АЛГОРИТМЫ НА ГРАФАХ

Алгоритм	Объект работы	Действие	Сложность		Доп. память	Полезн.
			Матрица	Список		
Поиск в ширину	Невзвеш-й граф	Поиск кратчайшего расстояния от одной вершины до всех остальных, определение количества компонент связности, определение двудольности графа	$O(N^2)$	$O(M)$	$O(N)$	10
Поиск в глубину	Невзвеш-й граф	Определение количества компонент связности, построение дерева поиска в глубину, поиск точек сочленения и мостов, определение двудольности графа	$O(N^2)$	$O(M)$	$O(N)$	8
Алгоритм Дейкстры	Взвеш-й граф без рёбер отрицат-го веса	Поиск кратчайшего расстояния от одной вершины до всех остальных	$O(N^2)$	$O(N^2)$,	$O(N)$	10
Алгоритм Форда-Беллмана	Взвешенный граф	Поиск кратчайшего расстояния от одной вершины до всех остальных	$O(N^3)$	$O(MN)$	$O(N)$	5
Алгоритм Флойда	Взвешенный граф	Поиск кратчайшего расстояния между каждой парой вершин	$O(N^3)$		$O(N^2)$	8

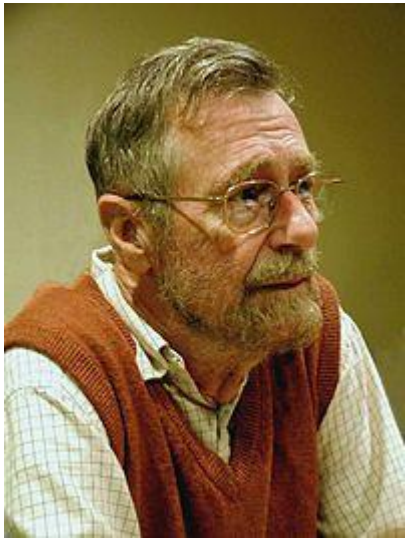
ОСНОВНЫЕ АЛГОРИТМЫ

- Нахождение кратчайших путей из одного источника: алгоритм Дейкстры.
- Построение минимального остова графа: алгоритм Крускала.
- Задача о лабиринте и поиск в глубину на неориентированном графе.

Алгоритм Дейкстры

АЛГОРИТМ ДЕЙКСТРЫ

- **Э́дсгер Ви́бе Де́йкстра** (*Edsger Wybe Dijkstra* [ˈɛtsxər ˈʋibə ˈdɛikstra]) (11 мая 1930, Роттердам, Нидерланды — 6 августа 2002) — нидерландский учёный, идеи которого оказали влияние на развитие компьютерной индустрии.



АЛГОРИТМ ДЕЙКСТРЫ

Алгоритм Дейкстры — алгоритм на графах, изобретённый нидерландским ученым Э. Дейкстрой в 1959 году. Находит кратчайшее расстояние от одной из вершин графа до всех остальных.

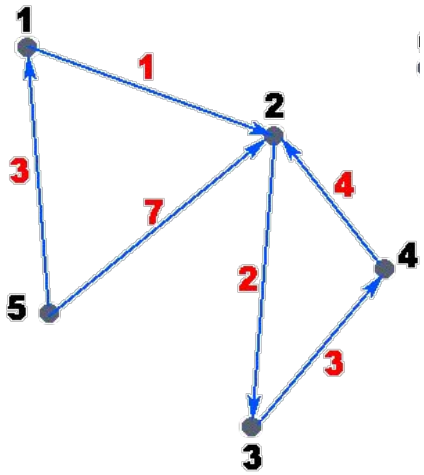
Алгоритм работает только для графов без рёбер отрицательного веса.

Алгоритм широко применяется в программировании и технологиях, например, его используют протоколы маршрутизации OSPF и IS-IS.

ФОРМУЛИРОВКА ЗАДАЧИ

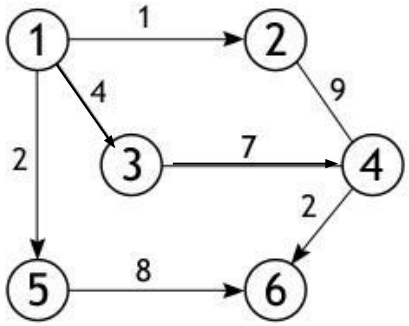
Дан взвешенный ориентированный граф без петель и дуг отрицательного веса. Найти кратчайшие пути от некоторой вершины a до всех остальных вершин этого графа

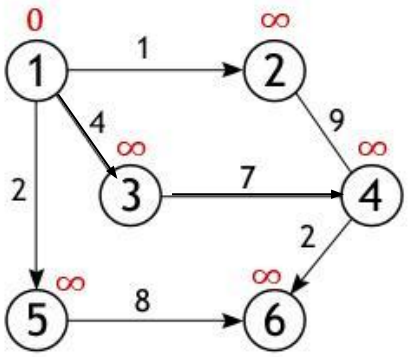
Граф называется ***взвешенным*** или ***нагруженным***, если каждому ребру поставлено в соответствии некоторое число ***w*** (вес).



