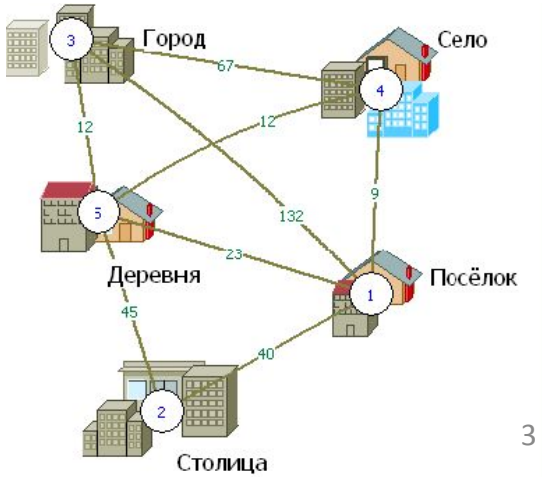
Алгоритмы на графах

Применение теории графов
Определения

Применение алгоритмов на графах

- КАРТОГРАФИЧЕСКИЕ СИСТЕМЫ

- Поиск

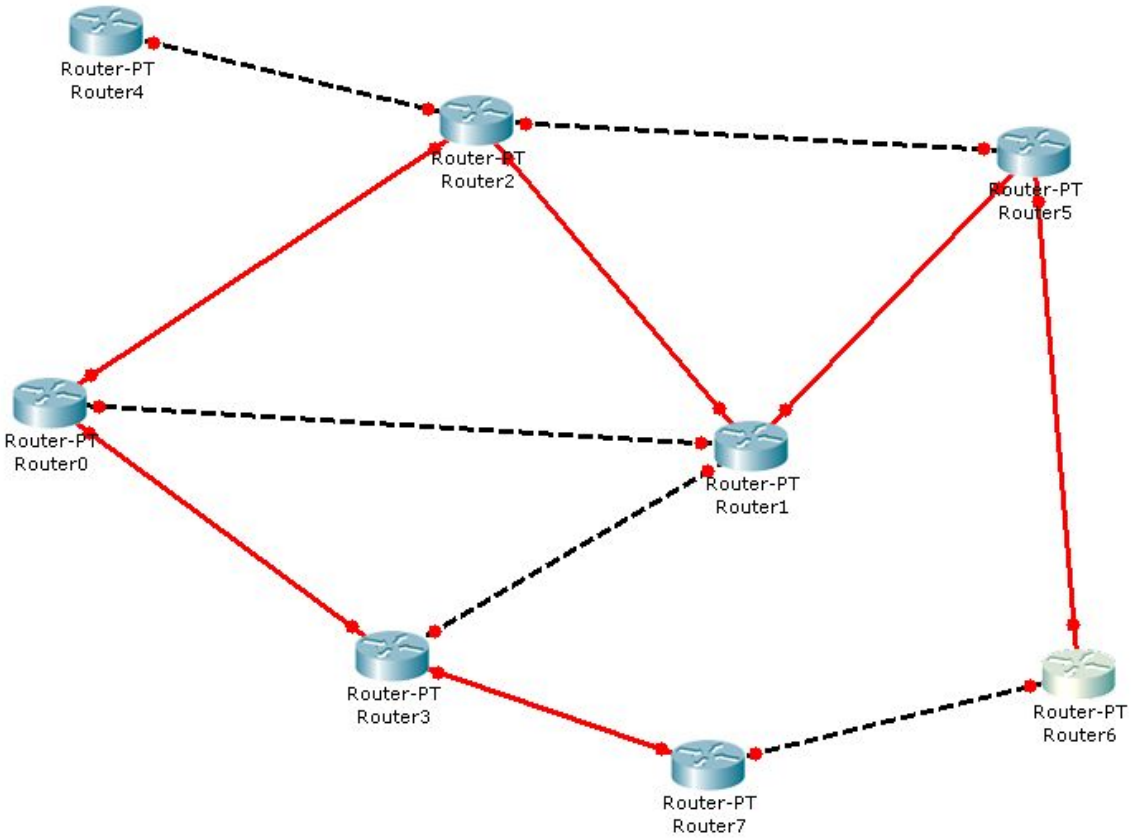


Применение алгоритмов на графах

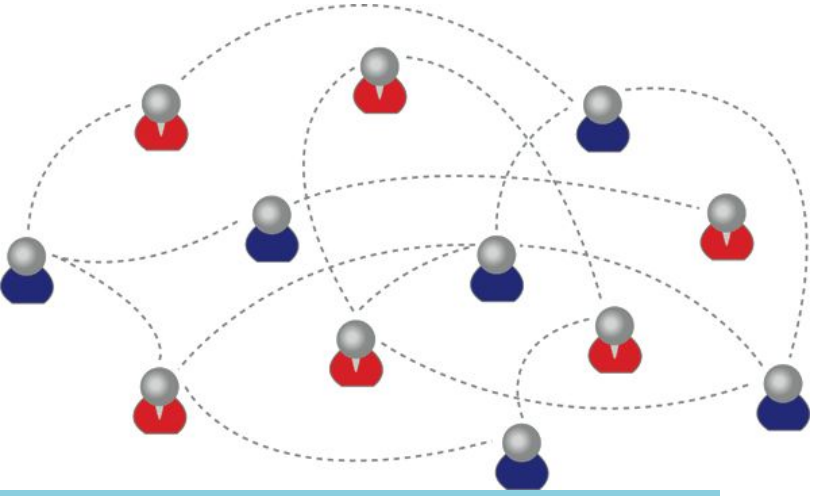
- Различные приложения для компьютерных игр

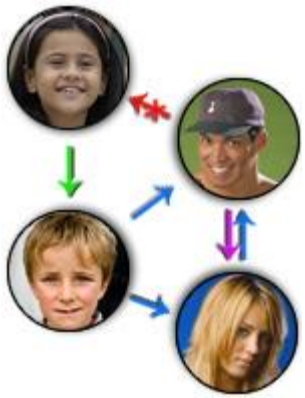
Применение алгоритмов на графах

- Маршрутизация в сетях

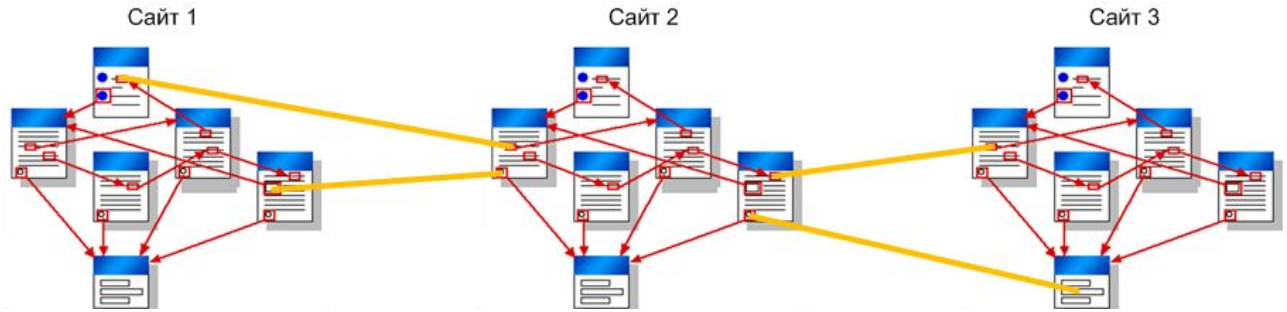


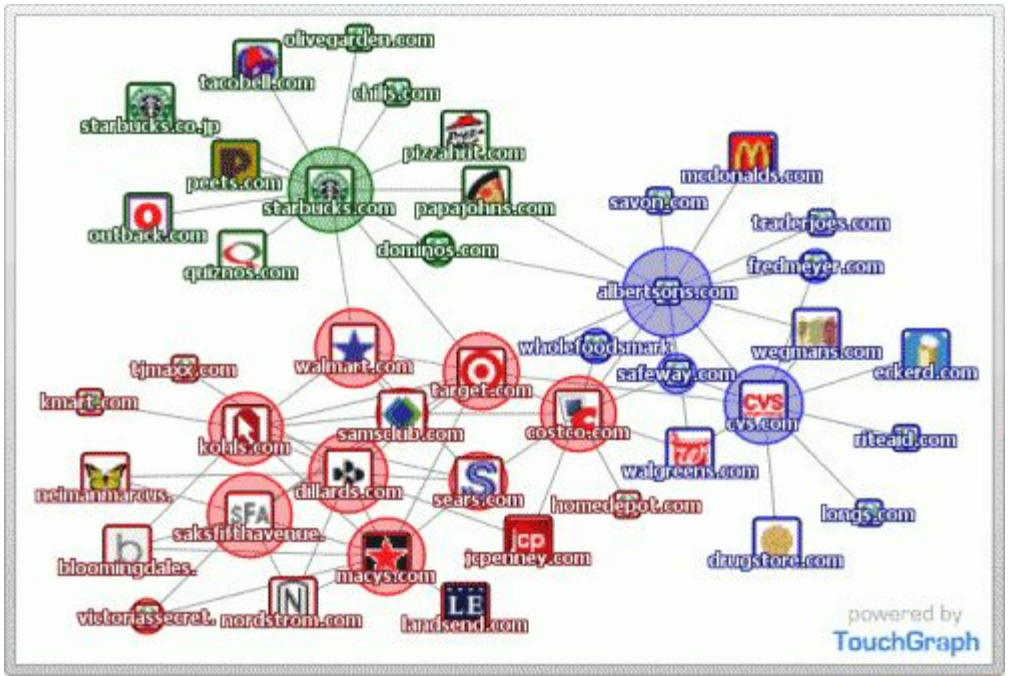
Социальные сети





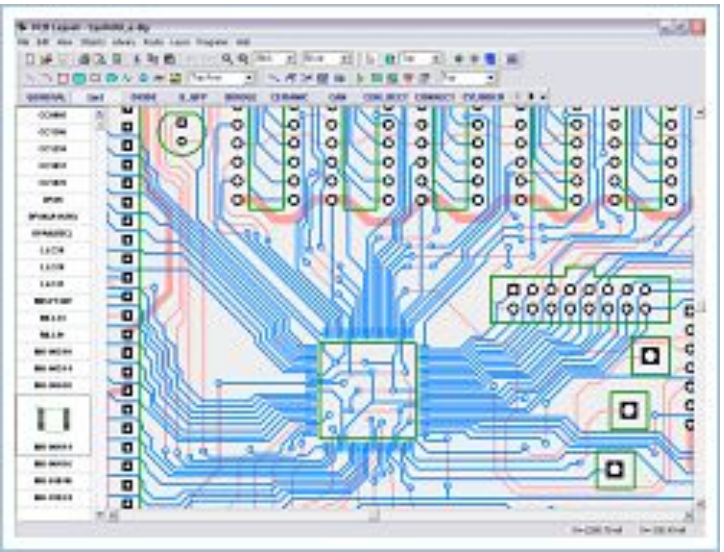
Поисковые системы

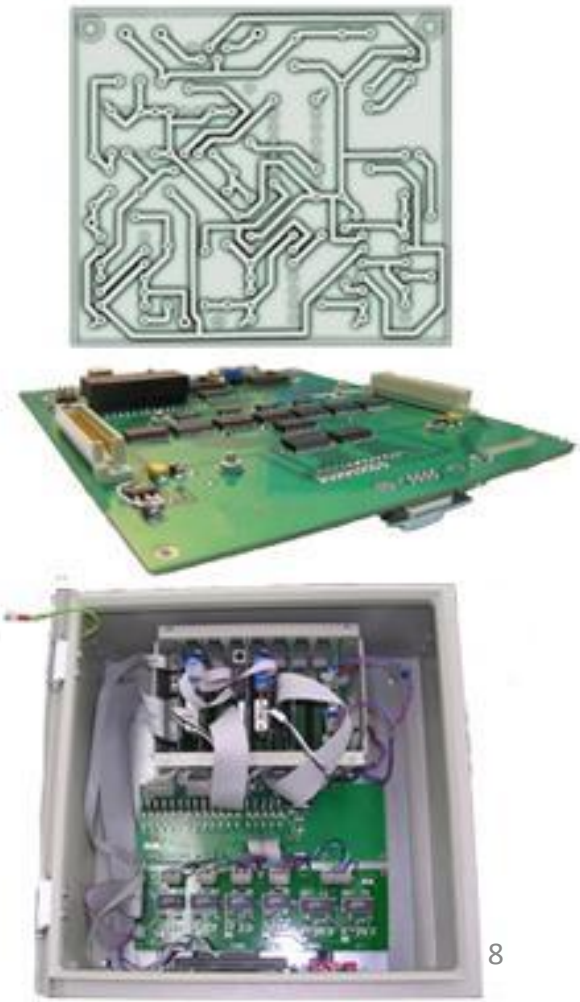




Применение алгоритмов на графах

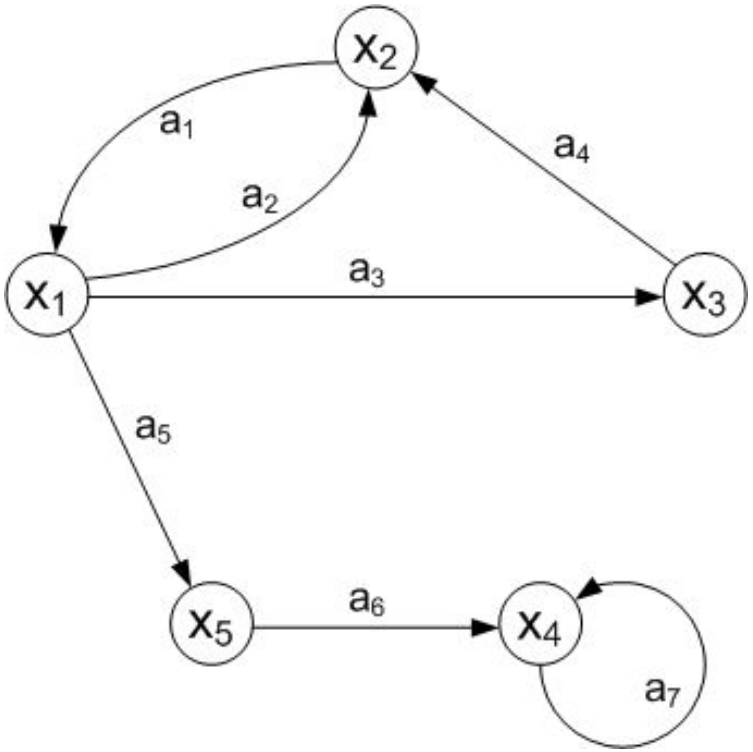
- Автоматизированная трассировка (разводка) печатных плат.





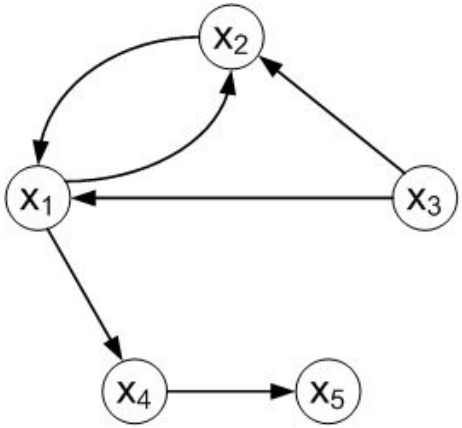
Определение графа

Графом G называется пара (X,A), где X(G) множество точек или ***вершин*** *(x_1, x_2, . . ., x_n)*, а A(G) множество *(a_1,a_2, . . ., a_m)* линий или ***ребер*** вида $a_k(x_i,x_j)$, соединяющих между собой все или часть этих точек из множества X.

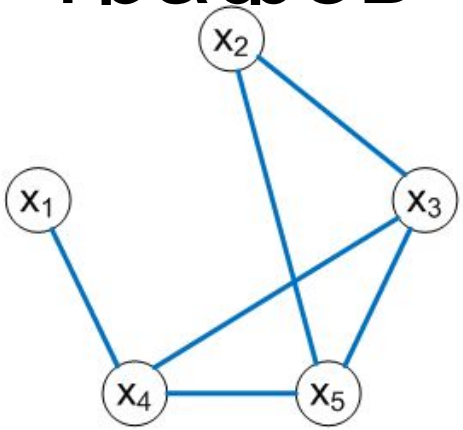


Множество вершин X	**Множество ребер A**
x_1	$a_1(x_2,x_1)$
x_2	$a_2(x_1,x_2)$
x_3	$a_3(x_1,x_3)$
x_4	$a_4(x_3,x_2)$
x_5	$a_5(x_1,x_5)$
	$a_6(x_5,x_4)$
	$a_7(x_4,x_4)$

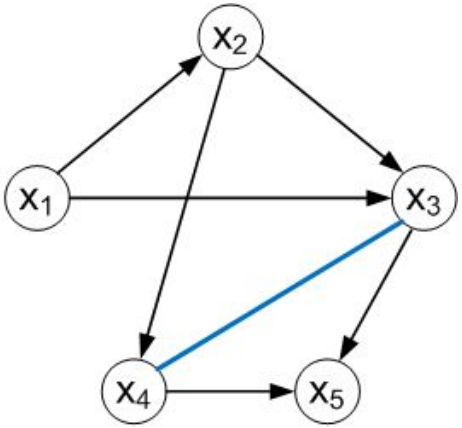
Задание ориентированных графов



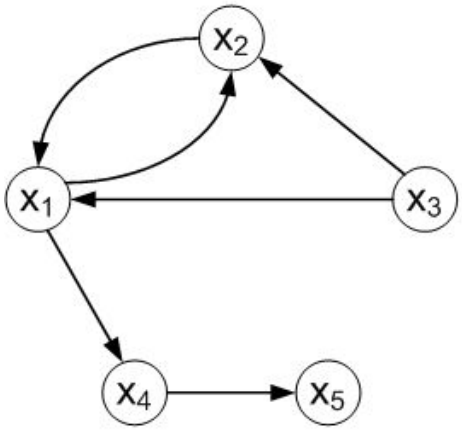
Ориентированный граф
(а)



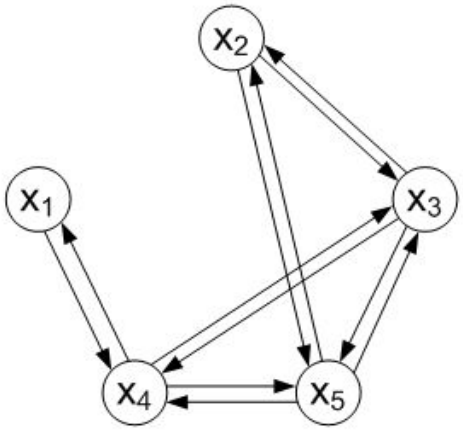
Неориентированный граф
(б)



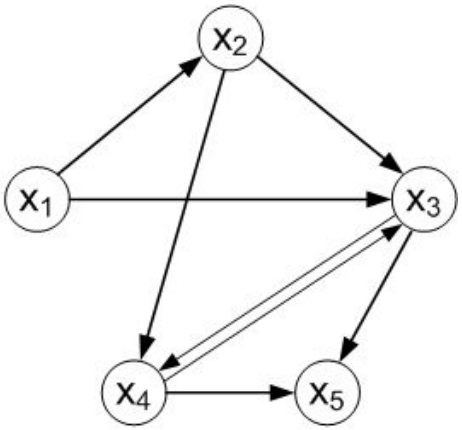
Смешанный граф
(в)



(а)



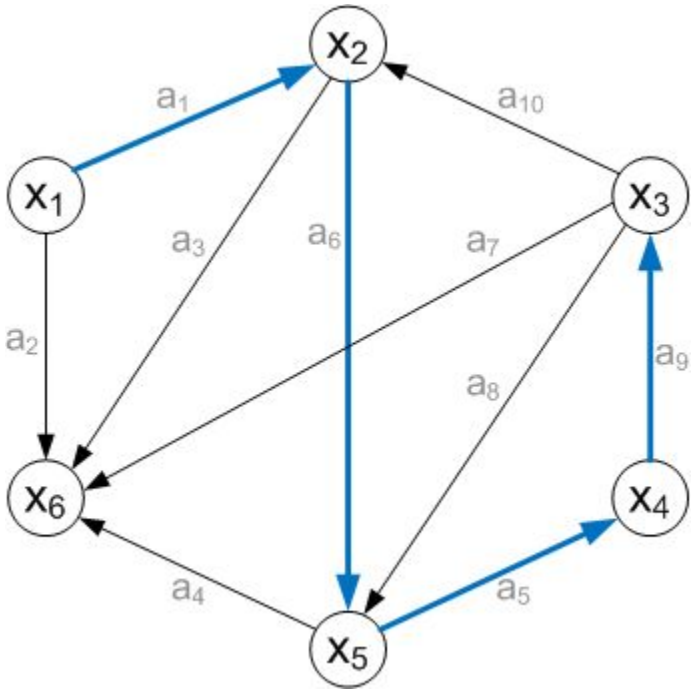
(б)



(в)

Пути и маршруты

- ***Путем*** (или *ориентированным маршрутом)* ориентированного графа называется последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей дуги.

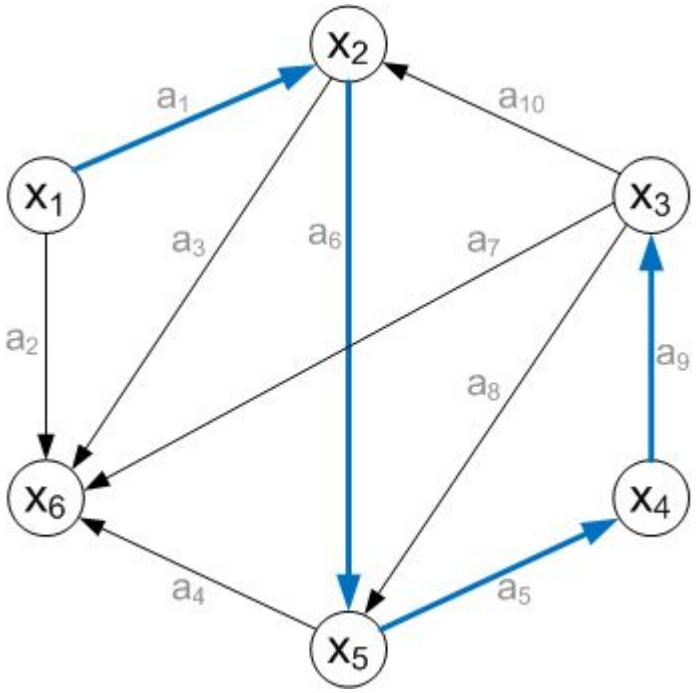


Пути на графе G

1) a_1, a_6, a_5, a_9
2) a_6, a_5, a_9, a_8, a_4
3) $a_1, a_6, a_5, a_9, a_{10}, a_6, a_4$

Орцепи

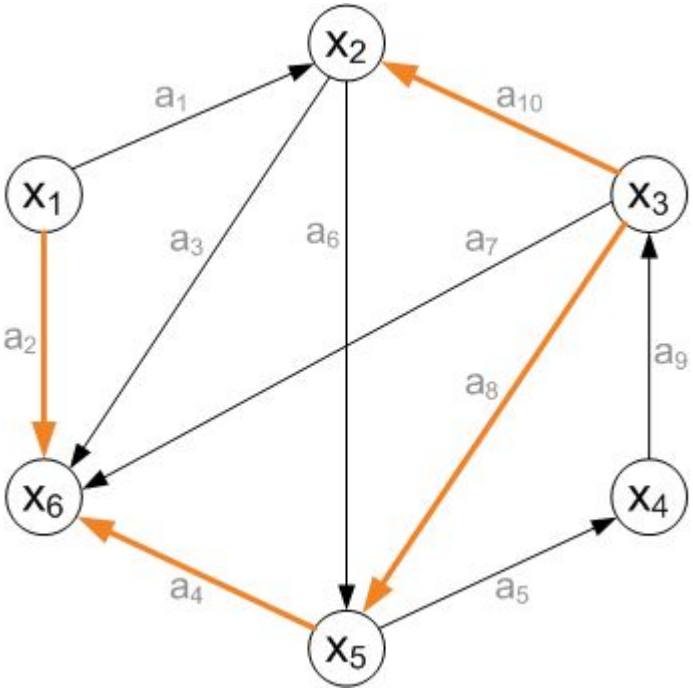
- ***Ориентированной цепью*** (или, короче, *орцепъю)* называется такой путь, в котором каждая дуга используется не больше одного раза.
- ***Простой орцепъю*** называется такой путь, в котором каждая вершина используется не более одного раза.



1) a_1, a_6, a_5, a_9 – простая орцепь
2) a_6, a_5, a_9, a_8, a_4 - орцепь
3) $a_1, \mathbf{a_6}, a_5, a_9, a_{10}, \mathbf{a_6}, a_4$

Маршрут

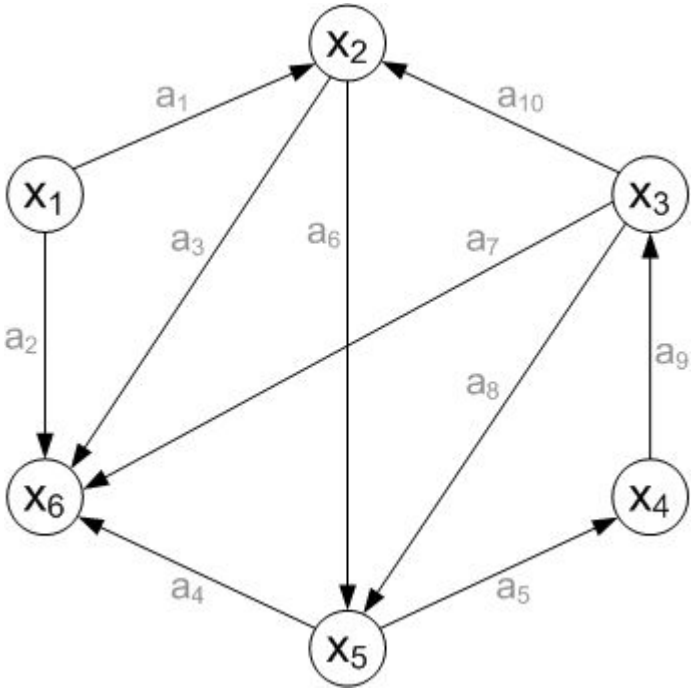
- *Маршрут* - это последовательность ребер $a_1, a_2, \ldots, a_q$ в которой каждое ребро a_i исключением, возможно, первого и последнего ребер, связано с ребрами a_{i-1} и a_{i+1} своими двумя концевыми вершинами.



Маршруты на графе G

1) $\overline{a_2}, \overline{a_4}, \overline{a_8}, \overline{a_{10}}$
2) $\overline{a_2}, \overline{a_7}, \overline{a_8}, \overline{a_4}, \overline{a_3}$
3) $\overline{a_{10}}, \overline{a_7}, \overline{a_4}, \overline{a_8}, \overline{a_7}, \overline{a_2}$

Пути и маршруты (продолжение)



Пути в виде посл-ти вершин

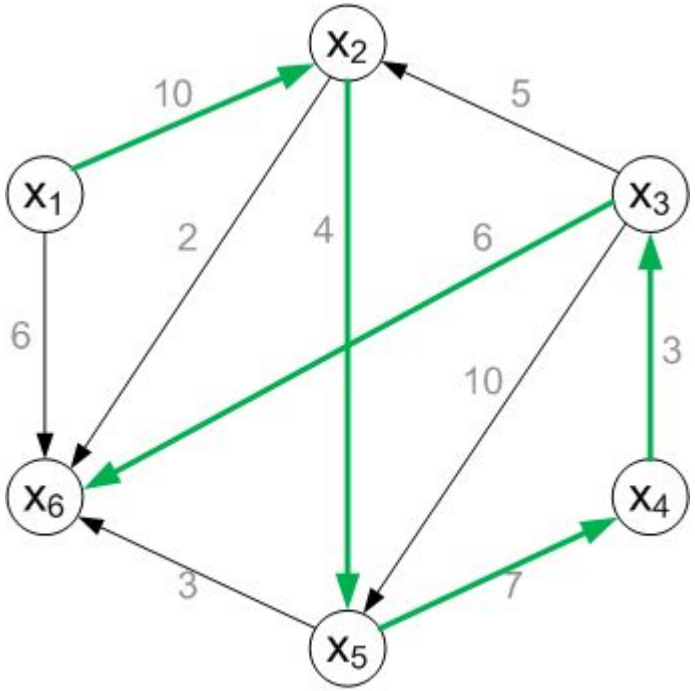
1) $a_1, a_6, a_5, a_9 \rightarrow x_1, x_2, x_5, x_4, x_3$

Маршруты в виде посл-ти вершин

2) $a_6, a_5, a_9, a_8, a_4 \rightarrow x_2, \mathbf{x_5}, x_4, x_3, \mathbf{x_5}, x_6,$

Веса и длина пути

- Если дуге (x_i,x_j) графа G ставится в соответствие некоторое число $c_{i,j}$, называемое *весом,* стоимостью или ценой дуги, тогда такой граф G называется *графом со взвешенными дугами*.



Граф со взвешенными дугами

Путь $\mu = \overline{a_1}, \overline{a_2}, \ldots, \overline{a_q}$

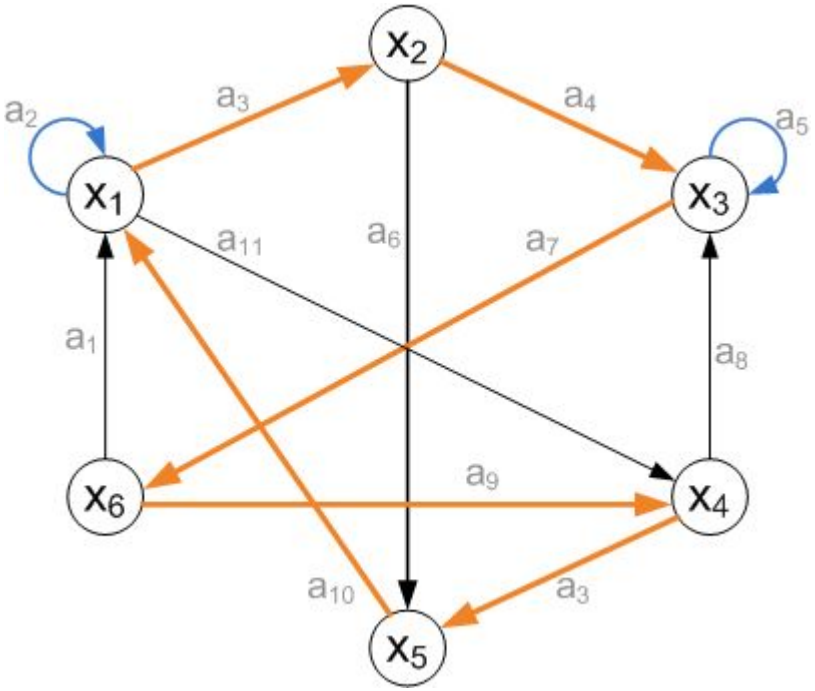
$$l(\mu) = \sum_{(x_i,x_j)\in\mu} c_{ij}.$$

Стоимость $l(\mu) = 10+4+7+3+6 = 30$

Длина $\mu = 1+1+1+1+1 = 5$

Петли, ориентированные циклы и циклы

- *Петлей* называется дуга, начальная и конечная вершины которой совпадают (a_2, a_5).
- Путь $a_1, a_2, \ldots, a_q$ называется *замкнутым,* если в нем начальная вершина дуги a_1 совпадает с конечной вершиной дуги a_q.



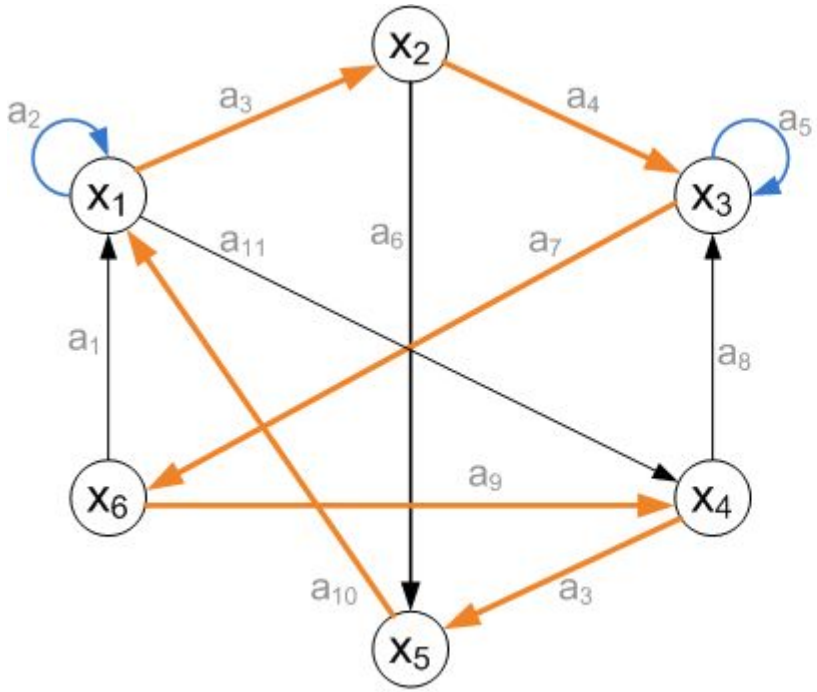
Замкнутые пути

1) a_3, a_6, a_{11}
2) $a_{10}, a_3, a_4, a_7, a_1, a_{11}, a_3$
3) $a_3, a_4, a_7, a_9, a_3, a_{10}$

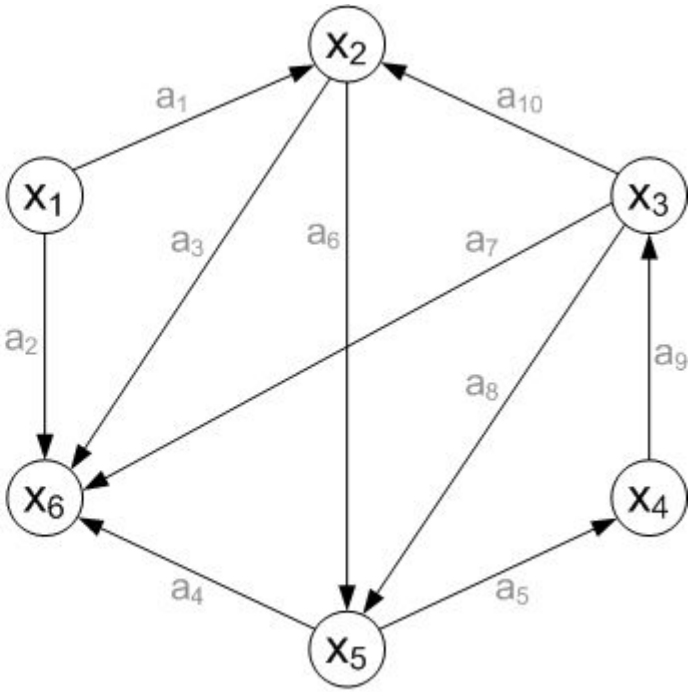
Замкнутые пути
1 и 2 – *контуры или простые орциклы*
3 – **гамельтонов** контур, который проходит через все вершины графа G.

Гамельтонов контур

- Контур, проходящий через все вершины графа, имеет особое значение и называется ***гамильтоновым*** контуром.



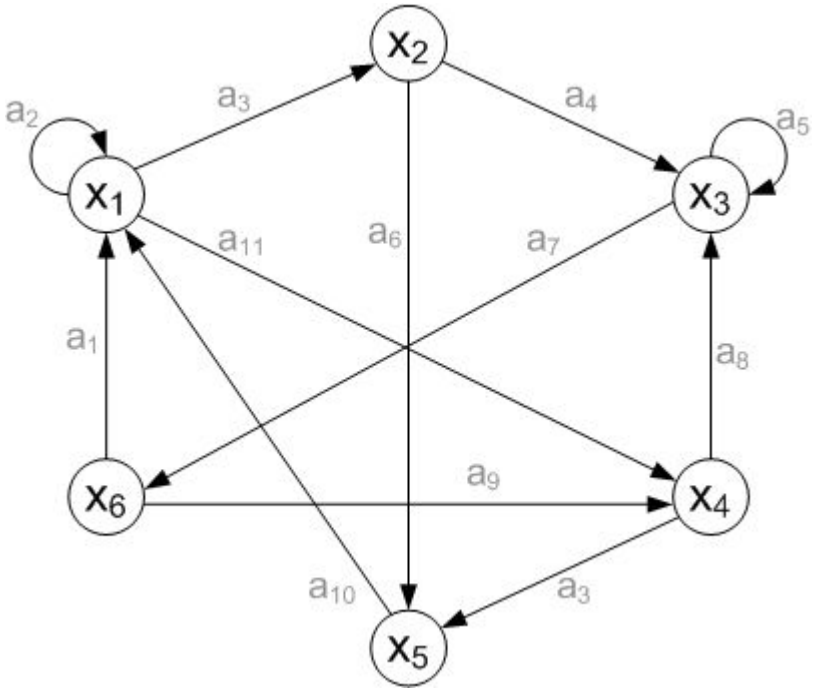
Есть гамельтонов контур



Нет гамельтонового контура

Замкнутый маршрут

- ***Замкнутый маршрут*** является неориентированным двойником замкнутого пути.
- Замкнутым маршрутом является маршрут $x_1, x_2, \ldots, x_q$ в котором совпадают начальная и конечная вершины, т. е. в котором $x_1 = x_q$.



Замкнутые маршруты

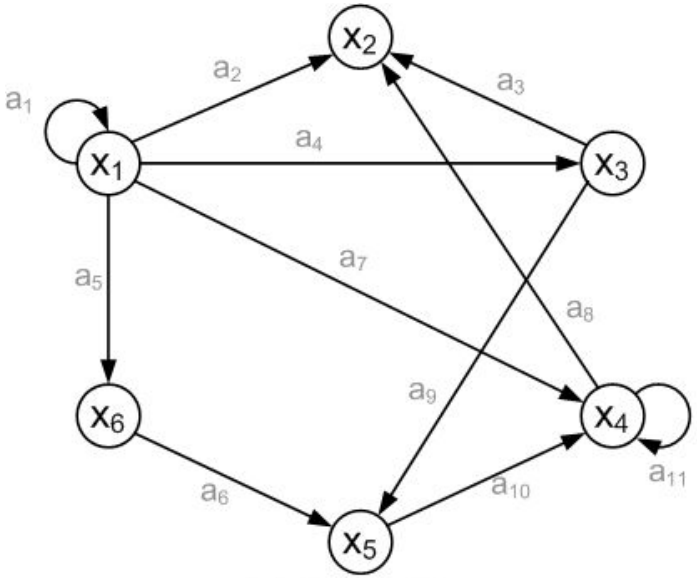
1) $\overline{a_1}, \overline{a_{11}}, \overline{a_9}$
2) $\overline{a_9}, \overline{a_1}, \overline{a_3}, \overline{a_4}, \overline{a_7}, \overline{a_1}, \overline{a_{12}}$

1) x_6, x_1, x_4, x_6
2) $x_4, x_6, x_1, x_2, x_3, x_6, x_1, x_4$

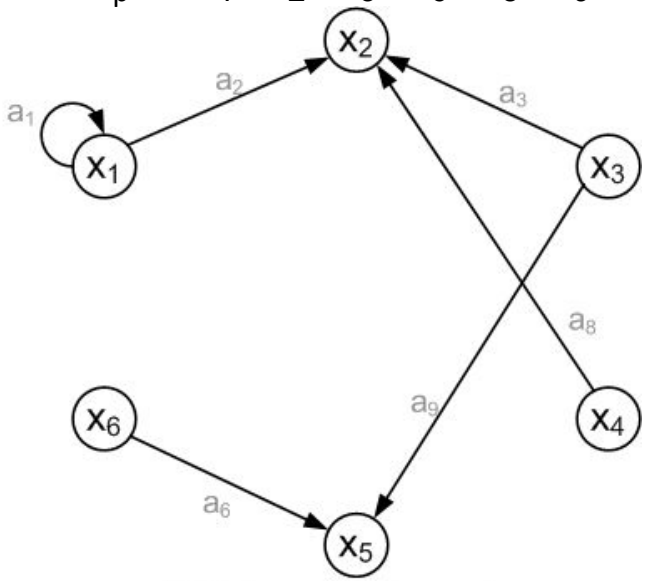
Подграфы (Остовный граф)

- Пусть дан граф $G=(X, A)$.
- ***Остовным подграфом*** G_p графа G называется граф (X, A_p), для которого $A_p \subset A$ исходного графа G. Т.о., *остовный подграф* имеет то же самое множество вершин, что и граф *G,* но множество дуг подграфа G_p является подмножеством множества дуг исходного графа.

$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}\}$ $A_p = \{a_1, a_2, a_3, a_6, a_8, a_9\}$



(а) Граф G

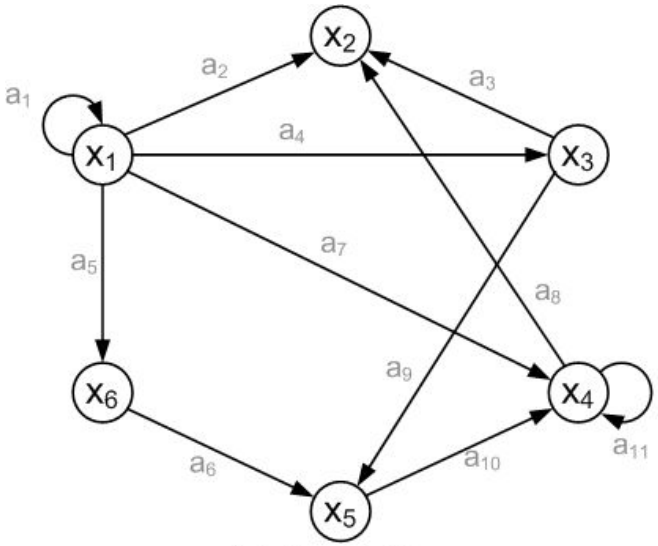


(б) Остовный граф

Порожденные графы

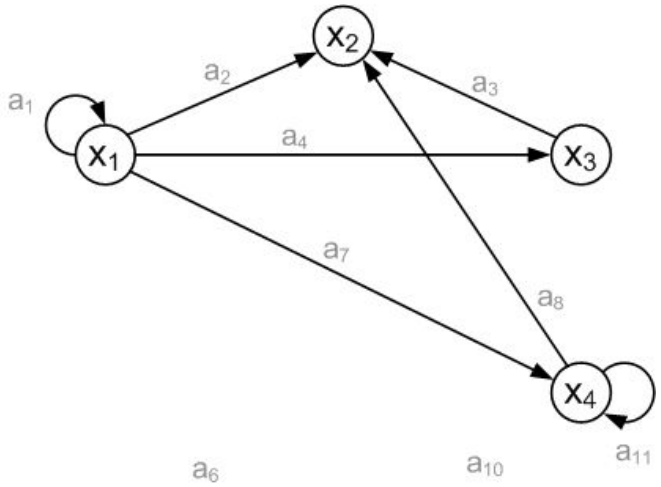
- Пусть дан граф $G = (X, K)$.
- ***Порожденным подграфом*** G_q, называется граф $G = (X_q, K_q)$, для которого X_q *подмножество множества* X и для каждой вершины $x_i \in X_q$, $A_q(x_i) = A_q(x_i) \cap X$
- Порожденный подграф состоит из подмножества вершин X_q множества вершин X исходного графа и всех таких дуг графа G, у которых конечные и начальные вершины принадлежат подмножеству X_q.

