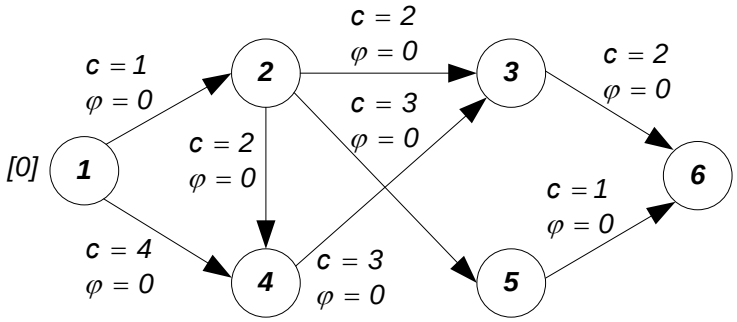


Рисунок 2

2) Пометим вершины и дуги сети (рис. 3), пользуясь следующим правилом: пусть v – помеченная вершина, w – непомеченная, тогда:

- если $w = \Gamma^+(v)$, то есть дуга направлена от вершины v к вершине w, и выполняется неравенство:

$$c(v,w) - \varphi(v,w) > 0,$$

вершине w присваивается метка, равная номеру вершины v, а дуге – значение «+»;

- если $w = \Gamma^{-}(v)$, то есть дуга направлена от вершины w к вершине v, и выполняется неравенство: $\varphi(v,w) > 0$, вершине w присваивается потенциал, равный номеру вершины v, а дуге – значение «–».

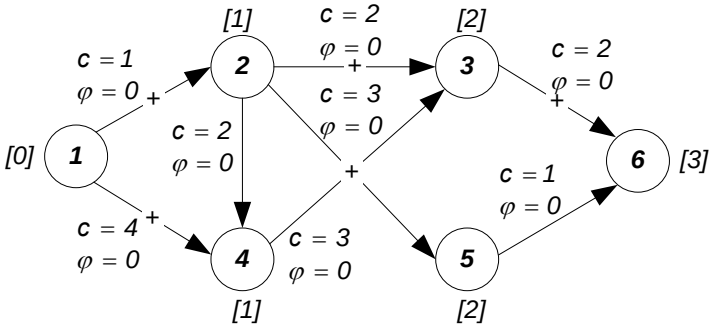


Рисунок 3

Вершина-сток получила метку, поэтому продолжаем решение.

3) Рассмотрим подграф исходного графа (рис. 4), такой, что метка вершины w соответствует номеру вершины v, где v – вершина-начало дуги (v, w).

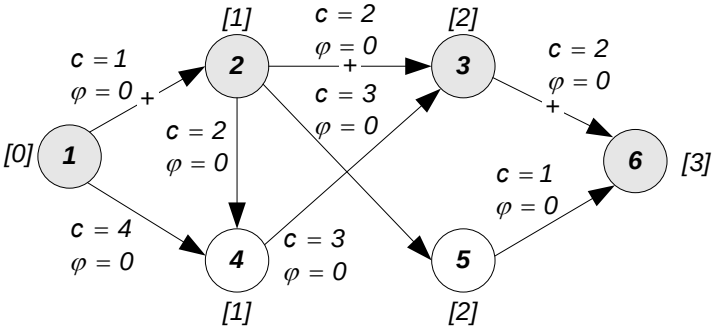


Рисунок 4

4) Перестроим поток в сети по правилу:

- если некоторая дуга e не принадлежит рассматриваемому подграфу, то величина нового потока составит $\varphi^{*}(e) = \varphi(e)$;
- если некоторая дуга e принадлежит рассматриваемому подграфу и имеет знак «+», то: $\varphi^{*}(e) = \varphi(e) + K_1$, где $K_1 = \min [c(e) - \varphi(e)]$;
- если некоторая дуга e принадлежит рассматриваемому подграфу и имеет знак «–», то: $\varphi^{*}(e) = \varphi(e) - K_2$, где $K_2 = \min [\varphi(e)]$.

В рассматриваемом случае (рис. 5) $\varphi^{*} = 0 + \min [1, 2, 2] = 1$.

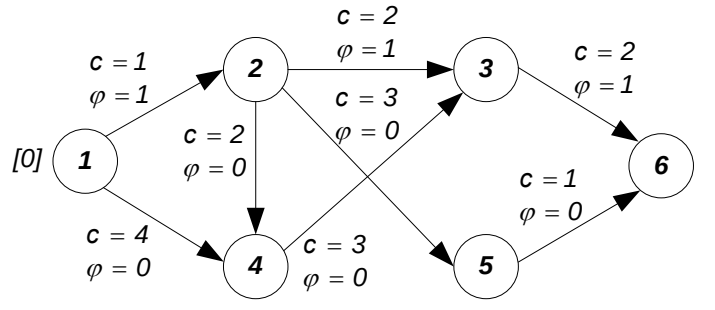


Рисунок 5

Далее осуществляем переход к пункту 2, по результатам выполнения которого представляется граф с заново размеченными вершинами (рис. 6):

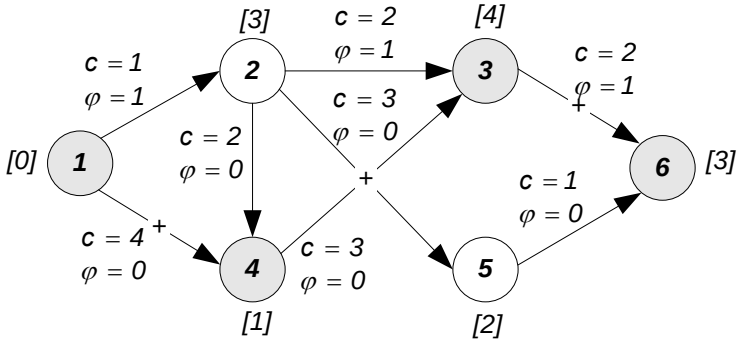


Рисунок 6

Вершина-сток получила метку, поэтому продолжаем решение. Зададим новый поток при рассмотрении подграфа, отраженного на рисунке 6.