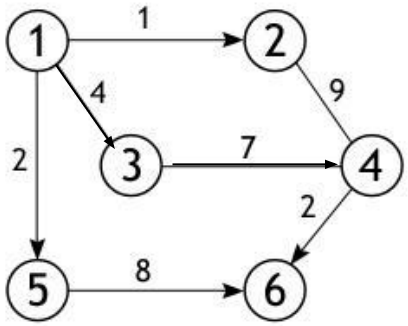
АЛГОРИТМ

Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.



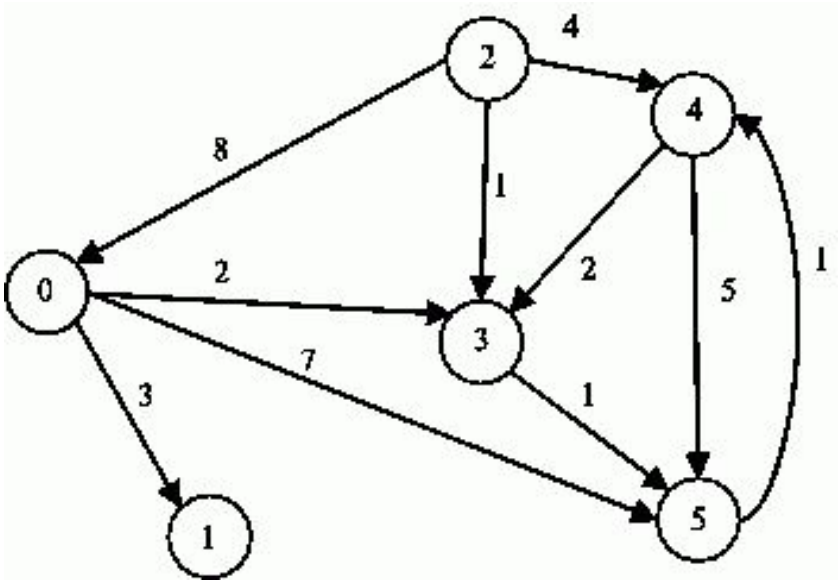


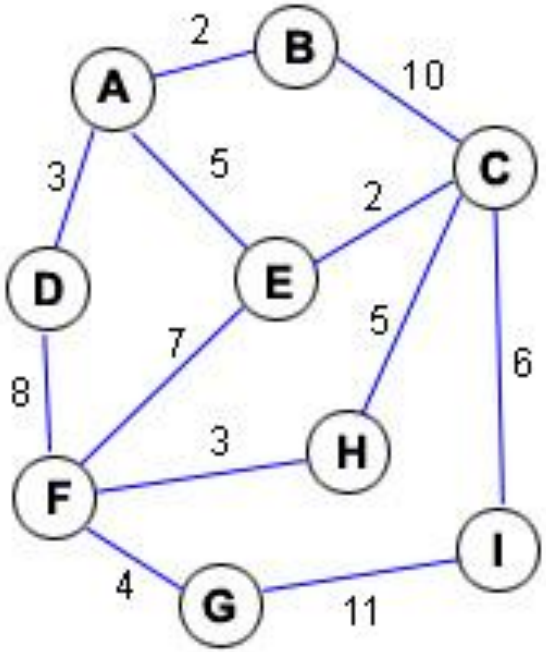
Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от *a* до других вершин пока неизвестны. Все вершины графа помечаются как непосещённые.