$X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$



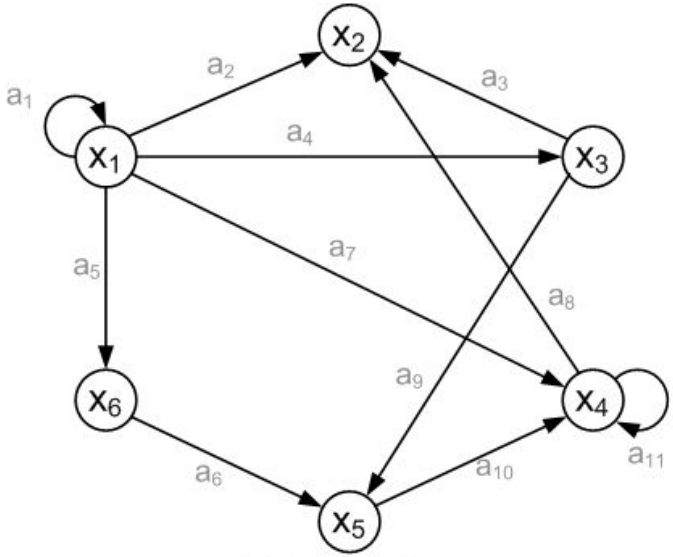
(а) Граф G

$X = \{x_1, x_2, x_3, x_4\}$

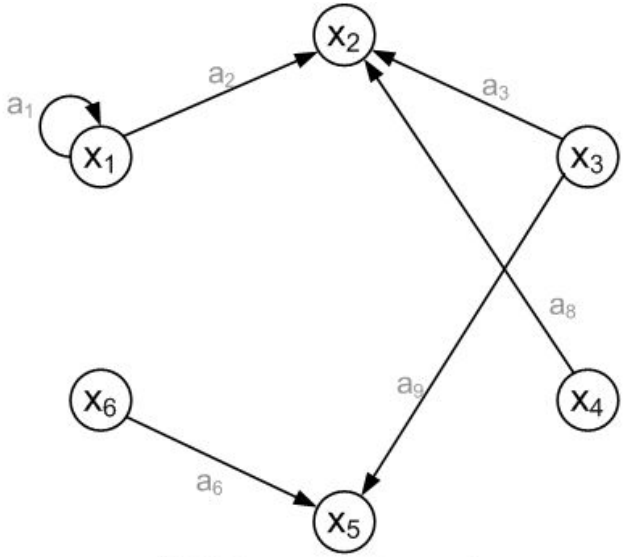


(в) Порожденный граф

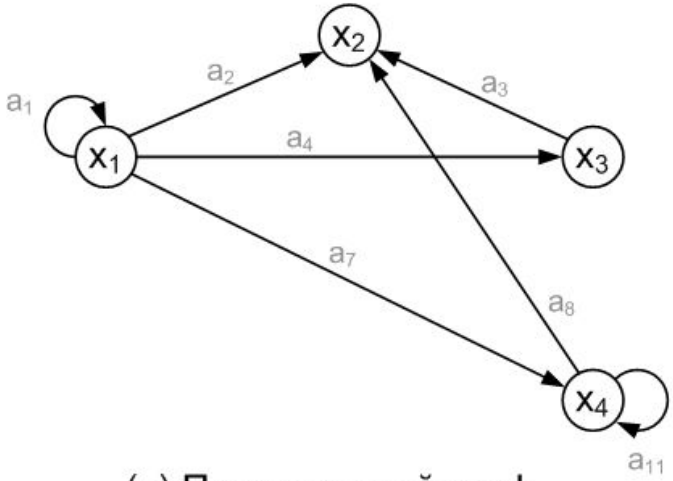
Пример подграфов



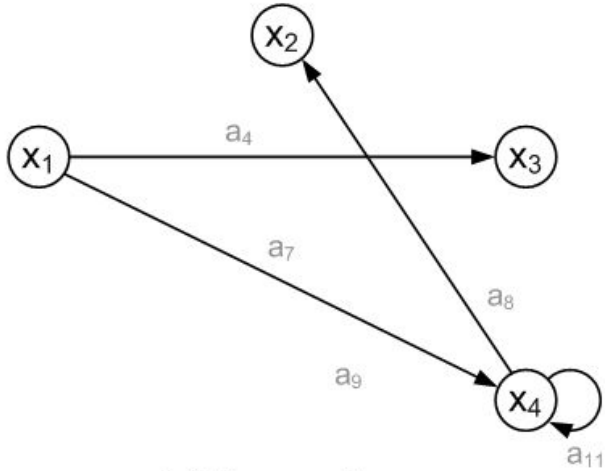
(а) Граф G



(б) Остовный граф



(в) Порожденный граф



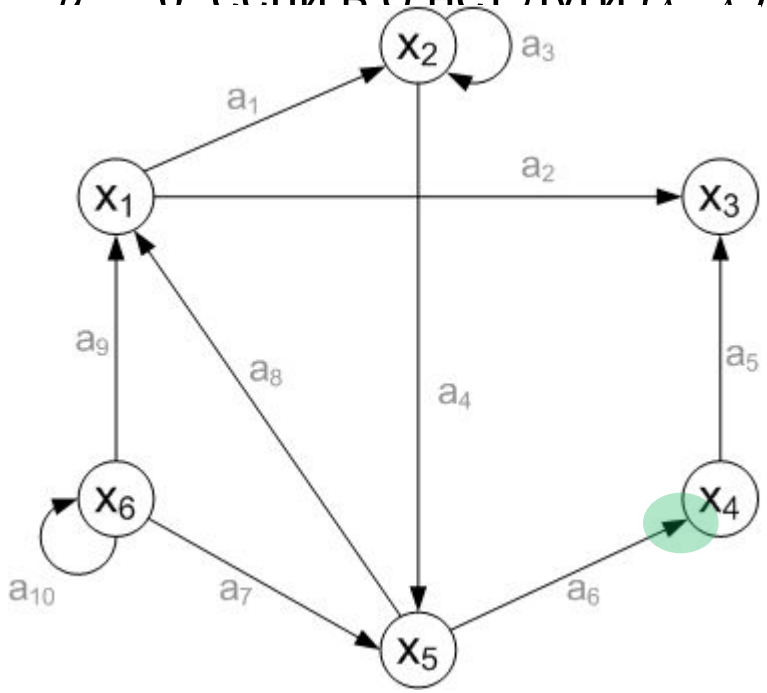
(г) Подграф

Хранение графов

Исправлено

Матрица смежности

- Пусть дан граф *G*, его *матрица смежности* обозначается через $A = |a_{ij}|$ и определяется следующим образом:
  - $a_{ij} = 1$, если в *G* существует дуга (x_i, x_j),
  - $a_{ij} = 0$, если в *G* нет дуги (x_i, x_j).



$A =$

	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	1	1	0	0	0
x_2	0	1	0	0	1	0
x_3	0	0	0	0	0	0
x_4	0	0	1	0	0	0
x_5	1	0	0	1	0	0
x_6	1	0	0	0	1	1

Полустепень исхода - строка

$d_{исход} = |K(x_5)| = 2$

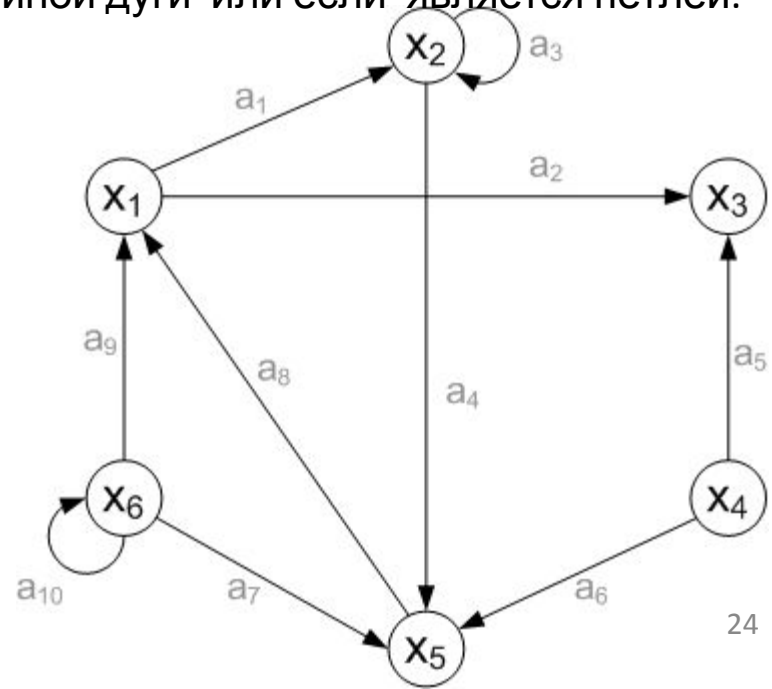
Полустепень исхода - столбец

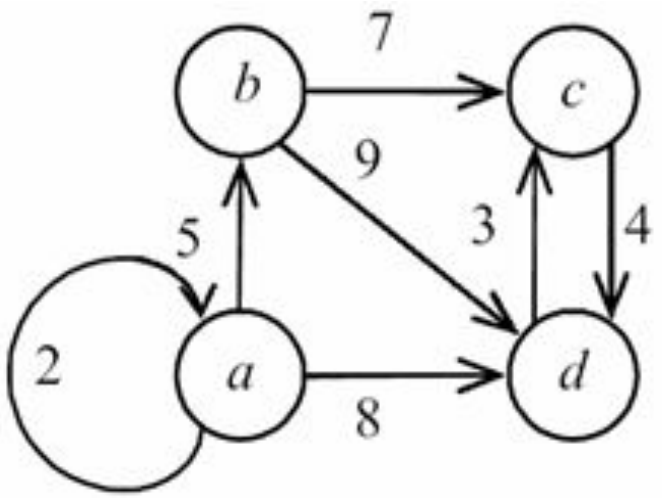
$d_{заход} = |K^{-1}(x_5)| = 2$

Матрица инцидентности

- Пусть дан граф *G* с *n* вершинами и *m* дугами.
- *Матрица инциденций (инцидентности)* графа *G* обозначается через $B = |b_{ij}|$ и является матрицей размерности $n \times m$, определяемой следующим образом:
  - $b_{ij} = 1$, если является начальной вершиной дуги ,
  - $b_{ij} = -1$, если является конечной вершиной дуги ,
  - $b_{ij} = 0$, если не является концевой вершиной дуги или если является петлей.

$$B = \begin{array}{c|cccccccccc} & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & a_8 & a_9 & a_{10} \\ \hline x_1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ x_2 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ x_4 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ x_5 & 0 & 0 & 0 & -1 & 0 & 1 & -1 & 1 & 0 & 0 \\ x_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$





Гра
ф

a	*a*	*a*	*b*	*b*	*c*	*d*
a	*b*	*d*	*c*	*d*	*d*	*c*
2	5	8	7	9	4	3

Список ребер
графа

	a	*b*	*c*	*d*
a	2	5	0	8
b	0	0	7	9
c	0	0	0	4
d	0	0	3	0

Матрица смежности
графа

	a	*b*	*c*	*d*
(*a*, *a*)	2	0	0	0
(*a*, *b*)	0	5	0	0
(*a*, *d*)	0	0	0	8
(*b*, *c*)	0	0	7	0
(*b*, *d*)	0	0	0	9
(*c*, *d*)	0	0	0	4
(*d*, *c*)	0	0	3	0

Матрица инцидентности
графа

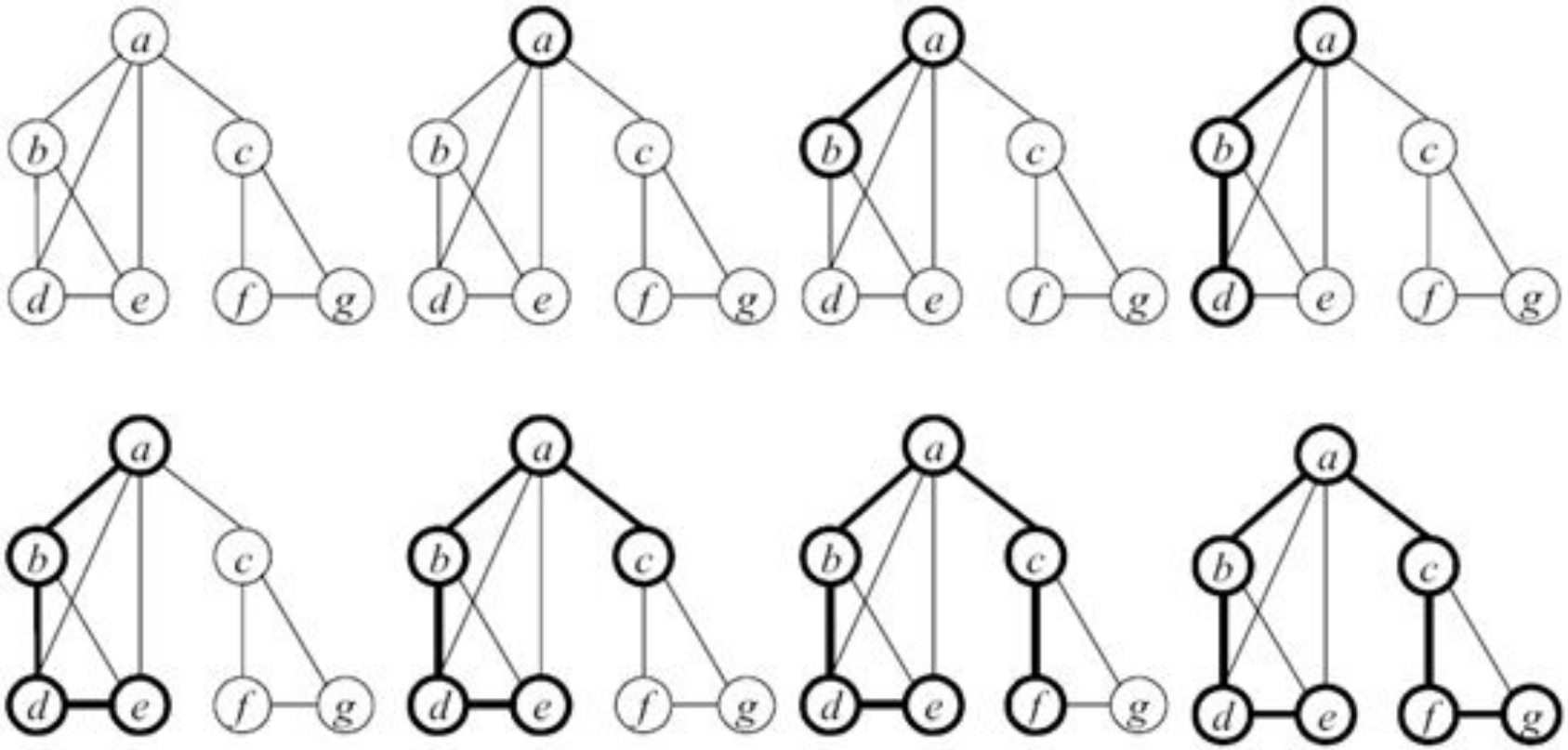
Алгоритмы на графах

- Обходные алгоритмы
  - Обход в глубину (Depth First Search, *DFS*);
  - Обход в ширину (Breadth First Search, *BFS*);
- Поиск кратчайшего пути
- Поиск максимального потока
- Поиск минимального остовного дерева

Поиск в глубину

- *Алгоритм поиска в глубину*
- Шаг 1. Всем вершинам графа присваивается значение не посещенная. Выбирается первая вершина и помечается как посещенная.
- Шаг 2. Для последней помеченной как посещенная вершины выбирается смежная вершина, являющаяся первой помеченной как не посещенная, и ей присваивается значение посещенная. Если таких вершин нет, то берется предыдущая помеченная вершина.
- Шаг 3. Повторить шаг 2 до тех пор, пока все вершины не будут помечены как посещенные.

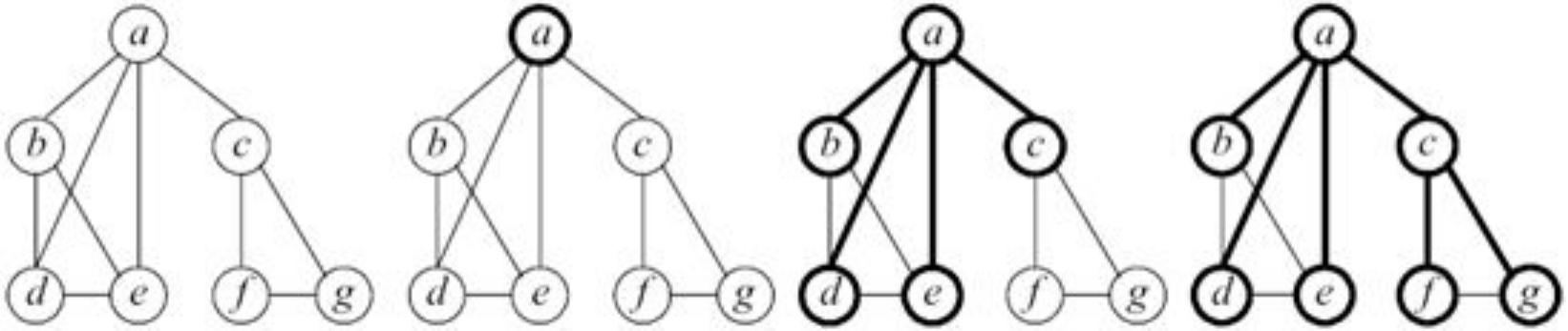
Пример поиска в глубину



Поиск в ширину

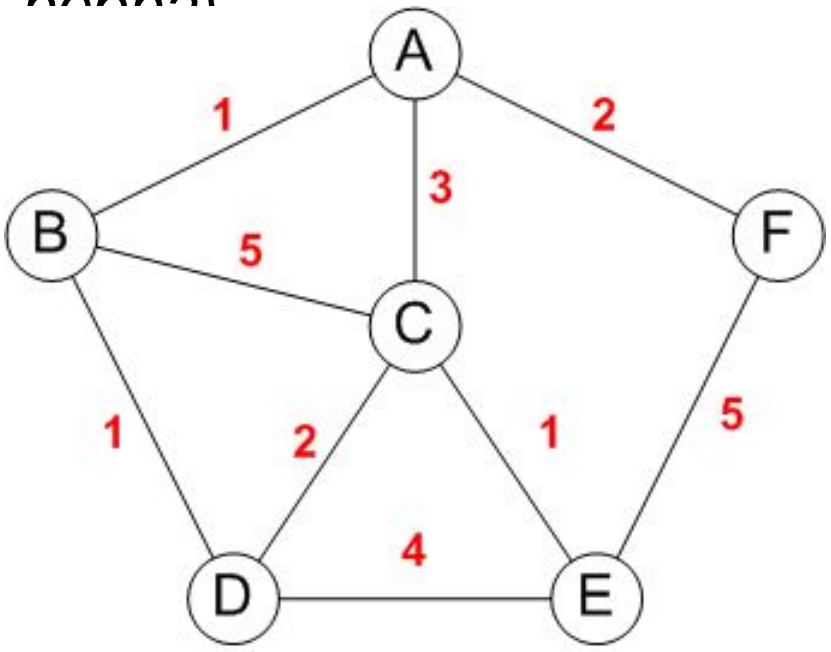
- *Алгоритм поиска в ширину*
- Шаг 1. Всем вершинам графа присваивается значение не посещенная. Выбирается первая вершина и помечается как посещенная (и заносится в очередь).
- Шаг 2. Посещается первая вершина из очереди (если она не помечена как посещенная). Все ее соседние вершины заносятся в очередь. После этого она удаляется из очереди.
- Шаг 3. Повторяется шаг 2 до тех пор, пока очередь не пуста.

Пример поиска в ширину



Взвешанный граф

- *Взвешенный граф* – граф, каждому ребру которого поставлено в соответствие некое значение (вес ребра)



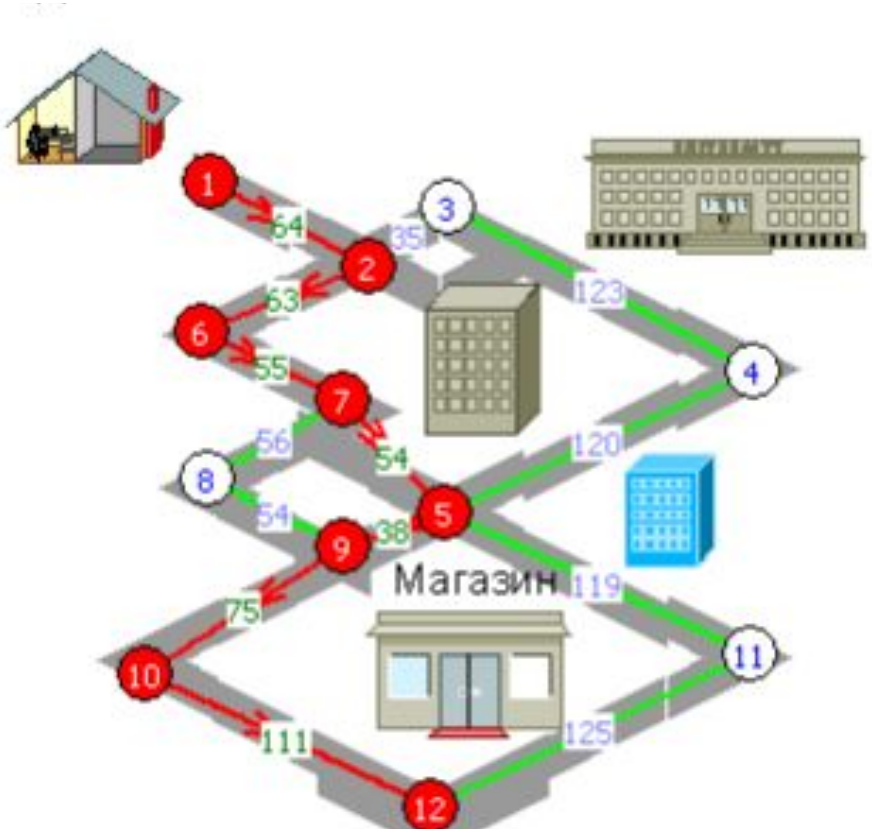
Взвешенный граф

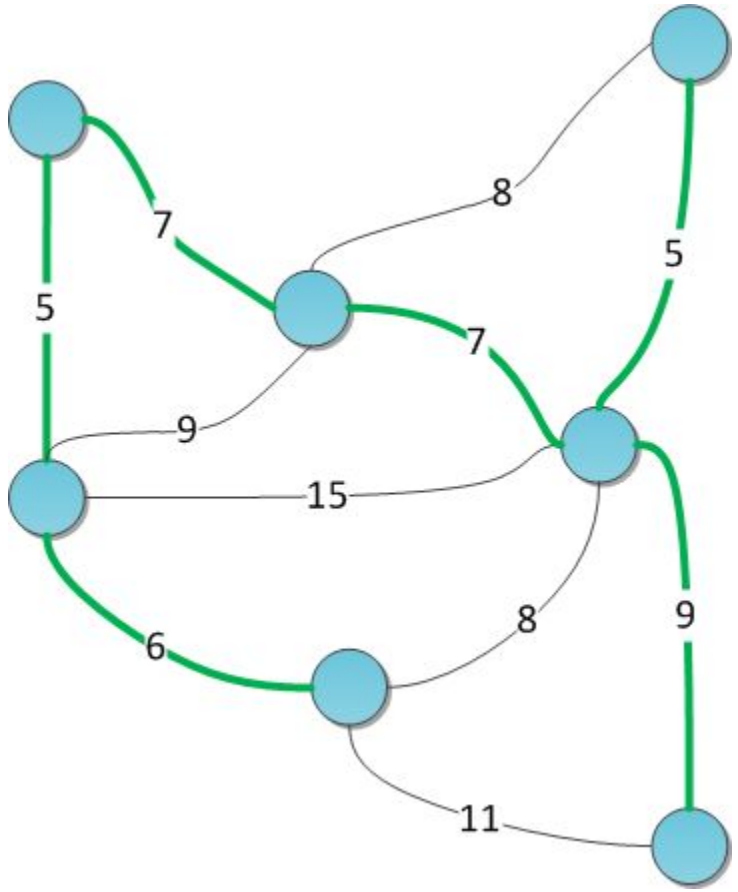
	A	B	C	D	E	F
A	0	1	∞	∞	∞	2
B	1	0	5	1	∞	∞
C	3	5	0	2	1	∞
D	∞	1	2	0	4	∞
E	∞	∞	1	4	0	5
F	2	∞	∞	∞	5	0

Матрица С

$$a_{i,j} = \begin{cases} \text{Вес дуги, если она существует} \\ 0, \text{ если i=j} \\ \infty \text{ ,если дуга не существует} \end{cases}$$

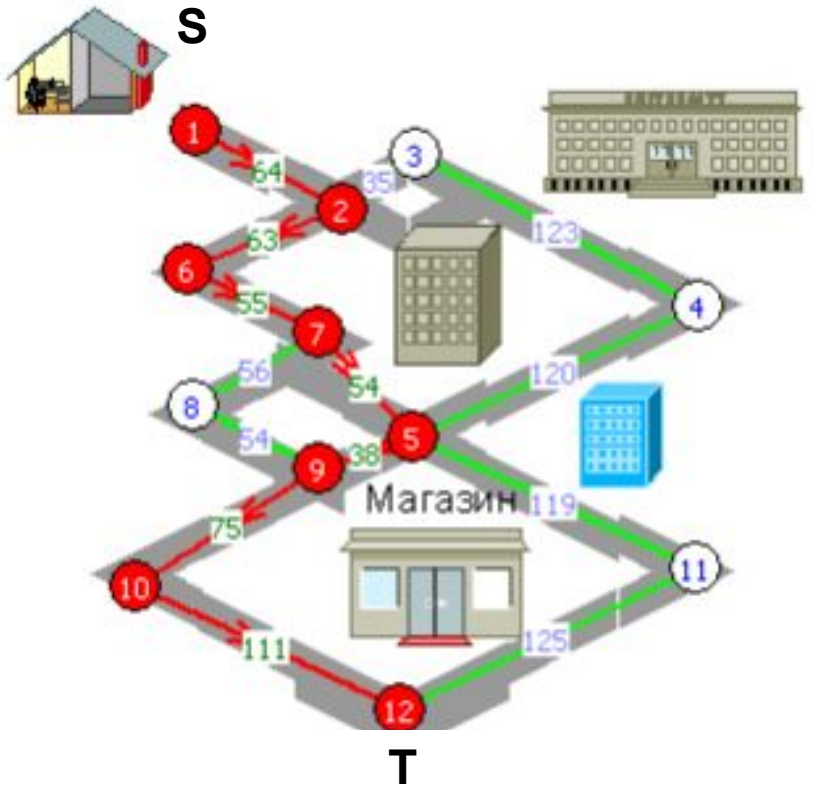
Примеры взвешенных графов





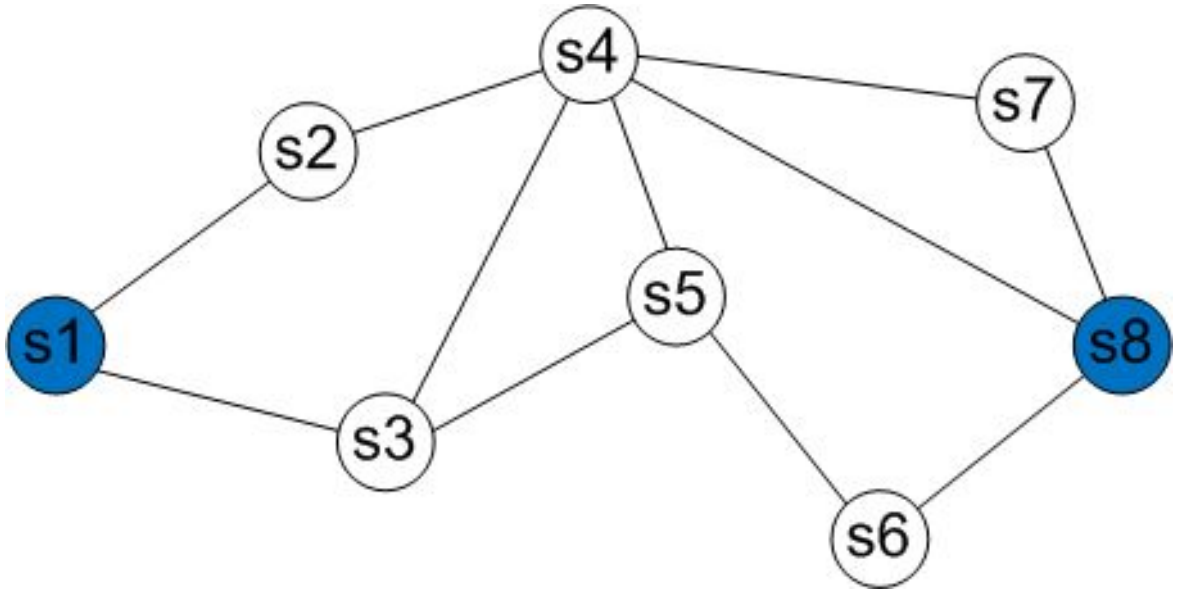
Задача поиска кратчайшего пути

- **Задача о кратчайшем пути** состоит в нахождении кратчайшего пути от заданной начальной вершины $s \in X$ до заданной конечной вершины $t \in X$, при условии, что такой путь существует, т.е. при условии $t \in R(s)$, где $R(s)$ - множество, достижимое из вершины s.



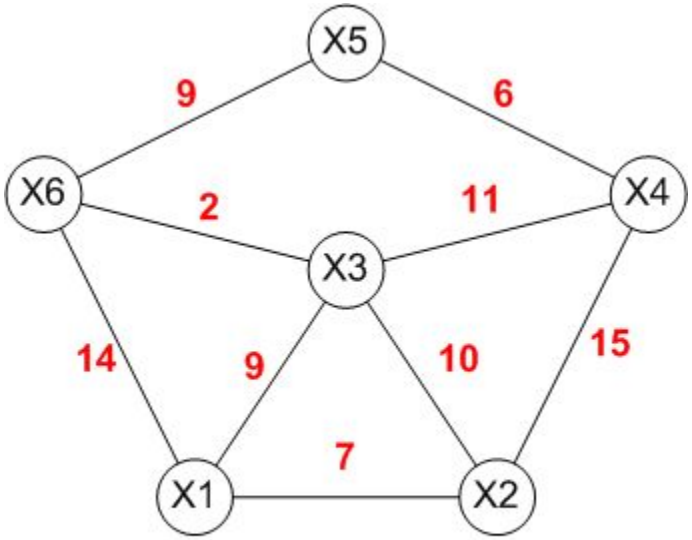
Поиск пути на графе

- Дано: непустой граф G=(V,E). Требуется найти путь между вершинами s=s1 и t=s8 графа (s не совпадает с t), содержащий минимальное количество промежуточных вершин (ребер), т.е. кратчайший путь.



Поиск кратчайшего пути (обобщения)

- Следующие задачи являются обобщениями сформулированной выше задачи о кратчайшем пути.
  - Для заданной начальной вершины *s* найти кратчайшие пути между *1* и всеми другими вершинами .
  - **Найти кратчайшие пути между всеми парами вершин.**



Взвешенный граф

Наиболее известные алгоритмы поиска кратчайшего пути

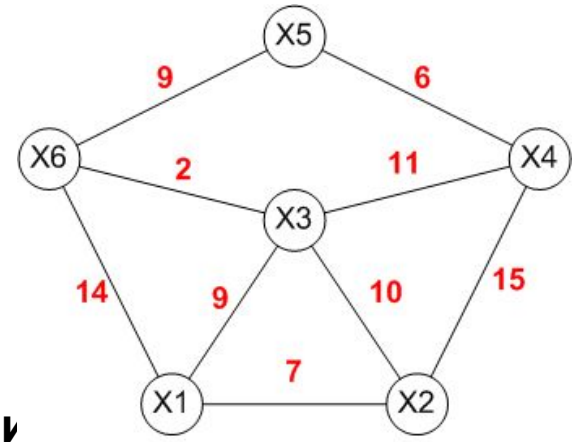
- *алгоритм Дейкстры*;
- алгоритм Белмана-Форда;
- переборные алгоритмы
  - волновой алгоритм.

Алгоритм Дейкстры

- *Алгоритм Дейкстры*
  - Шаг 1. Всем вершинам, за исключением первой, присваивается вес равный бесконечности, а первой вершине – 0.
  - Шаг 2. Все вершины не выделены.
  - Шаг 3. Первая вершина объявляется текущей.
  - Шаг 4. Вес всех невыделенных вершин пересчитывается по формуле: вес невыделенной вершины есть минимальное число из старого веса данной вершины, суммы веса текущей вершины и веса *ребра*, соединяющего текущую вершину с невыделенной.
  - Шаг 5. Среди невыделенных вершин ищется вершина с минимальным весом. Если таковая не найдена, то есть вес всех вершин равен бесконечности, то маршрут не существует. Следовательно, выход. Иначе, текущей становится найденная вершина. Она же выделяется.
  - Шаг 6. Если текущей вершиной оказывается конечная, то путь найден, и его вес есть вес конечной вершины.
  - Шаг 7. Переход на шаг 4.

Алгоритм Дейкстры

- **Формальное определение:**
  - Дан взвешенный ориентированный граф $G(X,E)$ **без петель и дуг отрицательного веса.** Найти кратчайшие пути от некоторой вершины s графа G до конечной вершины t (или всех остальных вершин) этого графа.



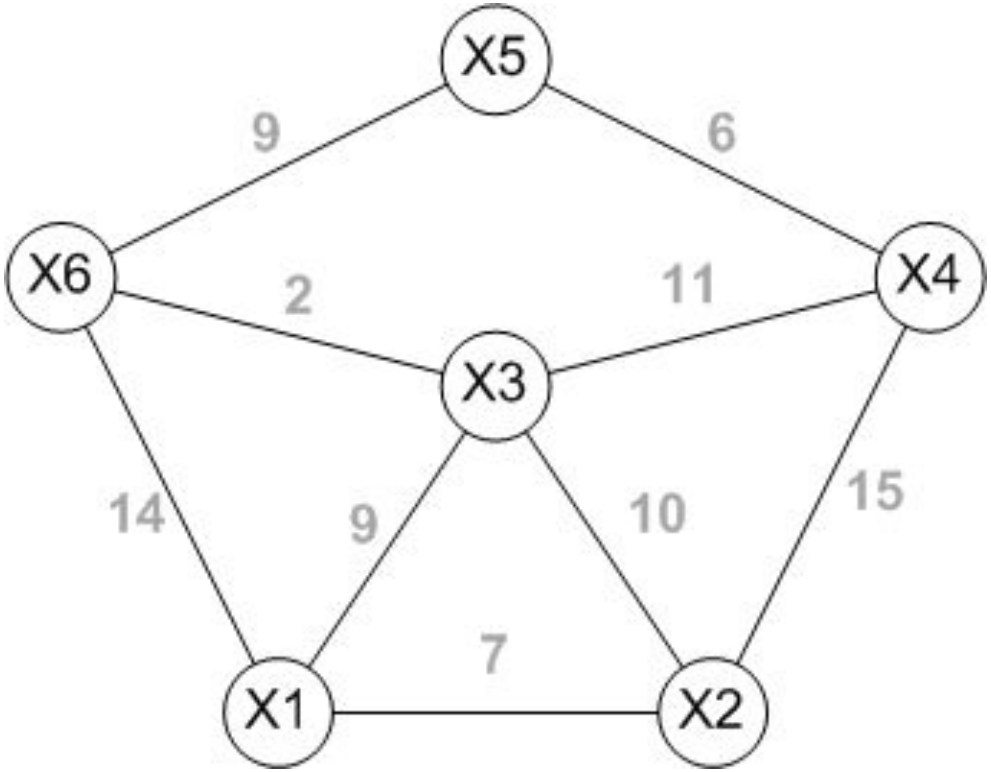
- **Неформальное описани**
  - Каждой вершине из X сопоставим метку — минимальное известное расстояние от этой вершины до *начальной вершины*.
  - Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.
  - Работа алгоритма завершается, когда все вершины посещены.

Алгоритм Дейкстры

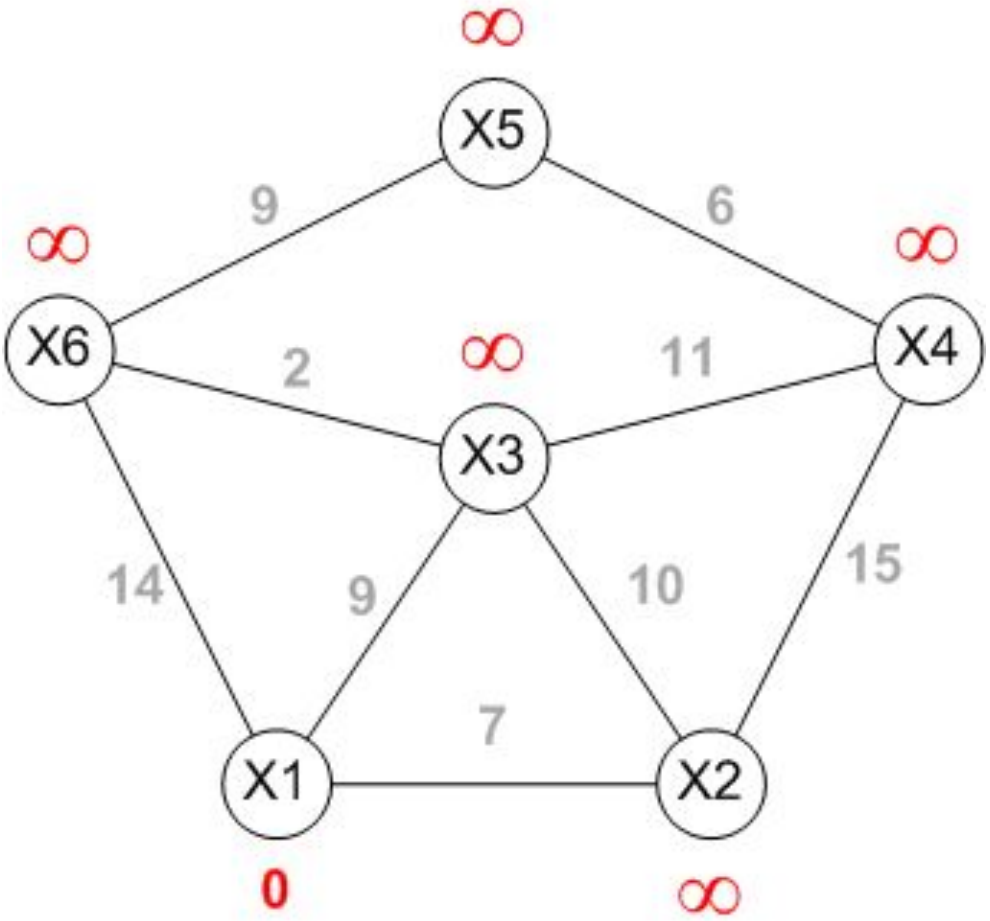
- **Шаг алгоритма:**
- Если все вершины посещены, алгоритм завершается.
- Иначе:
  - из ещё не посещённых вершин выбирается вершина *u*, имеющая минимальную метку;
  - рассматриваем всевозможные маршруты, в которых *u* является предпоследним пунктом;
  - вершины, в которые ведут рёбра из *u*, назовем *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;

Пример

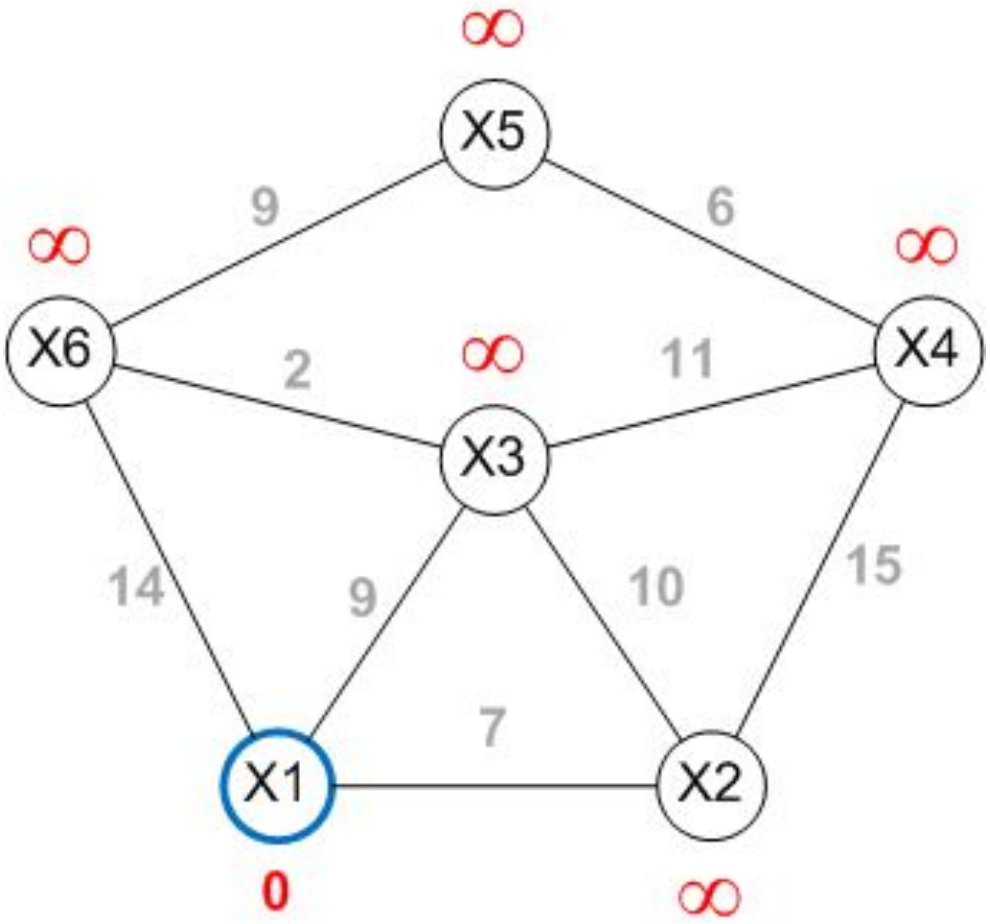
- Найти кратчайшие расстояния от X1-й вершины до всех остальных.