$K_1 = \min [4, 3, 1] = 1;$

$\varphi^*(1,4) = 0 + K_1 = 1;\ \varphi^*(4,3) = 0 + K_1 = 1;\ \varphi^*(3,6) = 1 + K_1 = 2$

$\varphi^* = \min [\varphi^*(1,4), \varphi^*(4,3), \varphi^*(3,6)] = 1$

5) Увеличим поток в сети на величину φ^* (рис. 7)

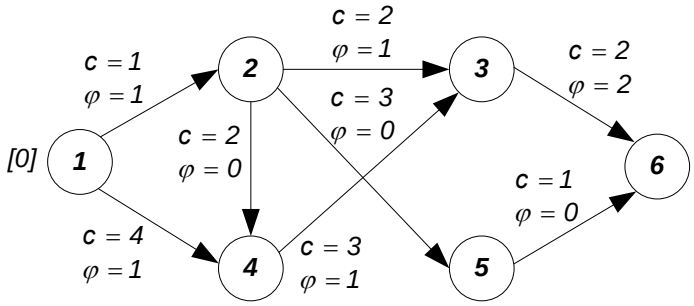


Рисунок 7

и осуществим переход к пункту 2, результат выполнения которого отражен на рисунке 8.

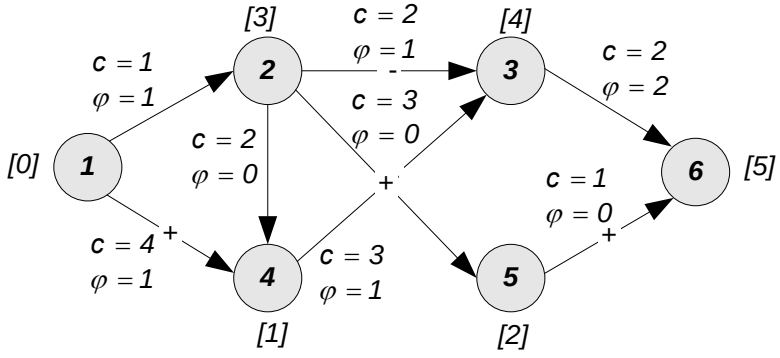


Рисунок 8

Вершине-стоку присвоен потенциал, поэтому продолжаем решение. Перестроим поток в сети:

$K_1 = \min [3, 2, 1, 3, 1] = 1;$

$\varphi^*(1,4) = 1 + K_1 = 2;\ \varphi^*(4,3) = 1 + K_1 = 2;\quad \varphi^*(3,2) = 1 + K_1 = 2;$

$\varphi^*(2,5) = 0 + K_1 = 1$; $\delta(5,6) = 0 + K_1 = 1$;

$\varphi^* = \min[\varphi^*(1,4), \varphi^*(4,3), \varphi^*(3,2), \varphi^*(2,5), \varphi^*(5,6)] = 1$.

6) Увеличиваем поток в сети на величину φ^* (рис. 9)

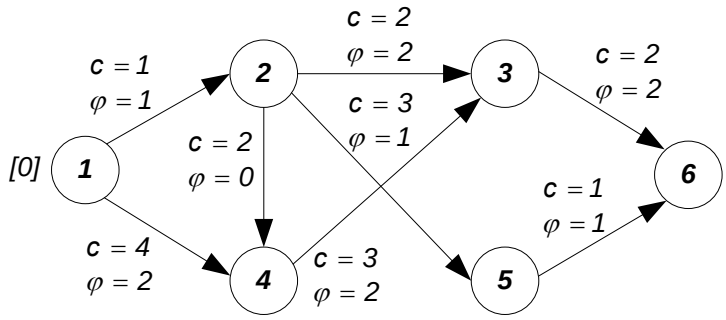


Рисунок 9

и осуществляем переход к пункту 2 (рис. 10):

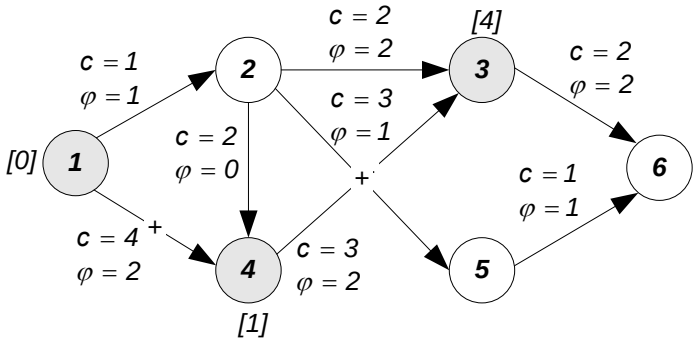


Рисунок 10

Вершина-сток не получила метку, что свидетельствует о том, что максимальный поток в рассматриваемой сети найден (рис. 11).

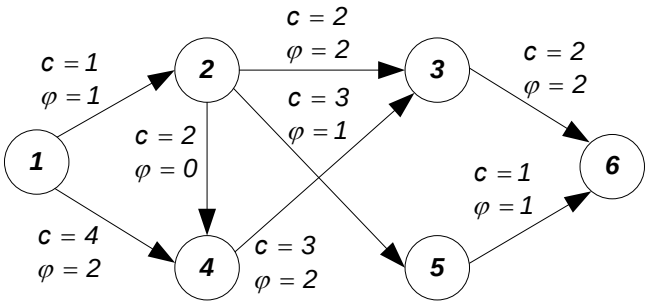


Рисунок 11

Таким образом, по результатам выполнения алгоритма получили, что множество достижимых из вершины-истока вершин S = {1, 3, 4}; множество недостижимых из вершины-источника вершин T = {2, 5, 6}. Минимальным орразрезом является P^+ : (3;6).