1
2
3
4
5
6
1
4
9
2
7
2
8

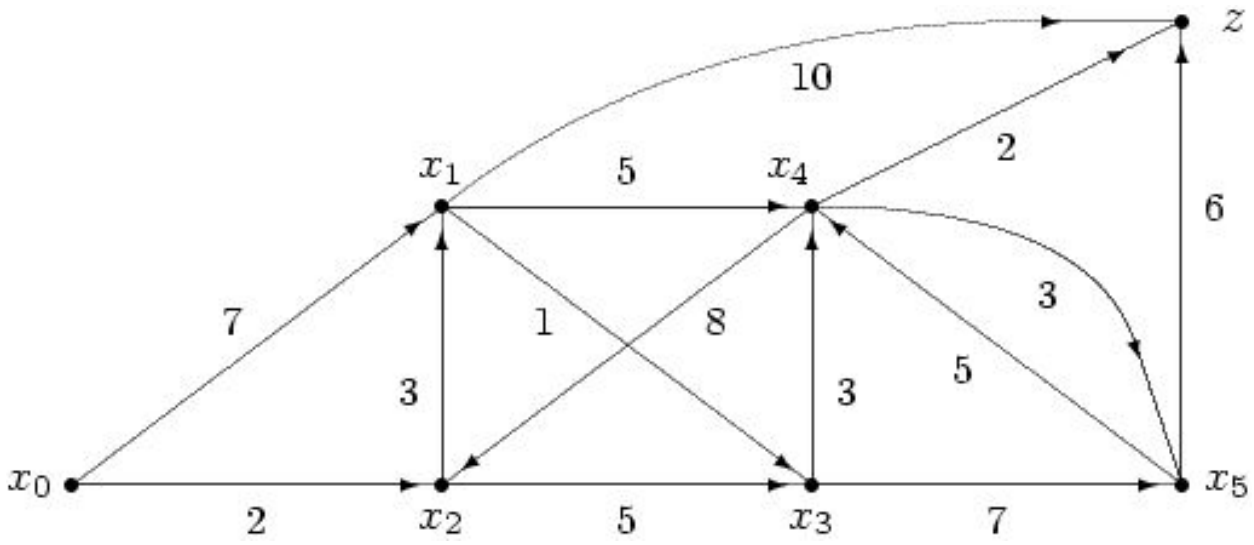
ШАГ АЛГОРИТМА

- Если все вершины посещены, алгоритм завершается.
- В противном случае, из ещё не посещённых вершин выбирается вершина *u*, имеющая минимальную метку.
  - Рассматриваем всевозможные маршруты из *u*. Вершины, в которые ведут рёбра из *u*, называются *соседями* этой вершины. Для каждого соседа вершины *u*, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки *u* и длины ребра, соединяющего *u* с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины.
  - Рассмотрев всех соседей, пометим вершину *u* как посещенную и повторим шаг алгоритма.


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

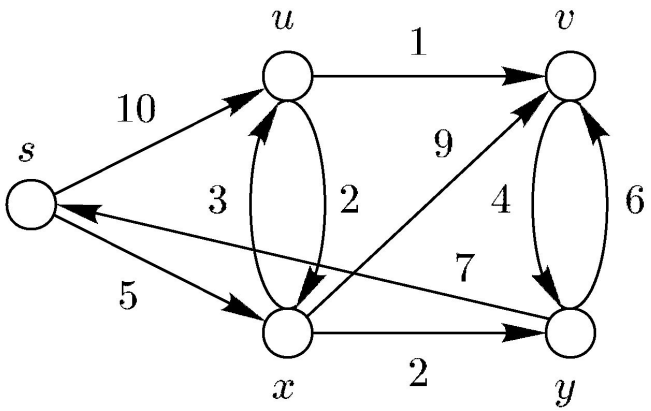

A
B
C
D
E
F
G
H
I
2
10
3
5
2
7
5
6
8
3
4
11

ПРИМЕР ПОИСКА КРАТЧАЙШЕГО ПУТИ НА ГРАФЕ (АЛГОРИТМ ДЕЙКСТРЫ)



x_0	x_1	x_2	x_3	x_4	x_5	z
0	∞	∞	∞	∞	∞	∞
	7	2	∞	∞	∞	∞
	5		7	∞	∞	∞
			6	10	∞	15
				9	13	15
					12	11

Задача. Для орграфа найти длины кратчайших путей от вершины **s** ко всем остальным вершинам и восстановить кратчайшие пути к ним.



s	x	y	u	v
0	∞	∞	∞	∞
0	**5**	∞	10	∞
0	**5**	**7**	8	14
0	**5**	**7**	**8**	13
0	**5**	**7**	**8**	**9**

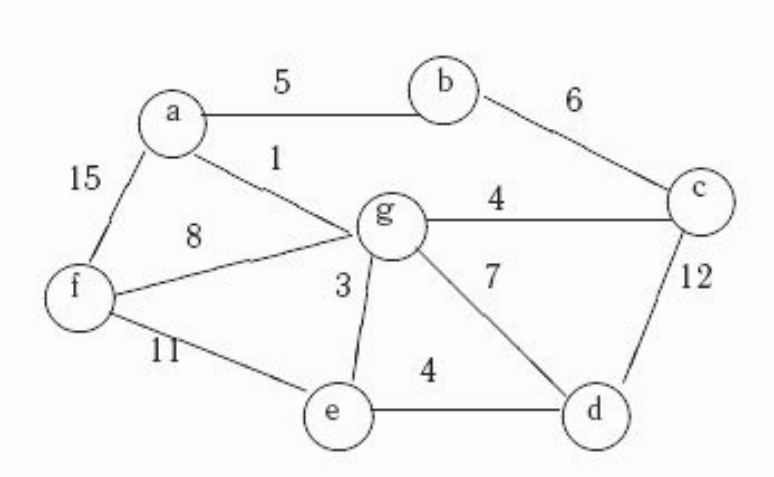
длины кратчайших путей от вершины s до остальных вершин будут равны

$$d(s,x)=5; \quad d(s,y)=7; \quad d(s,u)=8; \quad d(s,v)=9.$$

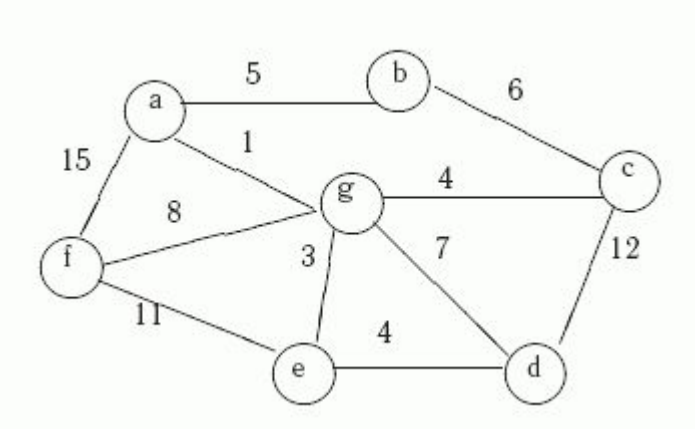
Кратчайшие пути будут соответственно равны

$$s \xrightarrow{5} x; \quad s \xrightarrow{5} x \xrightarrow{2} y; \quad s \xrightarrow{5} x \xrightarrow{3} u; \quad s \xrightarrow{5} x \xrightarrow{3} u \xrightarrow{1} v.$$