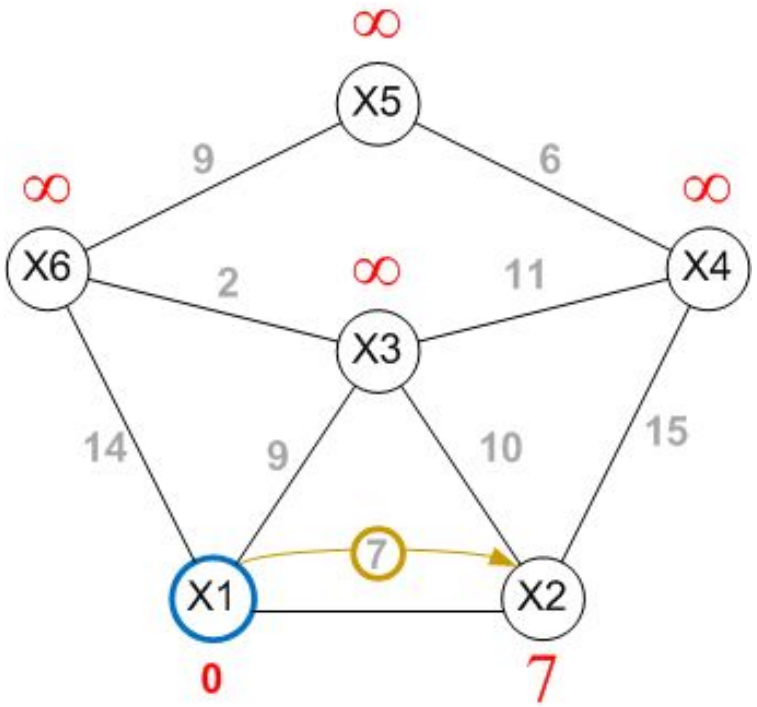
Пример

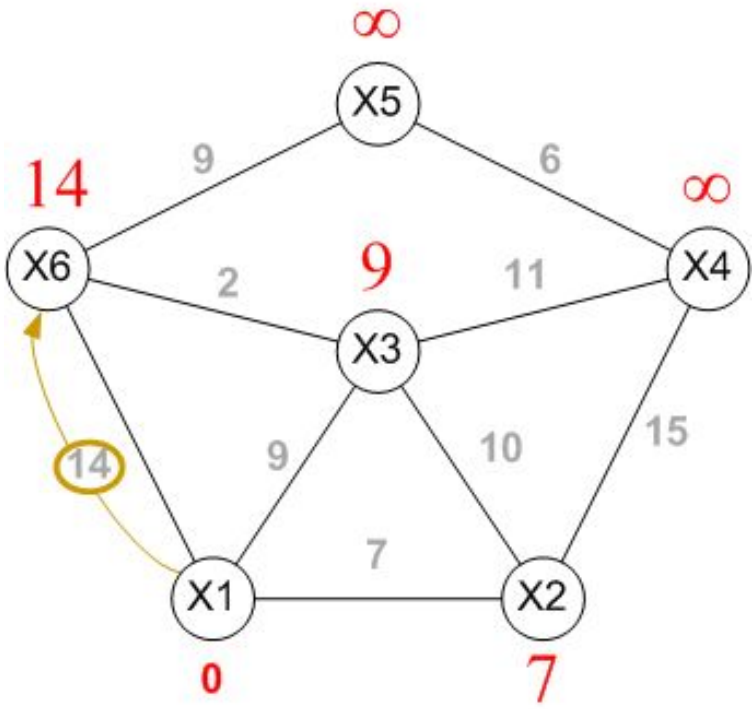


Пример

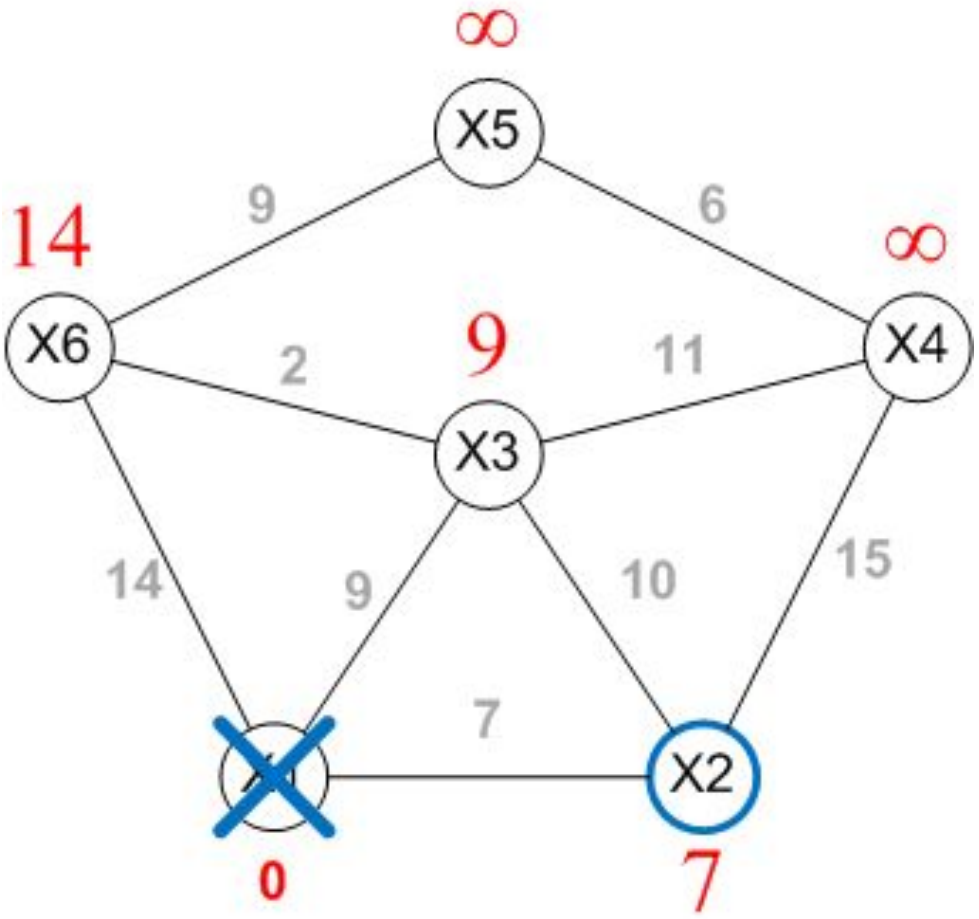


Пример

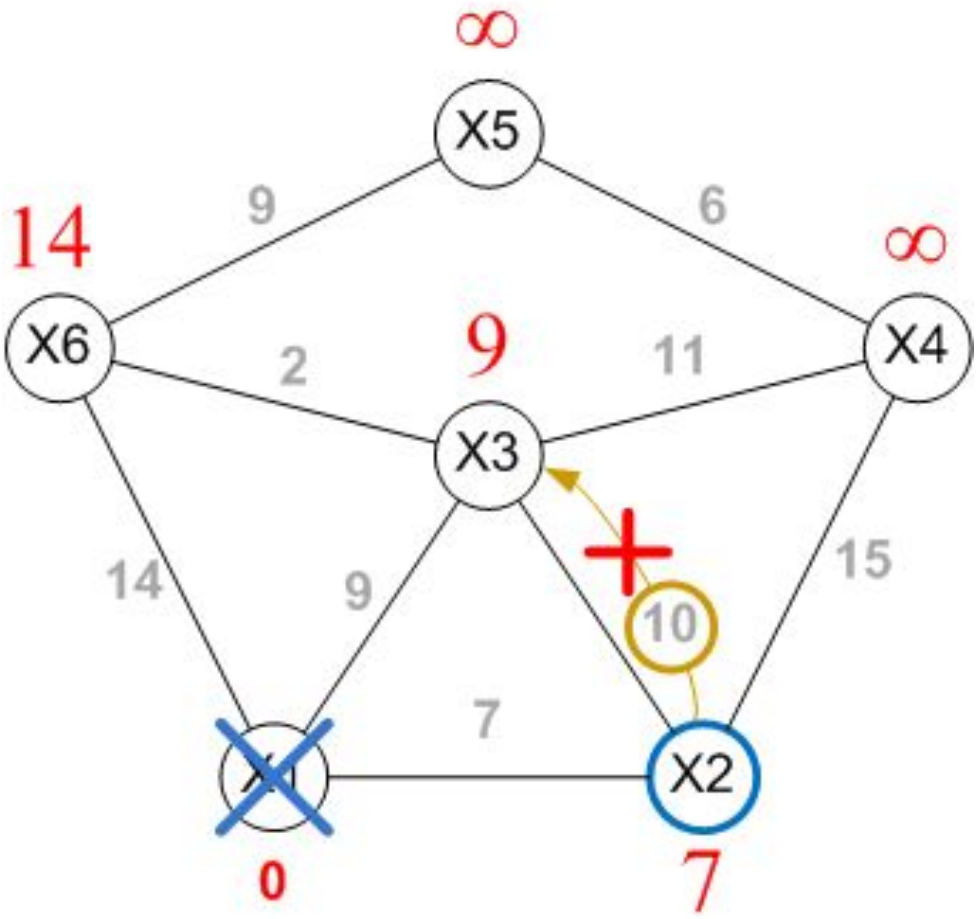

∞
X5
9
6
∞
∞
X6
2
∞
11
X4
X3
14
9
10
15
7
X1
X2
0
7


∞
X5
14
9
6
∞
X6
2
9
11
X4
X3
14
9
10
15
7
X1
X2
0
7

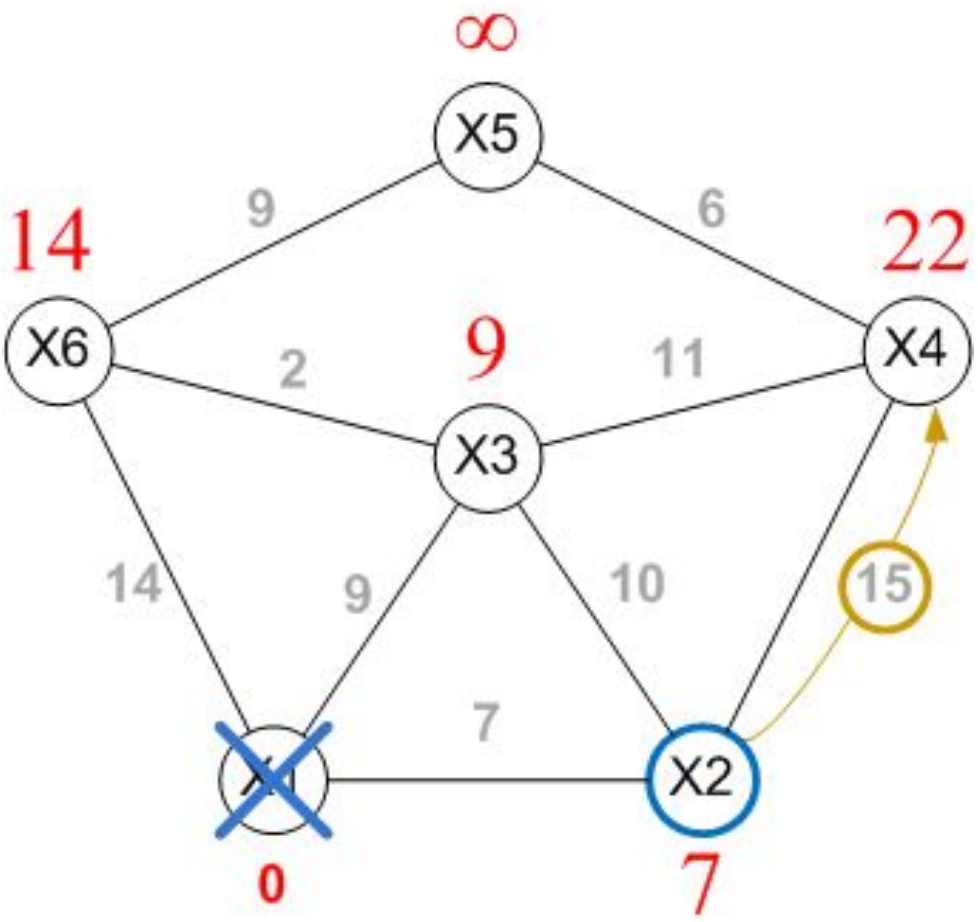
Пример



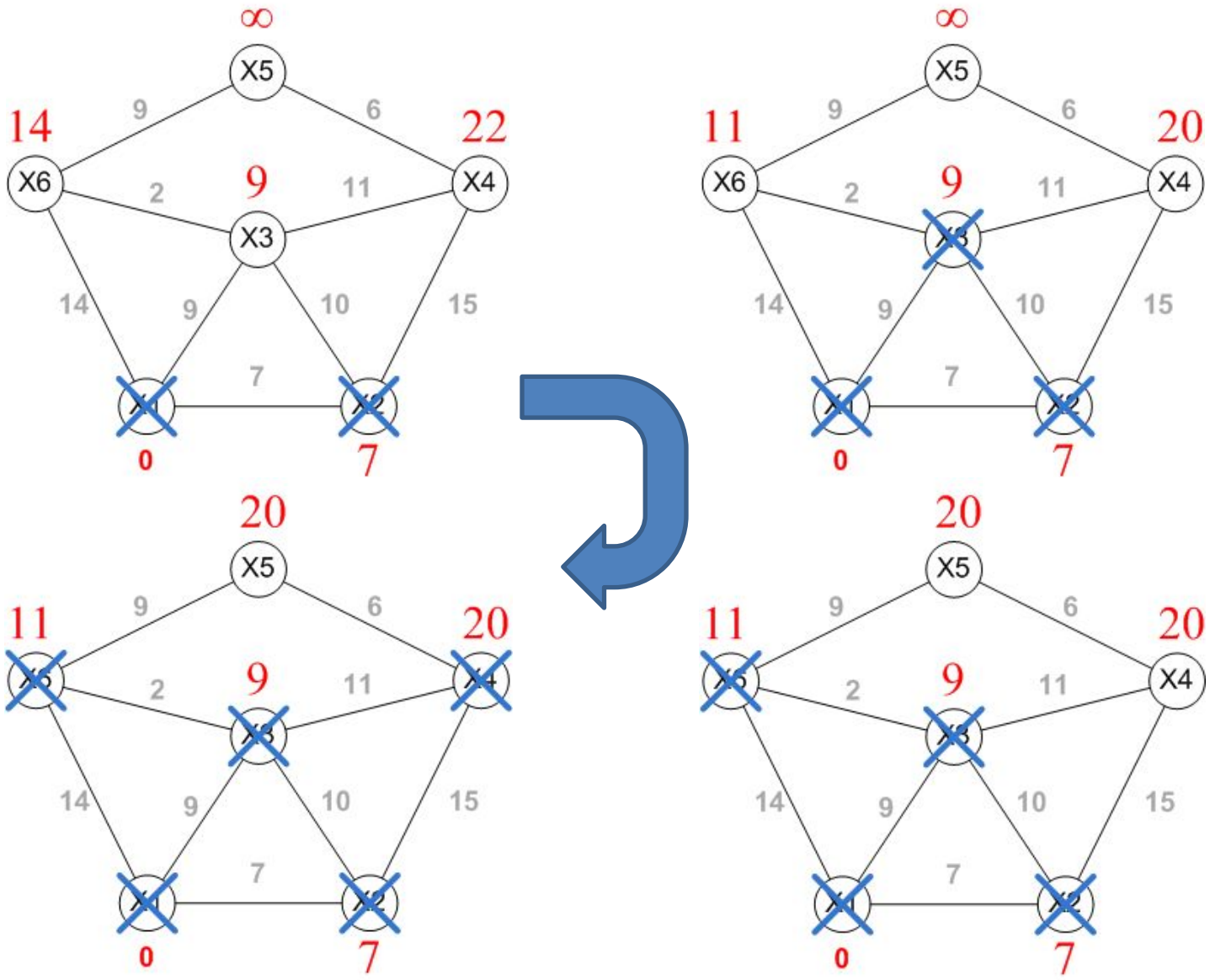
Пример



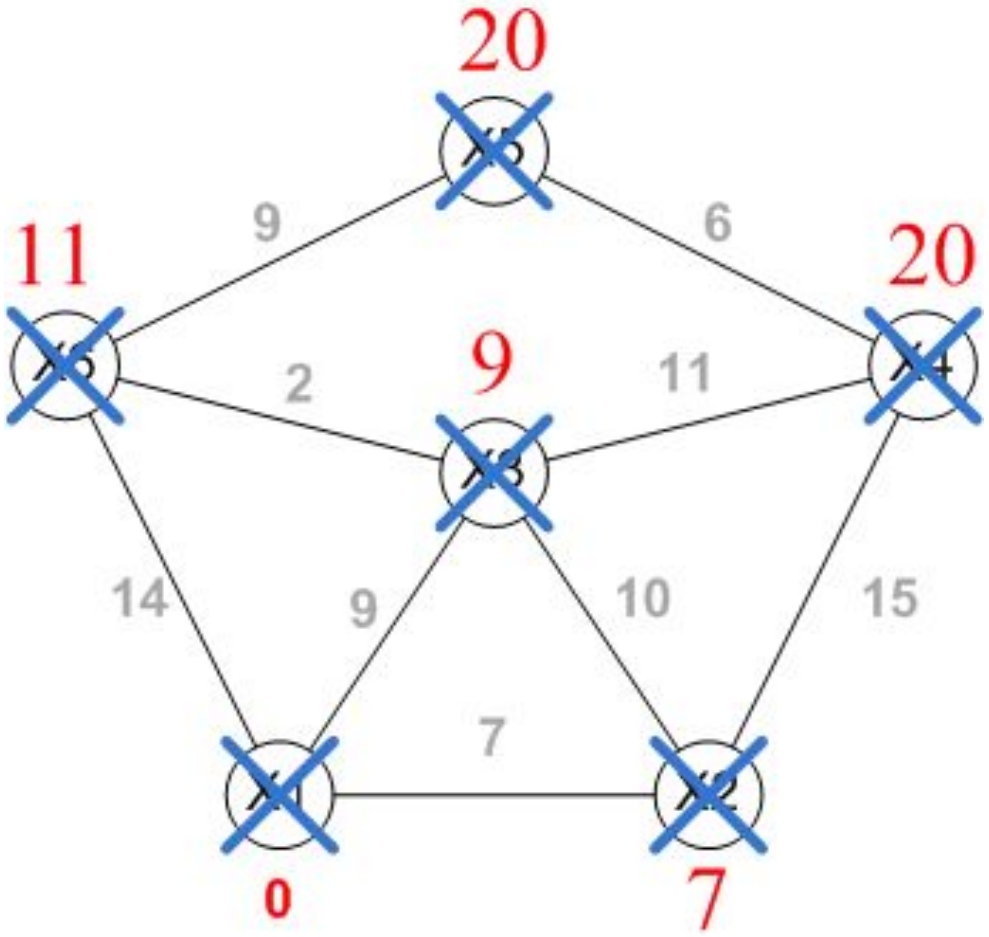
Привет



Пример


∞
X5
14
9
6
22
X6
2
9
11
X4
X3
14
9
10
15
7
0
7
∞
X5
11
9
6
20
X6
2
9
11
X4
14
9
10
15
7
0
7
20
X5
11
9
6
20
2
9
11
14
9
10
15
7
0
7
20
X5
11
9
6
20
X4
2
9
11
14
9
10
15
7
0
7

Пример



Применение алгоритмов поиска кратчайшего пути

- *Вариант 1.* Дана сеть автомобильных дорог, соединяющих города Московской области. Некоторые дороги односторонние. Найти кратчайшие пути от города Москва до каждого города области (если двигаться можно только по дорогам).
- *Вариант 2.* Имеется некоторое количество авиарейсов между городами мира, для каждого известна стоимость. Стоимость перелёта из A в B может быть не равна стоимости перелёта из B в A. Найти маршрут минимальной стоимости (возможно, с пересадками) от Копенгагена до Барнаула.

Алгоритм Белмана-Форда

Алгоритм Беллмана-Форда

- Дан ориентированный или неориентированный граф *G* со взвешенными рёбрами.
- Длиной пути назовём сумму весов рёбер, входящих в этот путь.
- Требуется найти кратчайшие пути от выделенной вершины *s* до всех вершин графа.

```
for v ∈ V
  do d[v] ← +∞
d[s] ← 0
for i ← 1 to |V| − 1
  do for (u, v) ∈ E
     if d[v] > d[u] + w(u,v)
       then d[v] ← d[u] + w(u, v)
return d
```

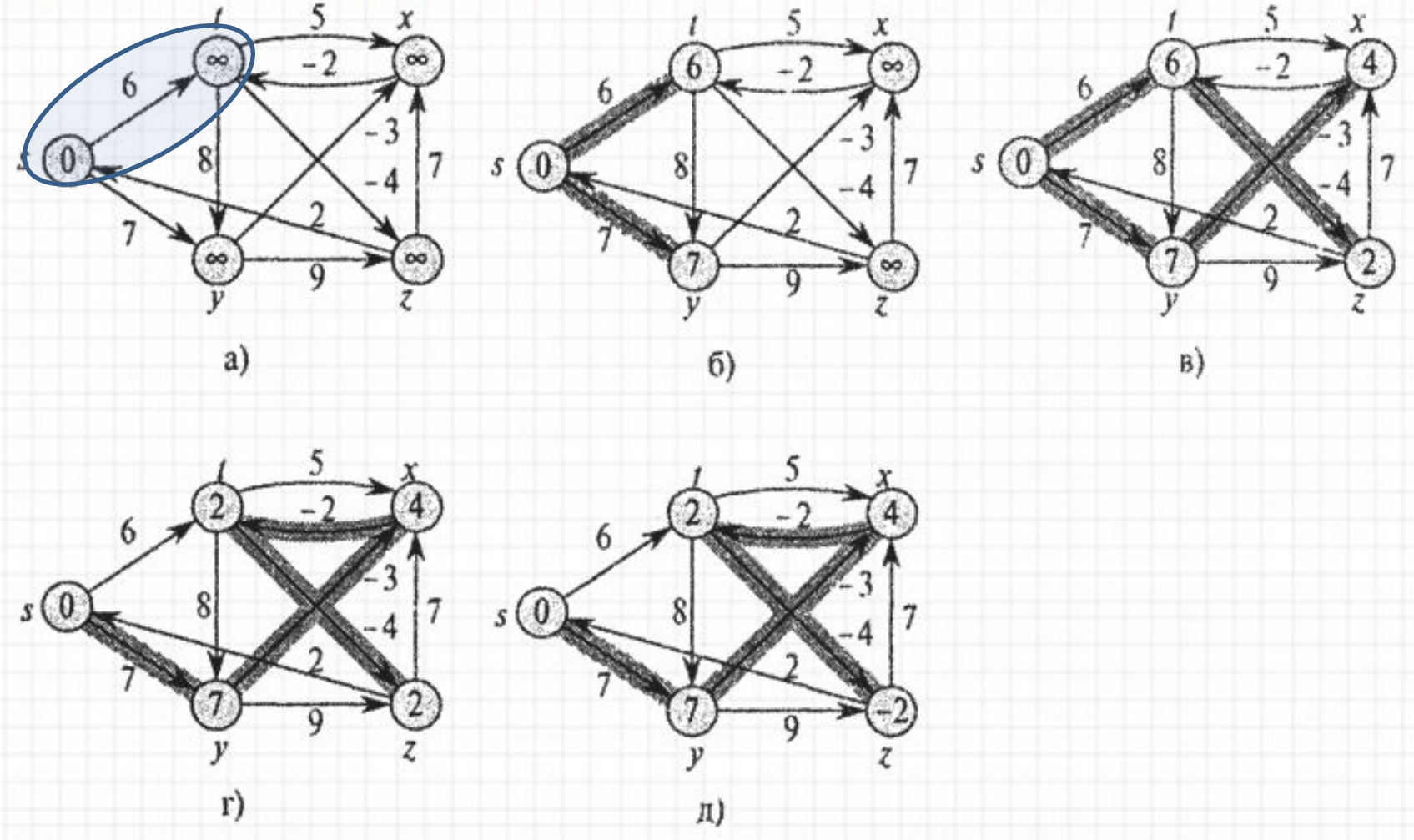
Присваиваем метки (∞) вершинам

Присваиваем метку 0 нач. вершине

Перебираем вершины и ребра

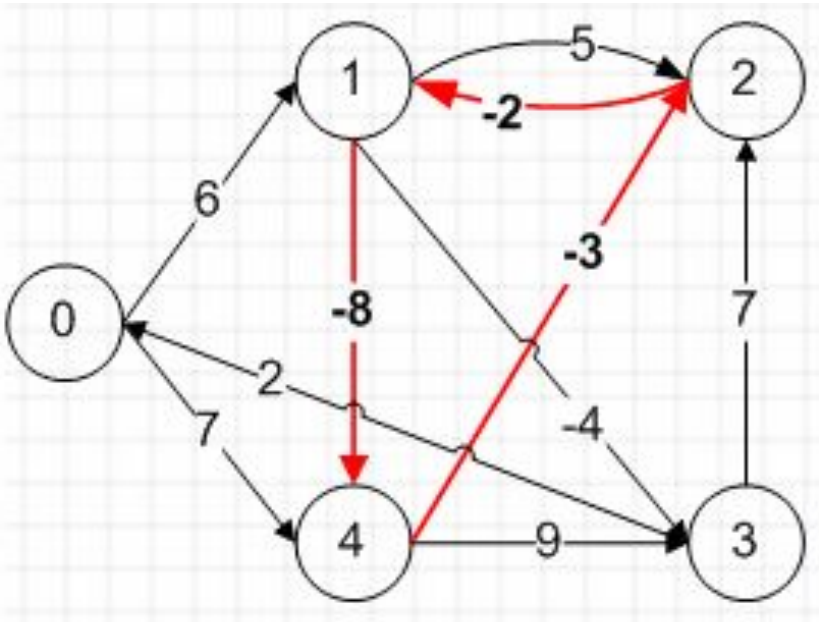
Уменьшаем метки вершин

Алгоритм Белмана-Форда



Отрицательный цикл

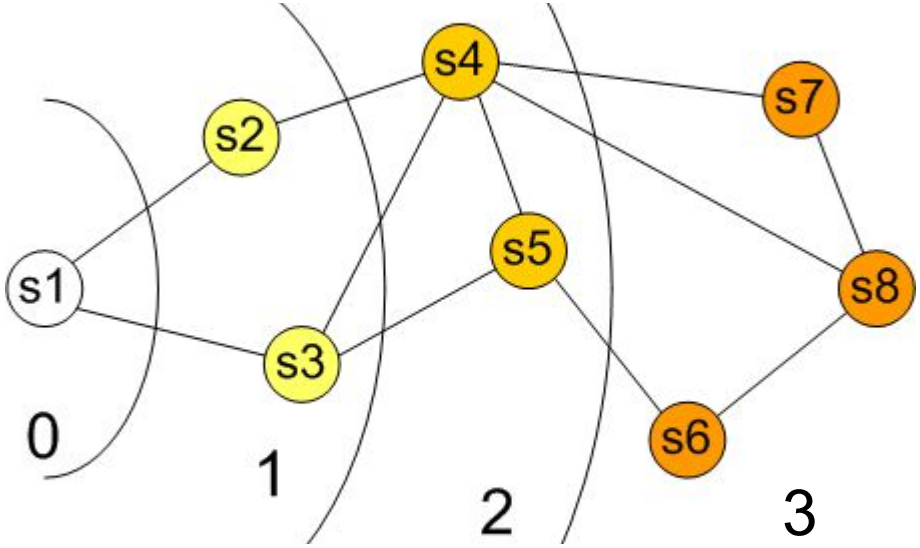
- Цикл, сумма весов рёбер которого отрицательна, называется **отрицательным циклом**



Волновой алгоритм

Волновой алгоритм

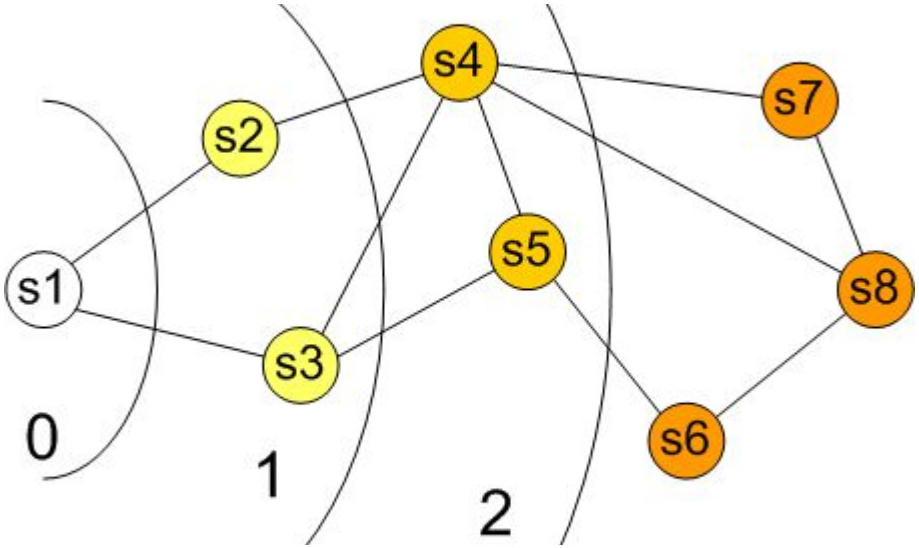
- Шаг 1. Обходим граф поиском в ширину, помечая длину пути на каждом шаге для рассматриваемых вершин



Первоначальный элемент s1

Волновой алгоритм

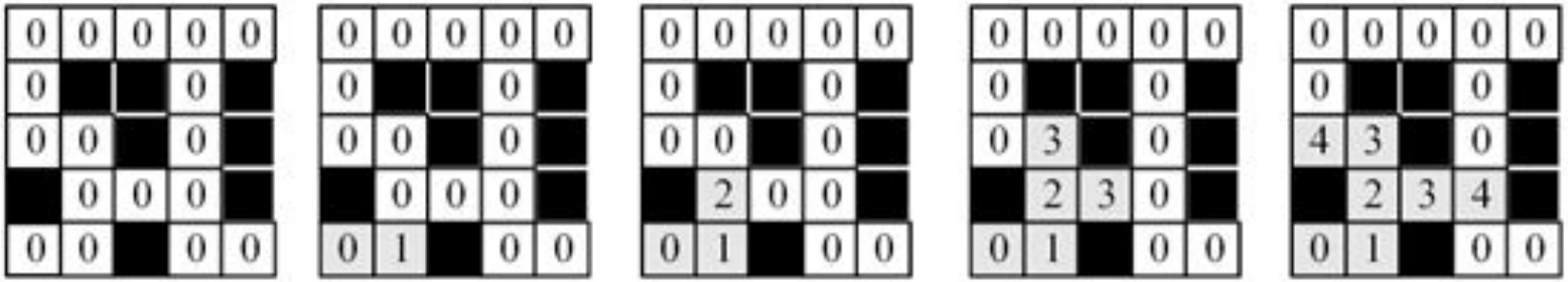
- Шаг 2. Восстанавливаем кратчайший путь:
  - среди соседей вершины t найдем любую вершину с волновой меткой T(t)-1, среди соседей последней - вершину с меткой T(t)-2, и т.д., пока не достигнем s. Найденная последовательность вершин определяет один из кратчайших путей из s в t.



s = s1

t = s8

Пример волнового алгоритма



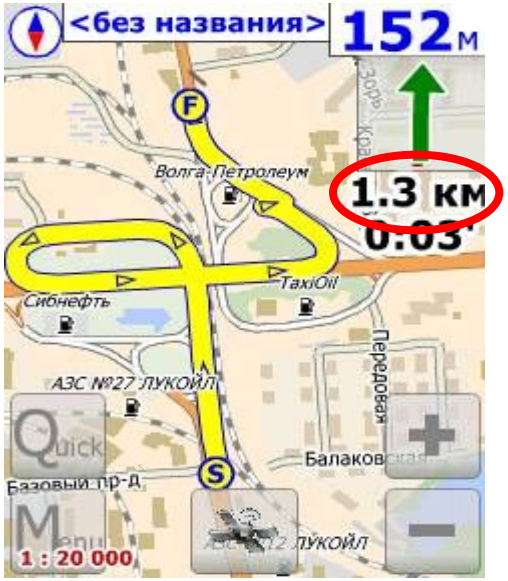
OptimalWay = (1,1), (1,2), (2,2), (2,3), (2,4), (3,4), (4,4), (5,4), (5,5);

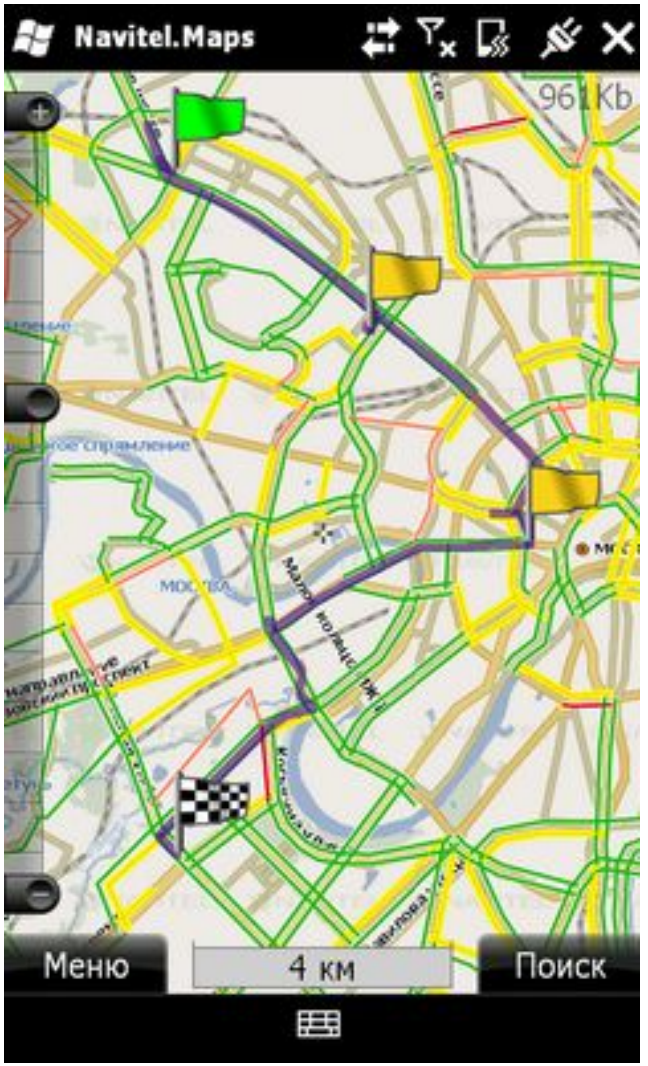
Length(Way) = 8

Применение методов поиска кратчайшего пути

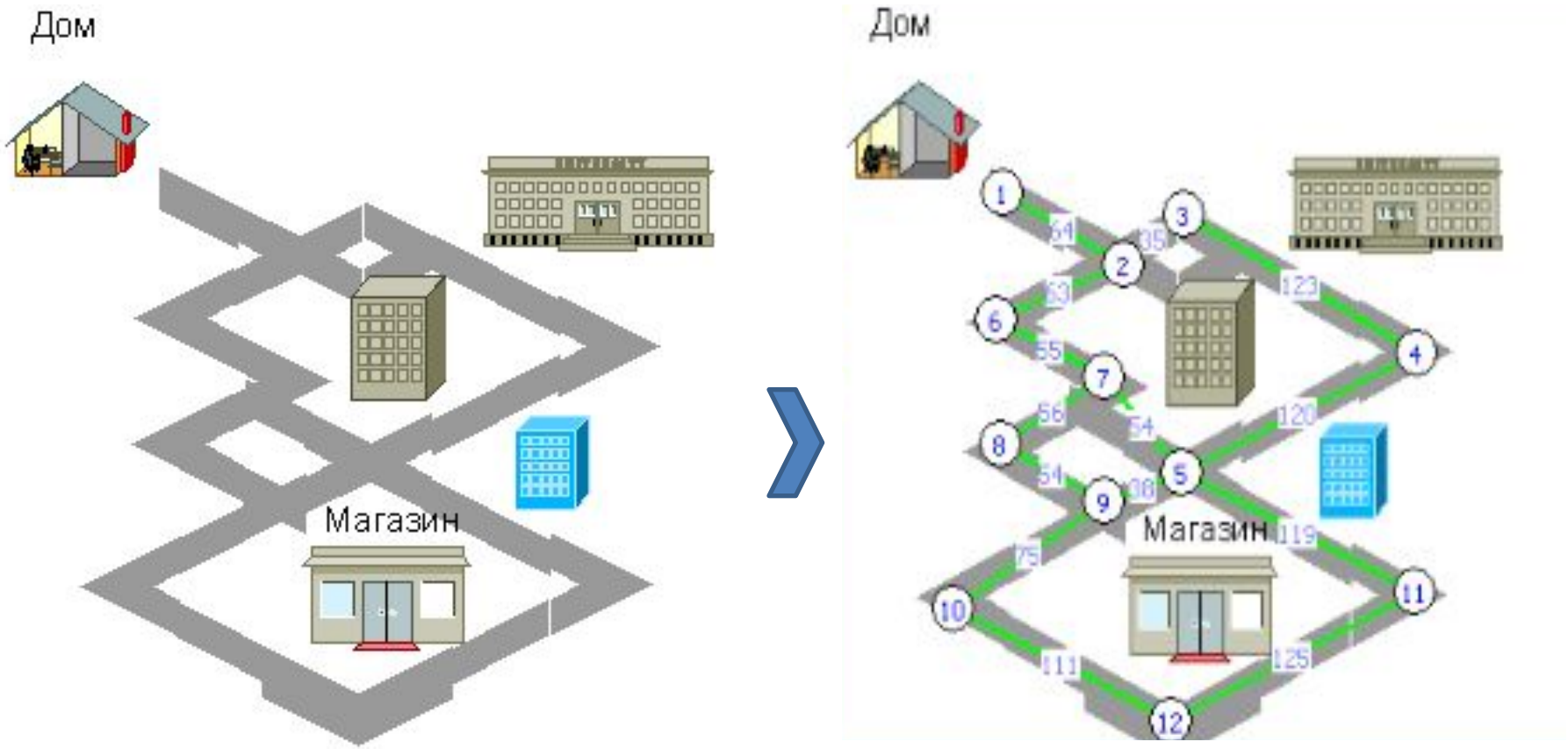
Примеры использования картографические и навигационные системы

- **Навигационная система** - это электронная система установленная на борту транспортного средства **в целях вычисления оптимального маршрута** движения.

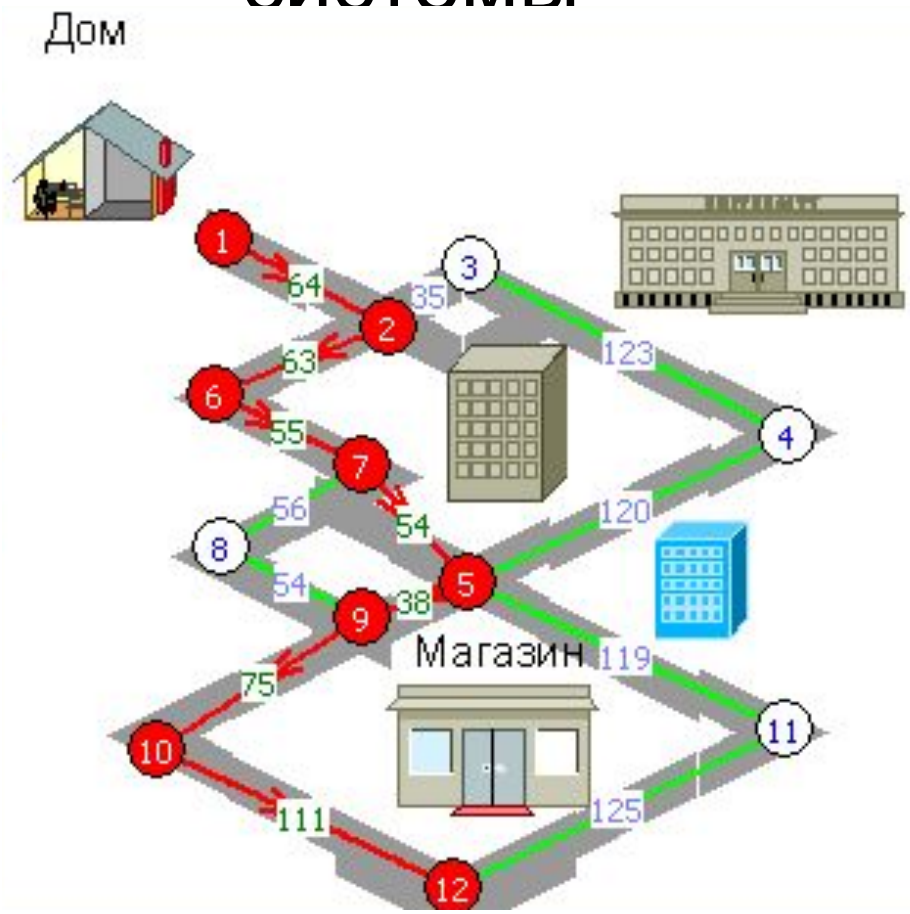




Примеры использования картографические и навигационные системы



Примеры использования картографические и навигационные системы



Кратчайший путь 64+63+55+54+38+75+111 = 460

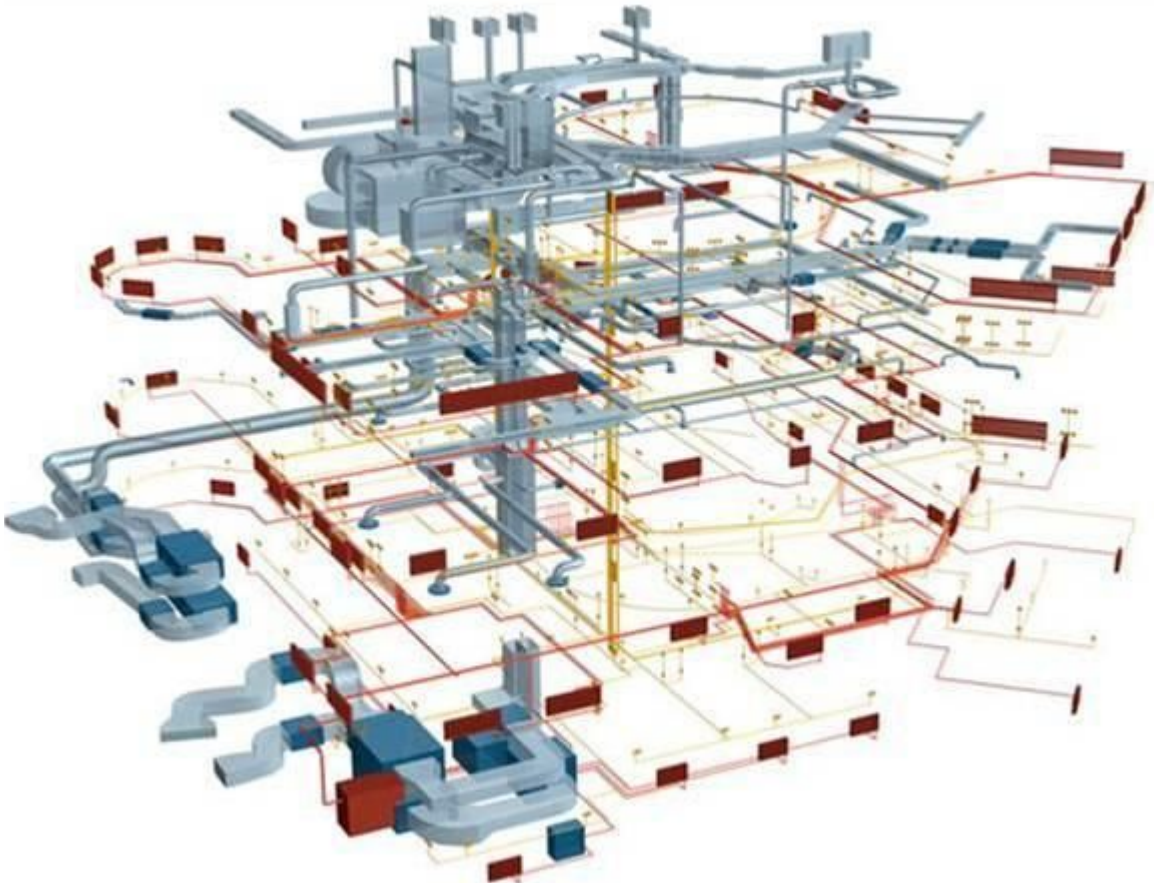
Примеры использования организация системы сетей различных типов

Прокладка инженерных сетей (трубопроводов, коммуникаций)

Прокладка электронных сетей (трубопроводов)

Поиск минимальных затрат на прокладку сети.

Примеры использования организация системы сетей различных типов



Примеры использования
поиск минимальных затрат при найме сотрудников

Дядя Петя:
С 8 до 10 – 200 рублей.
С 14 до 18 – 500 рублей
С 19 до 20 – 50 рублей

Робот
С 10 до 14 – 350 рублей
С 16 до 19 – 250 рублей

Наталья Ивановна
С 8 до 12 – 350 рублей
С 15 до 20 – 700 рублей

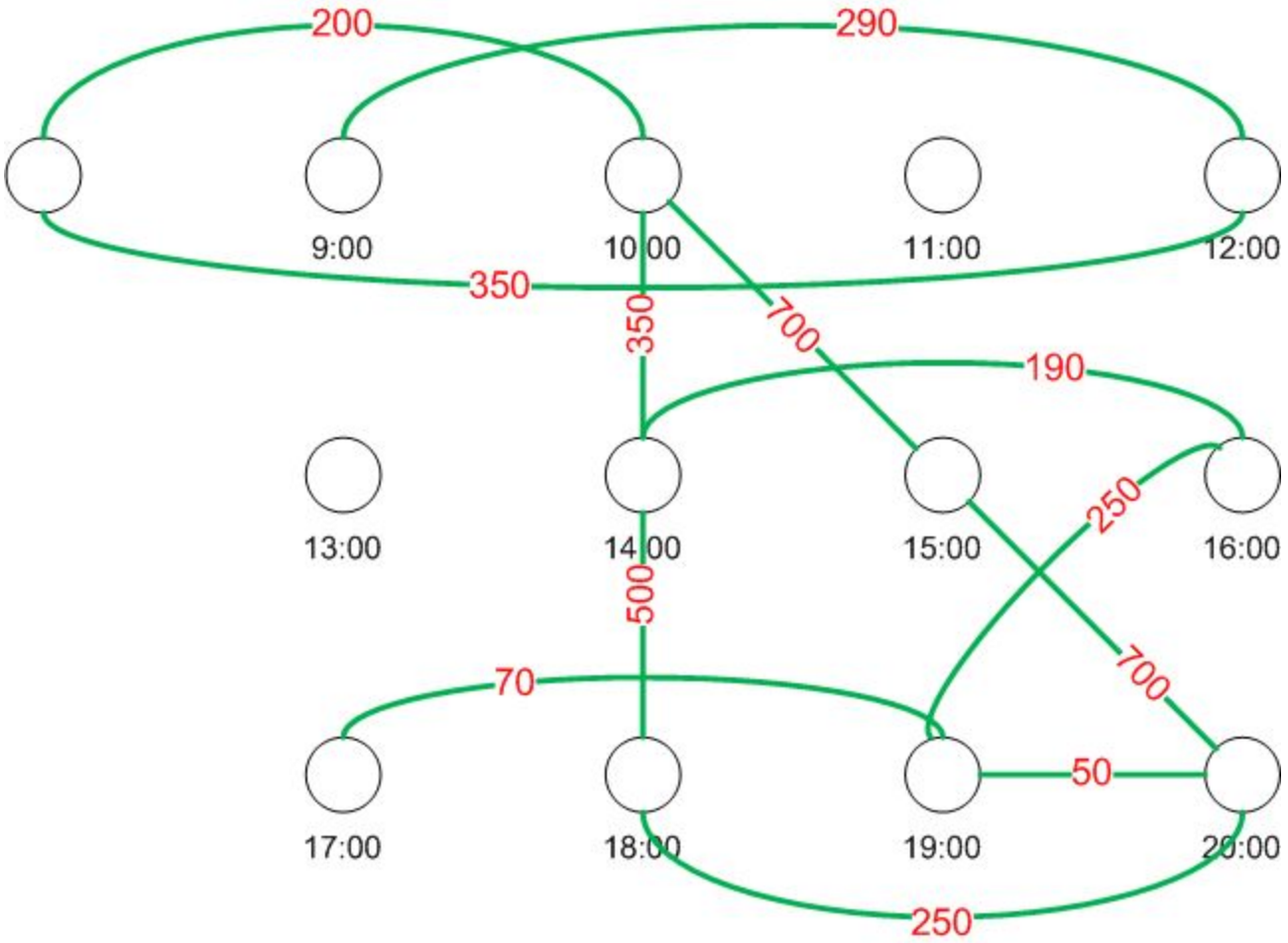
Василий Иванович
С 10 до 15 – 700 рублей
С 17 до 19 – 70 рублей

Мальчик Вова
С 9 до 12 – 290 рублей
С 14 до 16 – 190 рублей
С 18 до 20 – 250 рублей

Примеры использования поиск минимальных затрат при найме сотрудников

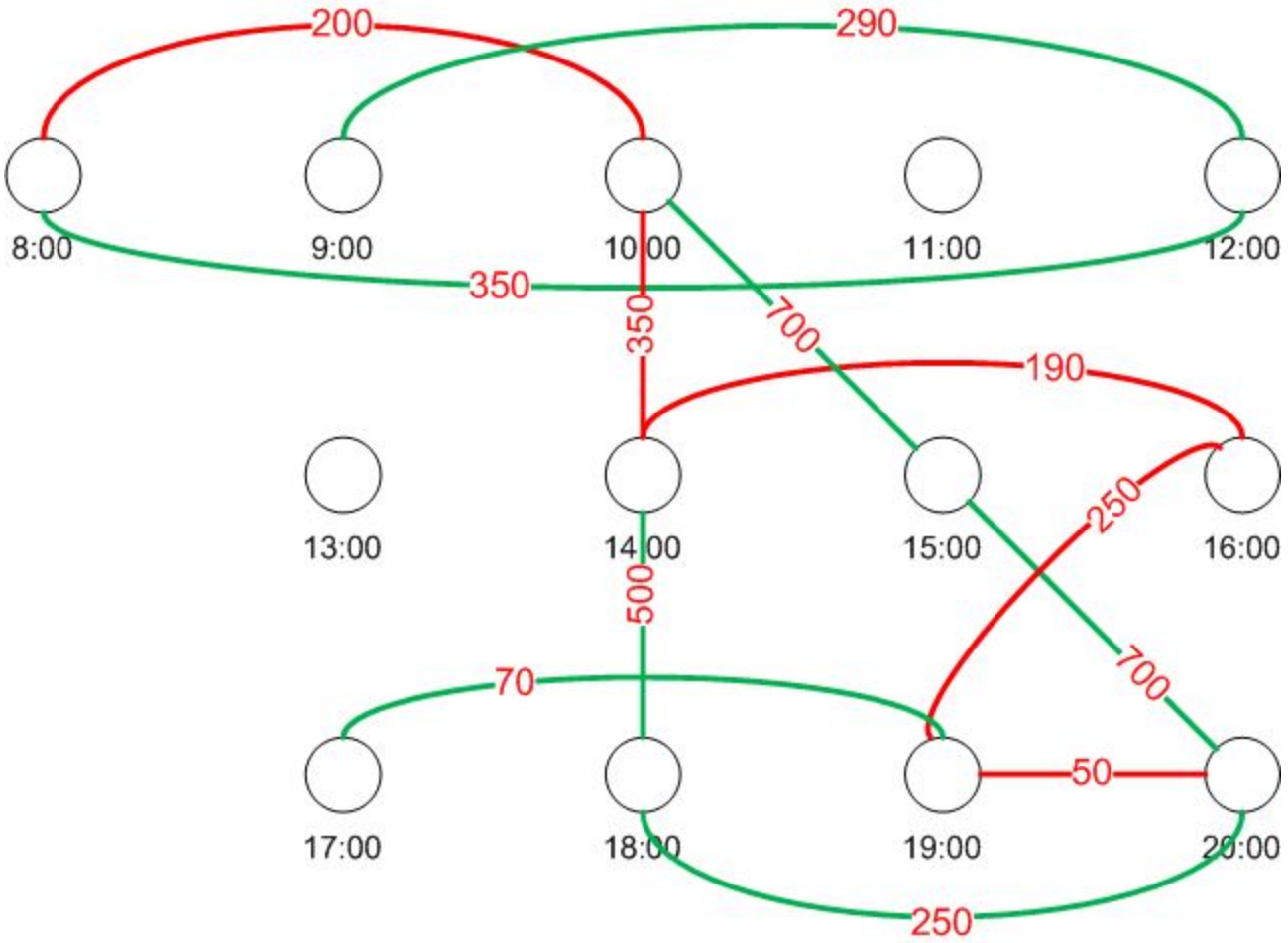
Время	С 8 до 9	С 9 до 10	С 10 до 11	С 11 до 12	С 12 до 13	С 13 до 14	С 14 до 15	С 15 до 16	С 16 до 17	С 17 до 18	С 18 до 19	С 19 до 20
Работники												
Дядя Петя	200 р						500 р					50 р
Наталья Ивановна		350 р							700 р			
Мальчик Вова			290 р				190 р				250 р	
Робот				350 р					250 р			
Василий Иванович					700 р					70 р		