Максимальный поток в сети равен 2.

Задания: Определить минимальный орразрез и найти максимальный поток в сети.

1. $\begin{pmatrix} 0 & 3 & 0 & 2 & 0 \\ 0 & 0 & 4 & 5 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 7. $\begin{pmatrix} 0 & 3 & 0 & 5 & 3 \\ 0 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 13. $\begin{pmatrix} 0 & 0 & 5 & 0 & 3 \\ 0 & 0 & 1 & 0 & 4 \\ 0 & 0 & 0 & 2 & 6 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 19. $\begin{pmatrix} 0 & 3 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 2 \end{pmatrix}$

2. $\begin{pmatrix} 0 & 2 & 5 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 5 & 2 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 8. $\begin{pmatrix} 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 2 & 6 & 1 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 14. $\begin{pmatrix} 0 & 1 & 8 & 4 & 0 \\ 0 & 0 & 5 & 6 & 0 \\ 0 & 0 & 0 & 3 & 5 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 20. $\begin{pmatrix} 0 & 5 & 0 & 3 & 0 \\ 0 & 0 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

3. $\begin{pmatrix} 0 & 2 & 1 & 4 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 6 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 9. $\begin{pmatrix} 0 & 4 & 0 & 5 & 1 \\ 0 & 0 & 1 & 4 & 2 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 15. $\begin{pmatrix} 0 & 3 & 5 & 0 & 1 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 3 & 6 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 21. $\begin{pmatrix} 0 & 5 & 0 & 3 & 0 \\ 0 & 0 & 0 & 5 & 3 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

4. $\begin{pmatrix} 0 & 2 & 1 & 5 & 1 \\ 0 & 0 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 10. $\begin{pmatrix} 0 & 0 & 2 & 4 & 5 \\ 0 & 0 & 4 & 8 & 0 \\ 0 & 0 & 0 & 7 & 6 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 16. $\begin{pmatrix} 0 & 1 & 0 & 6 & 3 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 22. $\begin{pmatrix} 0 & 1 & 1 & 0 & 4 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

5. $\begin{pmatrix} 0 & 2 & 0 & 4 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 1 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 11. $\begin{pmatrix} 0 & 6 & 0 & 4 & 0 \\ 0 & 0 & 4 & 8 & 4 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 17. $\begin{pmatrix} 0 & 5 & 0 & 4 & 1 \\ 0 & 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 23. $\begin{pmatrix} 0 & 5 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

6. $\begin{pmatrix} 0 & 3 & 1 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 12. $\begin{pmatrix} 0 & 1 & 3 & 3 & 1 \\ 0 & 0 & 5 & 3 & 2 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 18. $\begin{pmatrix} 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$ 24. $\begin{pmatrix} 0 & 3 & 4 & 0 & 3 \\ 0 & 0 & 3 & 0 & 6 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

Ответы

1. Обходы графов в ширину и глубину

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	2, 3		4, 5					
2	2, 3	4, 5	6, 7					
3	2, 3	4	5, 6					
4	2, 3	4	5, 6			7		
5	2, 3	4, 5						
6	2, 3	4, 5			6, 7			
7	2		1, 4, 5					
8	2	4, 5	1, 6, 7					
9	2	4	1, 5, 6					
10	2	4	1, 5, 6			7		
11	2	4, 5	1					
12	2	4, 5	1		6, 7			
13	3	1	4, 5					
14	3	1, 4, 5	6, 7					
15	3	1, 4	5, 6			7		
16	3	1, 4, 5			6, 7			
17	3	1, 4	5, 6					
18	3	1, 4, 5						
19	2		1, 5	3				
20	3	1, 5	6, 7	2				
21	3	1	5, 6	2				
22	3	1	5, 6	2		7		
23	3	1, 5		2				
24	3	1, 5		2	6, 7			
25	2, 3	4, 5			6, 7, 8			
26	2, 3	4, 5, 6						
27	2, 3	4, 5, 6	7					
28	2, 3, 7	4, 5, 6						
29	2, 7, 3	4, 5, 6	8					
30	2, 3	4, 5, 6	7, 8					

2. Алгоритм Тэрри

1. 3 – 2 – 4 – 1 – 5
2. 1 – 2 – 3 – 5 – 4
3. 5 – 2 – 4 – 1 – 3
4. 1 – 2 – 5 – 4 – 3
5. 2 – 5 – 4 – 3 – 1
6. 2 – 1 – 3 – 4 – 5

7. 4 – 3 – 2 – 5 – 1
8. 1 – 2 – 3 – 5 – 4
9. 3 – 5 – 4 – 2 – 1
10. 5 – 3 – 4 – 2 – 1
11. 1 – 3 – 2 – 4 – 5 – 6
12. 4 – 3 – 5 – 6 – 1 – 2
13. 5 – 4 – 3 – 6 – 1 – 2
14. 2 – 3 – 4 – 6 – 5 – 1
15. 6 – 4 – 5 – 3 – 2 – 1
16. 2 – 1 – 3 – 4 – 5 – 6
17. 6 – 5 – 2 – 1 – 3 – 4
18. 3 – 2 – 5 – 6 – 4 – 1
19. 5 – 6 – 2 – 4 – 3 – 1
20. 1 – 2 – 5 – 6 – 4 – 3
21. 1 – 2 – 3 – 4 – 5 – 6
22. 3 – 1 – 2 – 4 – 5 – 6
23. 4 – 3 – 5 – 1 – 2 – 6
24. 6 – 5 – 4 – 3 – 2 – 1
25. 2 – 3 – 5 – 6 – 4 – 1
26. 2 – 1 – 3 – 4 – 5
27. 5 – 3 – 4 – 1 – 2
28. 1 – 3 – 5 – 4 – 2
29. 3 – 5 – 4 – 2 – 1
30. 4 – 5 – 3 – 1 – 2