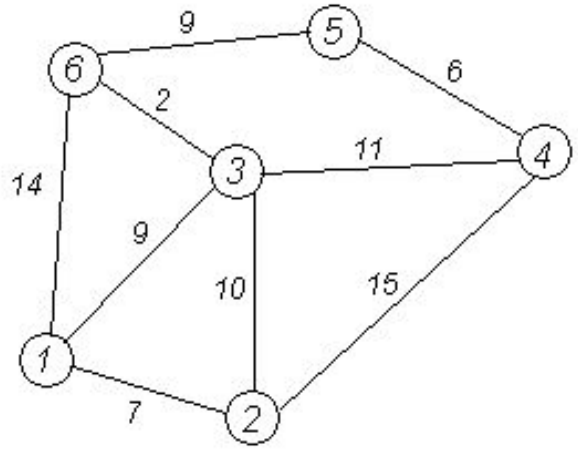
- **Задача**. Для н-графа найти длины кратчайших путей от вершины **a** ко всем остальным вершинам и восстановить кратчайшие пути к ним.

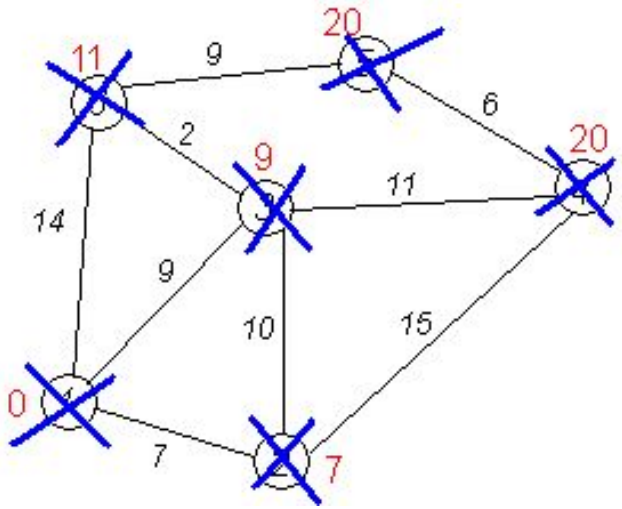


- **Задача**. Для н-графа найти длины кратчайших путей от вершины **a** ко всем остальным вершинам и восстановить кратчайшие пути к ним.



- **Задача**. Для н-графа найти длины кратчайших путей от вершины 1 ко всем остальным вершинам и восстановить кратчайшие пути к ним.





d(1,2)=7
d(1,3)=9
d(1,4)=2
0
d(1,5)=2
0
d(1,6)=1
1

Код программы алгоритма Дейкстры на Pascal:

```pascal
program DijkstraAlgorithm;
uses crt;
const V=6; inf=100000;
type vektor=array[1..v] of integer;
var start: integer;
const GR: array[1..V, 1..V] of integer=(
(0, 1, 4, 0, 2, 0),
(0, 0, 0, 9, 0, 0),
(4, 0, 0, 7, 0, 0),
(0, 9, 7, 0, 0, 2),
(0, 0, 0, 0, 0, 8),
(0, 0, 0, 0, 0, 0));
{
procedure Dijkstra(GR: array[1..V, 1..V]
var count, index, i, u, m, min: integer;
distance: vektor;
visited: array[1..V] of boolean;
begin
m:=st;
for i:=1 to V do
begin
distance[i]:=inf; visited[i]:=false;
end;
distance[st]:=0;
for count:=1 to V-1 do
begin
min:=inf;
for i:=1 to V do
if (not visited[i]) and (distance[i]<=min) then
begin
min:=distance[i]; index:=i;
end;
u:=index;
visited[u]:=true;
for i:=1 to V do
if (not visited[i]) and (GR[u, i]<>0) and (distance[u]<>inf) and
(distance[u]+GR[u, i]<distance[i]) then
distance[i]:=distance[u]+GR[u, i];
end;
write('Стоимость пути из начальной вершины до остальных:'); writel
for i:=1 to V do
if distance[i]<>inf then
writeln(m,' > ', i,' = ', distance[i])
else writeln(m,' > ', i,' = ', 'маршрут недоступен');
end;
{
{главное тело программы}
begin
clrscr;
write('Начальная вершина >> '); read(start);
Dijkstra(GR, start);
end.
```

```
progran
uses crt
const V
type ve
var star
const G
(0, 1, 4,
(0, 0, 0,
(4, 0, 0,
(0, 9, 7,
(0, 0, 0,
(0, 0, 0,
{_____
procedu
var cou
distance
visited:
begin
m:=st;
for i:=1
begin
distance
end;
distance
for cour
begin
min:=in
for i:=1 to V do
if (not visited[i]) and (distance[i]<=min) then
begin
min:=distance[i]; index:=i;
end;
u:=index;
visited[u]:=true;
for i:=1 to V do
if (not visited[i]) and (GR[u, i]<>0) and (distance[u]<>inf) and
(distance[u]+GR[u, i]<distance[i]) then
distance[i]:=distance[u]+GR[u, i];
end;
write('Стоимость пути из начальной вершины до остальных:'); writeln;
for i:=1 to V do
if distance[i]<>inf then
writeln(m,' > ', i,' = ', distance[i])
else writeln(m,' > ', i,' = ', 'маршрут недоступен');
end;
{главное тело программы}
begin
clrscr;
write('Начальная вершина >> '); read(start);
Dijkstra(GR, start);
end.
```