Примеры использования
поиск минимальных затрат при найме сотрудников



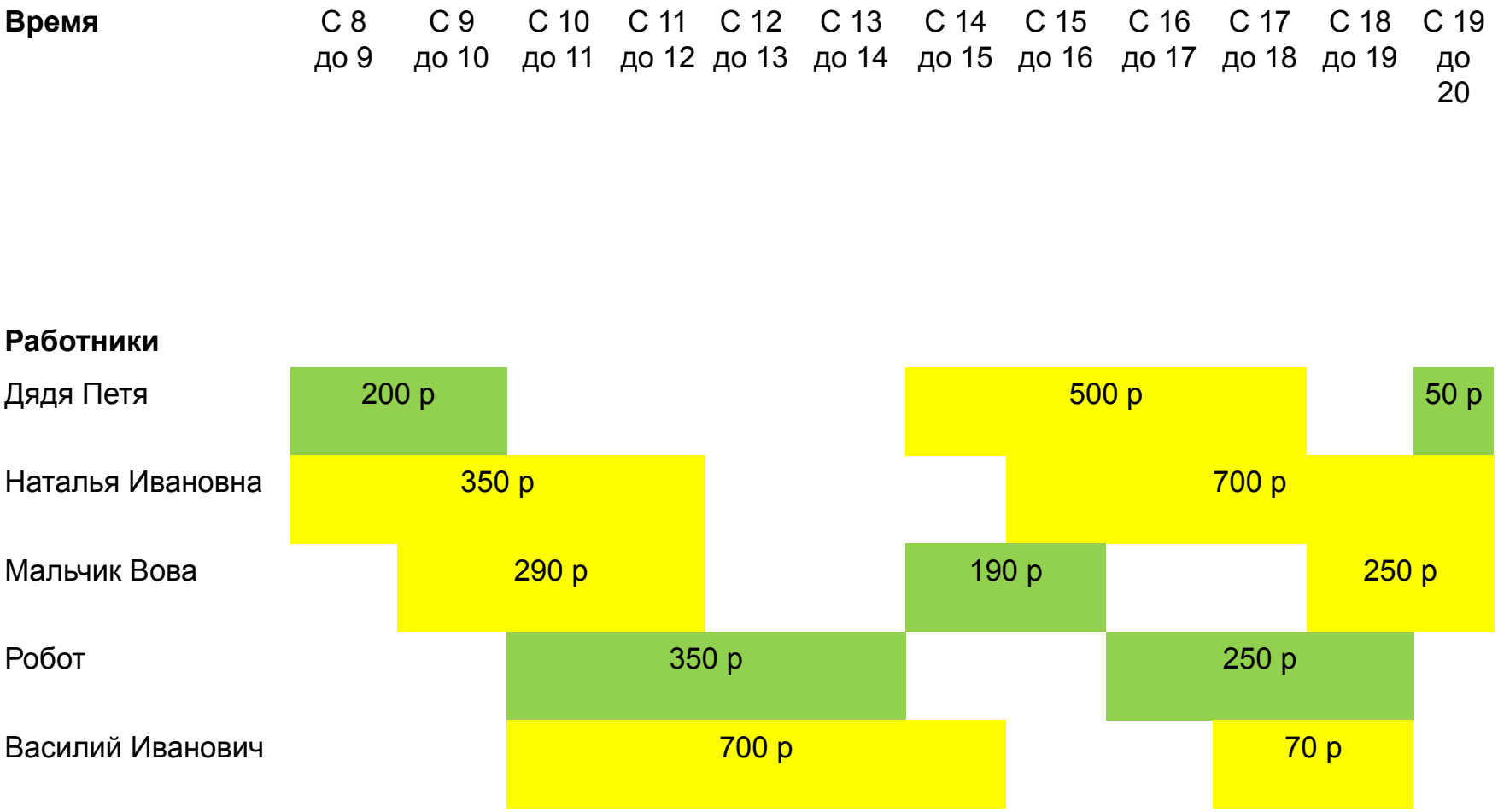
Граф на основе графика работы

Примеры использования
поиск минимальных затрат при найме сотрудников



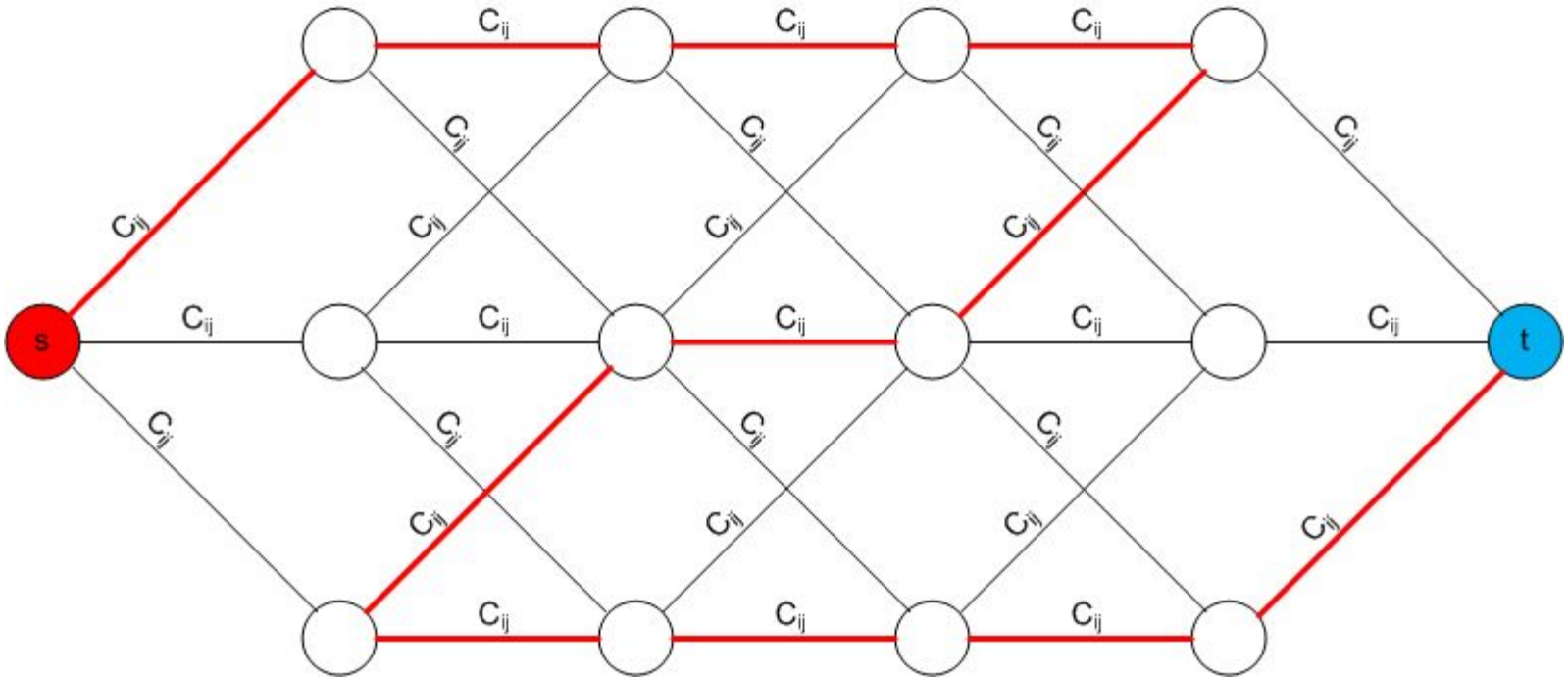
Найденный минимальный маршрут

Примеры использования поиск минимальных затрат при найме сотрудников



Применение к сетевому планированию и управлению

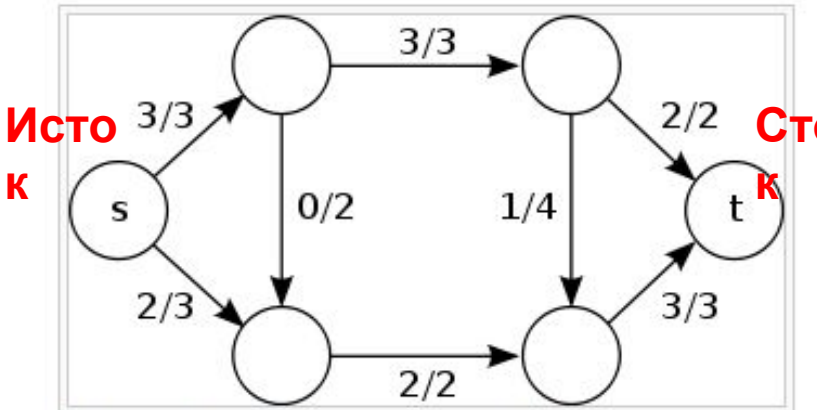
- Самый длинный путь называется



Задача о максимальном потоке

Задача о максимальном потоке (в транспортной сети)

- **Транспортной сетью** называется пара T=(G,C) , где G - *взвешенный орграф* , удовлетворяющий следующим условиям:
  a) нет петель;
  b) существует только одна вершина, не имеющая ни одного прообраза - это исток;
  c) существует только одна вершина, не имеющая ни одного образа - это сток;



Максимальный поток в транспортной сети. Числа обозначают потоки и пропускные способности.

- C - *функция, пропускных способностей дуг* ,которая является положительной вещественной функцией, определенной на множестве дуг графа.

Задача о максимальном потоке

- **Потоком** в транспортной сети Т называется неотрицательная вещественная функция, определенная на множестве дуг, удовлетворяющая условиям:
  - **ограниченности**: поток по любой дуге сети **не превосходит** пропускной способности этой дуги;
  - **сохранения**: суммарный поток , заходящий в любую вершину сети (кроме истока и стока) **равен** суммарному потоку , выходящему из этой вершины.
- **Дуга** сети называется **насыщенной**, если поток по этой дуге **равен** пропускной способности этой дуги.
- **Поток** по пути называется **полным**, если хотя бы одна дуга пути насыщена.
- **Величиной потока** называется **сумма значений этой функции** по всем выходным дугам сети (выходные дуги сети - это дуги, инцидентные стоку).Понятия потока и величины потока в сети часто путают, однако между ними существует различие: поток - это функция, а величина потока - число. Советуем это запомнить.

Алгоритм Форда-Фалкерсона (шаги 1-3)

1. Перенумеровать все вершины сети произвольным образом, кроме истока и стока.
2. Задать некоторый начальный поток в сети (например, нулевой).
3. Вершинам сети присвоить целочисленные метки, а дугам - знаки “+” и “-” по следующим правилам:
   a. вершине-истоку присвоить метку 0;
   b. находим **непомеченную** вершину W , **смежную помеченной** вершине V.
      1. Если **поток** по соединяющей вершины V-W дуге **меньше** **пропускной способности** этой дуги, то происходит **помечивание,** иначе переходим к рассмотрению следующей вершины**.**
      2. Если вершина W является **образом** помеченной вершины V , то происходит **прямое помечивание** : вершина W получает метку ,равную номеру вершины V, соединяющая вершины V-W дуга получает метку “ + ” и мы переходим к рассмотрению следующей вершины.
      3. Если вершина W не имеет ни одного помеченного прообраза, то происходит **обратное помечивание**: вершина W получает метку, равную номеру вершины V (являющейся в данном случае её образом), соединяющая вершины W-V дуга получает метку “ - ” и мы переходим к рассмотрению следующей вершины.
      4. Процесс помечивания продолжается до тех пор , пока все удовлетворяющие этим условиям вершины не получат метку.

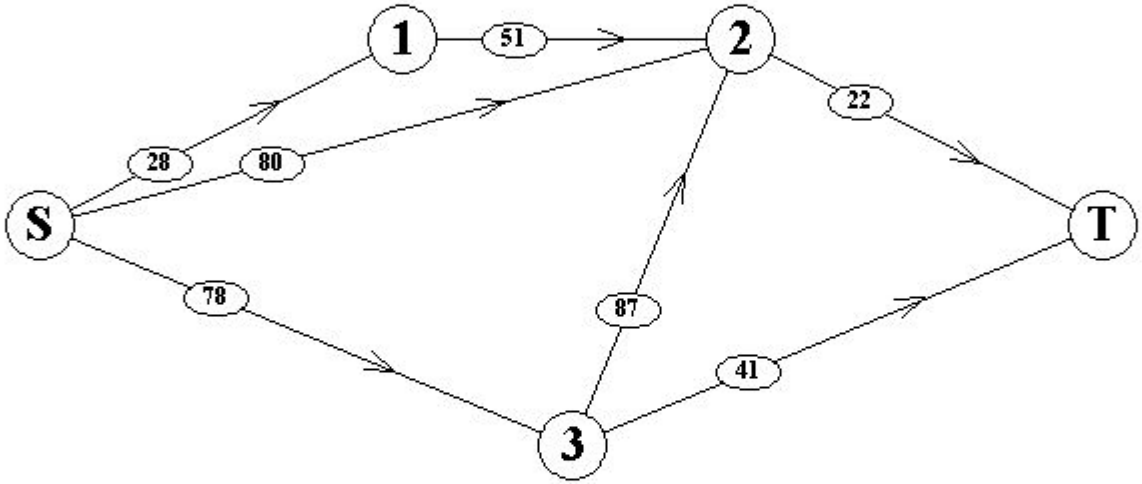
Алгоритм Форда-Фалкерсона (шаги 4-5)

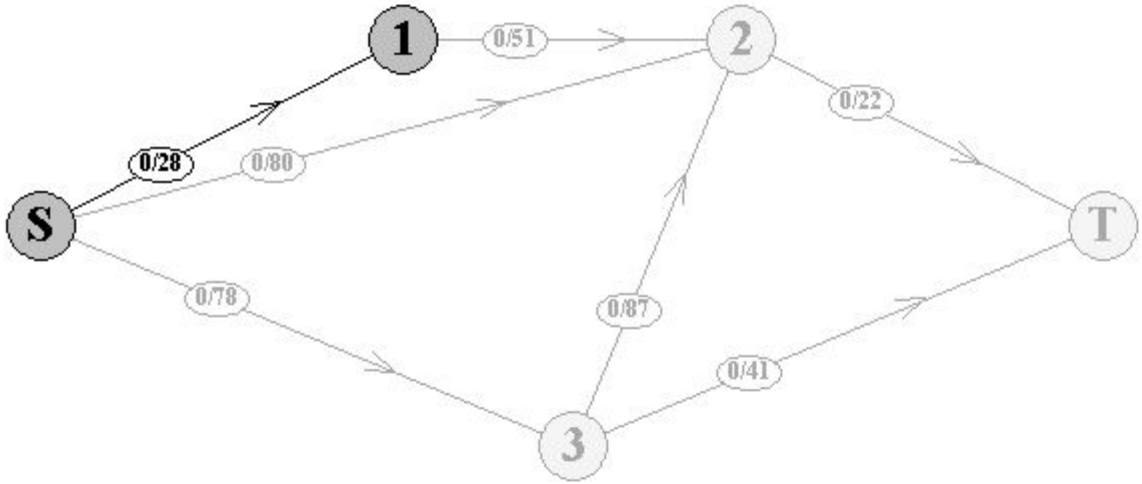
4. Если в результате процедуры помечивания вершина-сток метки не получила, то текущий поток - максимальный, задача решена. В противном случае, перейти к пункту **5**.
5. Рассмотреть последовательность вершин L=((сток)Xn,...,(исток)X0), метка каждой из которых равна номеру следующей за ней вершины, и множество дуг M, соединяющих соседние вершины из L. **Построить новый поток по схеме:**
   a. Если дуга принадлежит множеству M(смотри выше) и имеет знак “ + ”, то **новый поток по этой дуге = старый поток по этой дуге + К**
   b. Если дуга принадлежит множеству M и имеет знак “ - ”, то **новый поток по этой дуге = старый поток по этой дуге – К**
   c. Если дуга не принадлежит множеству M ,то **новый поток по дуге = старый поток по этой дуге.**
6. Перейти к п.**3.**

Алгоритм Форда-Фалкерсона (схема нахождения К)

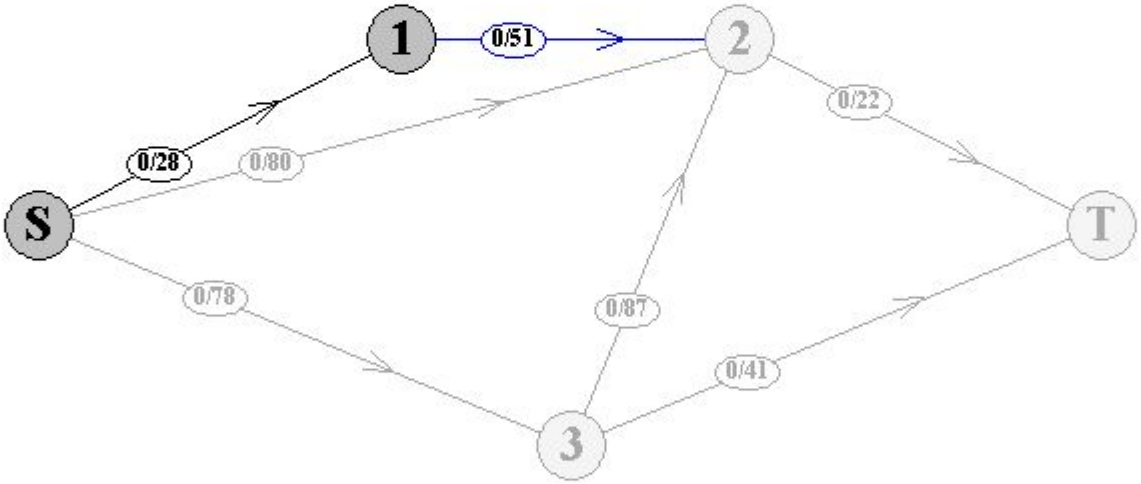
- Схема нахождения К:
- **К = min{ К1;К2 }, где**
- **Для нахождения К1** рассматриваются все дуги , принадлежащие множеству М и имеющие знак “ + ” и для каждой такой дуги вычисляется разность между пропускной способностью дуги и потоком по этой дуге. Затем из этих значений разностей выбирается минимальное значение и присваивается К1.
- **Для нахождения К2** рассматриваются все дуги , принадлежащие множеству М и имеющие знак “ - ” . Затем из этих дуг выбирается дуга с минимальным потоком и значение потока по этой дуге присваивается К2.

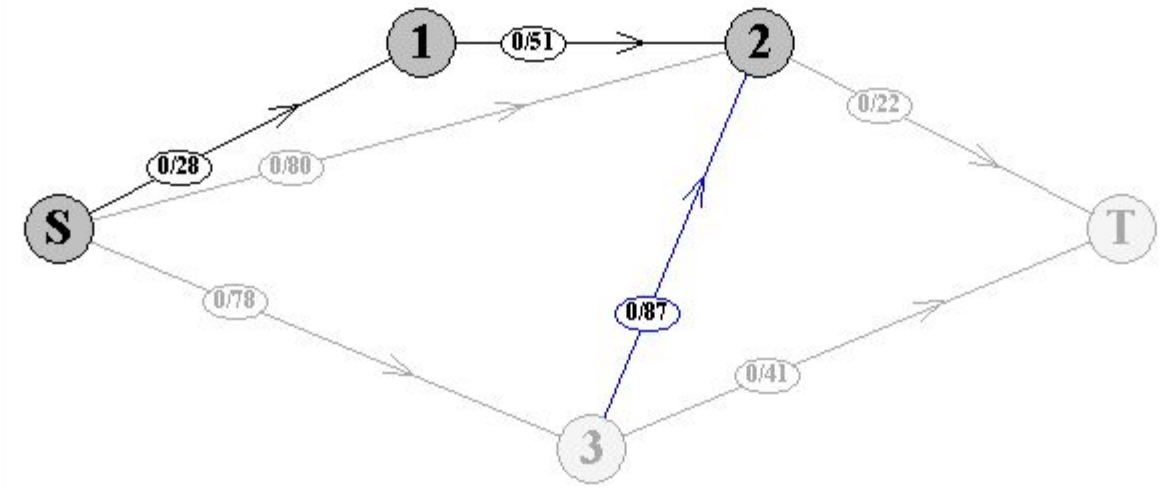
Пример



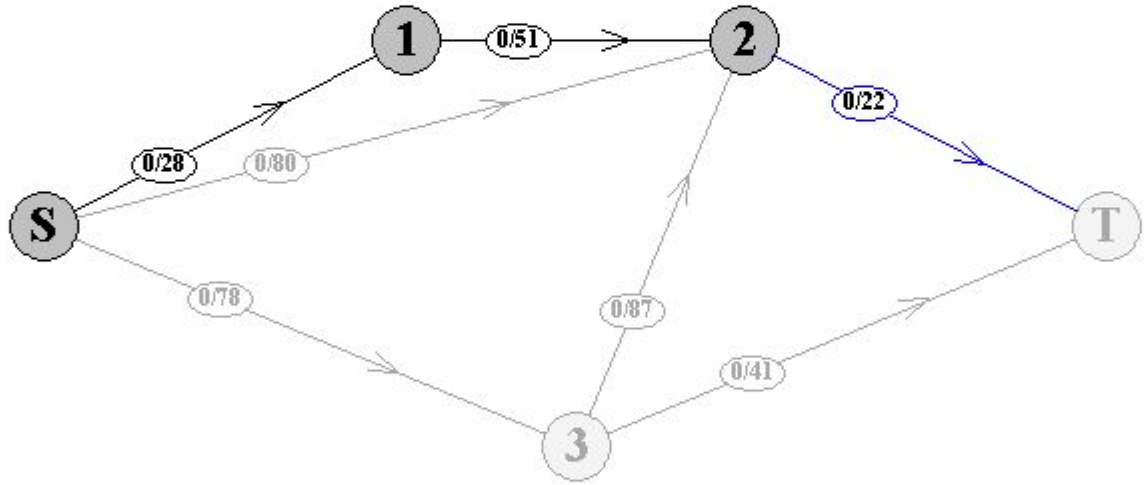


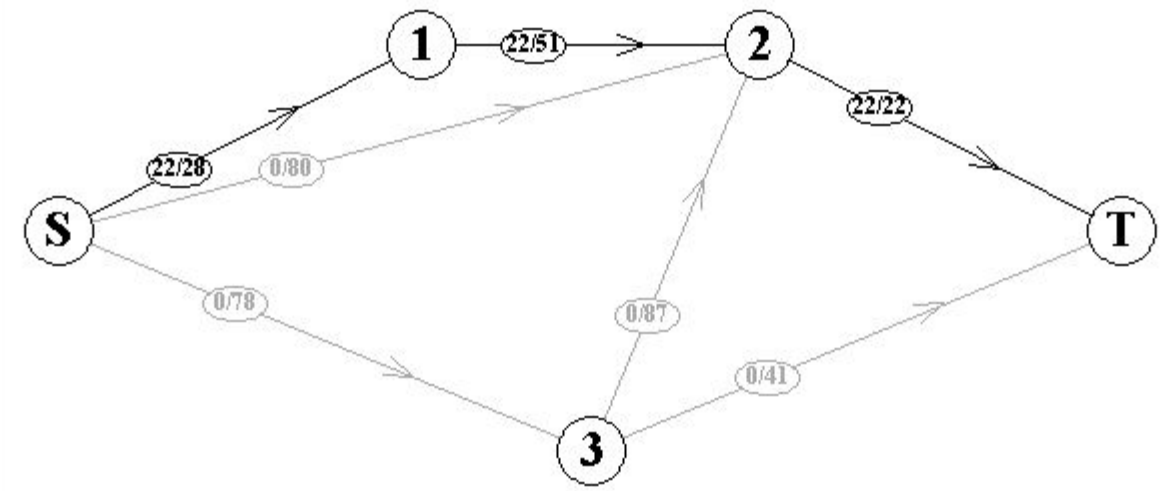
Пример


1
0/51
2
0/22
0/28
0/80
S
T
0/78
0/87
0/41
3

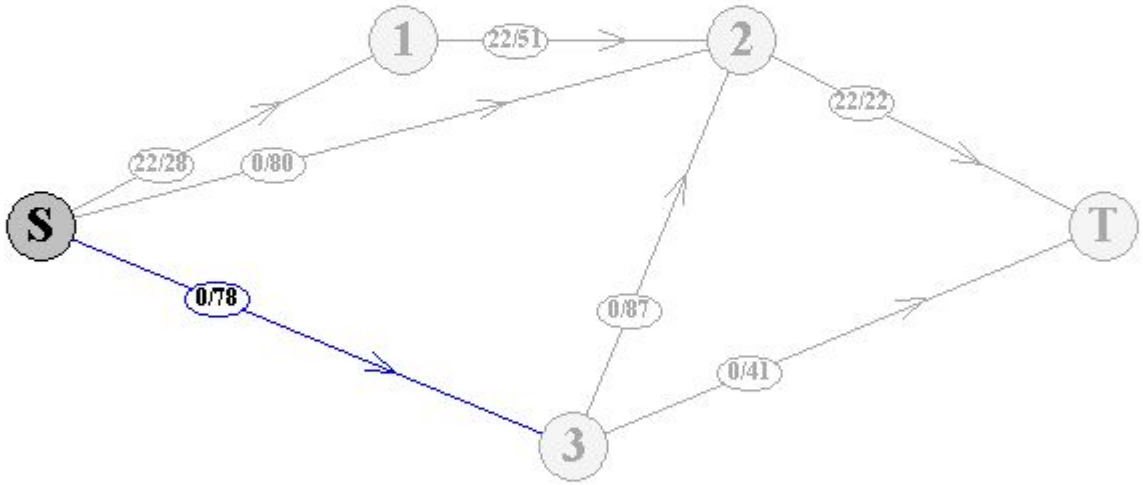

1
0/51
2
0/22
0/28
0/80
S
T
0/78
0/87
0/41
3

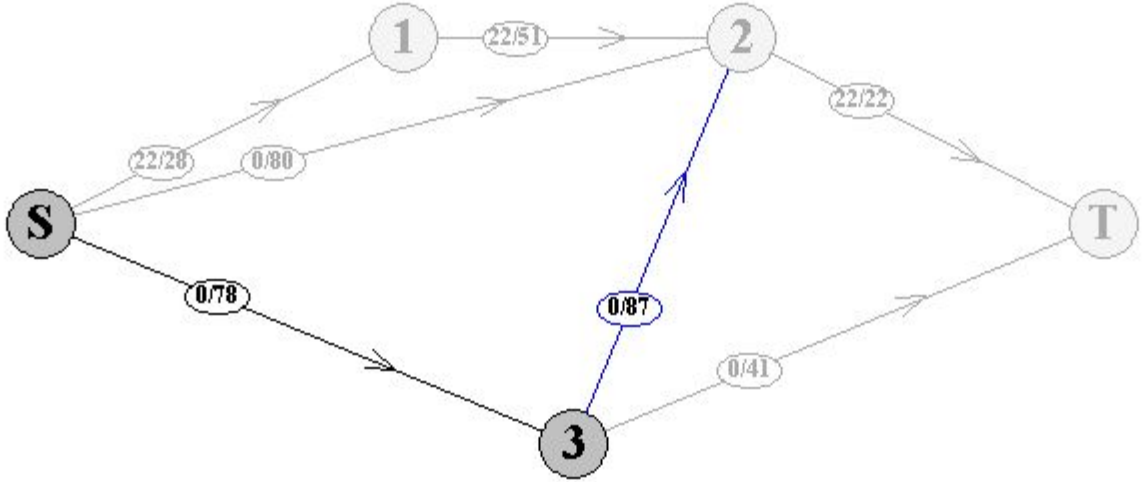
Пример


1
2
0/51
0/22
0/28
0/80
S
T
0/78
0/87
0/41
3

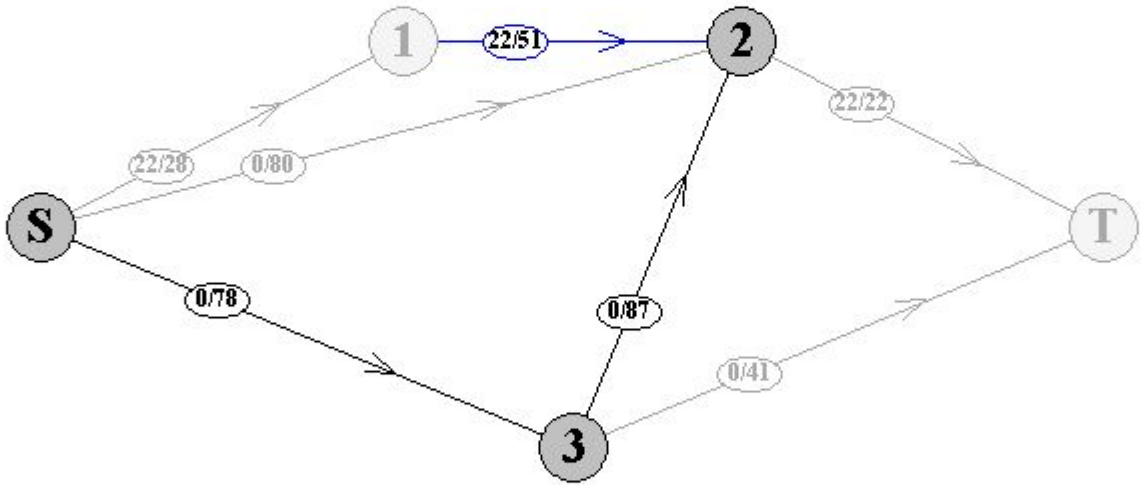

1
2
22/51
22/22
22/28
0/80
S
T
0/78
0/87
0/41
3

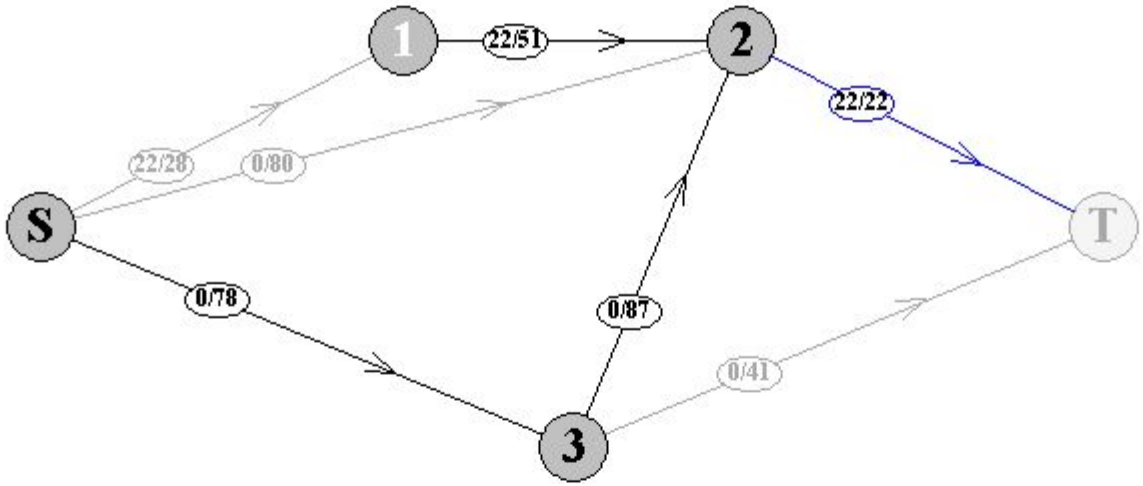
Пример


1
2
22/51
22/22
22/28
0/80
S
T
0/78
0/87
0/41
3

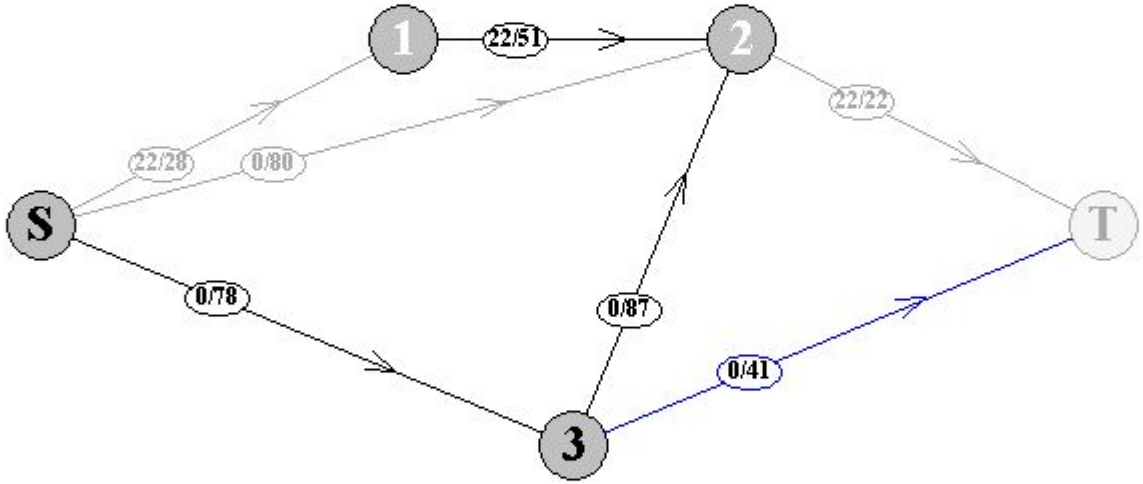

1
2
22/51
22/22
22/28
0/80
S
T
0/78
0/87
0/41
3

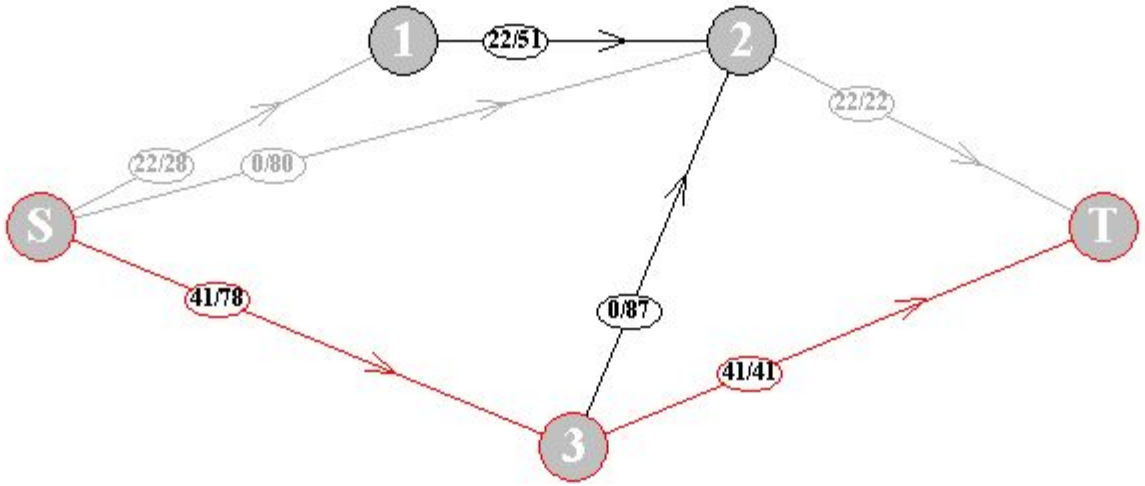
Пример


1
22/51
2
22/22
22/28
0/80
S
T
0/78
0/87
0/41
3

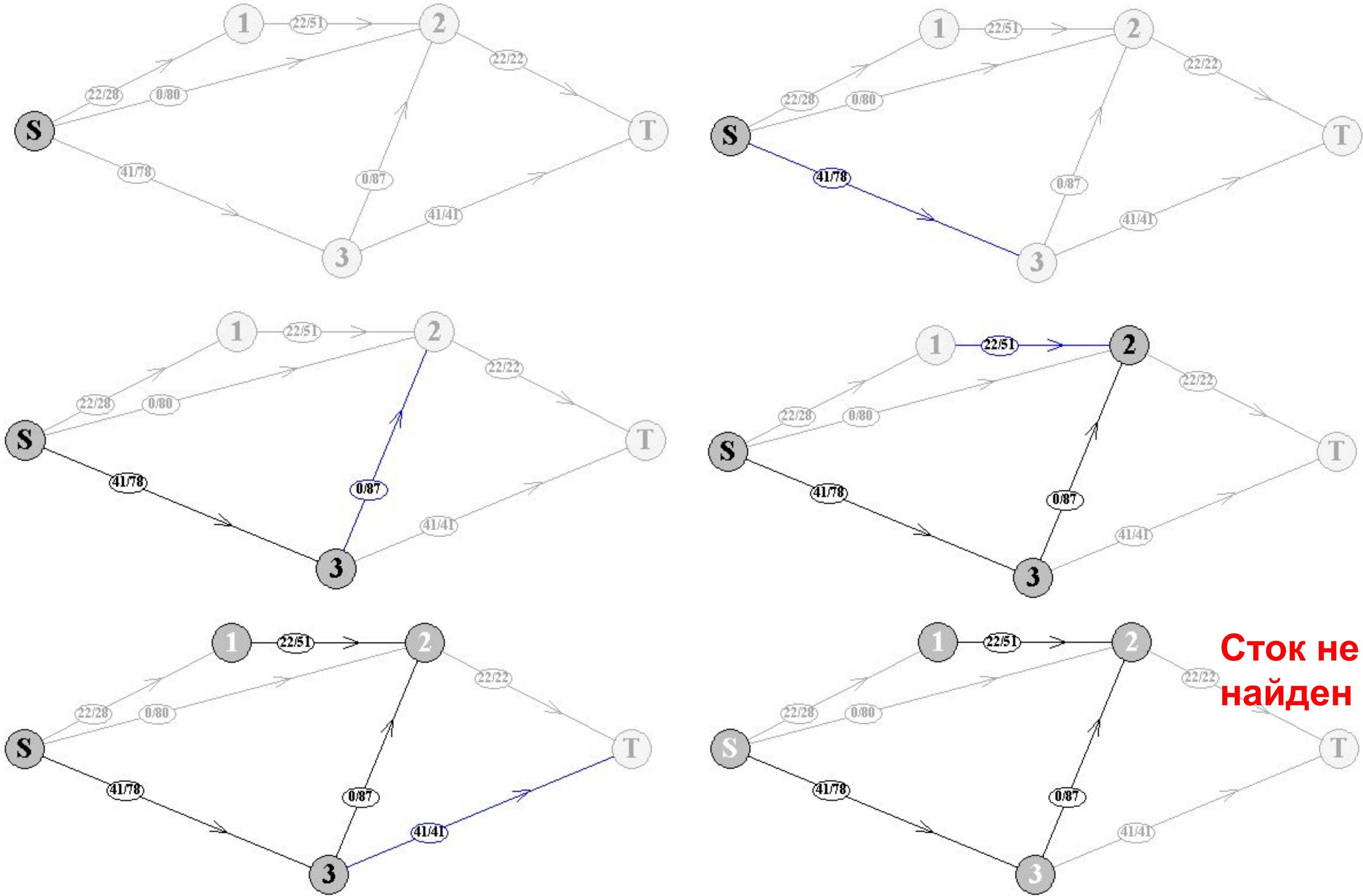

1
22/51
2
22/22
22/28
0/80
S
T
0/78
0/87
0/41
3

Пример

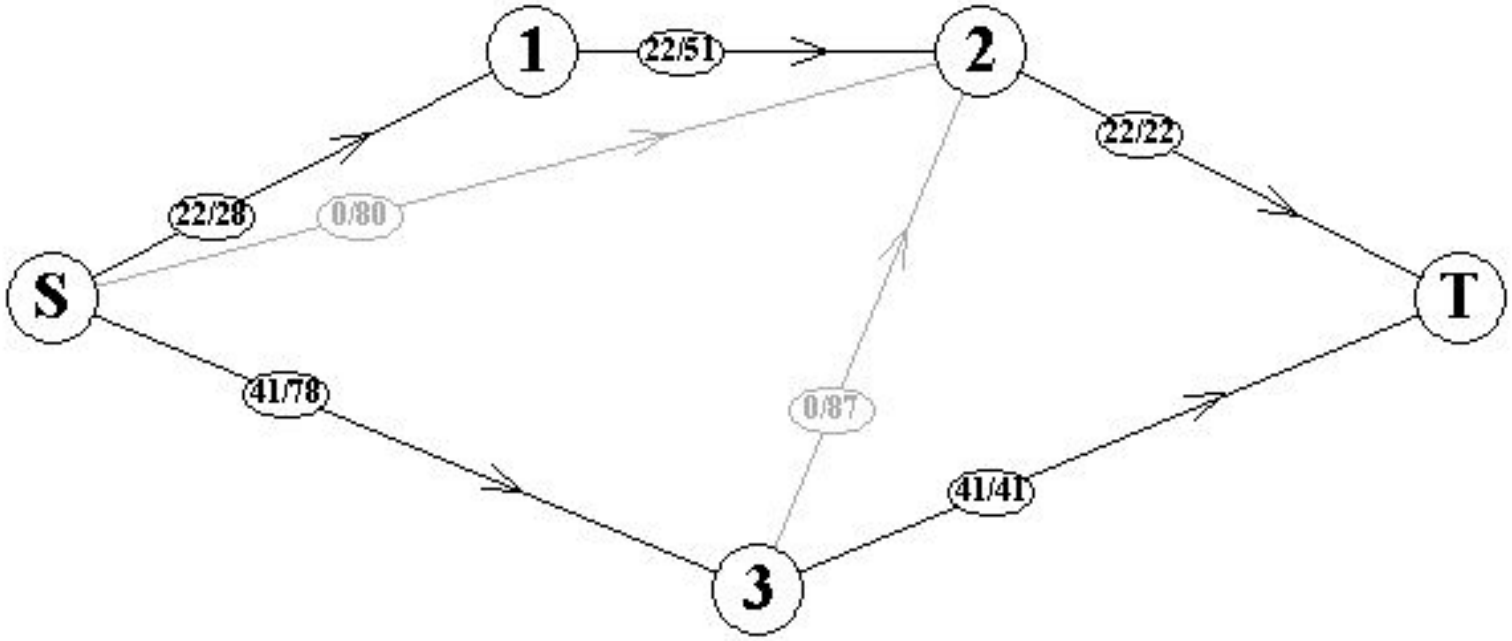

1
22/51
2
22/22
22/28
0/80
S
T
0/78
0/87
0/41
3


1
22/51
2
22/22
22/28
0/80
S
T
41/78
0/87
41/41
3

Пример



Алгоритм закончен



- Алгоритм закончен. Максимальный поток в данной сети равен 63.

http://rain.ifmo.ru/cat/view.php/vis/graph-flow-match/ford-fulkerson-2008

Примеры использования задачи поиска максимального потока

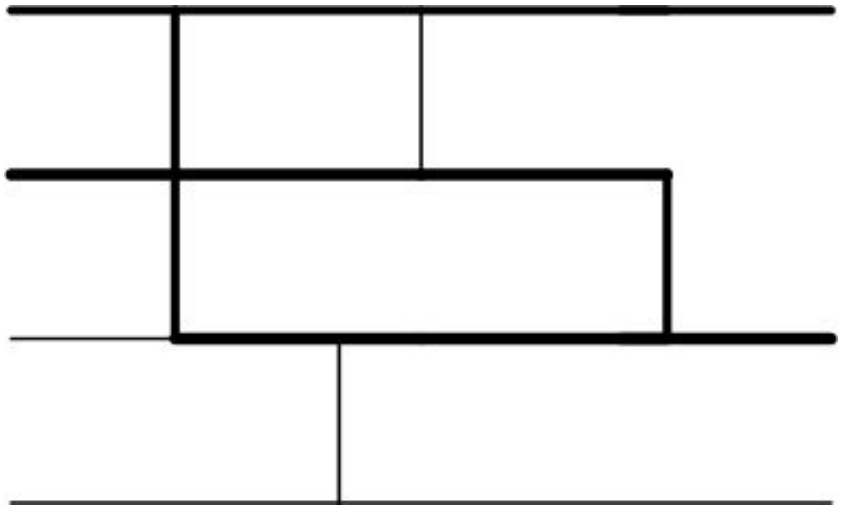
- Пример 1. Расчет пропускной способности компьютерной или дорожной сети
- Пример 2. Распределение работы между несколькими работниками