3. Матроиды. Жадные алгоритмы. Алгоритм Краскала

1. (2,5) (1,4) (3,4) (4,5)
2. (3,4) (4,5) (7,8) (4,7) (5,6) (2,3) (1,6)
3. (2,6) (4,5) (4,6) (3,4) (1,3)
4. (1,2) (3,6) (2,6) (4,5) (3,4)
5. (1,2) (6,8) (7,8) (1,8) (2,3) (5,6) (3,4)
6. (2,3) (1,4) (2,4) (1,5)
7. (1,6) (3,4) (4,5) (4,7) (5,6) (1,2) (7,8)
8. (1,2) (2,6) (4,5) (3,4) (5,6)
9. (2,6) (4,5) (4,6) (3,4) (1,3)
10. (1,8) (2,3) (3,4) (2,8) (6,8) (7,8) (4,5)
11. (2,4) (1,5) (2,3) (2,5)
12. (3,4) (7,8) (1,6) (3,8) (5,6) (2,3) (6,7)
13. (2,6) (4,6) (1,2) (1,3) (5,6)
14. (1,3) (3,6) (5,6) (2,6) (4,6)
15. (2,4) (5,6) (6,8) (1,8) (3,4) (1,2) (7,8)
16. (1,4) (1,5) (1,2) (2,3)
17. (4,5) (5,6) (7,8) (1,2) (4,7) (1,6) (2,3)
18. (3,4) (4,5) (4,6) (2,3) (1,2)
19. (2,6) (4,6) (1,2) (1,3) (5,6)
20. (1,8) (3,4) (1,2) (2,4) (5,6) (6,8) (7,8)
21. (1,3) (4,5) (2,4) (1,4)
22. (2,3) (3,8) (3,4) (4,7) (1,6) (4,5) (1,8)
23. (1,3) (5,6) (2,6) (4,6) (1,2)
24. (2,3) (1,2) (5,6) (2,6) (4,5)
25. (6,7) (1,2) (5,6) (1,8) (2,3) (4,5) (6,8)
26. (1,2) (2,5) (1,3) (2,4)
27. (1,2) (4,7) (1,6) (2,3) (3,4) (4,5) (1,8)
28. (2,3) (1,2) (5,6) (2,6) (4,5)
29. (1,3) (5,6) (2,6) (4,6) (1,2)
30. (2,4) (5,6) (1,8) (3,4) (1,2) (4,5) (7,8)

4. Нахождение кратчайшего пути в сети без контуров

1. Правильная нумерация: 0-0, 2-1, 3-2, 4-3, 1-4.
Потенциалы: 0-0, 1-1, 2-5, 3-6, 4-8. Кратчайший путь: 0 1 4.

2. Правильная нумерация: 0-0, 1-1, 2-2, 3-3, 4-4.
Потенциалы: 0-0, 1-1, 2-9, 3-5, 4-11. Кратчайший путь: 0 1 3 4.

3. Правильная нумерация: 1-0, 2-1, 3-2, 1-3, 4-4.
Потенциалы: 0-0, 1-8, 2-4, 3-13, 4-7. Кратчайший путь: 0 2 4.

4. Правильная нумерация: 0-0, 2-1, 3-2, 1-3, 4-4.
Потенциалы: 0-0, 1-8, 2-14, 3-10, 4-3. Кратчайший путь: 0 4.

5. Правильная нумерация: 2-0, 0-1, 1-2, 3-3, 4-4.
Потенциалы: 0-0, 1-4, 2-1, 3-8, 4-6. Кратчайший путь: 0 2 4.

6. Правильная нумерация: 3-0, 0-1, 1-2, 2-3, 4-4.
Потенциалы: 0-0, 1-6, 2-9, 3-4, 4-6. Кратчайший путь: 0 3 4.

7. Правильная нумерация: 2-0, 1-1, 3-2, 4-3, 0-4.
Потенциалы: 0-0, 1-4, 2-12, 3-2, 4-8. Кратчайший путь: 0 3 4.

8. Правильная нумерация: 2-0, 0-1, 1-2, 3-3, 4-4.
Потенциалы: 0-0, 1-9, 2-7, 3-16, 4-10. Кратчайший путь: 0 2 4.

9. Правильная нумерация: 1-0, 3-1, 0-2, 4-3, 2-4.
Потенциалы: 0-0, 1-7, 2-13, 3-3, 4-5. Кратчайший путь: 0 3 4.

10. Правильная нумерация: 1-0, 0-1, 3-2, 2-3, 4-4.
Потенциалы: 0-0, 1-2, 2-4, 3-5, 4-5. Кратчайший путь: 0 4.

11. Правильная нумерация: 1-0, 0-1, 3-2, 4-3, 2-4.
Потенциалы: 0-0, 1-6, 2-1, 3-3, 4-3. Кратчайший путь: 0 2 4.

12. Правильная нумерация: 2-0, 3-1, 4-2, 0-3, 1-4 .
Потенциалы: 0-0, 1-7, 2-1, 3-2, 4-5. Кратчайший путь: 0 2 4.

13. Правильная нумерация: 1-0, 0-1, 3-2, 4-3, 2-4.
Потенциалы: 0-0, 1-4, 2-1, 3-2, 4-7. Кратчайший путь: 0 1 4.