Алгоритм Крускала

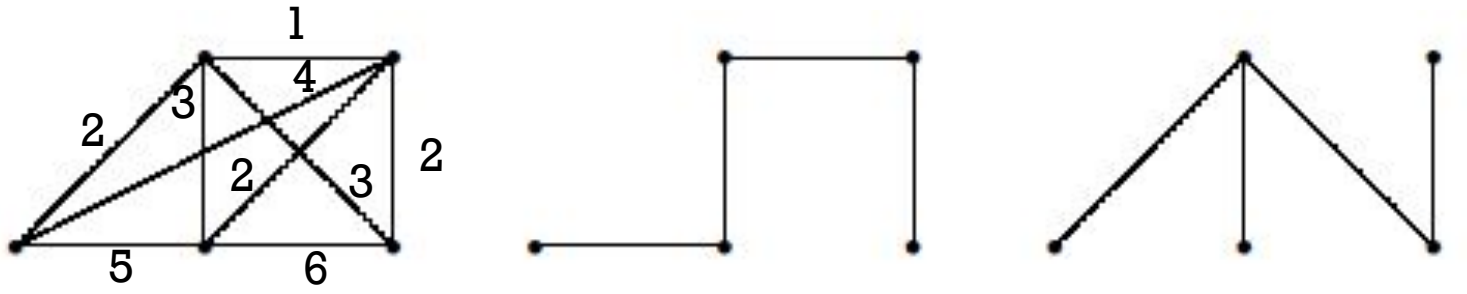
АЛГОРИТМ КРУСКАЛА (КРАСКАЛА)

- **Крускал, Джозеф Б.** (**Kruskal, Joseph B.**)
  29.01.1928 –19.09.2010
  Почетный профессор. Bell Labs, Lucent Technologies, Room 2C-281, Murray Hill, NJ 07974,

Алгоритм Крускала (или **алгоритм Краскала**) — алгоритм построения минимального остовного дерева взвешенного графа. Алгоритм впервые описан Джозефом Крускалом в 1956 году.

Минимальное остовное дерево

Минимальным остовным деревом называется остовное дерево с минимальным общим весом его ребер.



Пример связного графа и двух его остовных деревьев

ФОРМУЛИРОВКА ЗАДАЧИ

Для связного взвешенного графа G=(V,E) построить минимальный остов S=(V,T).

Вход: *связный граф* G=(V,E) и *функция* стоимости (весов) ребер c: E -> R.

Выход: *минимальный остов* S=(V,T).

ШАГИ АЛГОРИТМА

1. Создается список ребер E, содержащий длину ребра, номер исходной вершины ребра, номер конечной вершины ребра.
2. Данный список упорядочивается в порядке возрастания длин ребер.
3. Просматривается список E и выбирается из него ребро с минимальной длиной, еще не включенное в результирующее дерево S и не образующее цикла с уже построенными ребрами.
4. Если все вершины графа включены в дерево S и количество ребер на единицу меньше количества вершин, то алгоритм свою работу закончил. В противном случае осуществляется возврат к пункту 3.

ШАГИ АЛГОРИТМА

1. Пусть E содержит m ребер.
2. Упорядочим их *по* возрастанию стоимостей:

$$E = \{e_1, e_2, \ldots, e_i, \ldots, e_m\} \text{ так, что } c(e_1) \leq c(e_2) \leq \ldots \leq c(e_i) \leq \ldots \leq c(e_m)$$

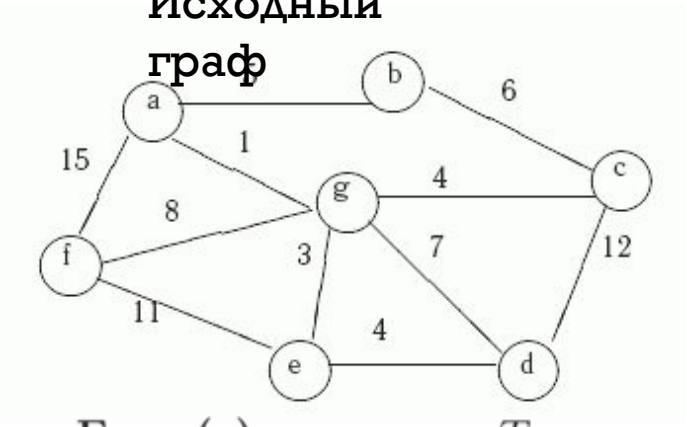
3. Последовательно для каждого i =1, … , m определим множество ребер T_i:

$$T_1 = \{e_1\}; \qquad \ldots T_i = \begin{cases} T_{i-1} \cup \{e_i\}, & \text{если во множестве } T_{i-1} \cup \{e_i\} \text{ нет циклов} \\ T_{i-1}, & \text{в противном случае} \end{cases}$$