- Пример 3. Задача поиска максимального потока минимальной стоимости

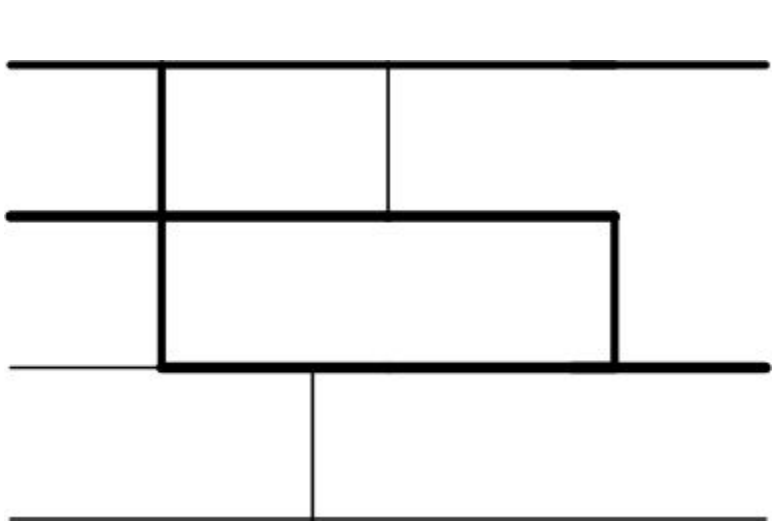
Пример 1

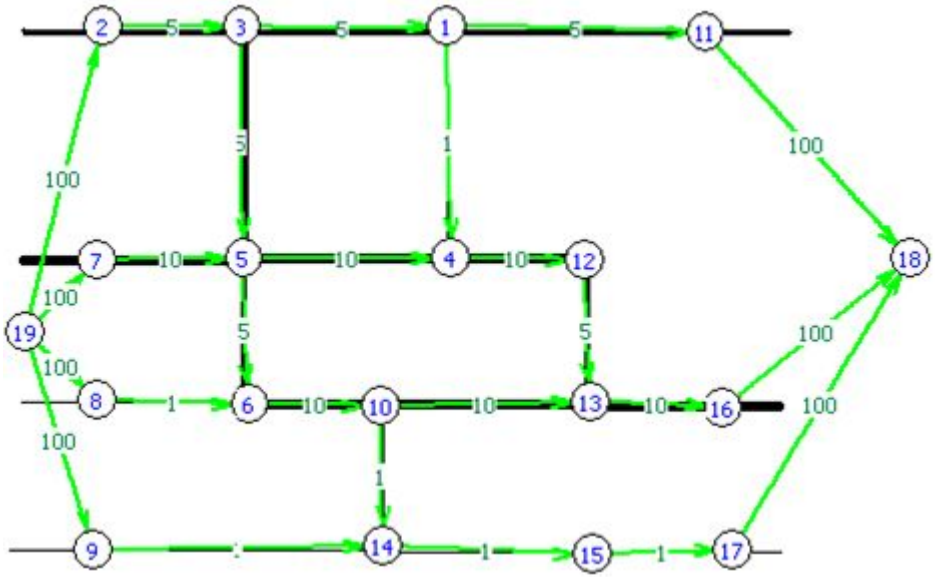
Расчет пропускной способности компьютерной или дорожной сети

Расчет пропускной способности компьютерной или дорожной сети



Расчет пропускной способности компьютерной или дорожной сети



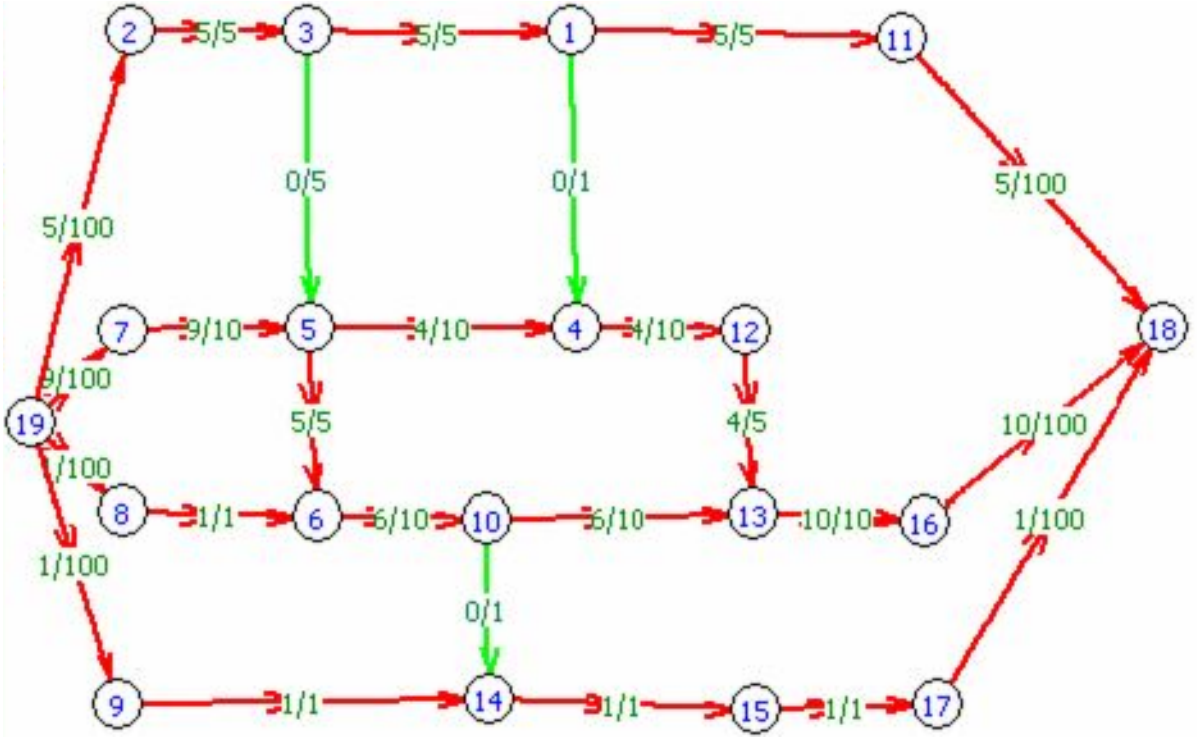


Граф на основе карты дорог

Расчет пропускной способности компьютерной или дорожной сети

Исток
5 +
9 +
1 +
1 +

= 16



Сток
5 +
10 +
1 +

= 16

Максимальная пропускная способность сети дорог

Пример 2

Распределение работы между несколькими работниками

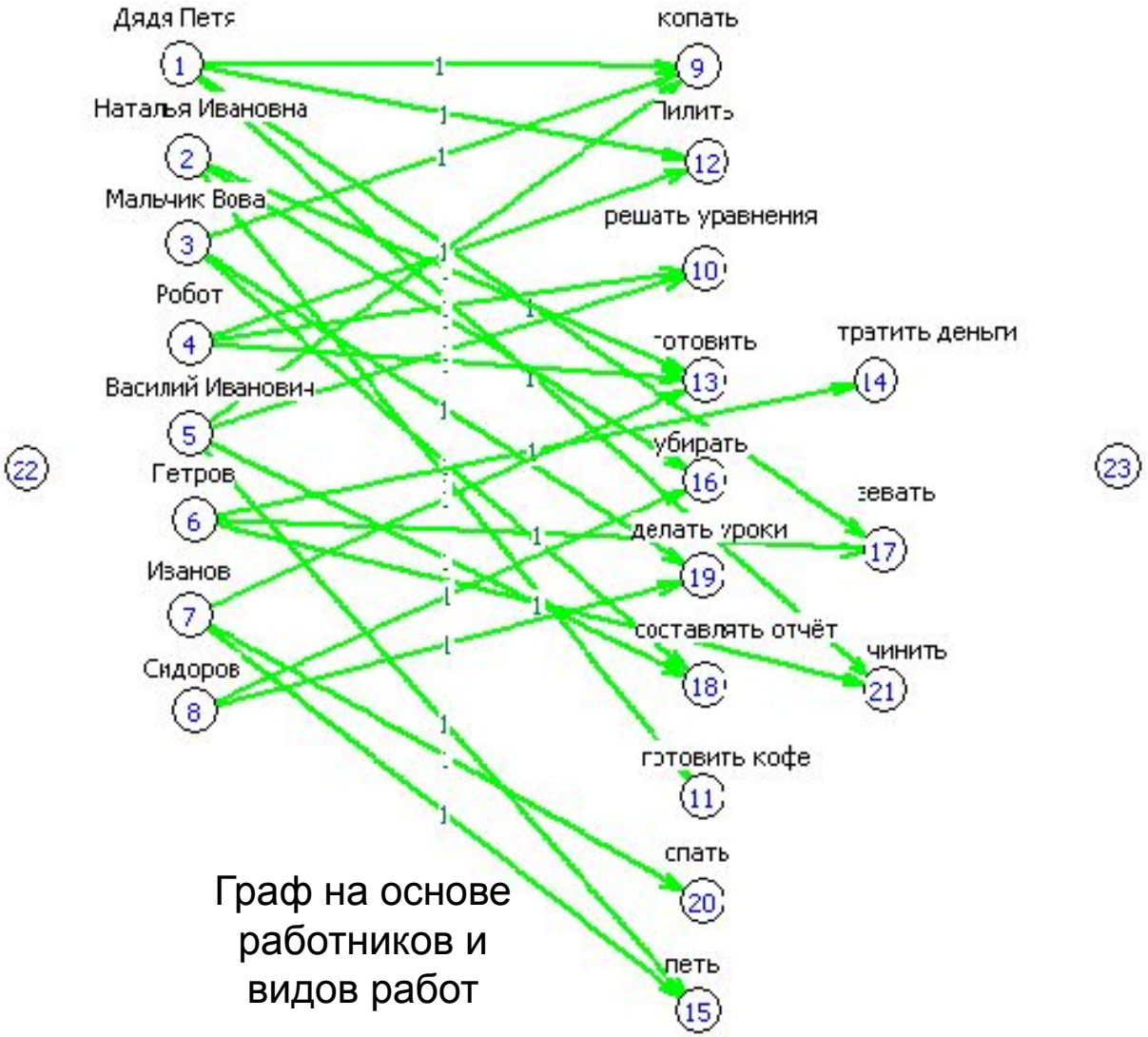
Распределение работы между несколькими работниками

Список работников	Список работ
1. Дядя Петя;	1. Копать;
2. Наталья Ивановна;	2. Пилить;
3. Мальчик Вова;	3. Решать уравнения;
4. Робот;	4. Готовить;
5. Василий Иванович;	5. Убирать;
6. Петров;	6. Делать уроки;
7. Иванов;	7. Составлять отчёт;
8.Сидоров.	8. Готовить кофе;
	9. Спать;
	10. Петь;
	11. Тратить деньги;
	12. Зевать;
	13. Чинить;

Распределение работ

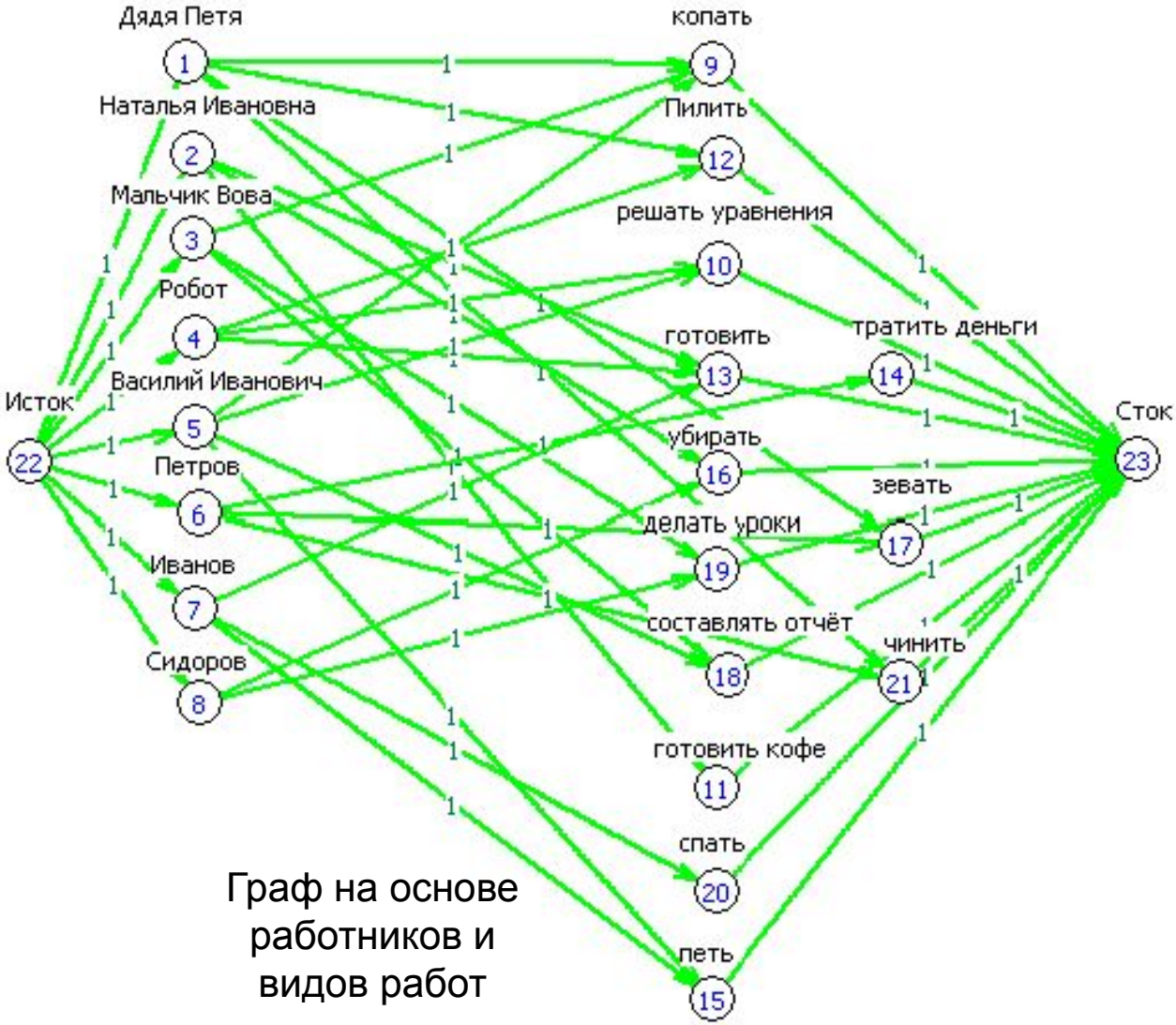
	Дядя Петя;	Наталья Ивановна;	Мальчик Вова;	Робот;	Василий Иванович;	Петров;	Иванов;	Сидоров;
Копать;	V		V		V			
Пилить;	V			V				
Решать уравнения;				V	V			
Готовить;		V		V				
Убирать;		V						V
Делать уроки;			V					V
Составлять отчёт;			V		V			
Готовить кофе;		V					V	
Спать;							V	
Петь;					V		V	
Тратить деньги;						V		
Зевать;	V					V		
Чинить;	V					V		

Распределение работы между несколькими работниками



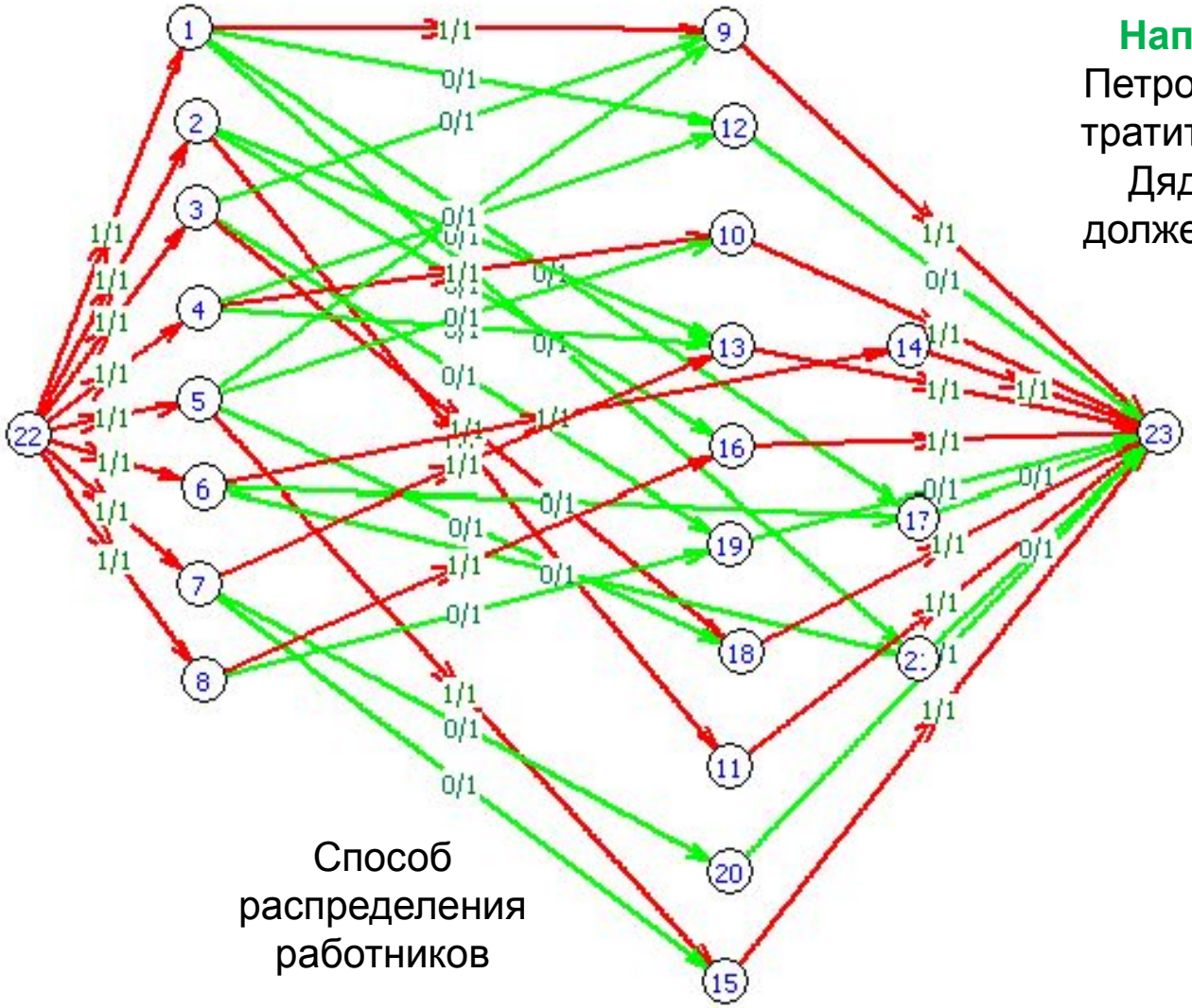
Граф на основе работников и видов работ

Распределение работы между несколькими работниками



Граф на основе работников и видов работ

Распределение работы между несколькими работниками



Способ распределения работников

Например: Петров должен тратить деньги, Дядя Петя должен копать.

Пример 3.

Задача поиска максимального пути минимальной стоимости

Задача поиска максимального пути минимальной стоимости

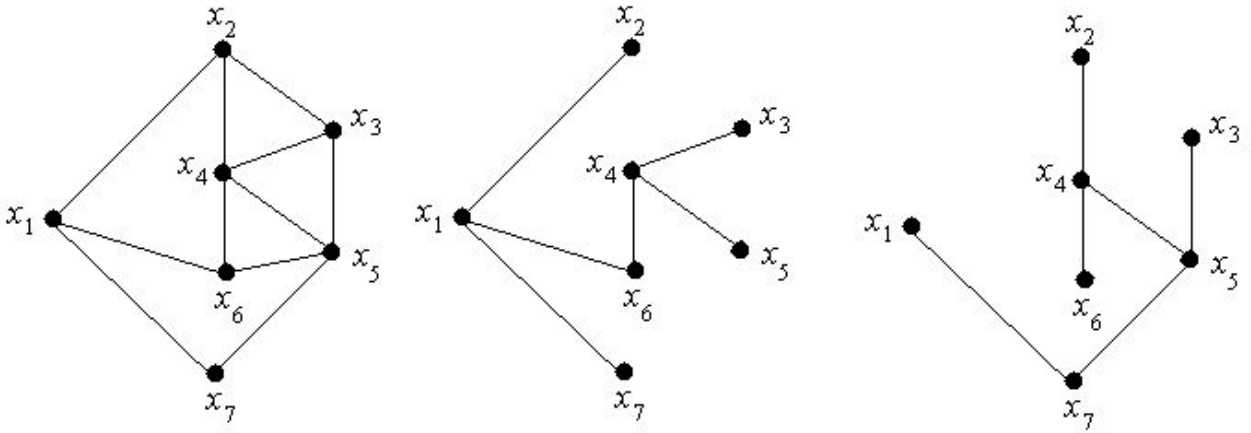
Необходимо узнать: сколько ящиков «Товара» в день вы можете подвозить в аэропорт?

Деревья

1. Деревья
2. Остовные деревья (остовы)
3. Задача поиска минимального остовного дерева
4. Алгоритмы

Деревья

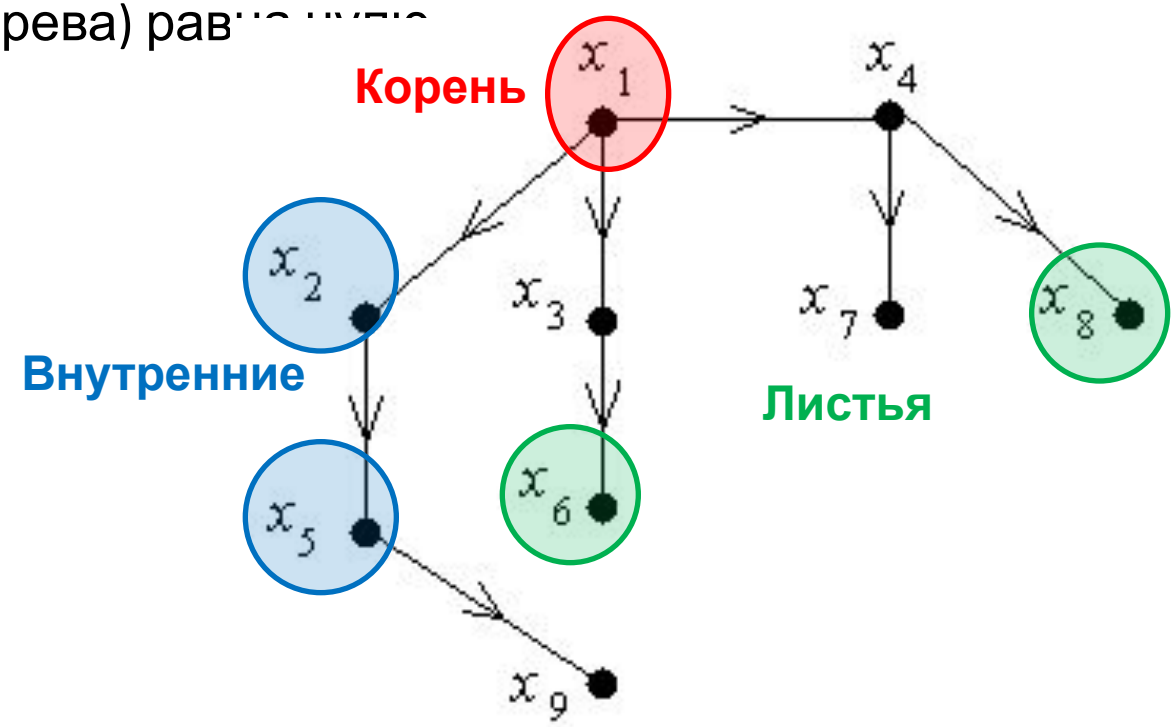
- **Неориентированное дерево** это:
  1. связный граф, содержащий *n* вершин и *n*-1 ребер;
  2. **связный граф, не имеющий циклов;**
  3. граф, в котором каждая пара вершин соединена одной и только одной простой цепью.
- **Остовным деревом (остовом)** неориентированного графа с n вершинами называется всякий остовной подграф графа *G*, являющийся деревом.



Граф G = (X,A) | Остов 1 графа G | Остов 2 графа G

Ориентированное дерево

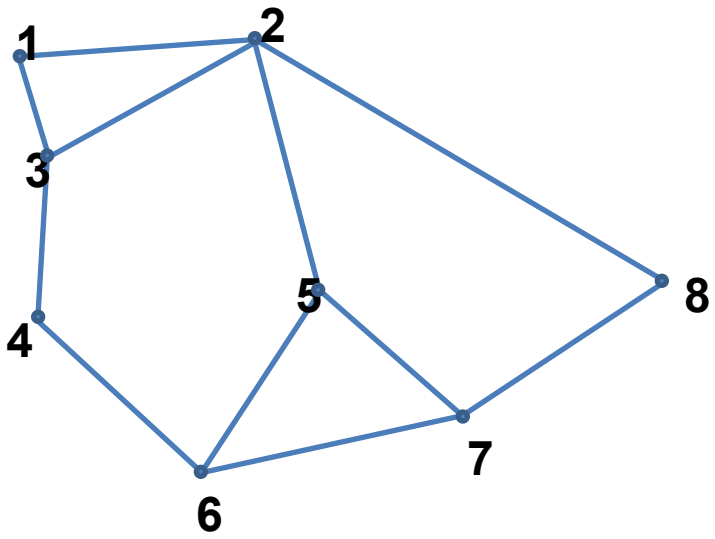
- **Ориентированное дерево** - это ориентированный граф без циклов, в котором полустепень захода каждой вершины, за исключением одной (начальной вершины x_1), равна единице, а полустепень захода вершины *x* (называемой корнем этого дерева) рав[illegible]



Ориентированное дерево

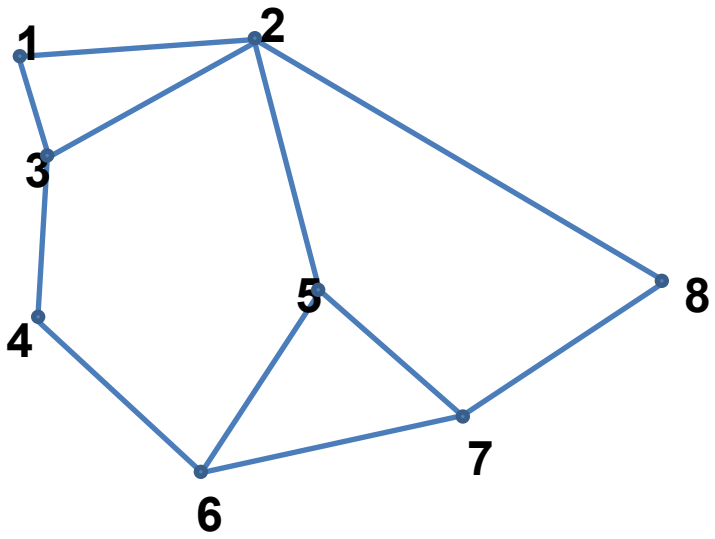
Цикломатической число

- ***Вопрос***: сколько ребер можно удалить из этого графа, чтобы получить остовое связанное дерево?



Цикломатической число G(8,11)

- ***Вопрос***: сколько ребер можно удалить из этого графа, чтобы получить остовое связанное дерево?



- ***Ответ***: Четыре ребра.
- Например, 2-3; 2-5; 6-7; 2-8 – это ребра, которые образуют циклы.

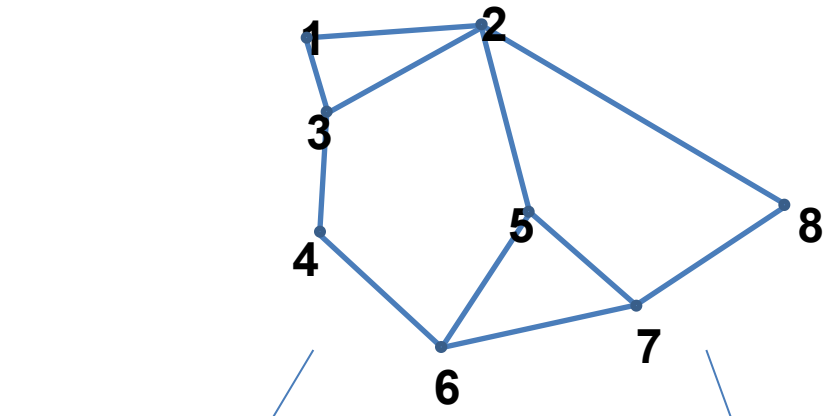
- Максимальное количество удаляемых ребер величина постоянная и зависит от количества вершин (n) и ребер (m) графа.

$$\gamma = 11 - 8 + 1 = 4$$

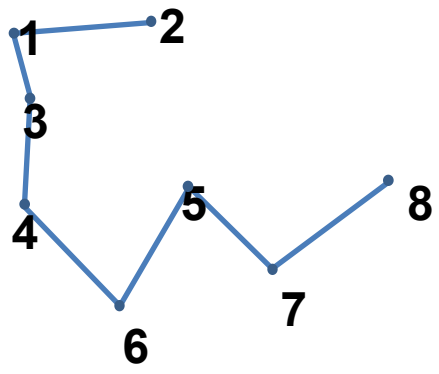
$$\gamma = m - n + 1,$$

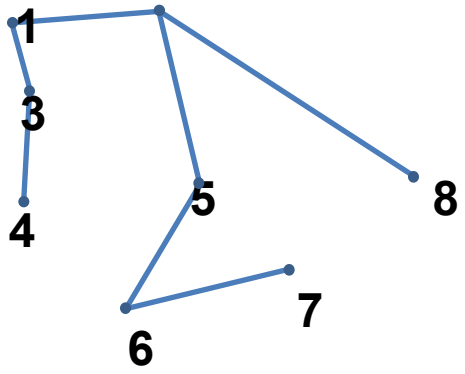
- Это величина получила название

Примеры возможного остовного связанного дерева.



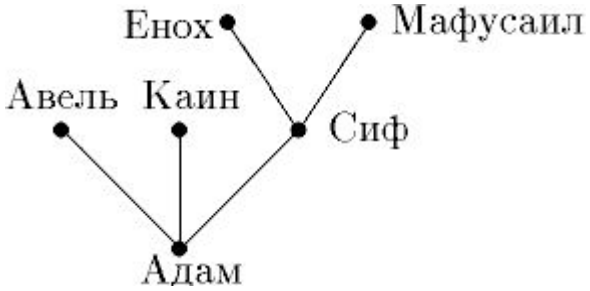
Граф G(X,A)

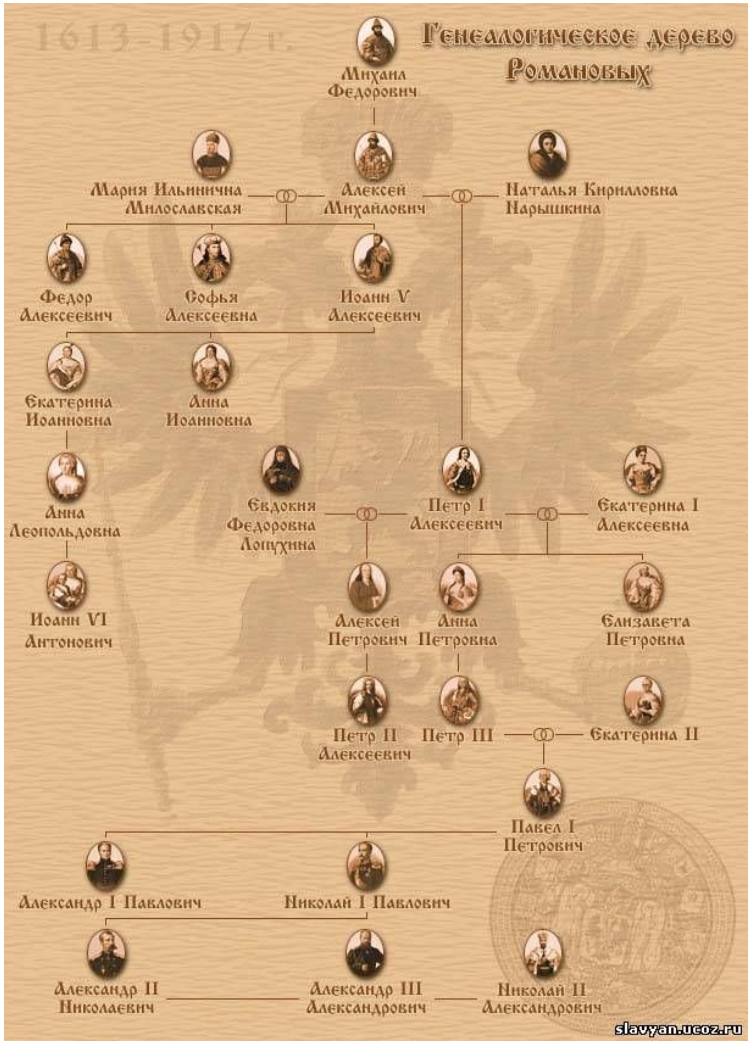




Возможные остовные связные деревья

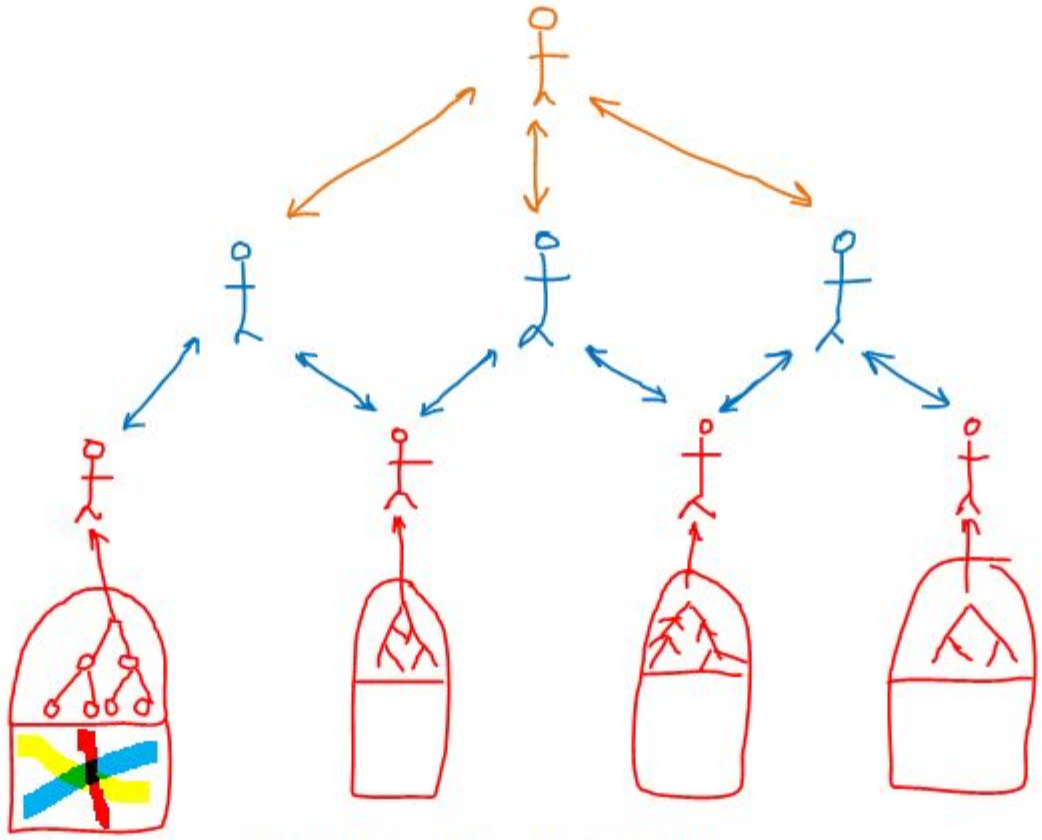
Пример. Генеалогическое дерево





Генеалогическое дерево Романовых

Пример. Иерархические структуры



Иерархическая структура управления в организациях

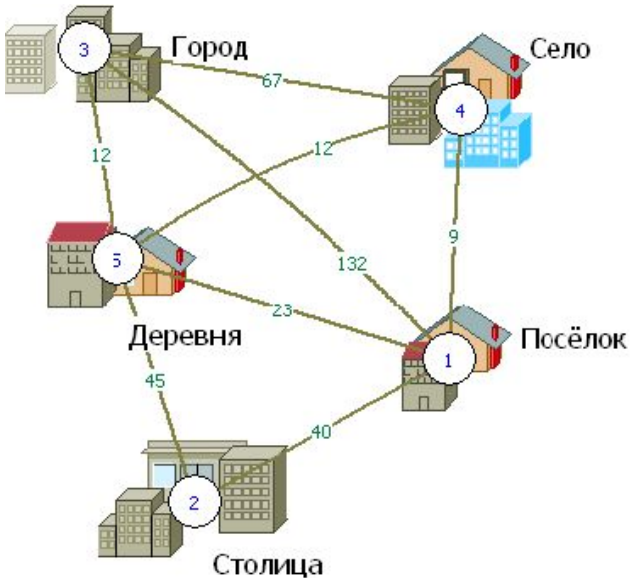
Задача поиска минимального остовного дерева (ПМОД)

Задача (ПМОД) поиска минимального остовного дерева

- **Постановка задачи:**
- Имеется *n* городов, которые нужно объединить в единую телефонную или транспортную сеть. Для этого достаточно проложить (*n*-1) телефонных линий или дорог между городами.

Задача (ПМОД) поиска минимального остовного дерева

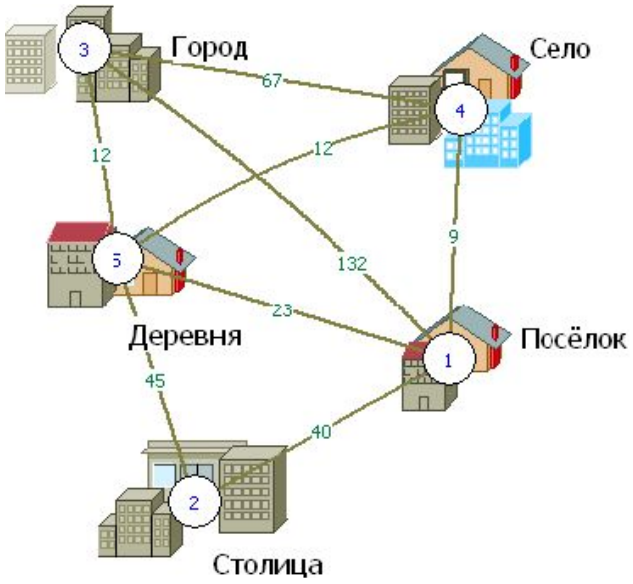
- **Постановка задачи:**
- Имеется *n* городов, которые нужно объединить в единую телефонную или транспортную сеть. Для этого достаточно проложить (*n*-1) телефонных линий или дорог между городами.
- Как соединить города так, чтобы суммарная стоимость соединений была минимальна?



Карта городов

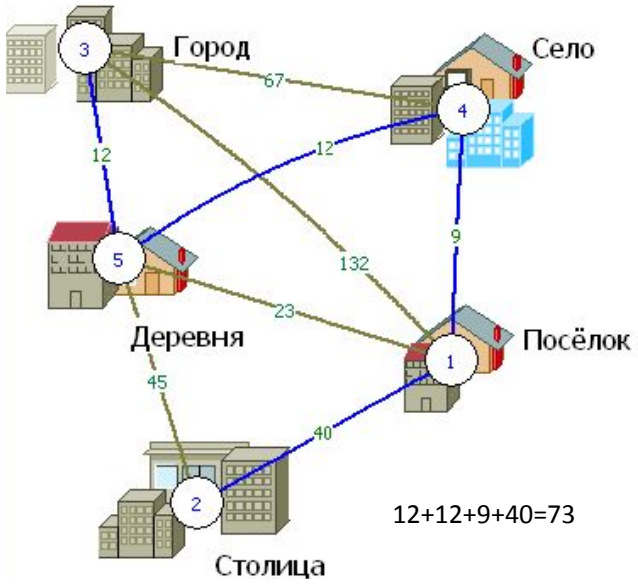
Задача (ПМОД) поиска минимального остовного дерева

- **Постановка задачи:**
- Имеется *n* городов, которые нужно объединить в единую телефонную или транспортную сеть. Для этого достаточно проложить (*n*-1) телефонных линий или дорог между городами.
- Как соединить города так, чтобы суммарная стоимость соединений была минимальна?



Карта городов

Наити минимальное оставное дерево

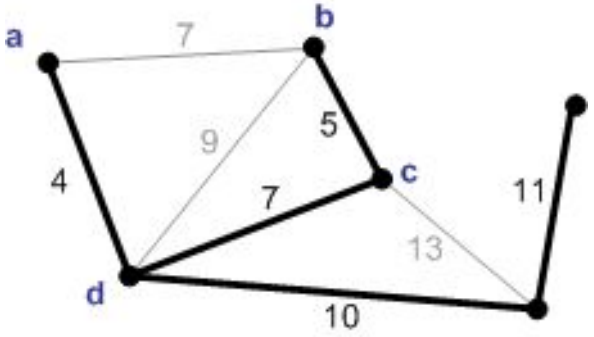


Минимальная стоимость постройки дорог

Задача (ПМОД)
Общая постановка задачи

- Дан связный, неориентированный граф $G = (X,A)$ с весами на ребрах.
- Для каждого ребра $a(u,v)$ однозначно определено некоторое вещественное число $w(u,v)$ — его вес.

- **Задача:** найти такой связный ациклический (без циклов) подграф $T \subset G$, содержащего все вершины, что суммарный вес его ребер будет минимален.



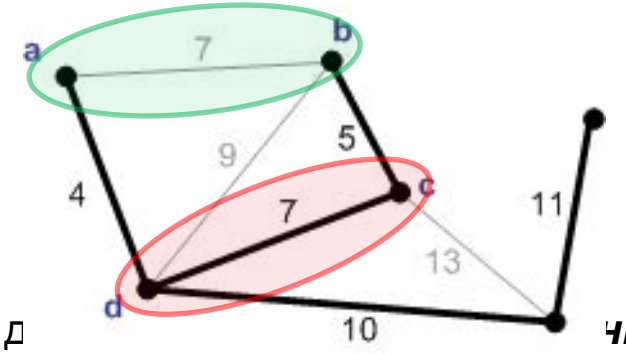
Минимальное остовное дерево. Суммарный вес дерева равен 37.

- Граф T является деревом и называется ***остовным*** или ***покрывающим деревом*** (*spanning tree*).
- Остовное дерево T, у которого суммарный вес его ребер $w(T) = \sum_{(u,v) \in T} w(u,v)$ минимален, называется ***минимальным остовным*** или ***минимальным покрывающим деревом*** (*minimum spanning tree*).

Задача (ПМОД)
Общая постановка задачи

- Дан связный, неориентированный граф $G = (X,A)$ с весами на ребрах.
- Для каждого ребра $a(u,v)$ однозначно определено некоторое вещественное число $w(u,v)$ — его вес.

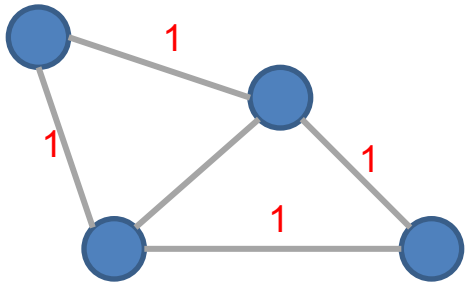
- **Задача:** найти такой связный ациклический (без циклов) подграф $T \subset G$, содержащего все вершины, что суммарный вес его ребер будет минимален.

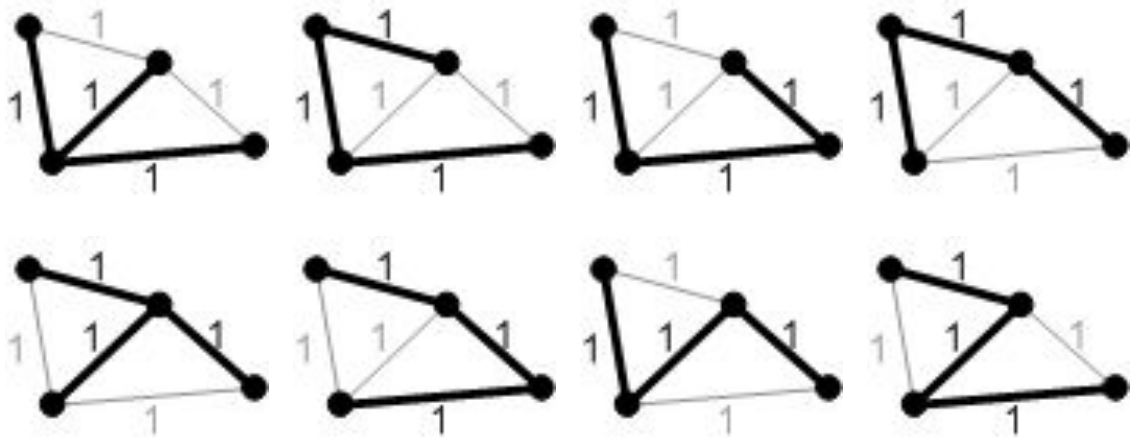


Минимальное остовное дерево. Суммарный вес дерева равен 37.

- Граф T является д ***чым*** или ***покрывающим деревом*** (*spanning tree*).
- Остовное дерево T, у которого суммарный вес его ребер $w(T) = \sum_{(u,v) \in T} w(u,v)$ минимален, называется ***минимальным остовным*** или ***минимальным покрывающим деревом*** (*minimum spanning tree*).

Пример





Все возможные минимальные остовные деревья для данного графа

Задача (ПМОД)
Алгоритмы решения

- Алгоритм Крускала
- Алгоритм Прима

Жадная стратегия

- Дан связный взвешенный неориентированный граф $G(X,A)$ с n вершинами.
- Искомый остов строится постепенно.
- Алгоритм использует некоторый ациклический подграф T исходного графа G, который называется ***промежуточным остовным лесом***.
- Очередное добавляемое ребро $a(u,v)$ выбирается так, чтобы не нарушить свойства: $T \cup \{a\}$ тоже должно быть подграфом минимального остова.
- **(!!! Т.е. не добавлять циклов в граф Т и быть минимальным по весу)**.
- ***Безопасным ребром*** относительно некоторой компоненты связности К из Т назовем ребро с минимальным весом, ровно один конец которого лежит в К.