14. Правильная нумерация: 1-0, 3-1, 0-2, 2-3, 4-4.
Потенциалы: 0-0, 1-4, 2-2, 3-5, 4-8. Кратчайший путь: 0 2 4.

15. Правильная нумерация: 4-0, 0-1, 1-2, 2-3, 3-4.
Потенциалы: 0-0, 1-3, 2-10, 3-7, 4-9. Кратчайший путь: 0 1 3 4.

16. Правильная нумерация: 2-0, 0-1, 5-2, 6-3, 1-4, 3-5, 4-6.
Потенциалы: 0-0, 1-6, 2-9, 3-7, 4-8, 5-15, 6-5. Кратчайший путь: 0 6.

17. Правильная нумерация: 4-0, 1-1, 2-2, 3-3, 5-4, 0-5, 6-6.
Потенциалы: 0-0, 1-4, 2-6, 3-8, 4-11, 5-8, 6-13. Кратчайший путь: 0 1 6.

18. Правильная нумерация: 4-0, 0-1, 2-2, 5-3, 3-4, 1-5, 6-6.
Потенциалы: 0-0, 1-9, 2-2, 3-16, 4-19, 5-7, 6-27.
Кратчайший путь: 0 1 3 4 6.

19. Правильная нумерация: 0-0, 1-1, 2-2, 3-3, 4-4, 5-5, 6-6.
Потенциалы: 0-0, 1-1, 2-2, 3-6, 4-9, 5-6, 6-9. Кратчайший путь: 0 6.

20. Правильная нумерация: 3-0, 0-1, 1-2, 2-3, 4-4, 6-5, 5-6.
Потенциалы: 0-0, 1-4, 2-5, 3-5, 4-1, 5-3, 6-8. Кратчайший путь: 0 2 4 6.
21. Правильная нумерация: 1-0, 0-1, 2-2, 3-3, 4-4, 5-5, 6-6.
Потенциалы: 0-0, 1-6, 2-4, 3-11, 4-7, 5-2, 6-14. Кратчайший путь: 0 1 6.
22. Правильная нумерация: 3-0, 0-1, 1-2, 5-3, 2-4, 4-5, 6-6.
Потенциалы: 0-0, 1-4, 2-8, 3-9, 4-10, 5-5, 6-7. Кратчайший путь: 0 1 6.
23. Правильная нумерация: 0-0, 1-1, 3-2, 4-3, 5-4, 2-5, 6-6.
Потенциалы: 0-0, 1-3, 2-7, 3-1, 4-8, 5-6, 6-6. Кратчайший путь: 0 6.
24. Правильная нумерация: 3-0, 0-1, 1-2, 2-3, 4-4, 6-5, 5-6.
Потенциалы: 0-0, 1-6, 2-4, 3-7, 4-6, 5-3, 6-8. Кратчайший путь: 0 5 6.
25. Правильная нумерация: 6-0, 0-1, 3-2, 1-3, 4-4, 2-5, 5-6.
Потенциалы: 0-0, 1-1, 2-9, 3-4, 4-5, 5-6, 6-8. Кратчайший путь: 0 1 6.
26. Правильная нумерация: 4-0, 0-1, 1-2, 2-3, 3-4, 6-5, 5-6.
Потенциалы: 0-0, 1-2, 2-8, 3-4, 4-5, 5-9, 6-12. Кратчайший путь: 0 1 5 6.
27. Правильная нумерация: 5-0, 0-1, 1-2, 2-3, 3-4, 4-5, 6-6.
Потенциалы: 0-0, 1-2, 2-4, 3-1, 4-7, 5-5, 6-12. Кратчайший путь: 0 2 6.
28. Правильная нумерация: 0-0, 1-1, 2-2, 3-3, 4-4, 5-5, 6-6.
Потенциалы: 0-0, 1-1, 2-4, 3-5, 4-6, 5-11, 6-4. Кратчайший путь: 0 1 6.
29. Правильная нумерация: 1-0, 0-1, 2-2, 3-3, 4-4, 5-5, 6-6.
Потенциалы: 0-0, 1-1, 2-6, 3-9, 4-5, 5-7, 6-5. Кратчайший путь: 0 1 6.
30. Правильная нумерация: 1-0, 0-1, 2-2, 3-3, 5-4, 6-5, 4-6.
Потенциалы: 0-0, 1-5, 2-2, 3-8, 4-14, 5-1, 6-9. Кратчайший путь: 0 5 6.

5. Алгоритм Беллмана-Форда

1. 0 – 1 – 2 – 3 – 5
2. 0 – 2 – 3 – 5
3. 0 – 1 – 3 – 2 – 4 – 5
4. 0 – 2 – 1 – 3 – 5
5. 0 – 2 – 4 – 5
6. 0 – 2 – 5 – 3 – 6
7. 0 – 2 – 1 – 5 – 3 – 4 – 6
8. 0 – 1 – 5 – 3 – 6
9. 0 – 2 – 5 – 3 – 4 – 6
10. 0 – 2 – 1 – 5 – 4 – 6
11. 0 – 2 – 1 – 3 – 4 – 6 – 5 – 7
12. 0 – 1 – 3 – 4 – 6 – 7
13. 0 – 1 – 3 – 5 – 7
14. 0 – 2 – 4 – 6 – 5 – 7
15. 0 – 2 – 1 – 3 – 5 – 7
16. 0 – 3 – 6
17. 0 – 1 – 3 – 4 – 6
18. 0 – 3 – 4 – 6
19. 0 – 2 – 5 – 6
20. 0 – 2 – 3 – 4 – 6
21. 0 – 1 – 2 – 3 – 4 – 5 – 6
22. 0 – 2 – 3 – 4 – 6
23. 0 – 1 – 2 – 4 – 5 – 6
24. 0 – 2 – 4 – 6
25. 0 – 1 – 2 – 3 – 4 – 6
26. 0 – 2 – 3 – 1 – 4 – 6
27. 0 – 1 – 4 – 3 – 5 – 6
28. 0 – 2 – 3 – 5 – 6
29. 0 – 1 – 4 – 6
30. 0 – 2 – 5 – 6