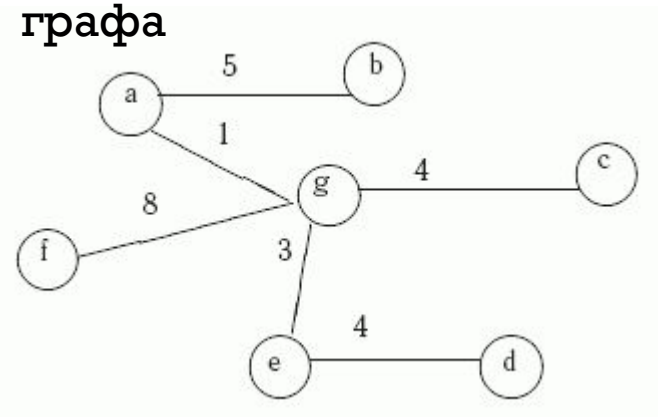
4. Положим $T=T_m$. Выдать в качестве результата *граф* S=(V,T).

ПРИМЕР РАБОТЫ АЛГОРИТМА КРУСКАЛА

Исходный граф



Минимальный остов графа

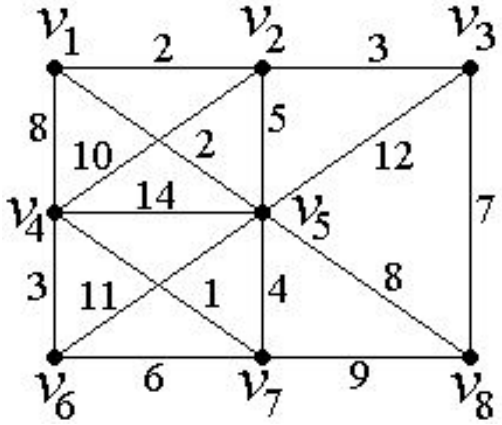


E	c(e)	входит	T_i
(a,g)	1	+	$T_1 = \{(a,g)\}$
(g,e)	3	+	$T_2 = \{(a,g),(g,e)\}$
(g,c)	4	+	$T_3 = \{(a,g),(g,e),(g,c)\}$
(e,d)	4	+	$T_4 = \{(a,g),(g,e),(g,c),(e,d)\}$
(a,b)	5	+	$T_5 = \{(a,g),(g,e),(g,c),(e,d),(a,b)\}$
(b,c)	6	−	$T_6 = T_5$
$(d,g))$	7	−	$T_7 = T_6$
(f,g)	8	+	$T_8 = \{(a,g),(g,e),(g,c),(e,d),(a,b),(f,g)\}$
(e,f)	11	−	$T_9 = T_8$
(c,d)	12	−	$T_{10} = T_9$
(a,f)	15	−	$T_{11} = T_{10}$

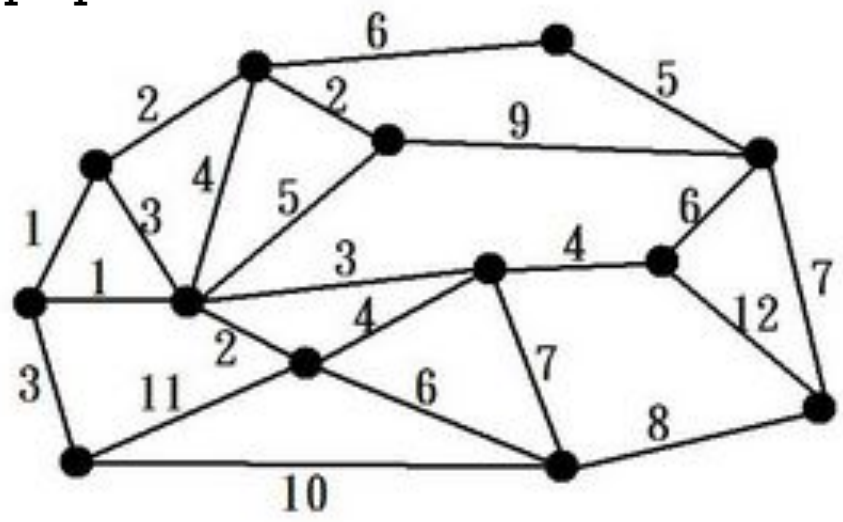
минимальный остов S=(V,T), где $T=T_8$ ={ (a,g), (g,e), (g,c), (e,d), (a,b), (f,g)}

ПРИМЕР РАБОТЫ АЛГОРИТМА КРУСКАЛА

Найти минимальный остов неориентированного взвешенного графа.