Общий алгоритм построения минимального остовного дерева

```
1: T ← 0
2: while (пока) T не является остовом
3:     do найти безопасное ребро (u,v) ∈ E для T
4:         T ← T ∪ {(u,v)}
5: return T
```

Алгоритм Буровки

0. BORUVKA(G,w)

1: T ← (X,0)

2: while (пока) в T больше одной компоненты связности

3: do CHOOSE-LEADERS(T)

4: FIND-SAFE-EDGES(G)

5: foreach (для каждого) лидера x`

6: добавить safe(x`) в T

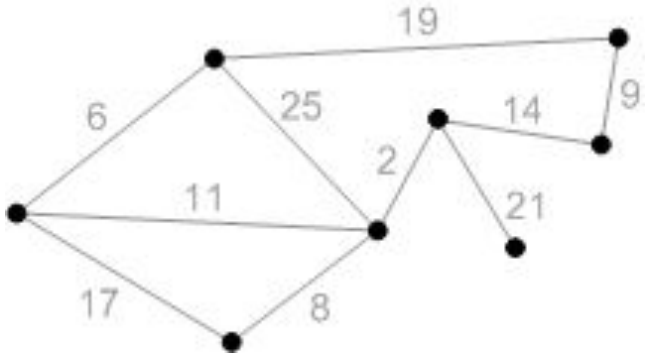
7: return A

0. Дан связный взвешенный неориентированный граф G(X;A) и на нем задана весовая функция w.
1. Пусть *T* – промежуточный остовный лес для графа *G*. На первом шаге *T* состоит из всех вершин *G* и пустого множества ребер.
2. Выбирается вершина «лидер» для каждой компоненты связности.
4. После того, как лидеры выбраны, для каждой компоненты связности находится безопасное для нее (для компоненты связности) ребро.
5.6. Как только все такие ребра отобраны, они добавляются к *T*. Процесс продолжается до тех пор, пока в *T* присутствует больше одной компоненты связности.

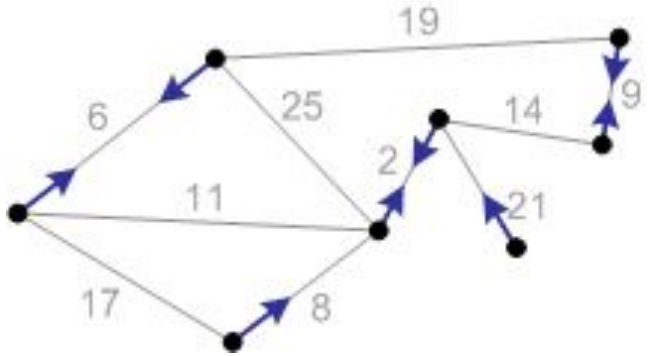
Вид возможной процедуры поиска безопасных ребер

```
FIND-SAFE-EDGES(G)
1: foreach (для каждого) лидера x'
2:   safe(x') ← ∞
3: foreach (для каждого) ребра (u,x) ∈ E
4:   u' ← leader(u)
5:   x' ← leader(x)
6:   if u' ≠ x' then
7:     if w(u,x) < w(safe(u')) then
8:       safe(u') ← (u,x)
9:     if w(u,x) < w(safe(x')) then
10:      safe(x') ← (u,x)
```

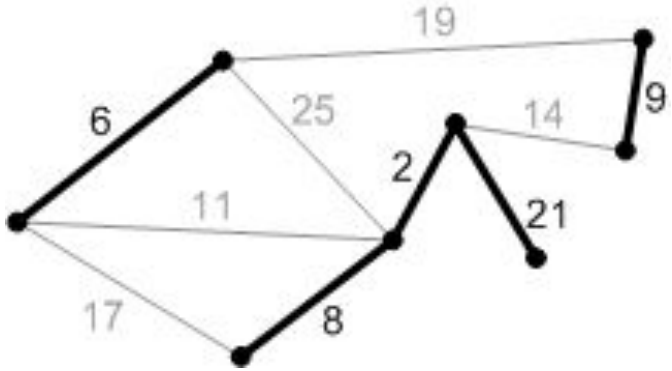
Пример работы алгоритма Буровки



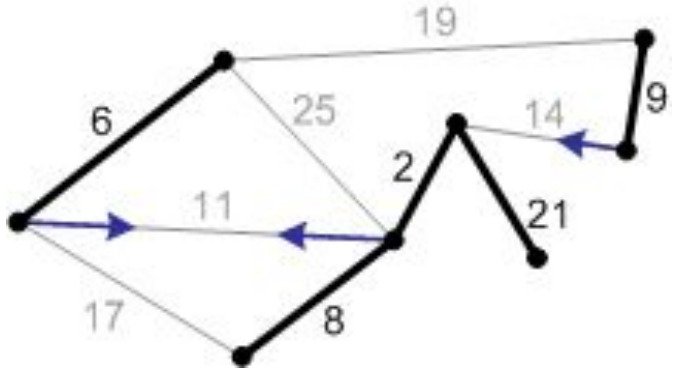
Фаза 1



Фаза 2

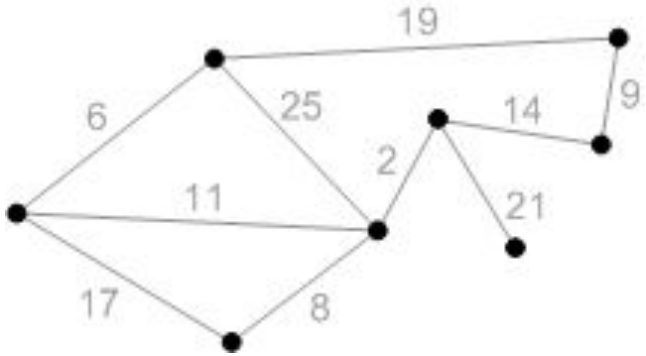


Фаза 3



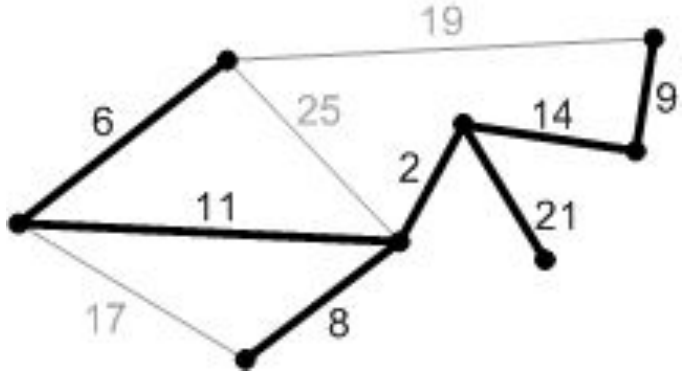
Фаза 4

Пример работы алгоритма Буровки



Фаза 1

Минимальное остовное дерево



Фаза 5

Алгоритм Крускала

0. KRUSKAL(*G*,*w*)

1: $T \leftarrow 0$
2: **foreach** (для каждой) вершины $x \in X[G]$
3: **do** Make-Set(*x*)
4: упорядочить ребра по весам
5: **for** $(u,x) \in A$ (в порядке возрастания веса)
6: **do if** Find-Set(*u*) ≠ Find-Set(*x*) **then**
7: $T \leftarrow T \cup \{(u,x)\}$
8: Union(*u*,*x*)
9: **return** *A*

`Make-Set(x)`	Создание множества из набора вершин
`Find-Set(x)`	Возвращает множество, содержащее данную вершину
`Union(u,x)`	Объединяет множества, содержащие данные вершины

Пример работы алгоритма Крускала

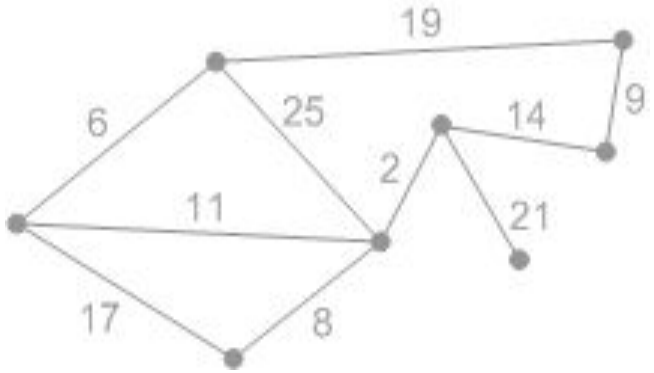


Рис 1.

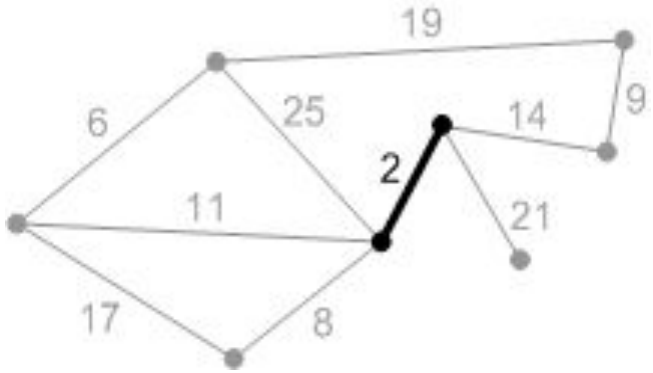


Рис 2.

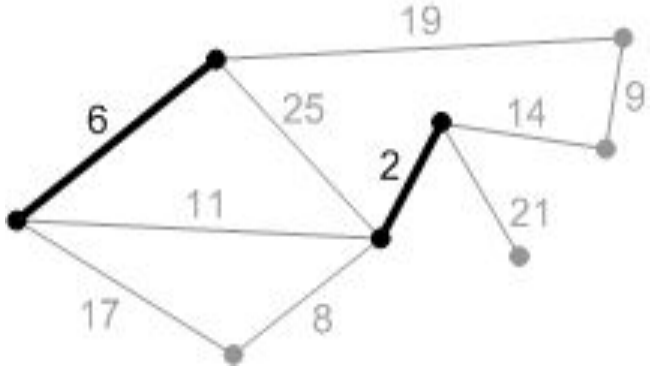


Рис 3.

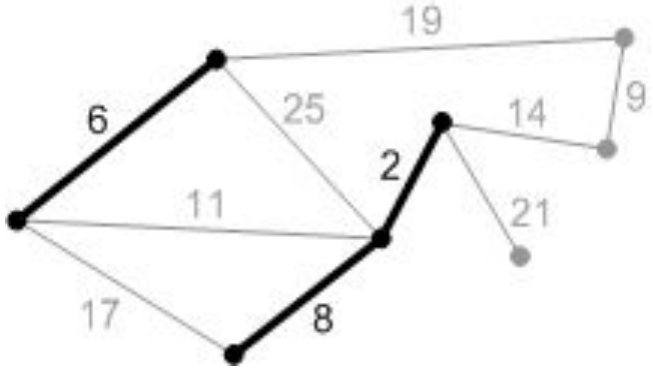


Рис 4.

Пример работы алгоритма Крускала

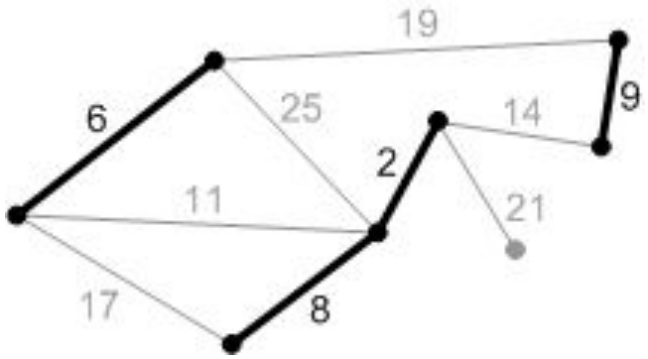


Рис 5.

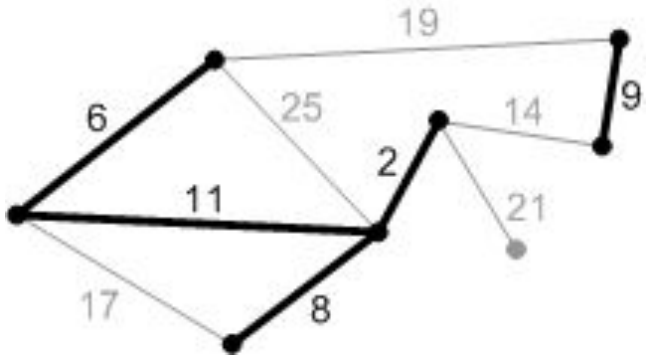


Рис 6.

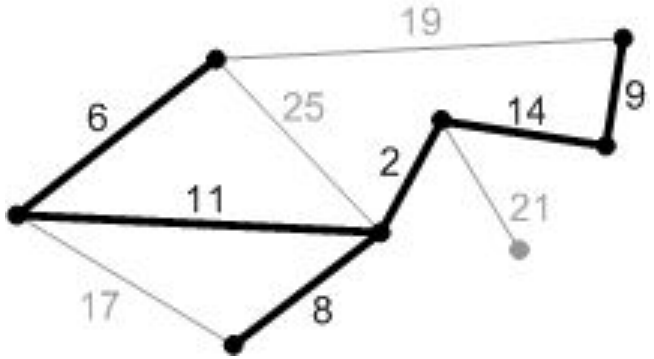


Рис 7.

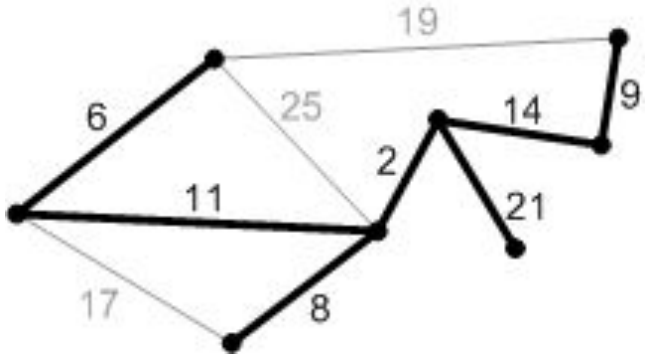


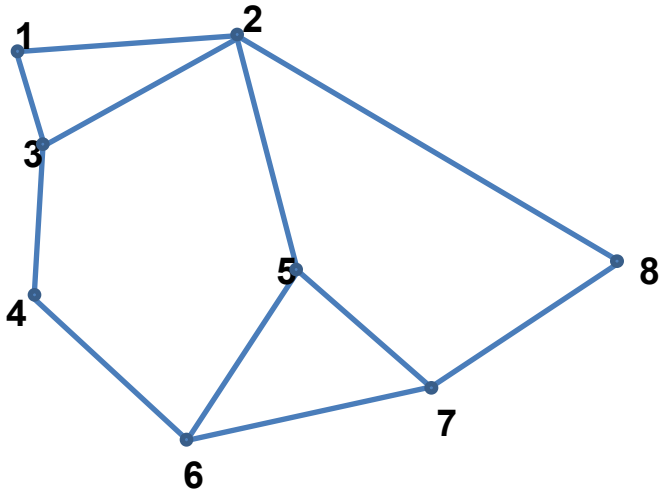
Рис 8.

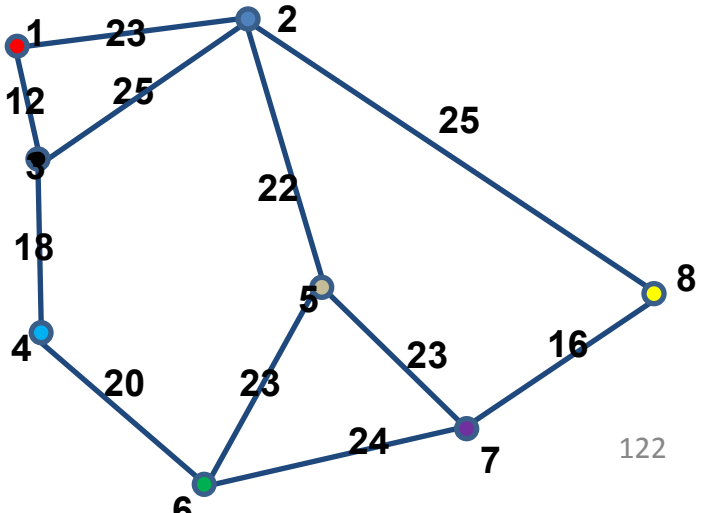
Минимальное остовное дерево

Алгоритм Крускала

Алгоритм Крускала

- Первоначально из графа удаляются все ребра. Каждая вершина такого графа помещается в одноэлементное подмножество.
- Ребра сортируются по возрастанию весов.
- Ребра последовательно, по возрастанию их весов, включаются в остовное дерево.
- Алгоритм заканчивают работу, когда все вершины будут объединены в одно множество, при этом оставшиеся ребра не включаются в остовное дерево.





Матрица смежности

	1	2	3	4	5	6	7	8
1	0	23	12	10000	10000	10000	10000	10000
2	23	0	25	10000	22	10000	10000	35
3	12	25	0	18	10000	10000	10000	10000
4	10000	10000	18	0	10000	20	10000	10000
5	10000	22	10000	10000	0	23	14	10000
6	10000	10000	10000	20	23	0	24	10000
7	10000	10000	10000	10000	14	24	0	16
8	10000	35	10000	10000	10000	10000	16	0

Алгоритм Крускала. Шаг 1.

Раскрасить все вершины в разные цвета.

Для этого создадим массив C(8).

C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)
1	2	3	4	5	6	7	8

Алгоритм Крускала. Шаг 2.

В матрице смежности найти самое короткое неиспользованное ребро (минимального веса).

	1	2	3	4	5	6	7	8
1	0	23	**12**	10000	10000	10000	10000	10000
2	23	0	25	10000	22	10000	10000	35
3	12	25	0	18	10000	10000	10000	10000
4	10000	10000	18	0	10000	20	10000	10000
5	10000	22	10000	10000	0	23	14	10000
6	10000	10000	10000	20	23	0	24	10000
7	10000	10000	10000	10000	23	24	0	16
8	10000	35	10000	10000	10000	10000	16	0

Алгоритм Крускала. Шаг 3.

Проверяем, можно ли использовать это ребро (вершины должны быть разного цвета).

	1	2	3	4	5	6	7	8
1	0	23	**12**	10000	10000	10000	10000	10000
2	23	0	25	10000	22	10000	10000	35
3	12	25	0	18	10000	10000	10000	10000
4	10000	10000	18	0	10000	20	10000	10000
5	10000	22	10000	10000	0	23	14	10000
6	10000	10000	10000	20	23	0	24	10000
7	10000	10000	10000	10000	23	24	0	16
8	10000	35	10000	10000	10000	10000	16	0

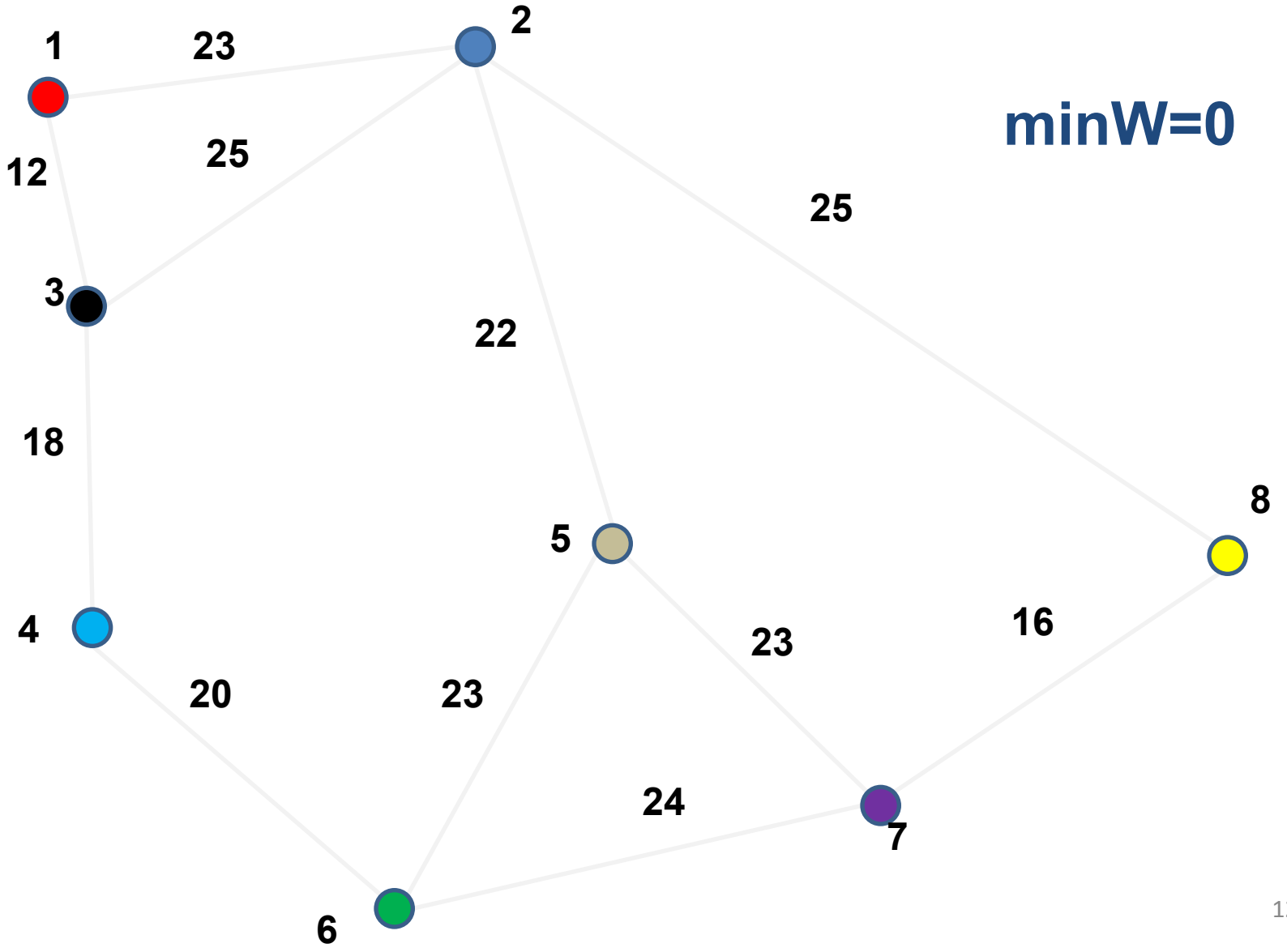
Алгоритм Крускала. Шаг 4.

Добавляем вес найденного ребра к весу остового дерева и перекрашиваем вершины в один цвет (меньший по номеру).

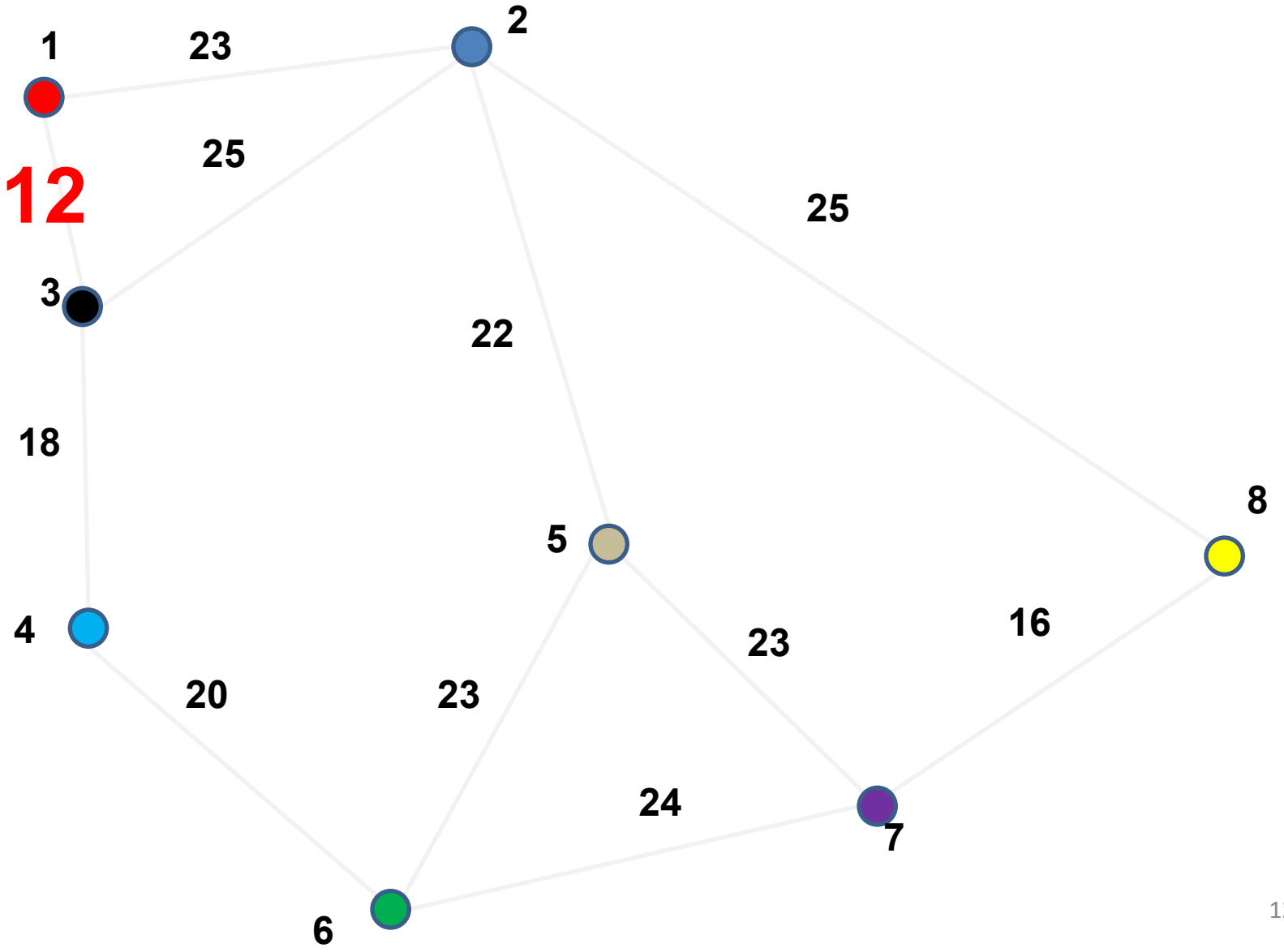
C(1)	C(2)	C(3)	C(4)	C(5)	C(6)	C(7)	C(8)
1	2	1	4	5	6	7	8

Вернуться к шагу №2.

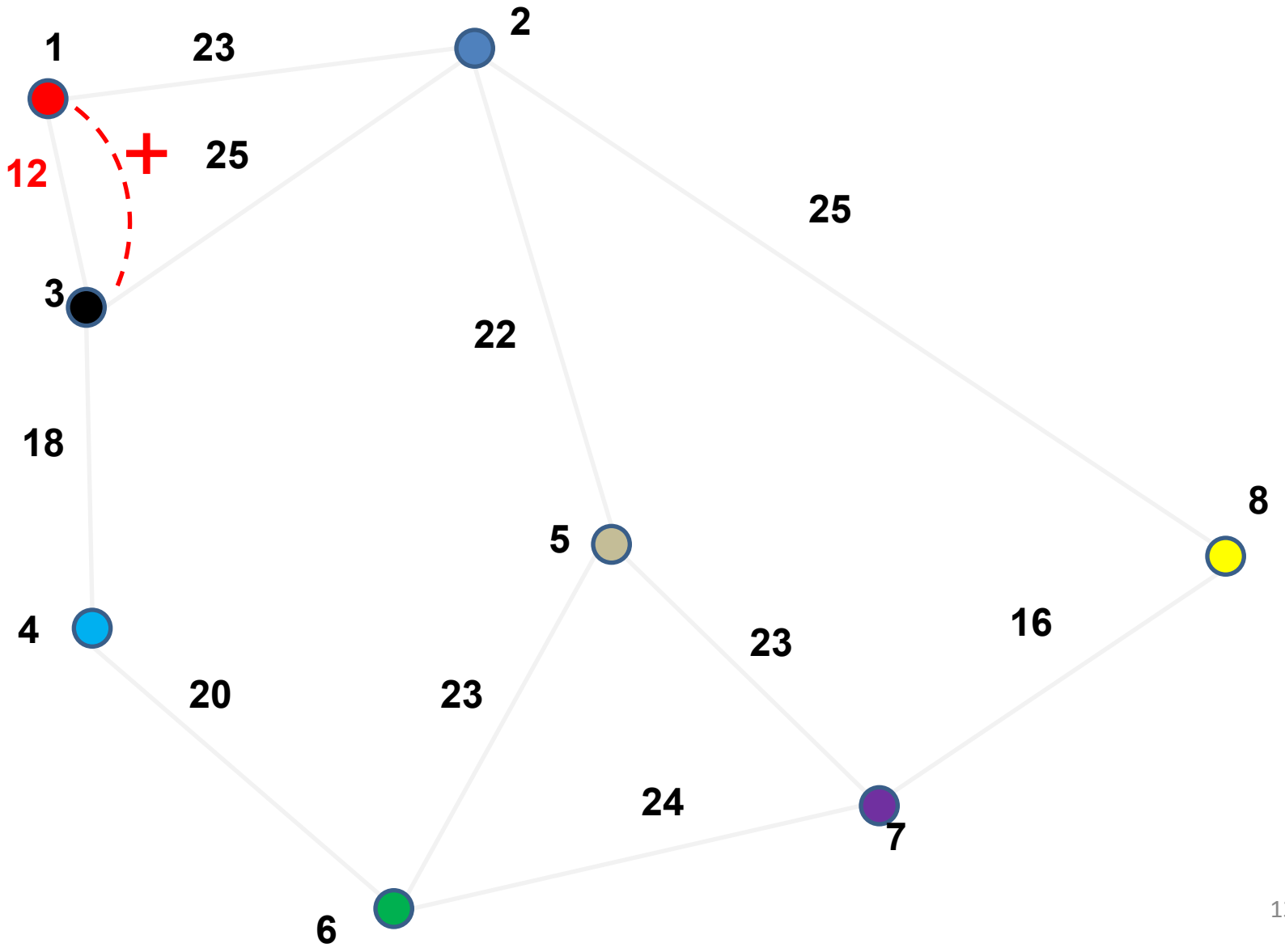
Алгоритм на графе. Шаг 1.



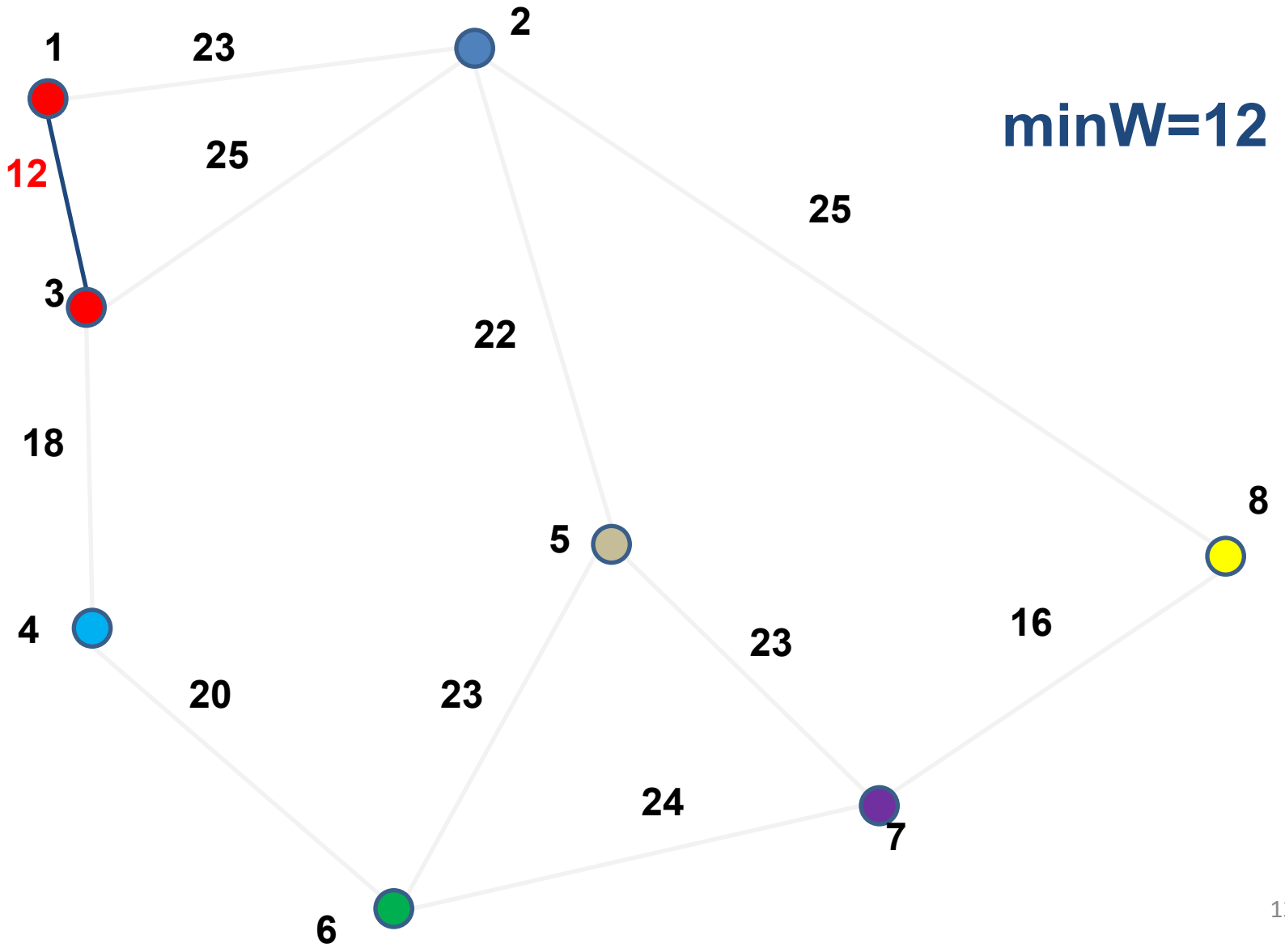
Шаг 2.



Шаг 3.

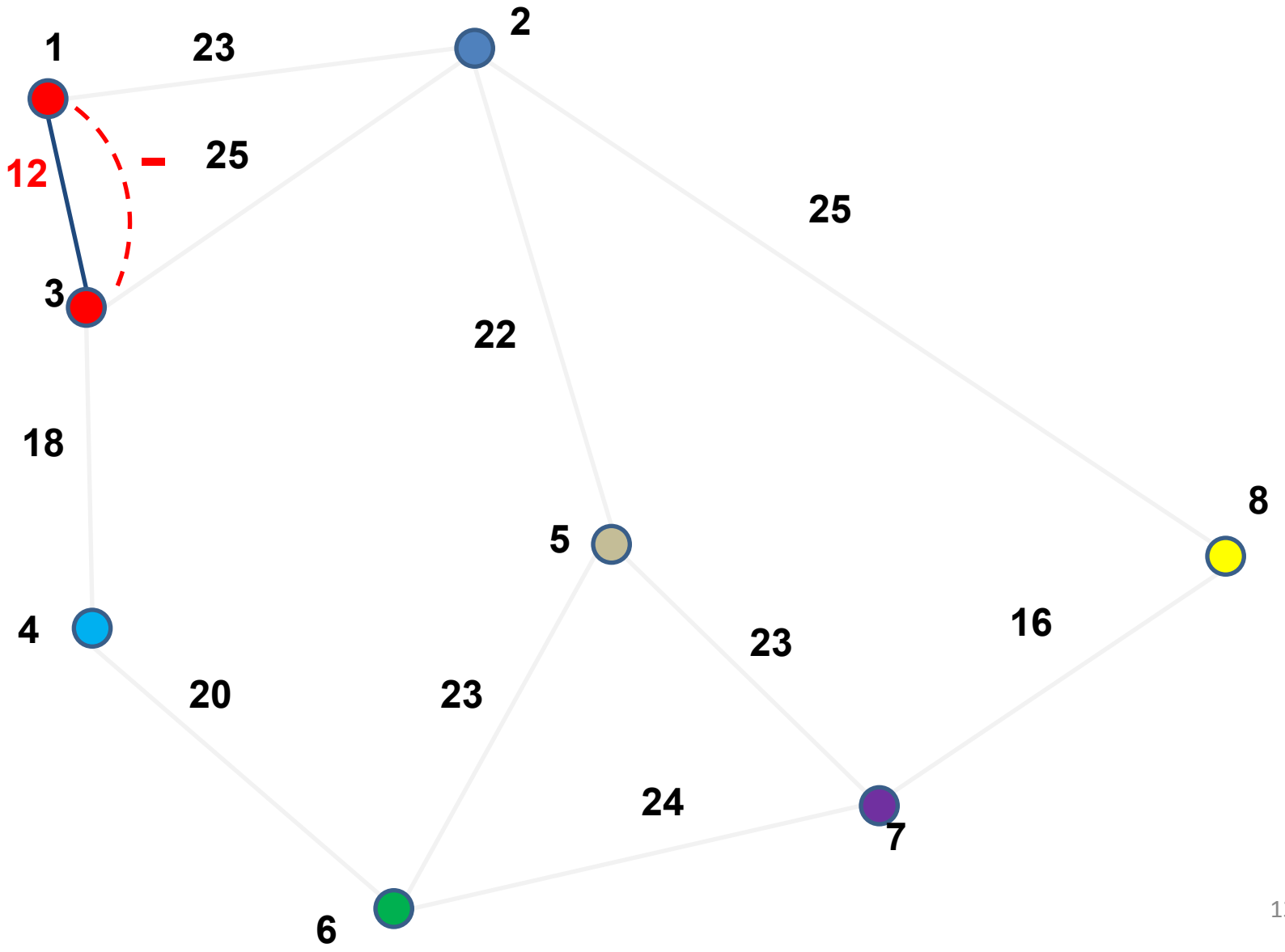


Шаг 4.

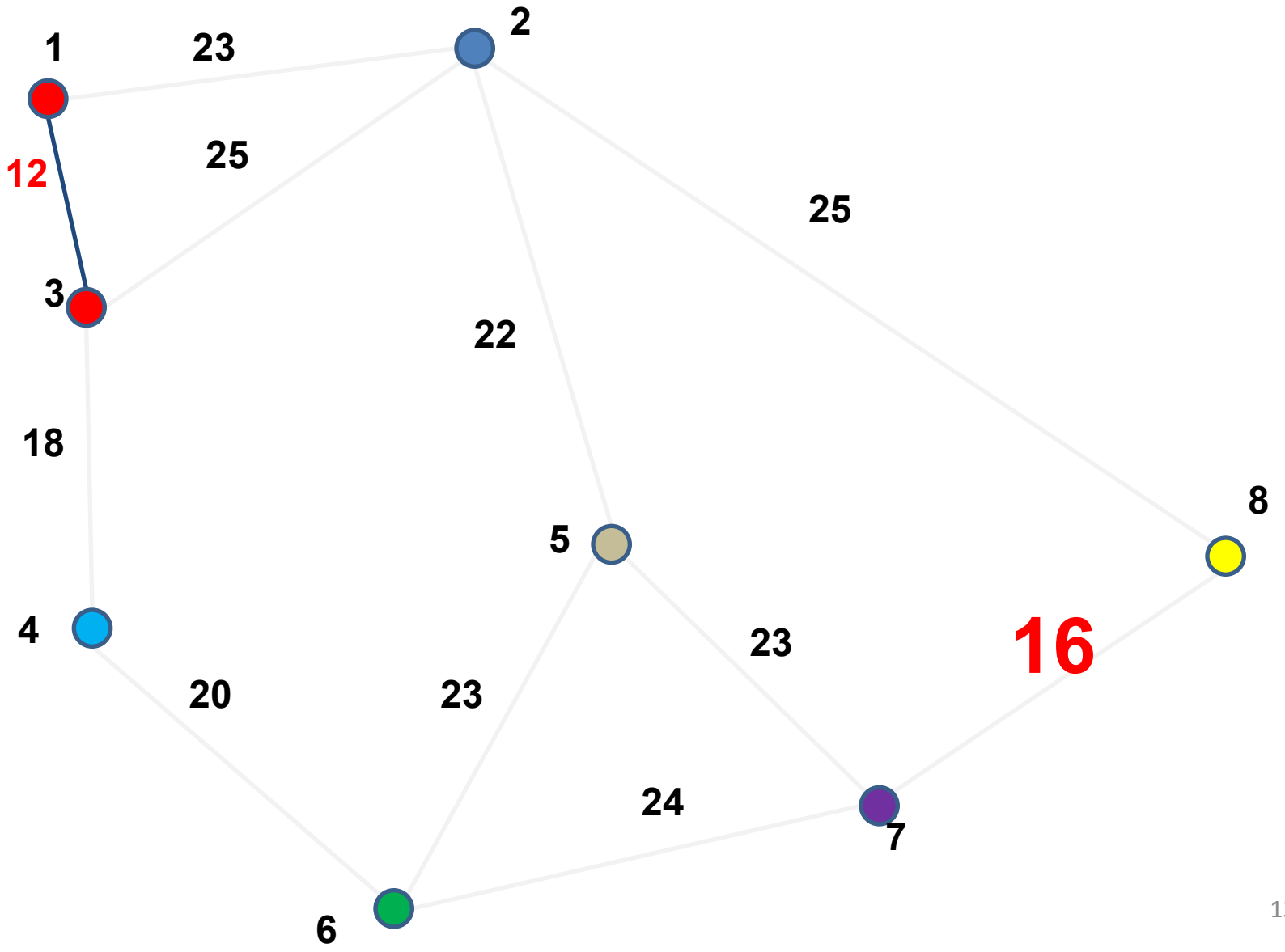


minW=12

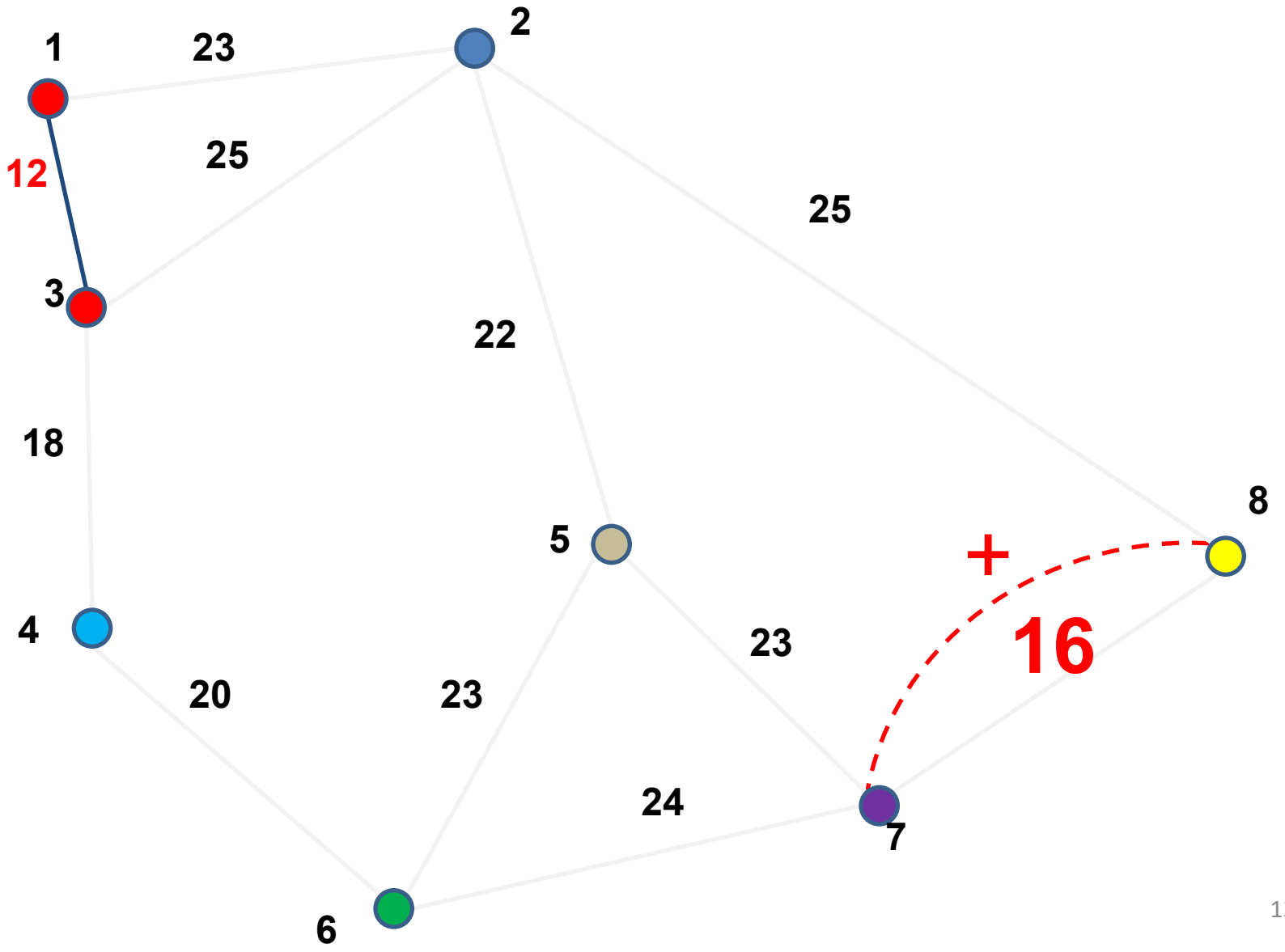
Шаг 2.