6. Алгоритм Дейкстра

1. $1 \to 4 \to 5$
Длина пути: 7

2. $1 \to 2 \to 4 \to 3 \to 5$
Длина пути: 12

3. $1 \to 2 \to 3 \to 5$
Длина пути: 7

4. $1 \to 4 \to 3 \to 5$
Длина пути: 6

5. $1 \to 3 \to 5$
Длина пути: 5

6. $1 \to 2 \to 5$
Длина пути: 7

7. $1 \to 4 \to 5$
Длина пути: 5

8. $1 \to 3 \to 5$
Длина пути: 3

9. $1 \to 2 \to 5$
Длина пути: 4

10. $1 \to 2 \to 4 \to 5$
Длина пути: 7

11. $1 \to 4 \to 5$
Длина пути: 5

12. $1 \to 2 \to 3 \to 5$
Длина пути: 4

13. $1 \to 4 \to 3 \to 5$
Длина пути: 6

14. $1 \to 2 \to 5$
Длина пути: 2

15. $1 \to 4 \to 3 \to 5$
Длина пути: 7

16. $1 \to 2 \to 3 \to 5$
Длина пути: 11

17. $1 \to 2 \to 4 \to 5$
Длина пути: 5

18. $1 \to 5$
Длина пути: 1

19. $1 \to 2 \to 4 \to 5$
Длина пути: 5

20. $1 \to 2 \to 5$
Длина пути: 2

21. $1 \to 3 \to 4 \to 5$
Длина пути: 10

22. $1 \to 5$
Длина пути: 3

23. $1 \to 3 \to 5$
Длина пути: 5

24. $1 \to 4 \to 5$
Длина пути: 4

25. $1 \to 2 \to 5$
Длина пути: 6

26. $1 \to 2 \to 3 \to 4 \to 5$
Длина пути: 11

27. $1 \to 4 \to 2 \to 5$
Длина пути: 8

28. $1 \to 3 \to 5$
Длина пути: 2

29. $1 \to 4 \to 5$
Длина пути: 3

30. $1 \to 5$
Длина пути: 1

7. Алгоритм Флойда – Уоршалла

№	Матрица кратчайших расстояний T_{ij}	Матрица маршрутов H_{ij}	№	Матрица кратчайших расстояний T_{ij}	Матрица маршрутов H_{ij}
1	2	3	4	5	6
1	$\begin{pmatrix} 0 & 9 & 7 & 3 & 9 \\ \infty & 0 & 3 & \infty & 1 \\ \infty & \infty & 0 & \infty & 2 \\ \infty & \infty & 4 & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 4 & 4 & 4 \\ 0 & 0 & 3 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**16**	$\begin{pmatrix} 0 & 1 & 3 & 5 & 2 \\ \infty & 0 & \infty & 4 & 1 \\ \infty & \infty & 0 & 2 & 7 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 3 & 2 & 2 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
2	$\begin{pmatrix} 0 & 8 & 14 & 6 & 1 \\ \infty & 0 & \infty & \infty & 3 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 2 & 8 & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 4 & 4 & 4 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**17**	$\begin{pmatrix} 0 & 1 & 4 & 2 & 7 \\ \infty & 0 & 3 & 1 & 6 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 2 & 4 \\ 0 & 0 & 3 & 4 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