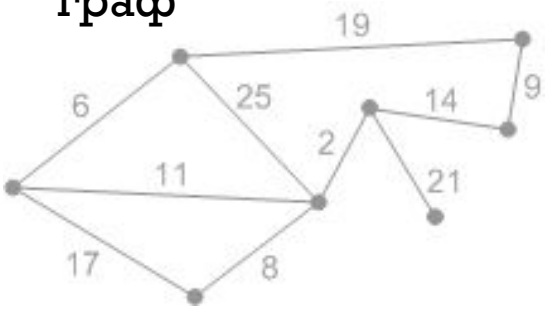


Д/З Найти минимальный остов неориентированного взвешенного графа.

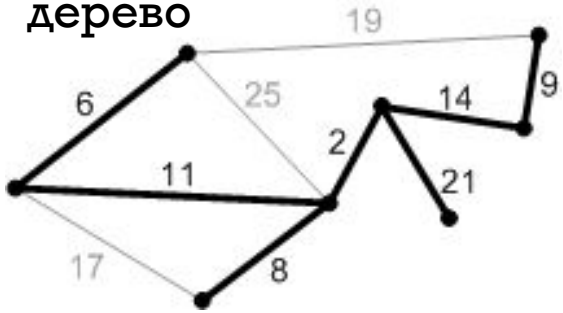


Задача. Найти минимальный остов неориентированного взвешенного графа.

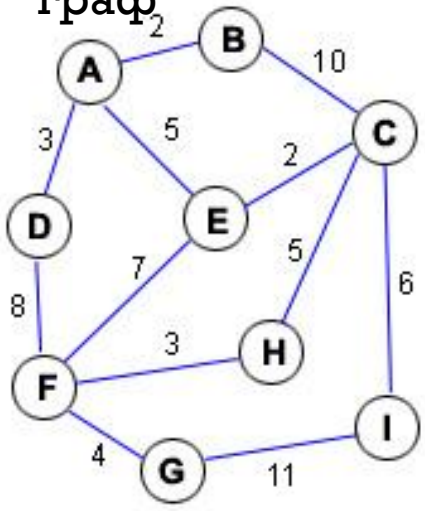
Исходный граф



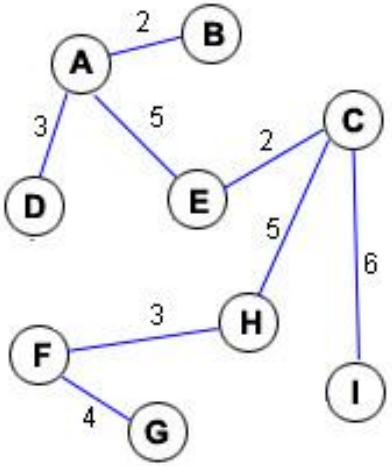
Минимальное остовное дерево



Исходный граф



Минимальное остовное дерево



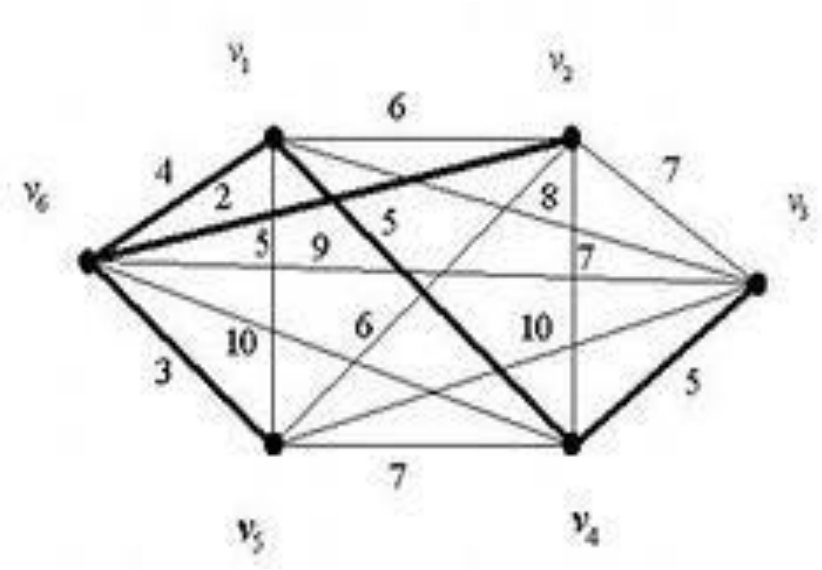
- Вариант 1

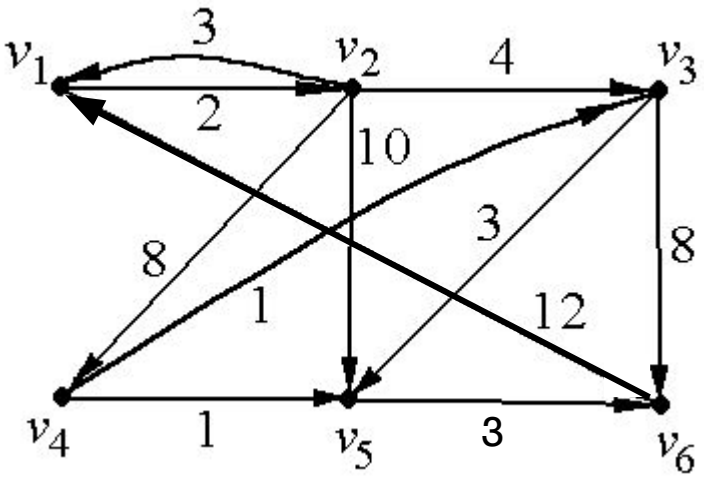
1. Найти минимальный остов взвешенного графа
2. Найти минимальное расстояние от вершины v3 до остальных вершин