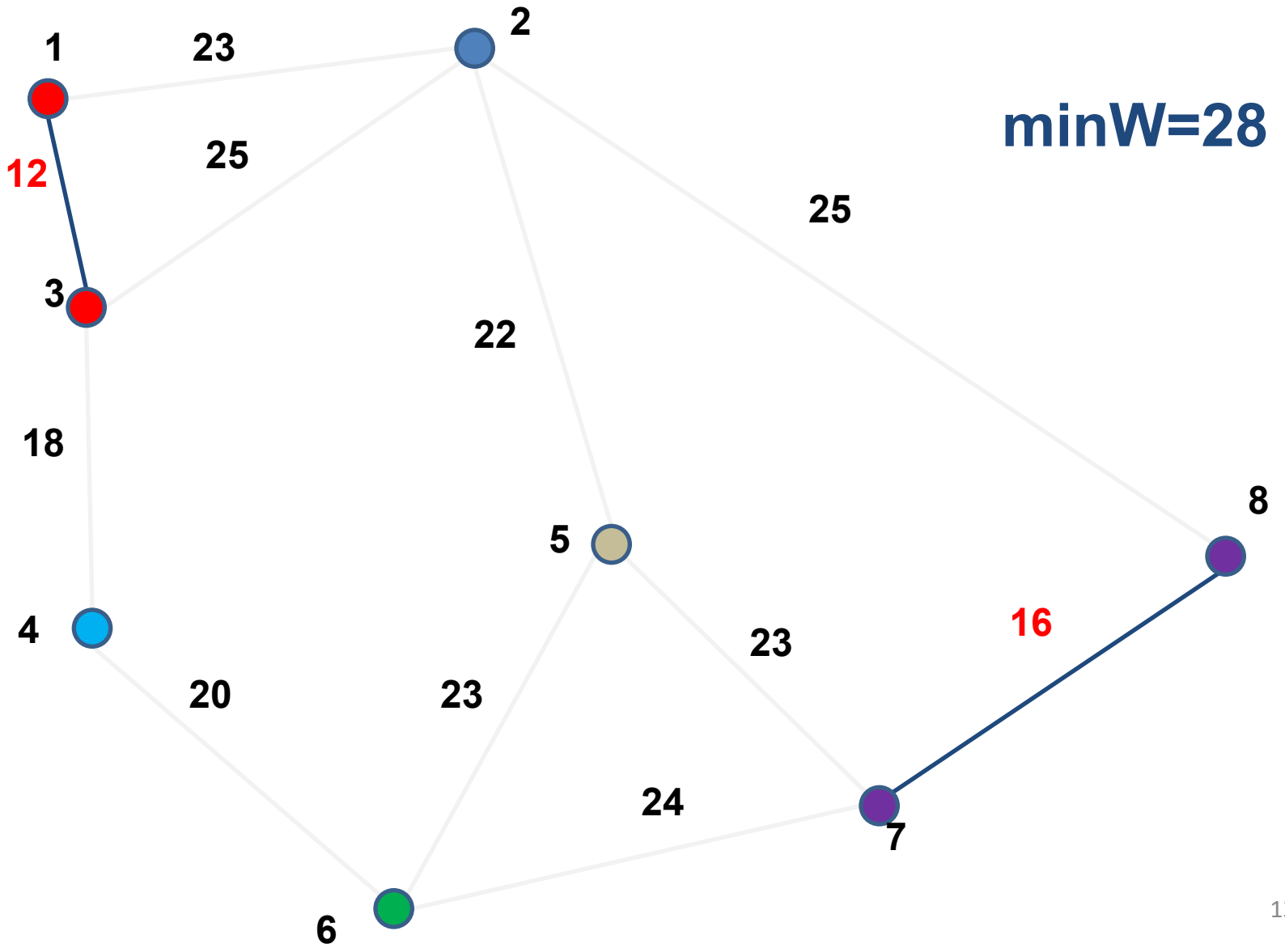
Шаг 2.



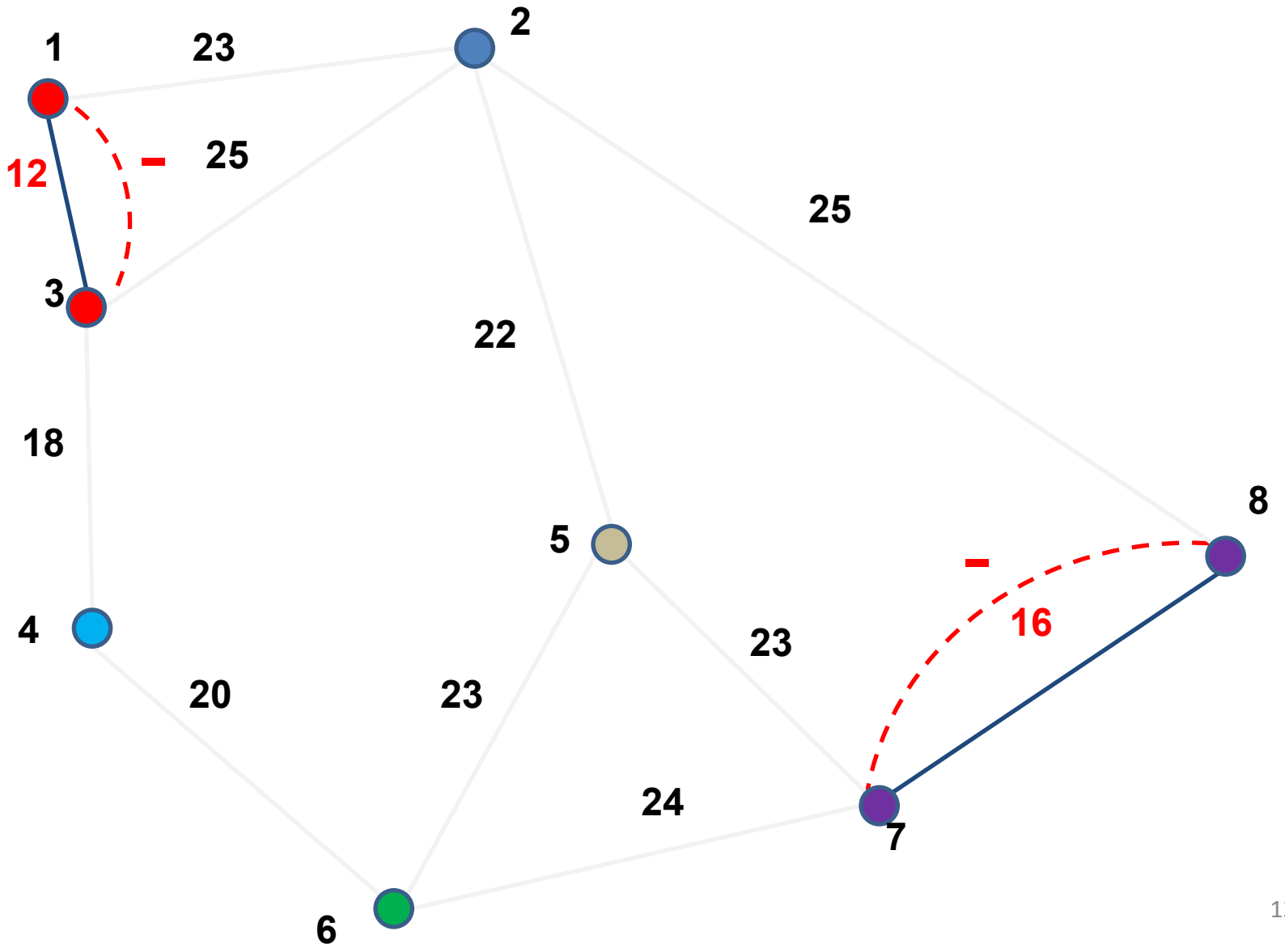
Шаг 3.



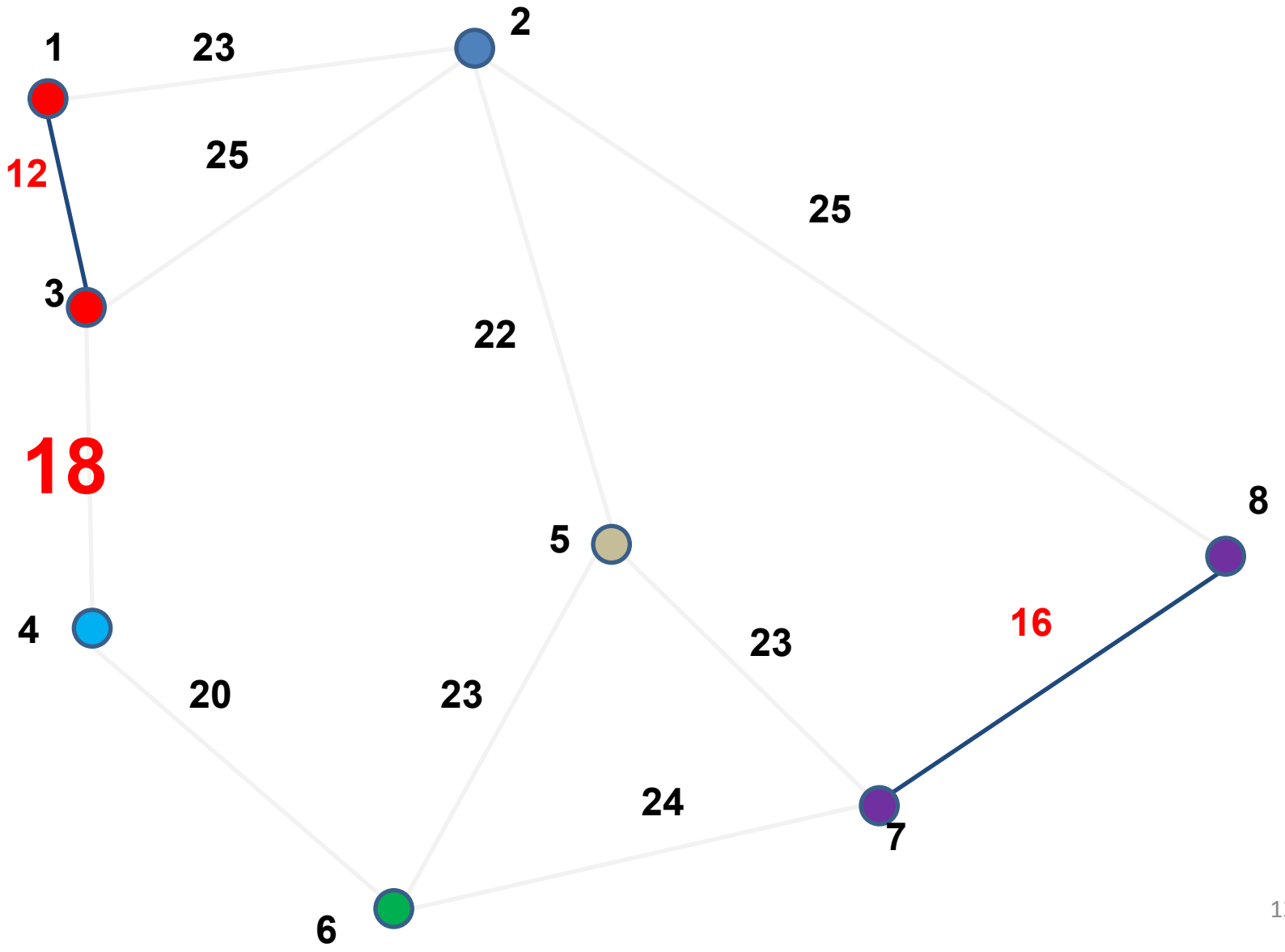
Шаг 4.



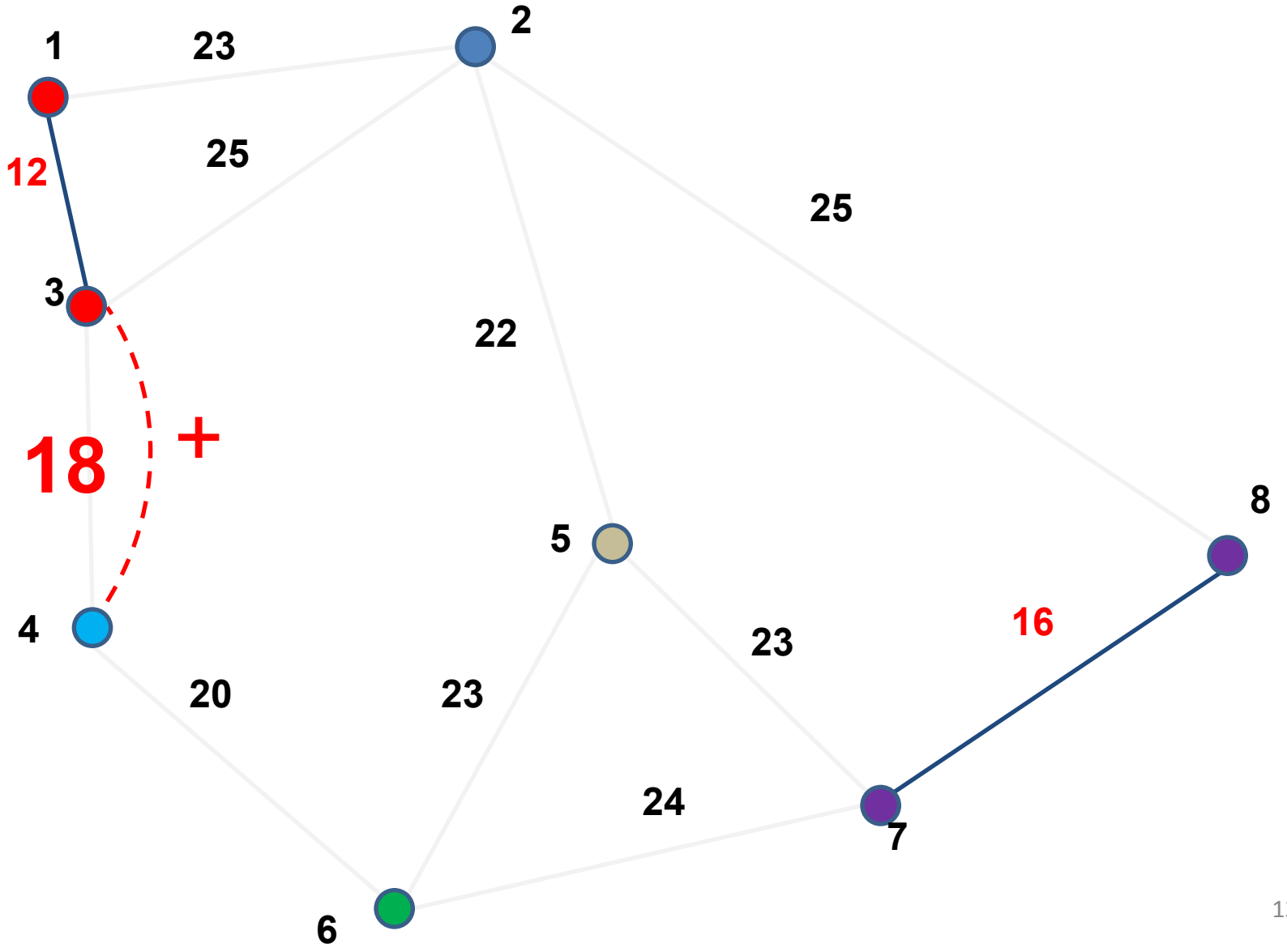
Шаг 2.



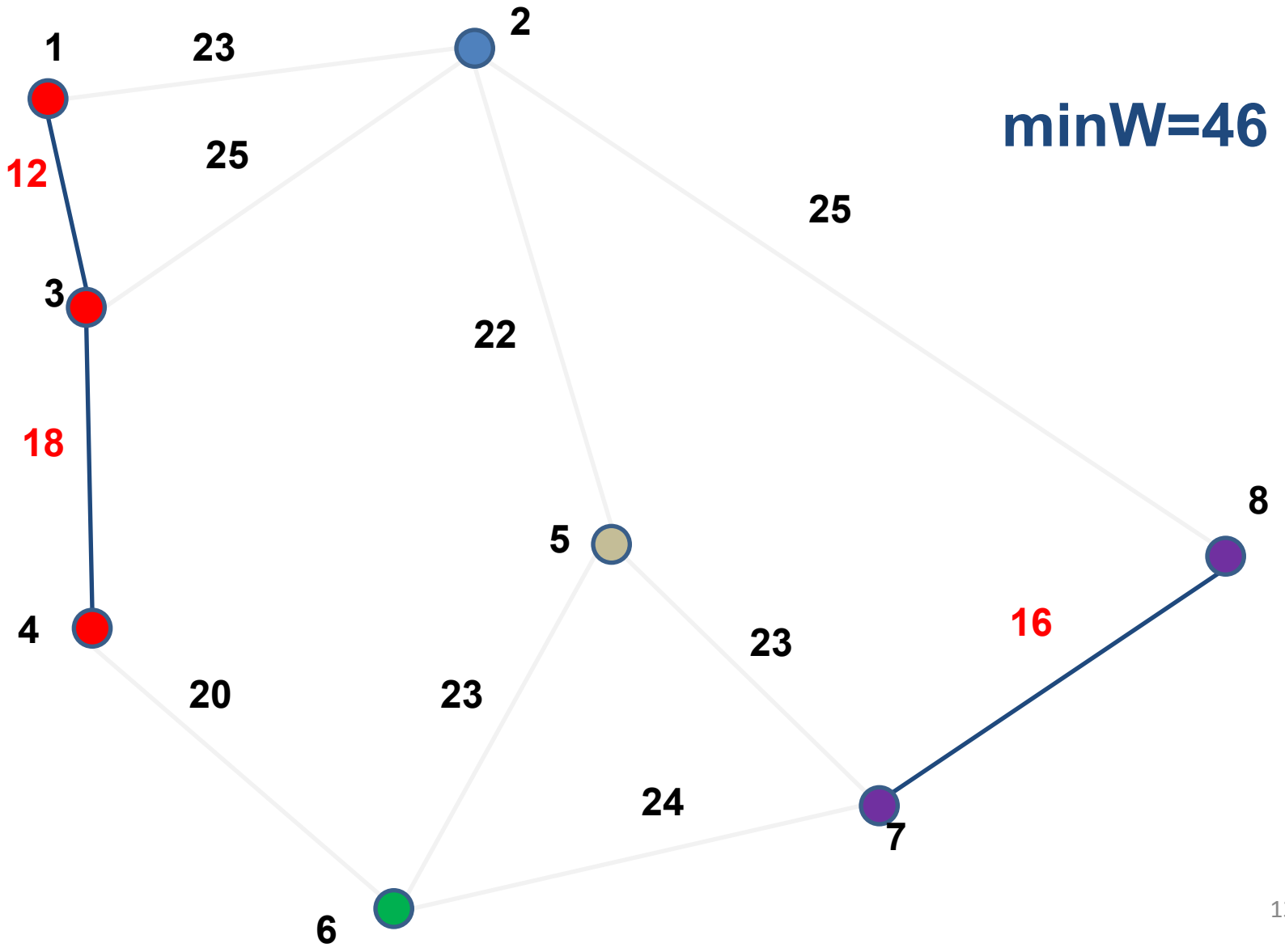
Шаг 2.



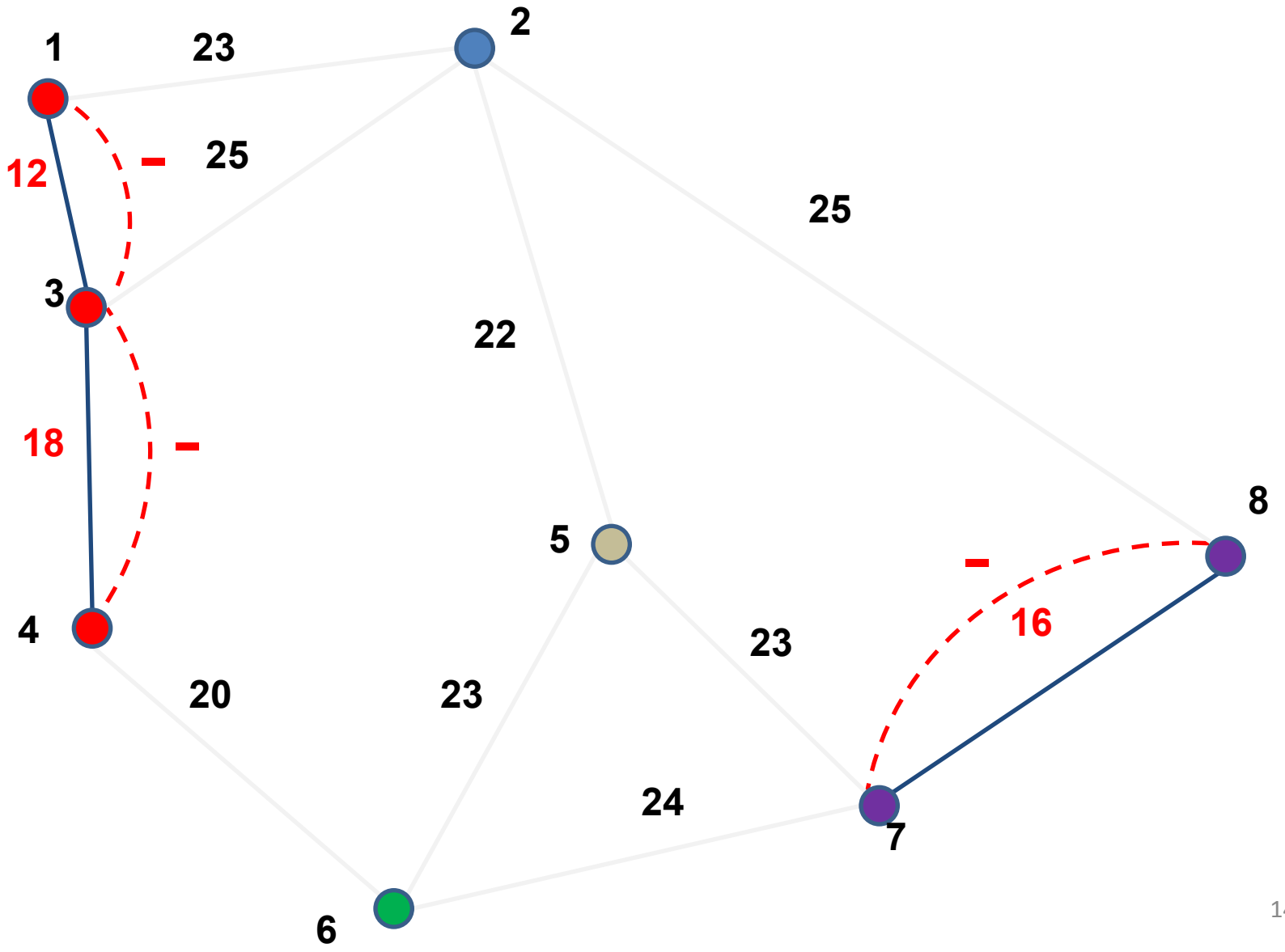
Шаг 3.



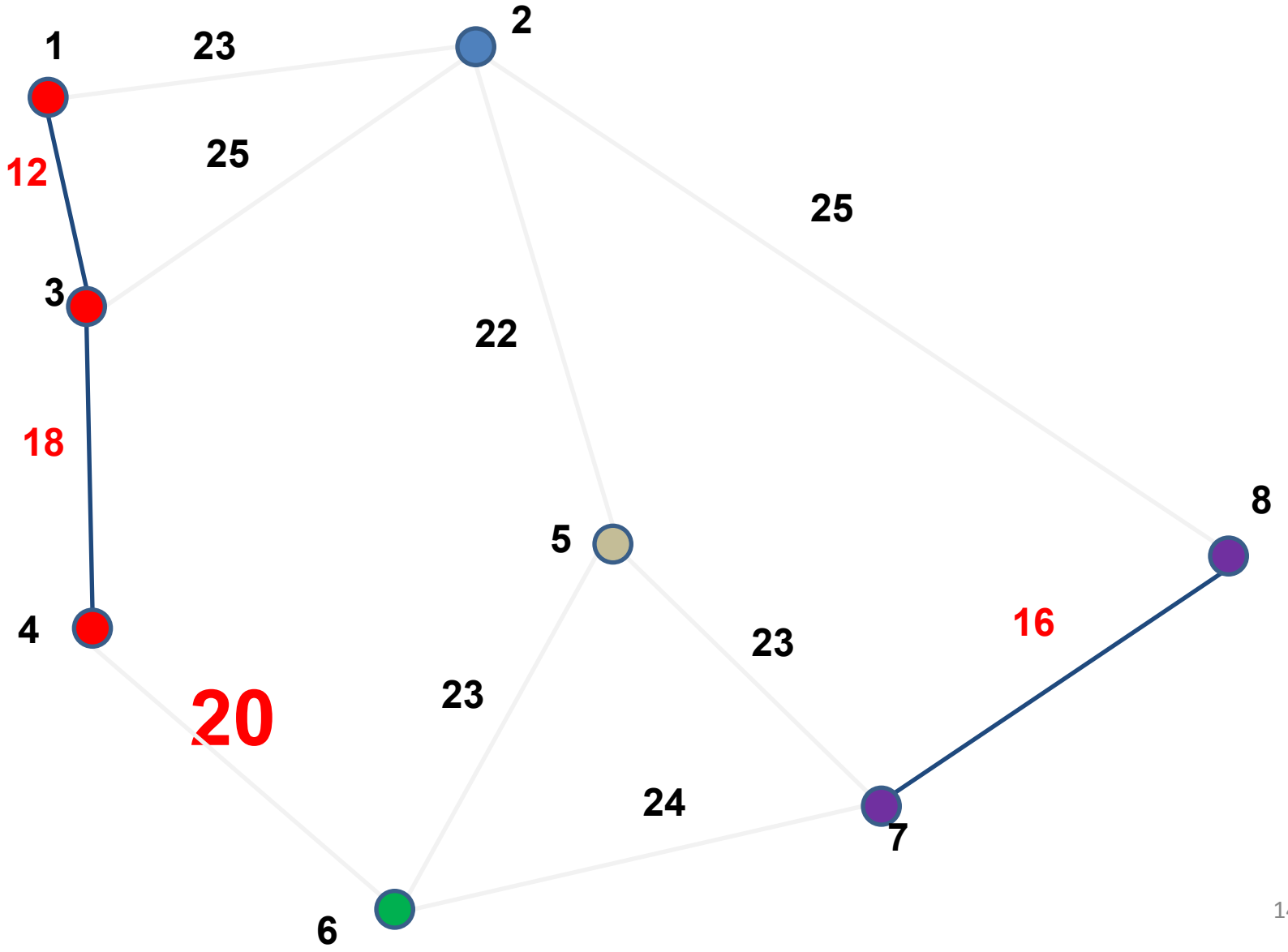
Шаг 4.



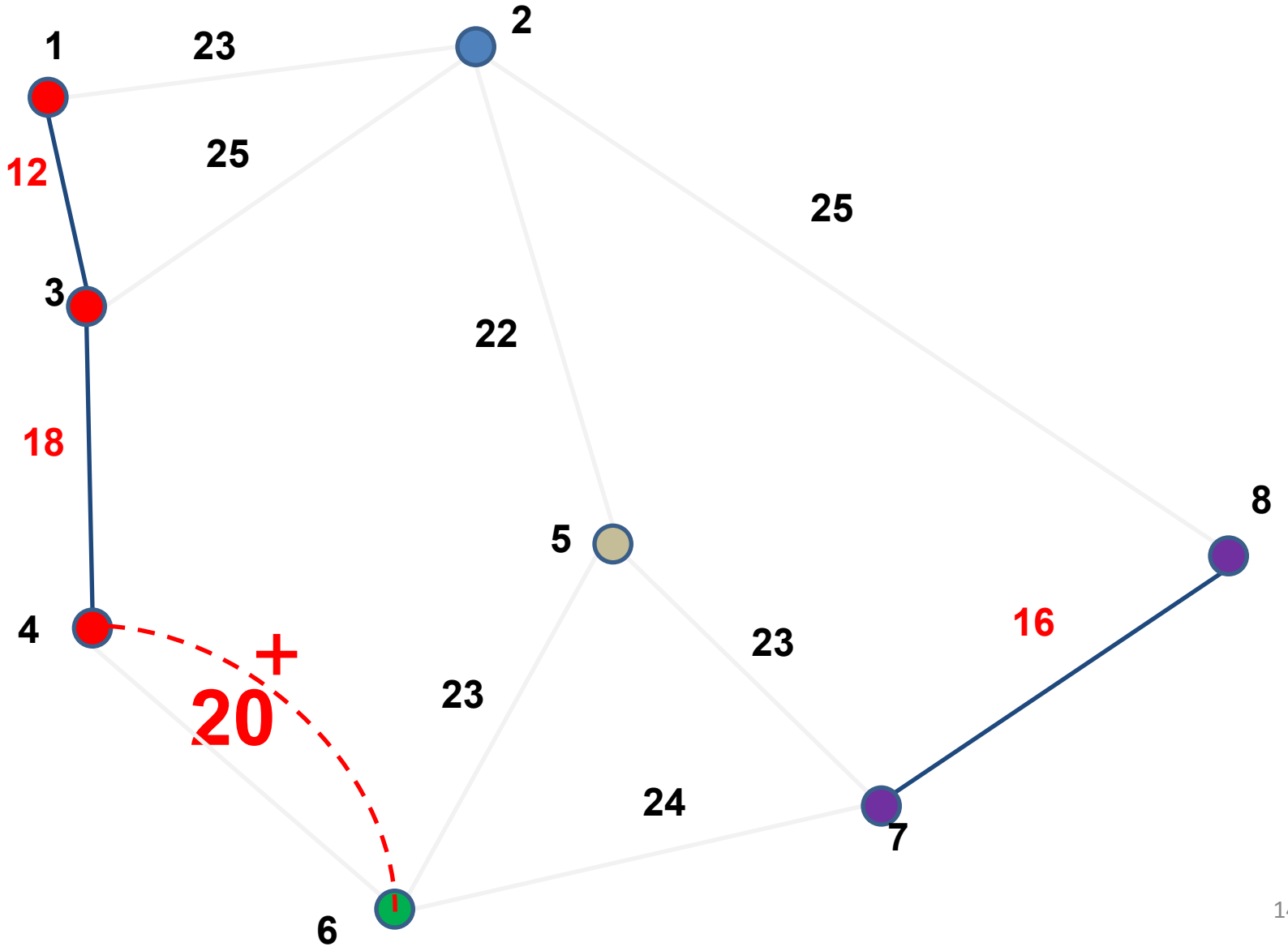
Шаг 2.



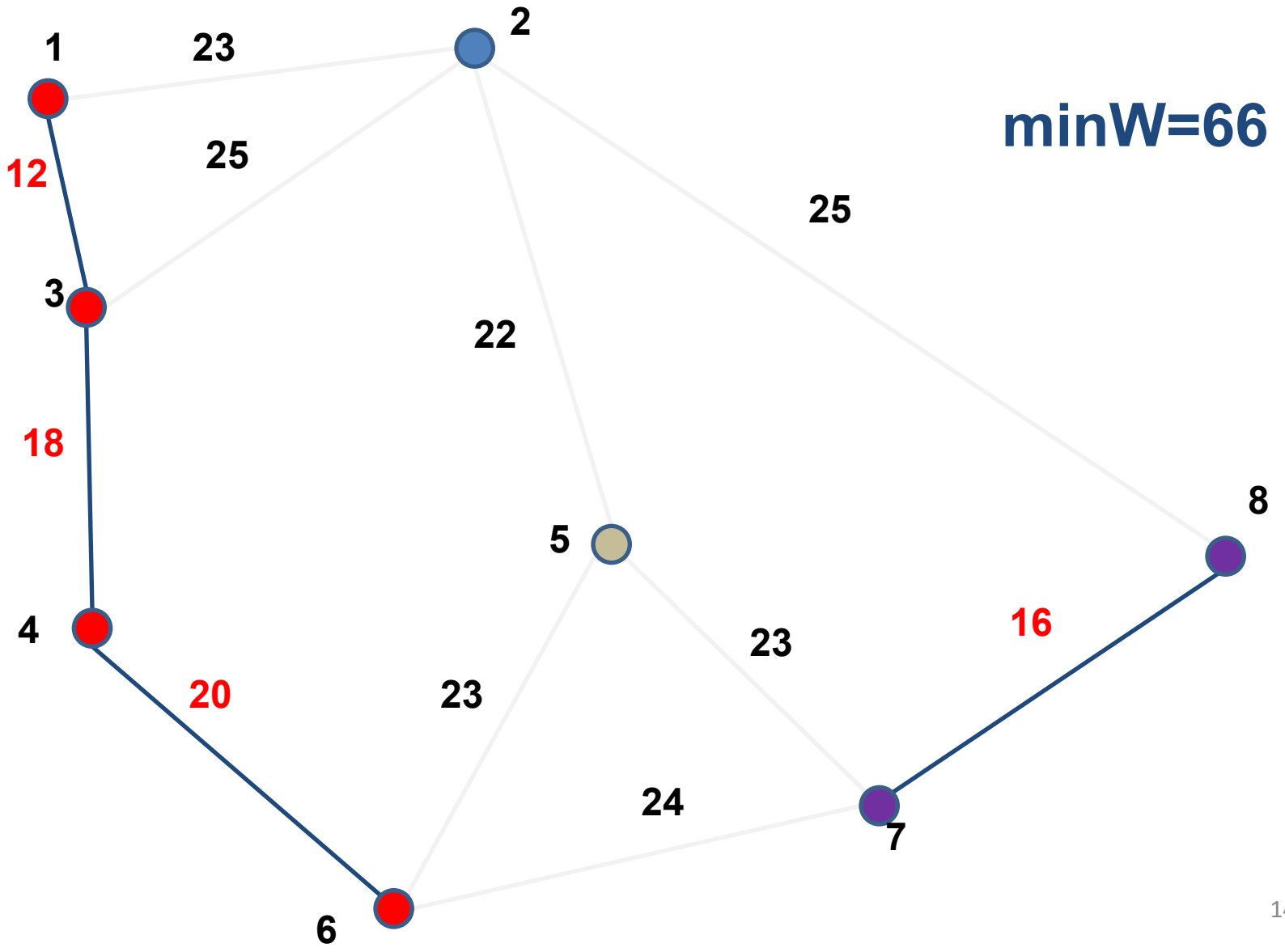
Шаг 2.



Шаг 3.

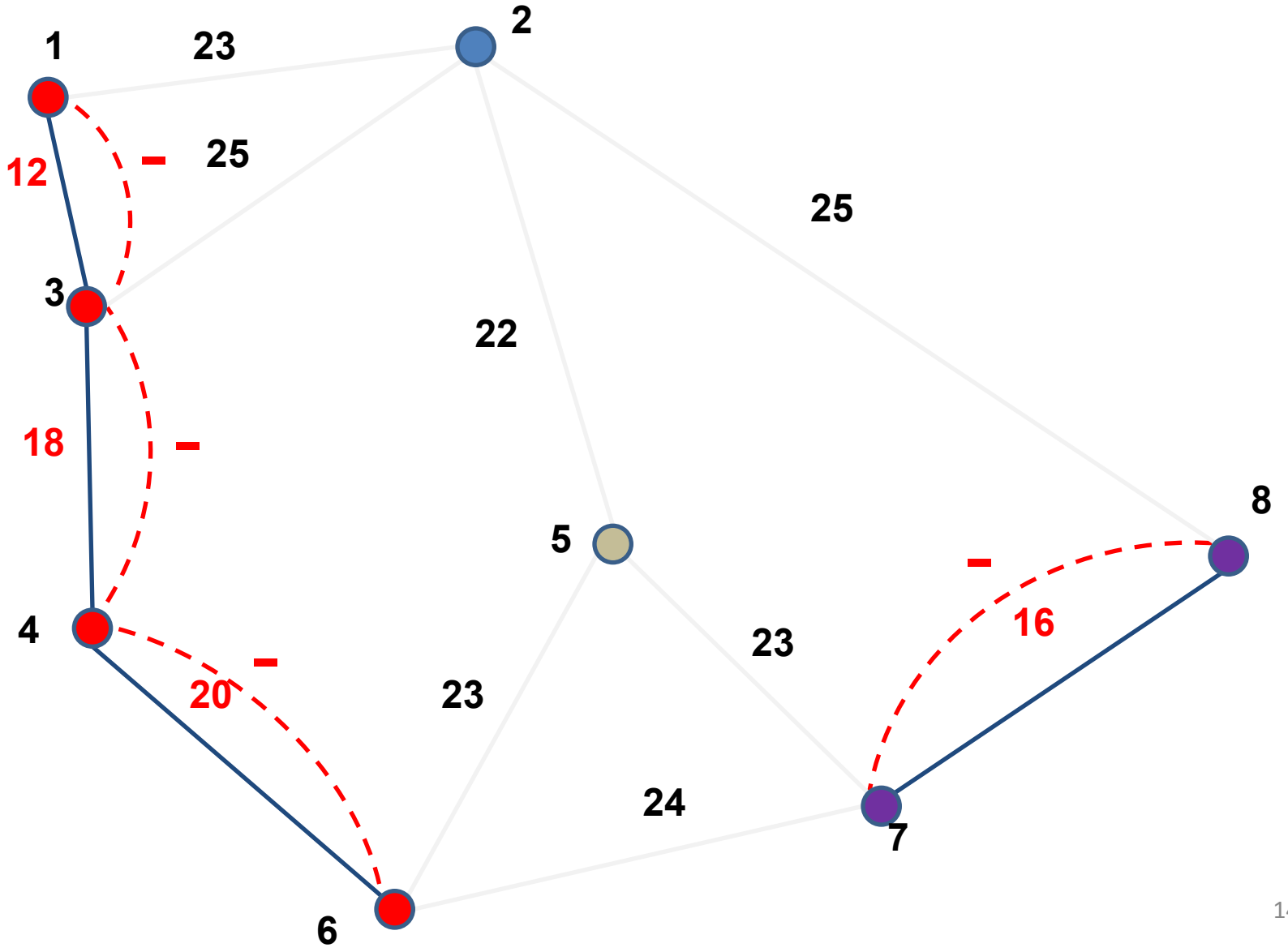


Шаг 4.

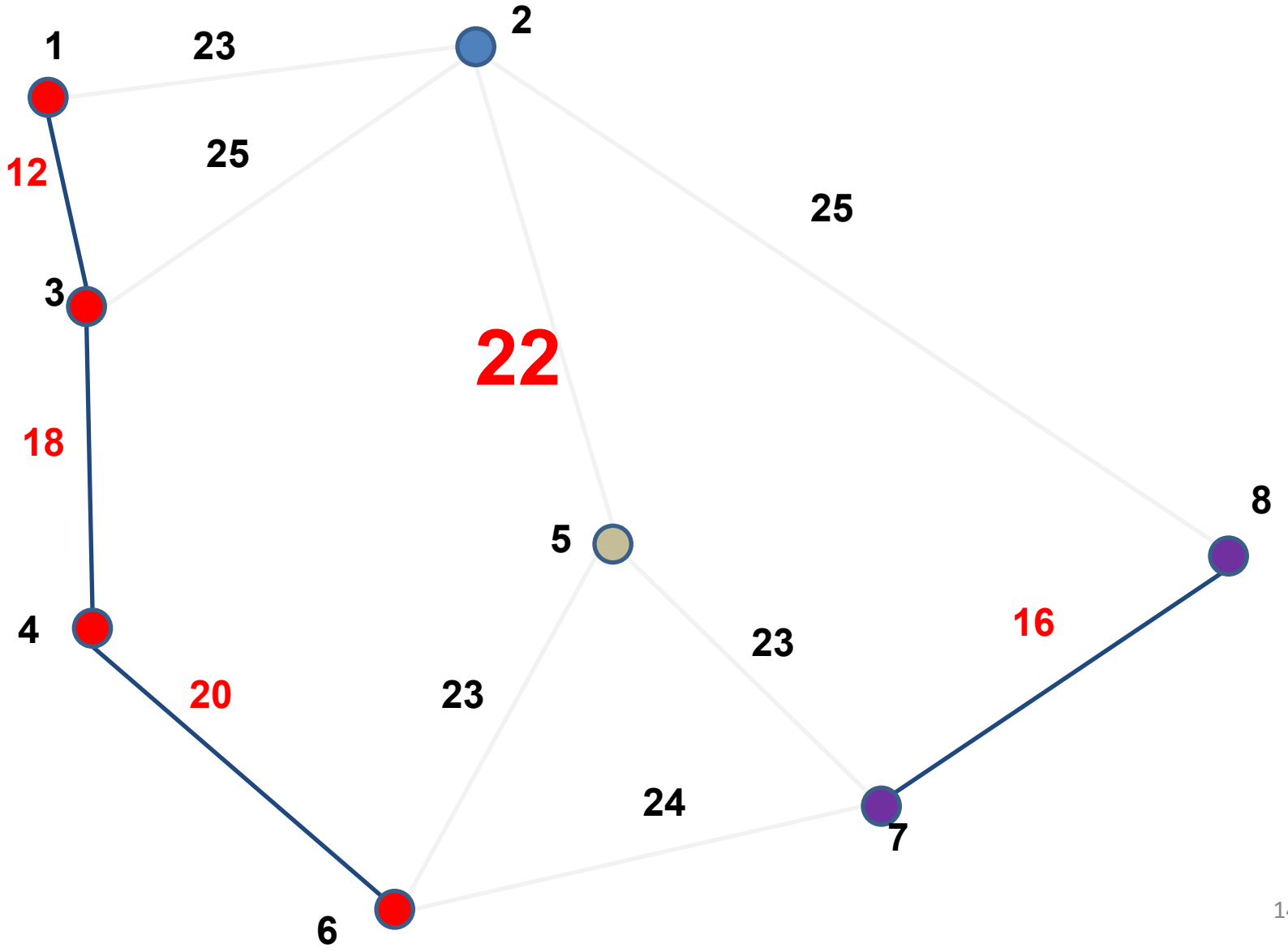


minW=66

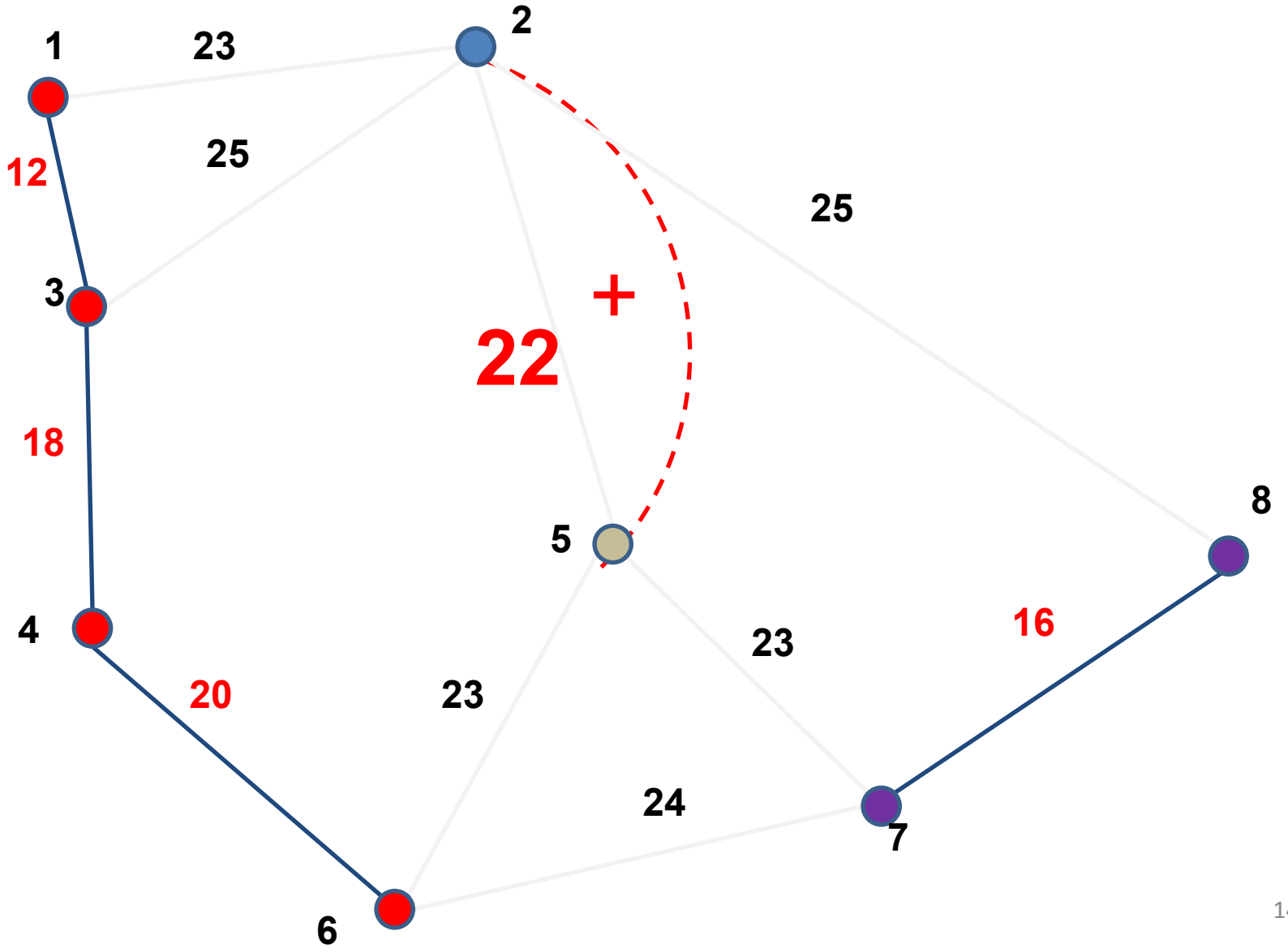
Шаг 2.