1	2	3	4	5	6
3	$\begin{pmatrix} 0 & 1 & 8 & 1 & 7 \\ \infty & 0 & 7 & \infty & 10 \\ \infty & \infty & 0 & \infty & 3 \\ \infty & \infty & \infty & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 4 & 4 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**18**	$\begin{pmatrix} 0 & 1 & 10 & \infty & 14 \\ \infty & 0 & 9 & \infty & 13 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 0 & 3 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
4	$\begin{pmatrix} 0 & 9 & 10 & 8 & 11 \\ \infty & 0 & 1 & \infty & 2 \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 5 & 0 & 6 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 4 & 3 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**19**	$\begin{pmatrix} 0 & 9 & \infty & 2 & 14 \\ 8 & 0 & \infty & 10 & 5 \\ 7 & 16 & 0 & 9 & 4 \\ 15 & 7 & \infty & 0 & 12 \\ 3 & 12 & \infty & 5 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 4 & 3 & 4 & 4 \\ 5 & 0 & 3 & 5 & 5 \\ 5 & 5 & 0 & 5 & 5 \\ 5 & 2 & 0 & 0 & 2 \\ 1 & 4 & 0 & 2 & 0 \end{pmatrix}$
5	$\begin{pmatrix} 0 & 4 & 6 & 1 & 3 \\ \infty & 0 & \infty & \infty & 3 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 3 & 5 & 0 & 2 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 4 & 4 & 4 & 4 \\ 0 & 0 & 3 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**20**	$\begin{pmatrix} 0 & \infty & 7 & 2 & 4 \\ \infty & 0 & 11 & 6 & 8 \\ \infty & \infty & 0 & \infty & 1 \\ \infty & \infty & 5 & 0 & 2 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 4 & 4 & 4 \\ 0 & 0 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
6	$\begin{pmatrix} 0 & 5 & 15 & 10 & 16 \\ 14 & 0 & 10 & 5 & 11 \\ 4 & 9 & 0 & 14 & 1 \\ 9 & 14 & 5 & 0 & 6 \\ 3 & 8 & 18 & 13 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 4 & 2 & 4 \\ 5 & 0 & 4 & 4 & 4 \\ 5 & 5 & 0 & 5 & 5 \\ 5 & 5 & 3 & 0 & 3 \\ 1 & 1 & 4 & 2 & 0 \end{pmatrix}$	**21**	$\begin{pmatrix} 0 & 5 & 1 & 2 & 5 \\ 8 & 0 & 9 & 10 & 5 \\ 7 & 12 & 0 & 9 & 4 \\ 11 & 3 & 12 & 0 & 8 \\ 3 & 8 & 4 & 5 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 4 & 3 & 4 & 3 \\ 5 & 0 & 5 & 5 & 5 \\ 5 & 5 & 0 & 5 & 5 \\ 5 & 2 & 5 & 0 & 2 \\ 1 & 4 & 1 & 1 & 0 \end{pmatrix}$
7	$\begin{pmatrix} 0 & 3 & 5 & 9 & 10 \\ \infty & 0 & 2 & 6 & 7 \\ \infty & 7 & 0 & 5 & 6 \\ \infty & 2 & 4 & 0 & 1 \\ \infty & 1 & 3 & 7 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 2 & 4 \\ 0 & 0 & 3 & 4 & 4 \\ 0 & 5 & 0 & 4 & 4 \\ 0 & 5 & 5 & 0 & 5 \\ 0 & 2 & 2 & 2 & 0 \end{pmatrix}$	**22**	$\begin{pmatrix} 0 & 1 & 2 & 7 & 4 \\ \infty & 0 & 1 & 6 & 3 \\ \infty & \infty & 0 & 5 & 2 \\ \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 3 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 3 & 3 \\ 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \end{pmatrix}$
8	$\begin{pmatrix} 0 & 7 & 1 & 2 & 2 \\ \infty & 0 & \infty & \infty & 8 \\ \infty & \infty & 0 & \infty & 1 \\ \infty & 5 & 9 & 0 & 10 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 4 & 3 & 4 & 3 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 2 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**23**	$\begin{pmatrix} 0 & 1 & 3 & 2 & 2 \\ \infty & 0 & 3 & \infty & 1 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & 1 & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 4 & 4 & 2 \\ 0 & 0 & 3 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
9	$\begin{pmatrix} 0 & 5 & 6 & 9 & 8 \\ 6 & 0 & 1 & 4 & 3 \\ 5 & 10 & 0 & 14 & 2 \\ 7 & 12 & 2 & 0 & 4 \\ 3 & 8 & 9 & 12 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 2 & 3 \\ 5 & 0 & 3 & 4 & 3 \\ 5 & 5 & 0 & 5 & 5 \\ 5 & 5 & 3 & 0 & 3 \\ 1 & 1 & 2 & 2 & 0 \end{pmatrix}$	**24**	$\begin{pmatrix} 0 & 1 & 4 & \infty & 4 \\ \infty & 0 & 3 & \infty & 3 \\ \infty & \infty & 0 & \infty & 4 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 0 & 2 \\ 0 & 0 & 3 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

1	2	3	4	5	6
10	$\begin{pmatrix} 0 & 1 & 4 & 5 & 1 \\ \infty & 0 & 3 & 4 & 5 \\ \infty & 7 & 0 & 5 & 6 \\ \infty & 2 & 5 & 0 & 1 \\ \infty & 1 & 4 & 5 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 2 & 5 \\ 0 & 0 & 3 & 4 & 4 \\ 0 & 5 & 0 & 4 & 4 \\ 0 & 5 & 5 & 0 & 5 \\ 0 & 2 & 2 & 2 & 0 \end{pmatrix}$	**25**	$\begin{pmatrix} 0 & \infty & 5 & 3 & 5 \\ 9 & 0 & 6 & 4 & 6 \\ 4 & \infty & 0 & 7 & 1 \\ 5 & \infty & 2 & 0 & 2 \\ 3 & \infty & 8 & 6 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 4 & 4 & 4 \\ 5 & 0 & 4 & 4 & 4 \\ 5 & 0 & 0 & 5 & 5 \\ 5 & 0 & 3 & 0 & 5 \\ 1 & 0 & 4 & 1 & 0 \end{pmatrix}$
11	$\begin{pmatrix} 0 & 6 & 6 & 1 & 9 \\ \infty & 0 & \infty & \infty & 4 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & 5 & 5 & 0 & 9 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 4 & 4 & 4 & 5 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**26**	$\begin{pmatrix} 0 & 1 & 3 & 4 & 5 \\ \infty & 0 & 2 & 3 & 4 \\ \infty & 10 & 0 & 5 & 6 \\ \infty & 5 & 7 & 0 & 1 \\ \infty & 4 & 6 & 7 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 2 & 4 \\ 0 & 0 & 3 & 4 & 4 \\ 0 & 5 & 0 & 4 & 4 \\ 0 & 5 & 5 & 0 & 5 \\ 0 & 2 & 2 & 2 & 0 \end{pmatrix}$
12	$\begin{pmatrix} 0 & 2 & 6 & 3 & 7 \\ 11 & 0 & 7 & 4 & 8 \\ 4 & 6 & 0 & 7 & 1 \\ 7 & 9 & 3 & 0 & 4 \\ 3 & 5 & 9 & 6 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 4 & 4 & 4 \\ 5 & 0 & 4 & 4 & 4 \\ 5 & 5 & 0 & 5 & 5 \\ 5 & 5 & 3 & 0 & 3 \\ 1 & 1 & 4 & 1 & 0 \end{pmatrix}$	**27**	$\begin{pmatrix} 0 & 1 & 3 & 1 & 6 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & 3 \\ \infty & \infty & 2 & 0 & \infty \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 4 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
13	$\begin{pmatrix} 0 & 3 & 7 & 9 & 11 \\ 6 & 0 & 4 & 6 & 8 \\ 2 & 5 & 0 & 2 & 4 \\ 3 & 6 & 10 & 0 & 14 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 2 & 3 & 3 \\ 3 & 0 & 3 & 3 & 3 \\ 1 & 1 & 0 & 4 & 5 \\ 1 & 1 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	**28**	$\begin{pmatrix} 0 & 1 & 2 & 5 & 3 \\ \infty & 0 & \infty & 4 & 3 \\ \infty & \infty & 0 & 3 & 8 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 3 & 2 & 5 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$
14	$\begin{pmatrix} 0 & 9 & 18 & 16 & 11 \\ 10 & 0 & 9 & 7 & 2 \\ 9 & 18 & 0 & 25 & 1 \\ 11 & 20 & 2 & 0 & 3 \\ 8 & 17 & 26 & 4 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 4 & 2 & 2 \\ 5 & 0 & 4 & 4 & 5 \\ 5 & 5 & 0 & 5 & 5 \\ 5 & 5 & 3 & 0 & 3 \\ 1 & 1 & 4 & 2 & 0 \end{pmatrix}$	**29**	$\begin{pmatrix} 0 & 1 & \infty & 2 & 7 \\ \infty & 0 & \infty & \infty & \infty \\ 7 & 8 & 0 & 9 & 4 \\ 8 & 3 & \infty & 0 & 5 \\ 3 & 4 & \infty & 5 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 0 & 5 & 5 \\ 5 & 2 & 0 & 0 & 5 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$
15	$\begin{pmatrix} 0 & \infty & 2 & 5 & 3 \\ 13 & 0 & 6 & 4 & 7 \\ 7 & \infty & 0 & 12 & 1 \\ 9 & \infty & 2 & 0 & 3 \\ 6 & \infty & 8 & 11 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 3 & 4 & 3 \\ 5 & 0 & 4 & 4 & 4 \\ 5 & 0 & 0 & 5 & 5 \\ 5 & 0 & 3 & 0 & 3 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$	**30**	$\begin{pmatrix} 0 & 4 & 2 & 5 & 10 \\ \infty & 0 & \infty & 4 & 7 \\ \infty & \infty & 0 & 3 & 8 \\ \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 2 & 3 & 3 & 4 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