- Вариант 2

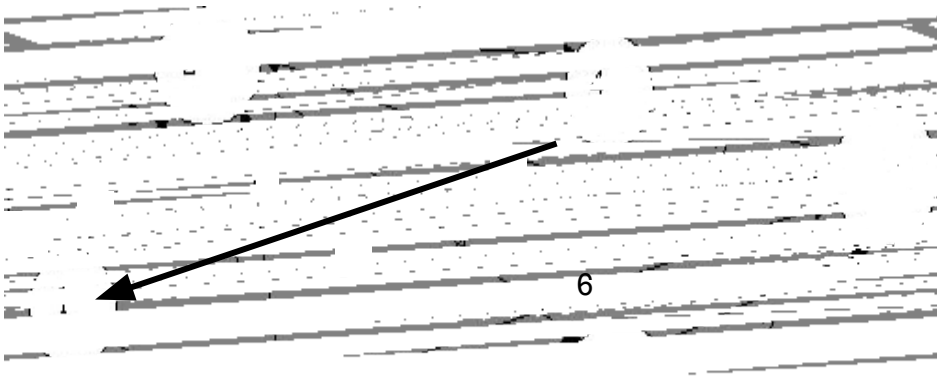
1. Найти минимальное расстояние от вершины 6 до остальных вершин
2. Найти минимальный остов взвешенного графа

Вариант 1

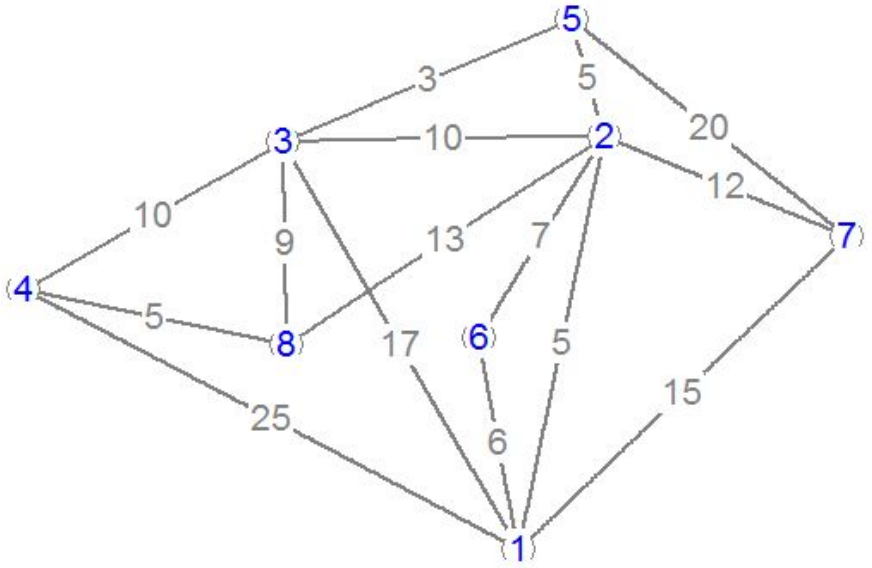


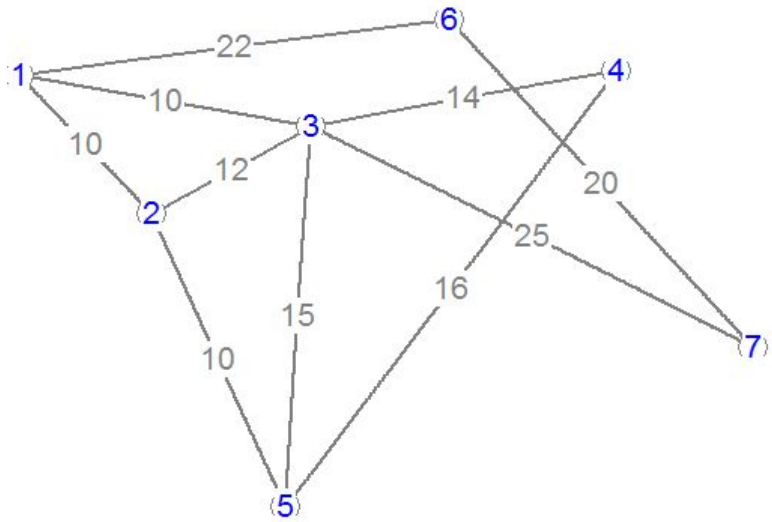


Вариант 2



4




6
22
1
4
10
14
3
10
12
20
2
25
16
15
7
10
5