8. Алгоритм Форда – Фалкерсона

1. S = {1}; T = {2, 3, 4, 5};
   P^+: (1, 2); (1, 4)
   Максимальный поток: 5

2. S = {1, 3}; T = {2, 3, 4, 5};
   P^+: (3, 4); (3, 5)
   Максимальный поток: 7

3. S = {1}; T = {2, 3, 4, 5};
   P^+: (1, 2); (1, 3); (1, 4)
   Максимальный поток: 7
4. S = {1, 4}; T = {2, 3, 5};
   P^+: (1, 2); (1, 3); (1, 4); (4, 5)
   Максимальный поток: 7
5. S = {1}; T = {2, 3, 4, 5};
   P^+: (1, 2); (1, 4)
   Максимальный поток: 6
6. S = {1, 2, 3}; T = {4, 5};
   P^+: (1, 5); (3, 4)
   Максимальный поток: 6
7. S = {1, 2, 3, 4}; T = {5};
   P^+: (1, 5); (2, 5); (4, 5)
   Максимальный поток: 7
8. S = {1}; T = {2, 3, 4, 5};
   P^+: (1, 3); (1, 5)
   Максимальный поток: 6
9. S = {1, 2, 3, 4}; T = {5};
   P^+: (1, 5); (2, 5); (4, 5)
   Максимальный поток: 6
10. S = {1, 4}; T = {2, 3, 5};
    P^+: (1, 3); (1, 5); (4, 5)
    Максимальный поток: 10
11. S = {1}; T = {2, 3, 4, 5};
    P^+: (1, 2); (1, 4)
    Максимальный поток: 10
12. S = {1, 3}; T = {2, 3, 4, 5};
    P^+: (1, 2); (1, 4); (1, 5); (3, 4)
    Максимальный поток: 7
13. S = {1}; T = {2, 3, 4, 5};
    P^+: (1, 3); (1, 5)
    Максимальный поток: 8
14. S = {1, 2, 3, 4}; T = {5};
    P^+: (3, 5); (4, 5)
    Максимальный поток: 9
15. S = {1}; T = {2, 3, 4, 5};
    P^+: (1, 2); (1, 3); (1, 5)
    Максимальный поток: 9
16. S = {1, 4}; T = {2, 3, 5};
    P^+: (1, 2); (1, 5); (4, 5)
    Максимальный поток: 8
17. S = {1}; T = {2, 3, 4, 5};
    P^+: (1, 2); (1, 4); (1, 5)
    Максимальный поток: 10
18. S = {1}; T = {2, 3, 4, 5};
    P^+: (1, 3); (1, 4); (1, 5)
    Максимальный поток: 6
19. S = {1}; T = {2, 3, 4, 5};
    P^+: (1, 2); (1, 3); (1, 4)
    Максимальный поток: 8
20. S = {1}; T = {2, 3, 4, 5};
    P^+: (1, 2); (1, 4)
    Максимальный поток: 8
21. S = {1, 2, 4}; T = {3, 5};
    P^+: (2, 5); (4, 5)
    Максимальный поток: 6
22. S = {1, 2}; T = {3, 4, 5};
    P^+: (1, 5); (2, 4); (2, 5)
    Максимальный поток: 7
23. S = {1, 2}; T = {3, 4, 5};
    P^+: (1, 5); (2, 4); (2, 5)
    Максимальный поток: 7
24. S = {1, 3}; T = {2, 4, 5};
    P^+: (1, 2); (1, 5); (3, 4)
    Максимальный поток: 9