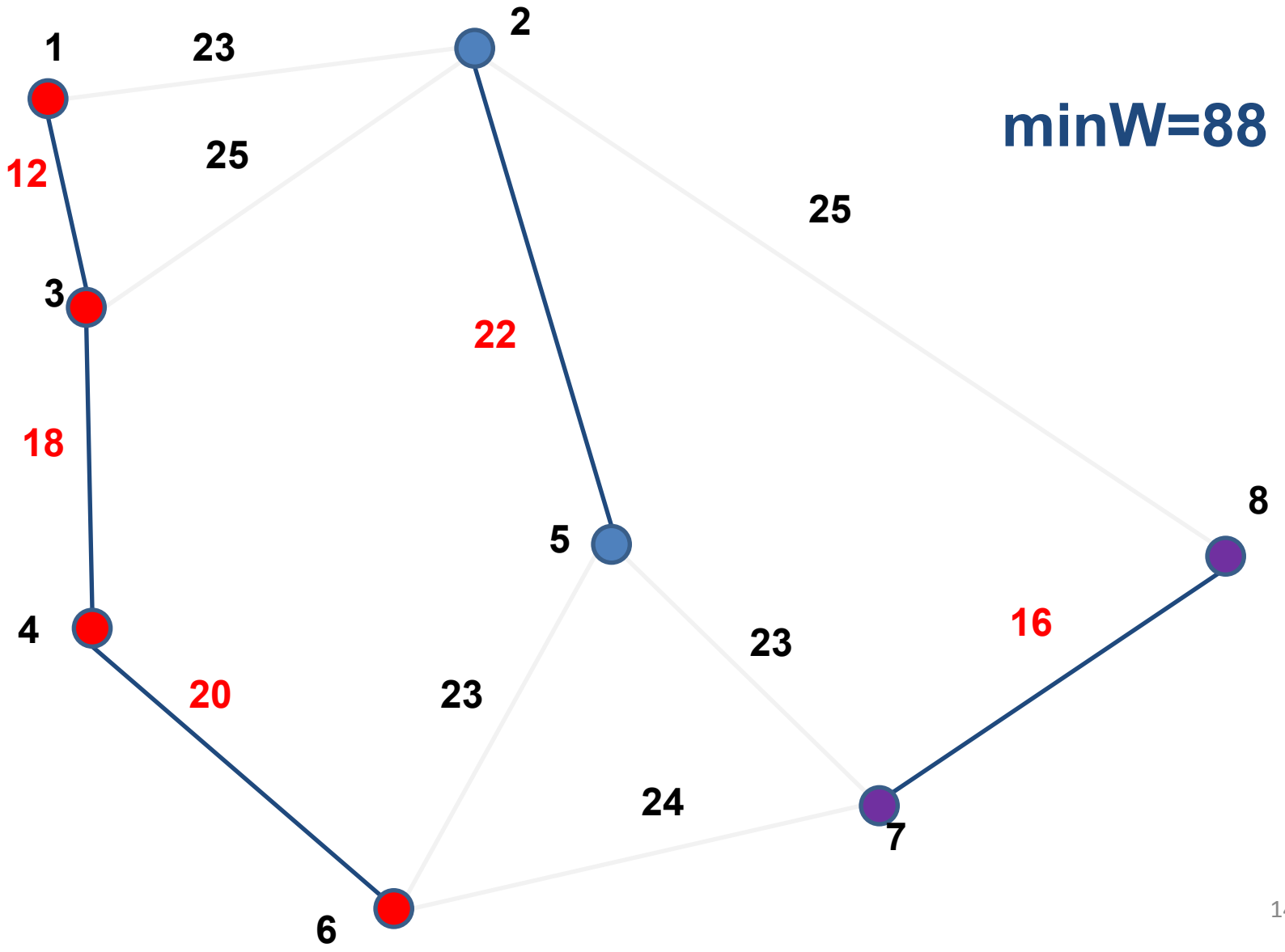
Шаг 2.



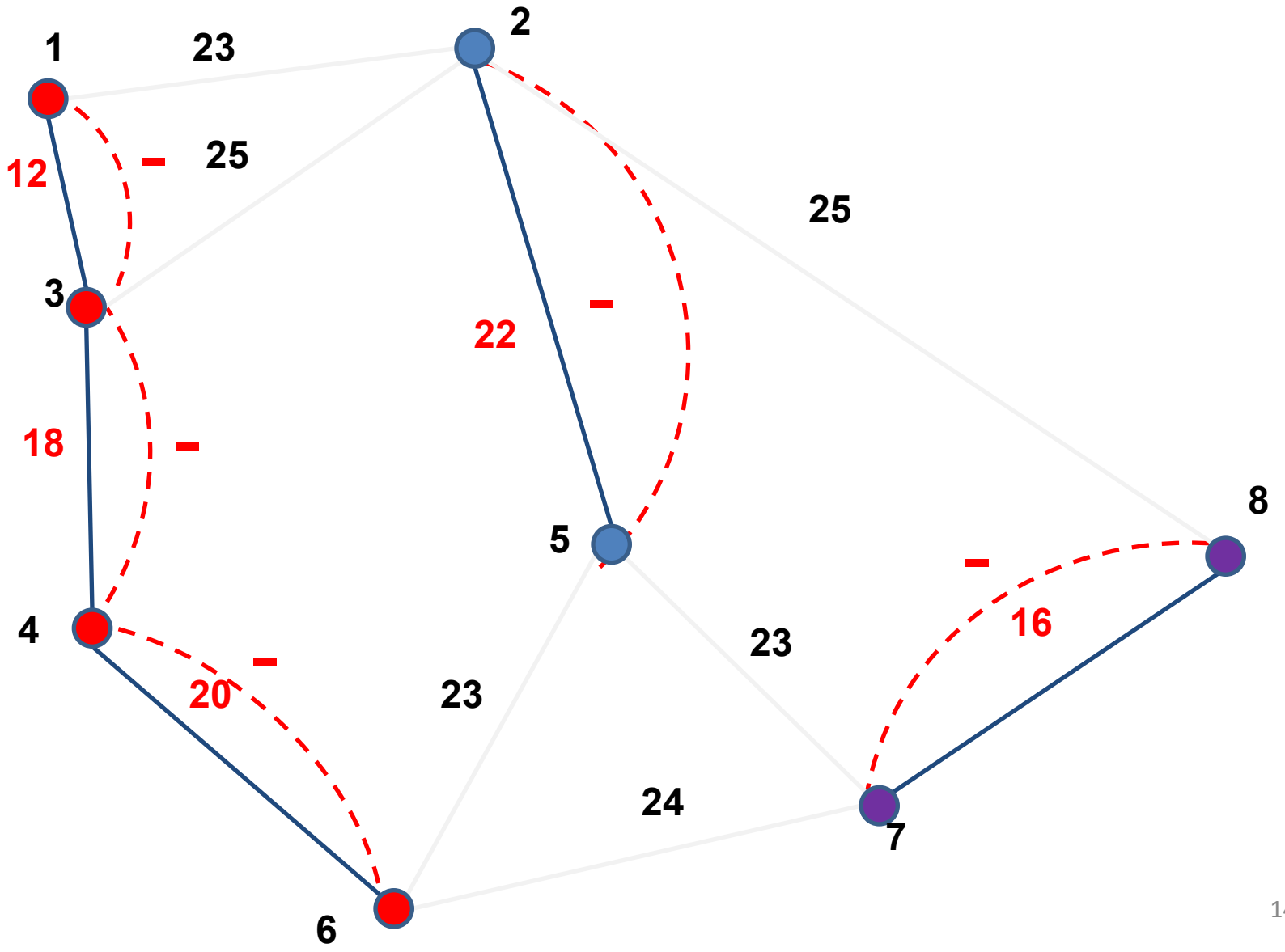
Шаг 3.



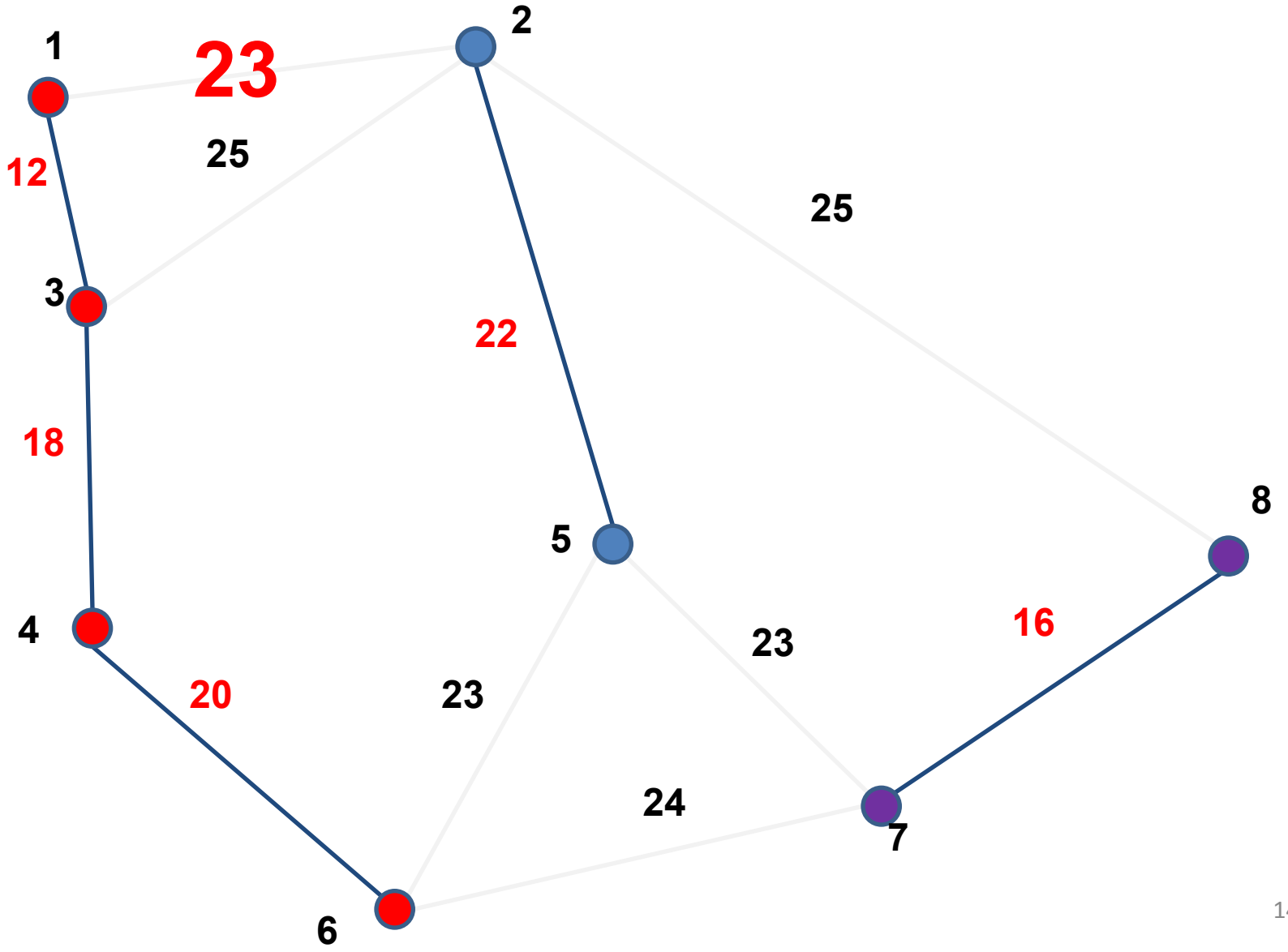
Шаг 4.



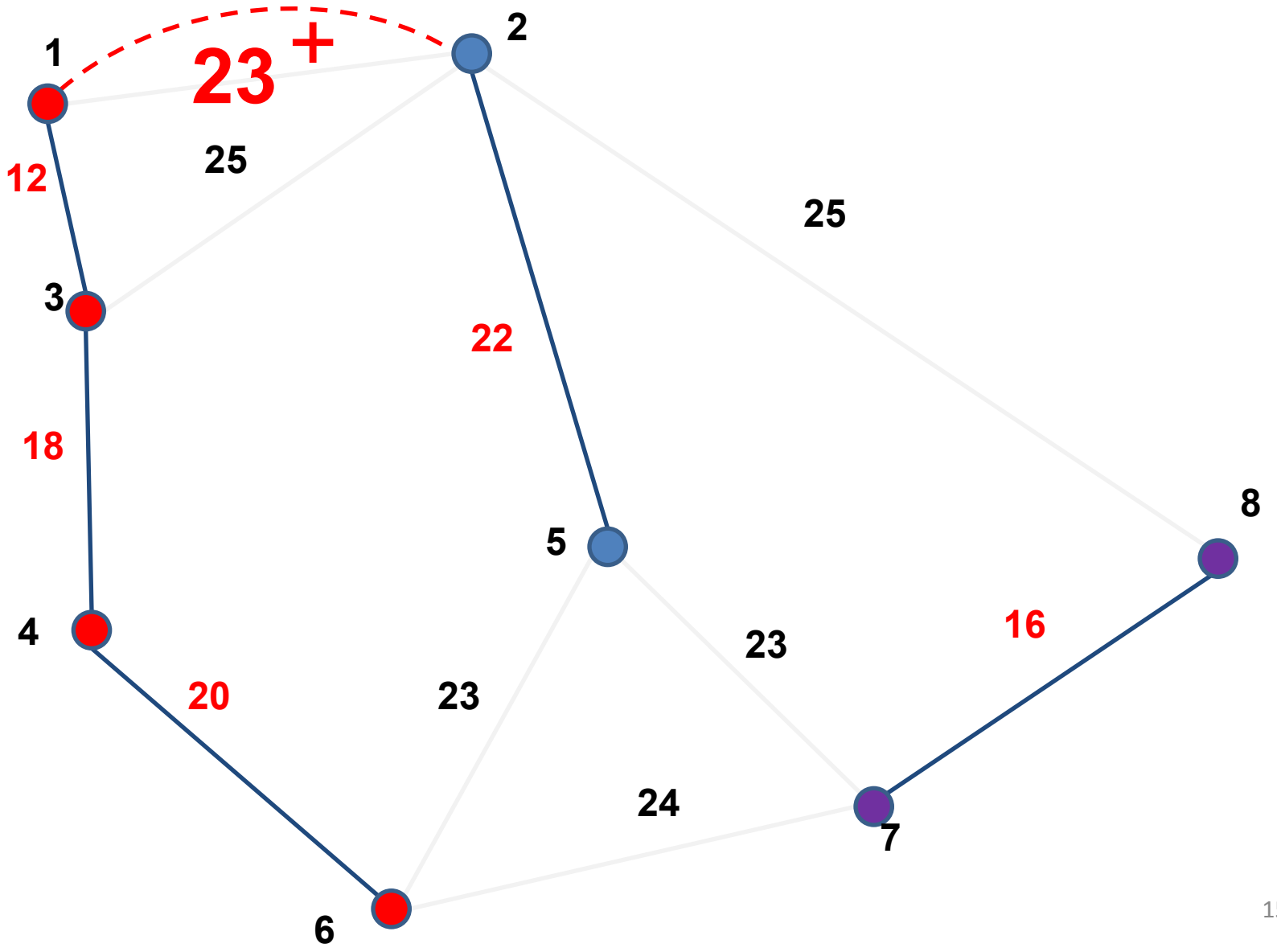
Шаг 2.



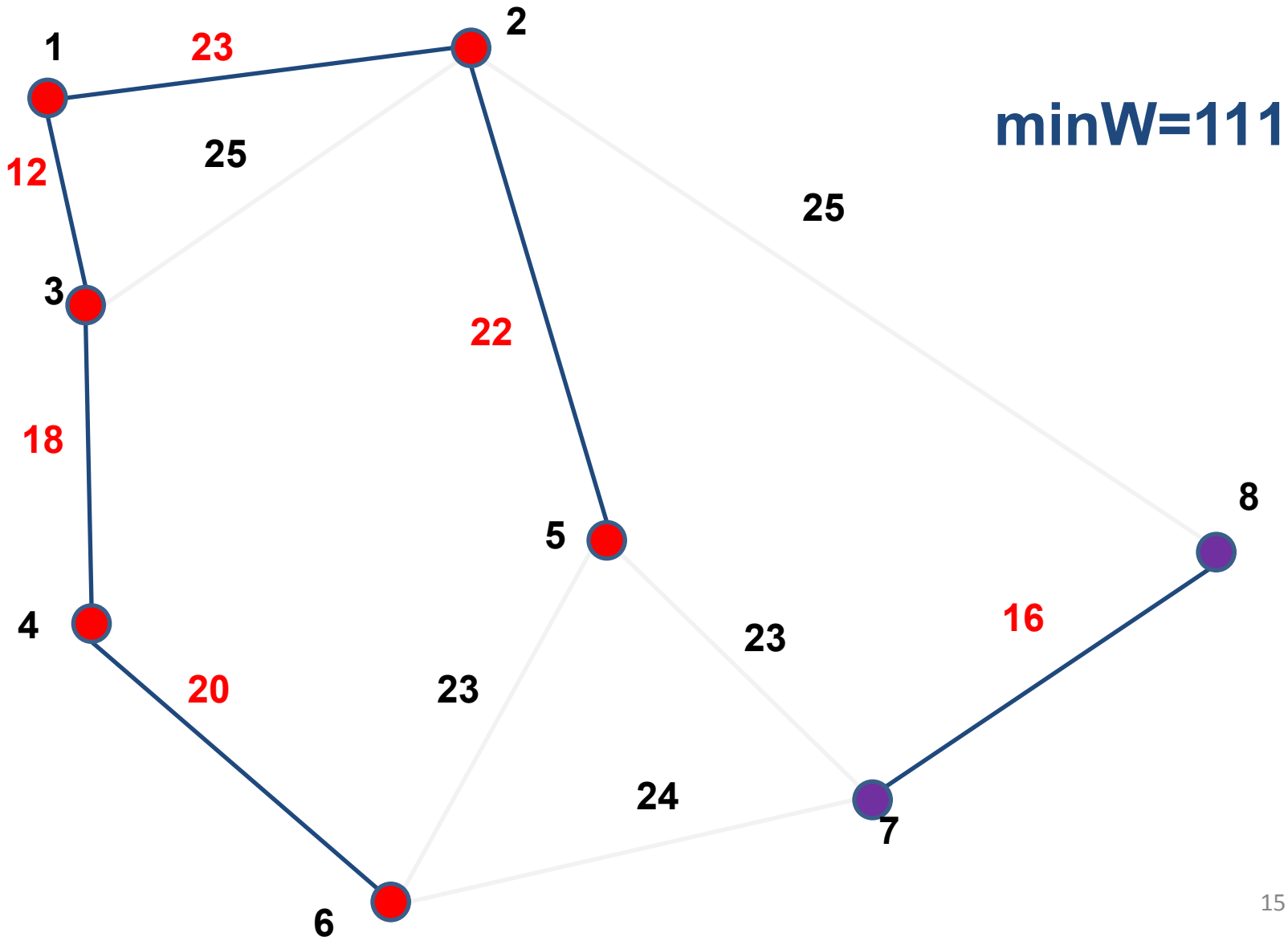
Шаг 2.



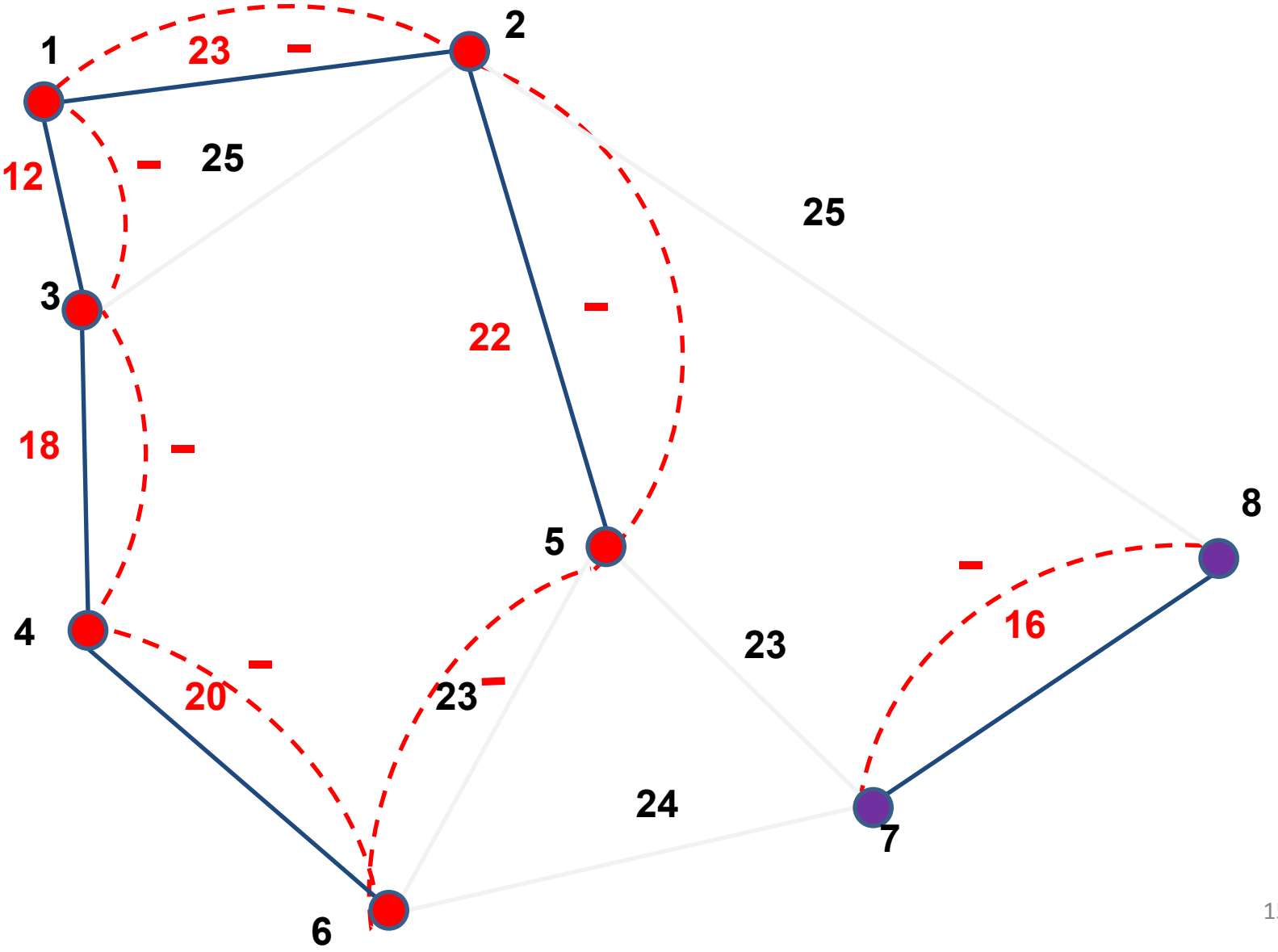
Шаг 3.



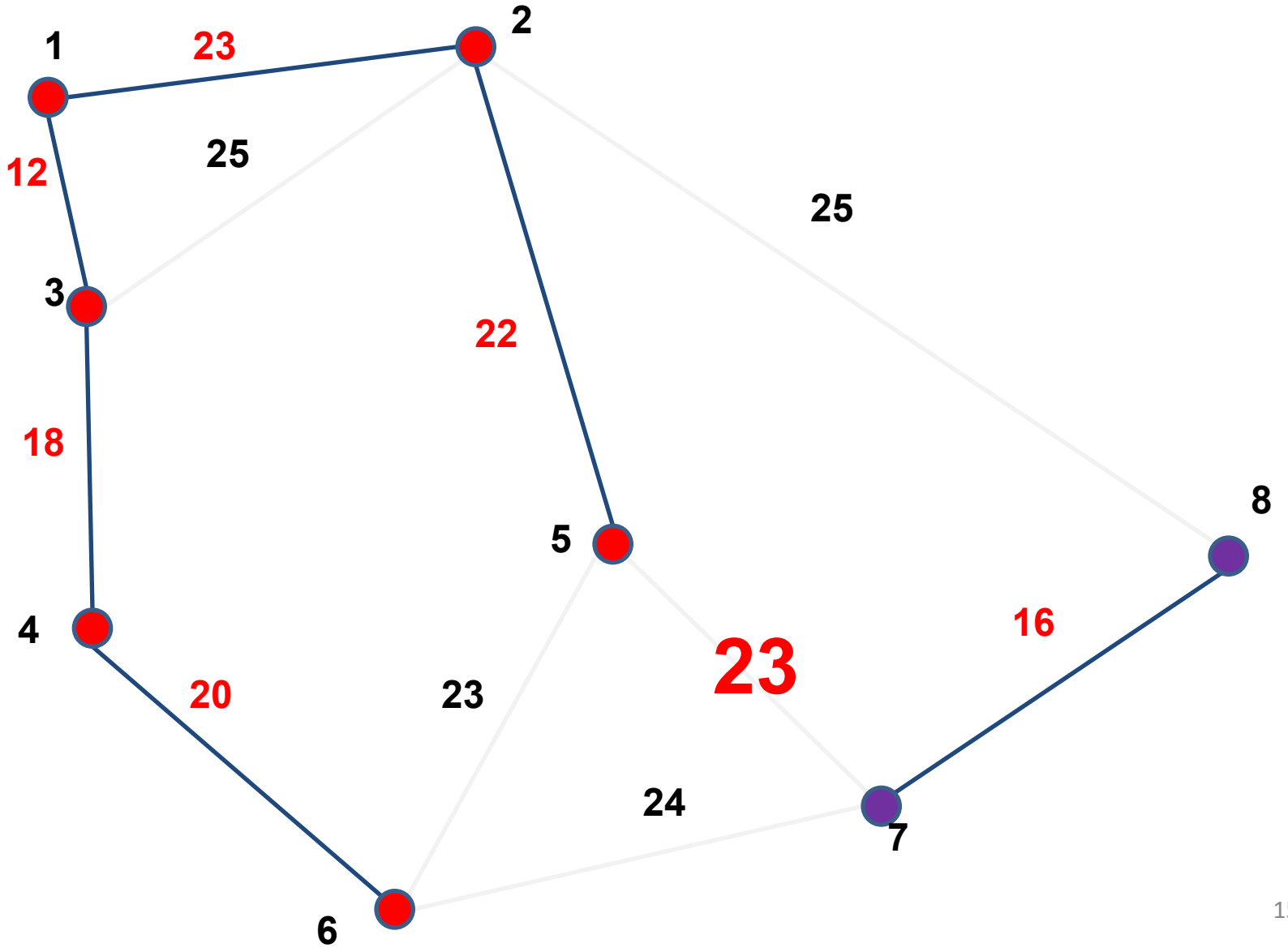
Шаг 4.



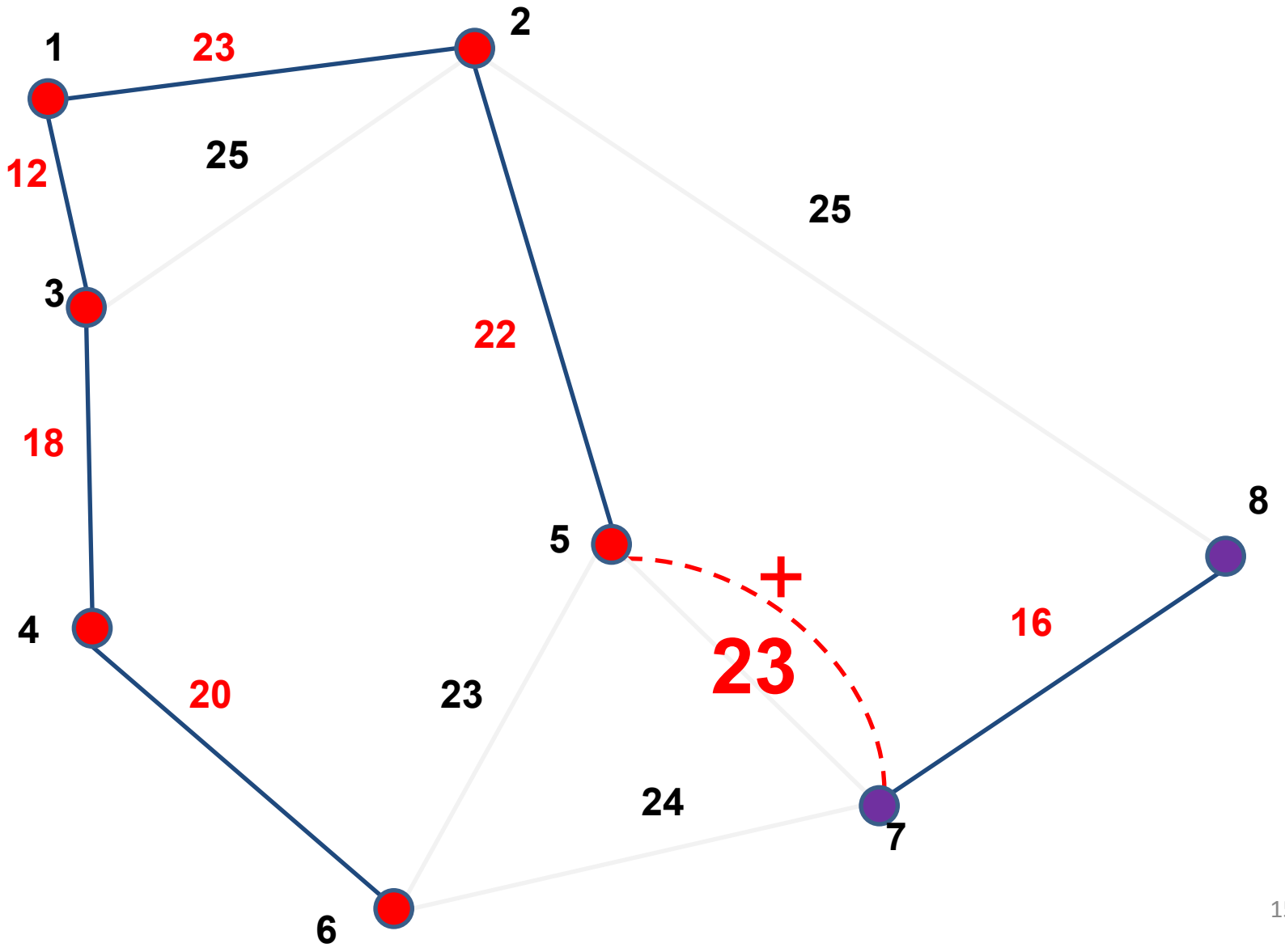
Шаг 2.



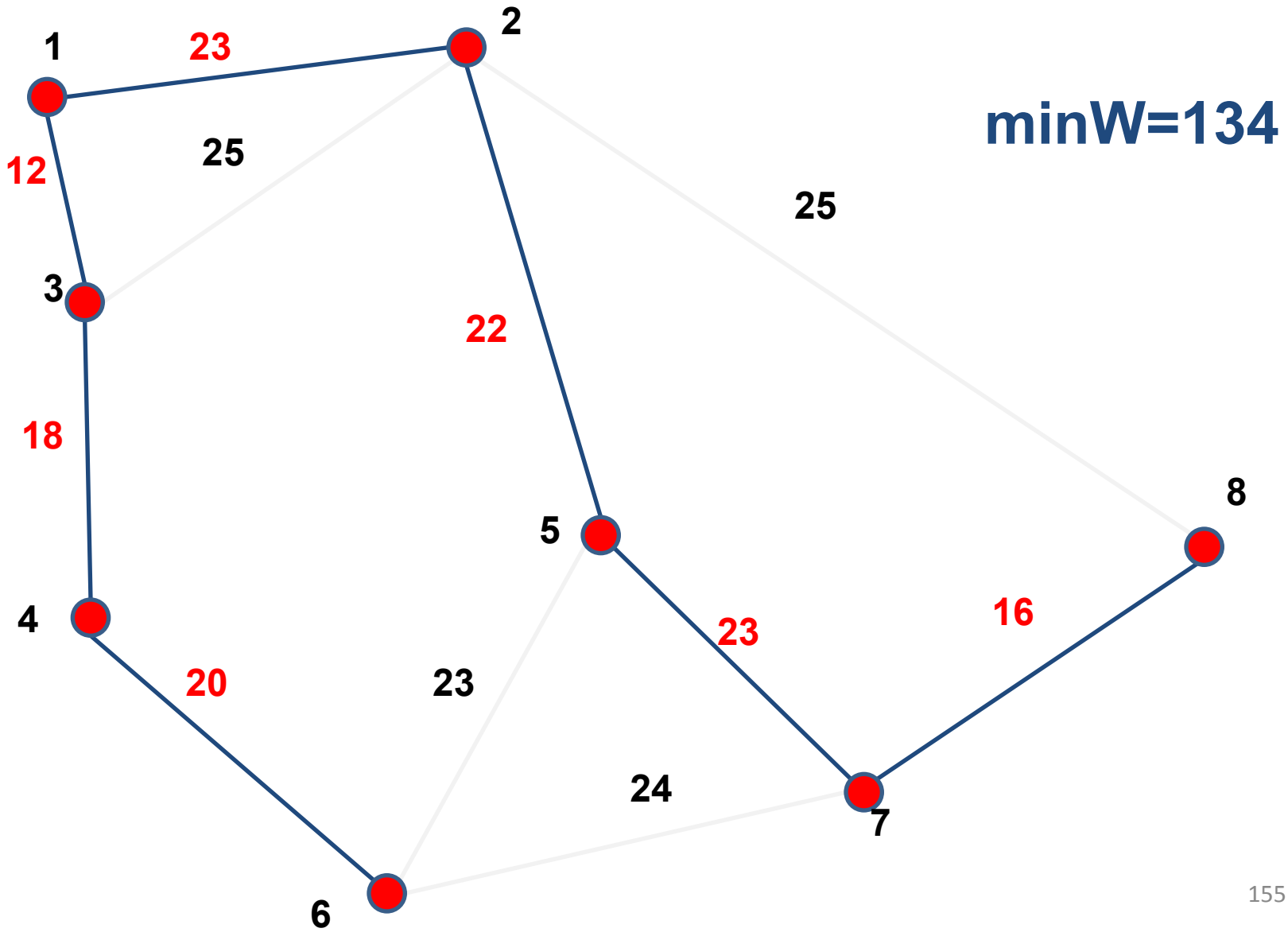
Шаг 2.



Шаг 3.

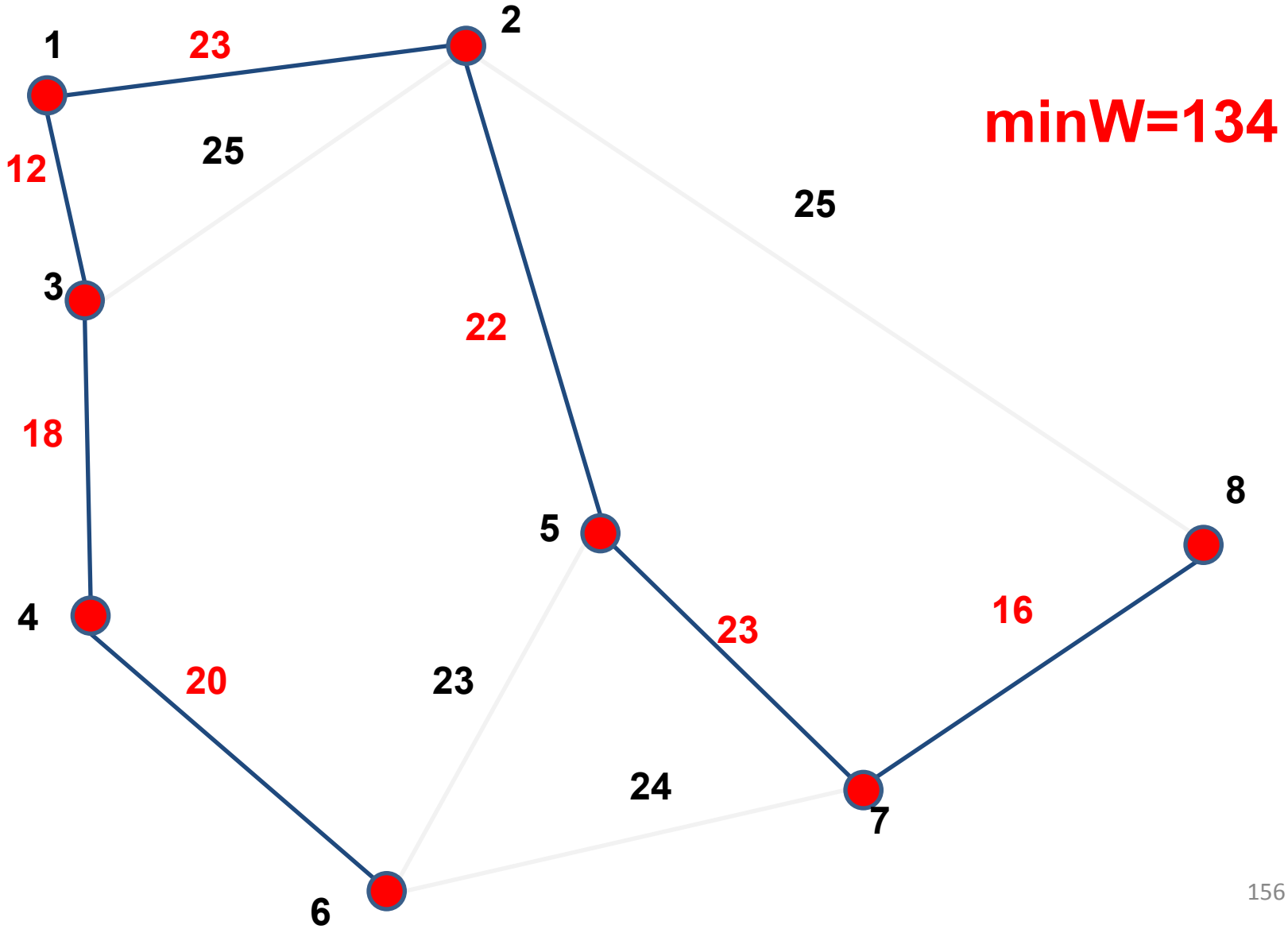


Шаг 4.



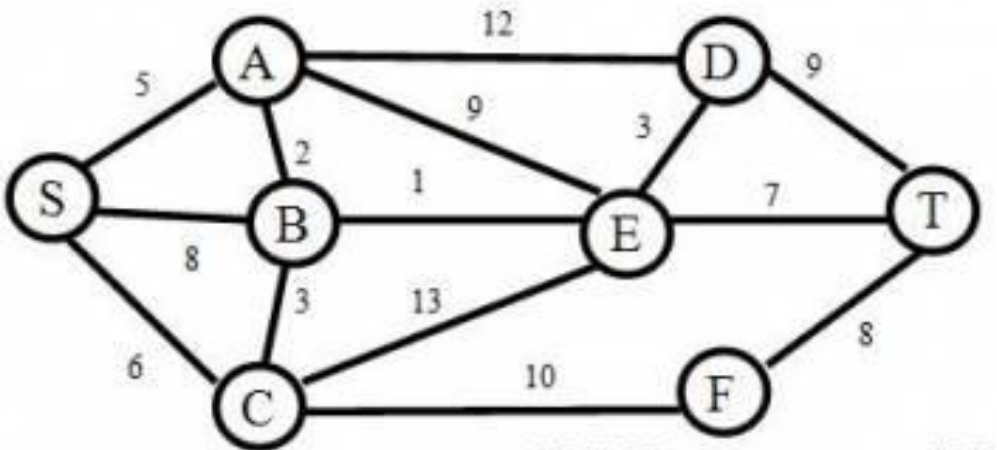
minW=134

Задача решена.



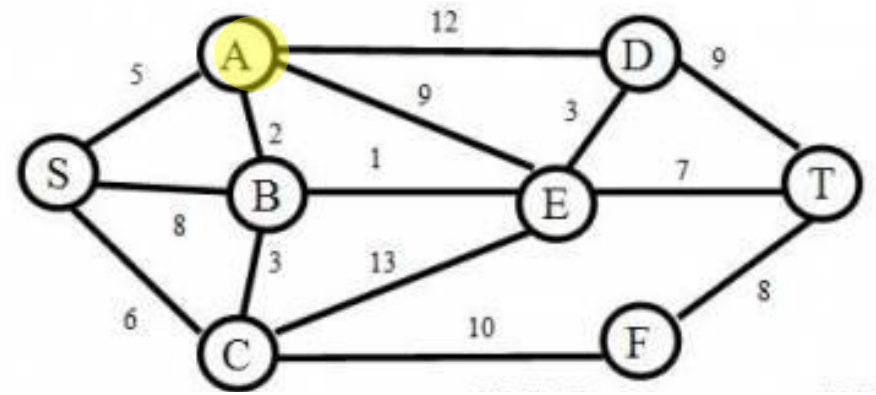
Алгоритм Прима

- Задан неориентированный взвешенный граф.
- Найти остовое дерево минимальной стоимости посредством алгоритма Прима.
- **Описание алгоритма:**
  - Построение начинается с дерева, включающего в себя одну (произвольную) вершину.
  - На каждом шаге алгоритма к текущему дереву присоединяется самое лёгкое из рёбер, соединяющих вершину из построенного дерева, и вершину не из дерева.



	A	B	C	D	E	F	T	S
A	0	2	0	12	9	0	0	5
B	2	0	3	0	1	0	0	8
C	0	3	0	0	13	10	0	6
D	12	0	0	0	3	0	9	0
E	9	1	13	3	0	0	7	0
F	0	0	10	0	0	0	8	0
T	0	0	0	9	7	8	0	0
S	5	8	6	0	0	0	0	0

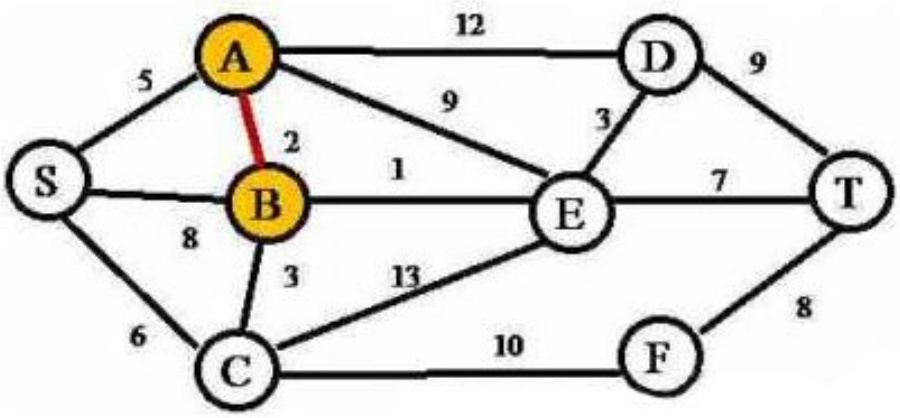
Алгоритм Прима



Находится в остове

*		A	B	C	D	E	F	T	S
*	A	0	2	0	12	9	0	0	5
	B	2	0	3	0	1	0	0	8
	C	0	3	0	0	13	10	0	6
	D	12	0	0	0	3	0	9	0
	E	9	1	13	3	0	0	7	0
	F	0	0	10	0	0	0	8	0
	T	0	0	0	9	7	8	0	0
	S	5	8	6	0	0	0	0	0

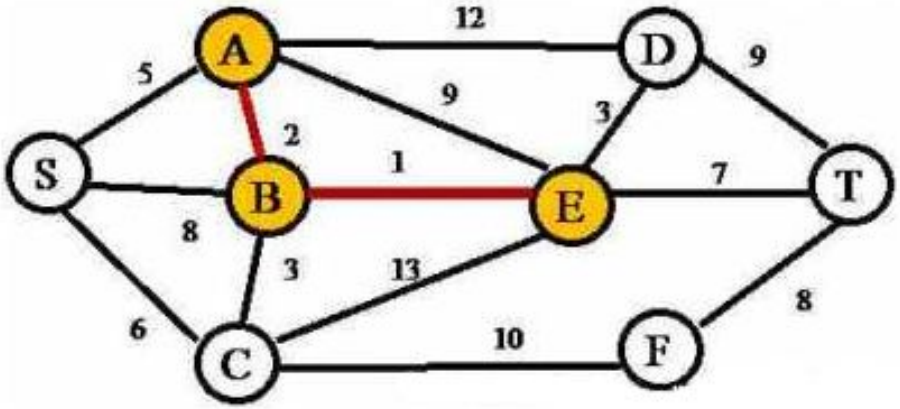
Алгоритм Прима



Находится в остове

*		A	B	C	D	E	F	T	S
*	A	0	~~2~~	0	12	9	0	0	5
*	B	~~2~~	0	3	0	1	0	0	8
	C	0	3	0	0	13	10	0	6
	D	12	0	0	0	3	0	9	0
	E	9	1	13	3	0	0	7	0
	F	0	0	10	0	0	0	8	0
	T	0	0	0	9	7	8	0	0
	S	5	8	6	0	0	0	0	0

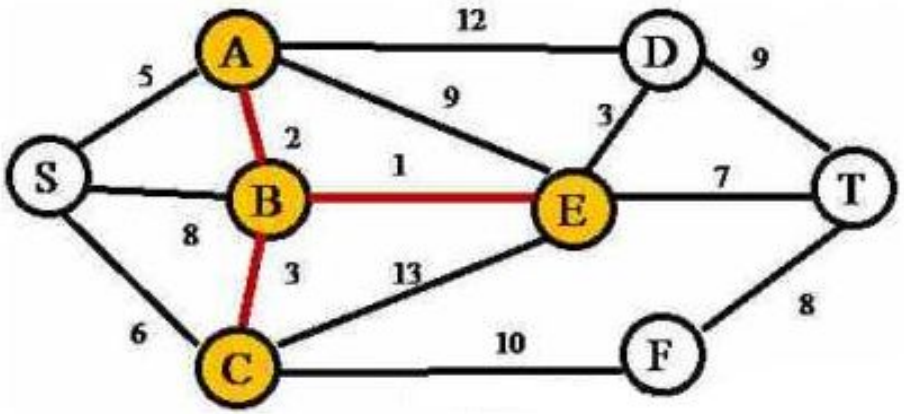
Алгоритм Прима



Находится в остове

*		A	B	C	D	E	F	T	S
*	A	0	~~2~~	0	12	9	0	0	5
*	B	~~2~~	0	3	0	~~1~~	0	0	8
	C	0	3	0	0	13	10	0	6
	D	12	0	0	0	3	0	9	0
*	E	9	~~1~~	13	3	0	0	7	0
	F	0	0	10	0	0	0	8	0
	T	0	0	0	9	7	8	0	0
	S	5	8	6	0	0	0	0	0

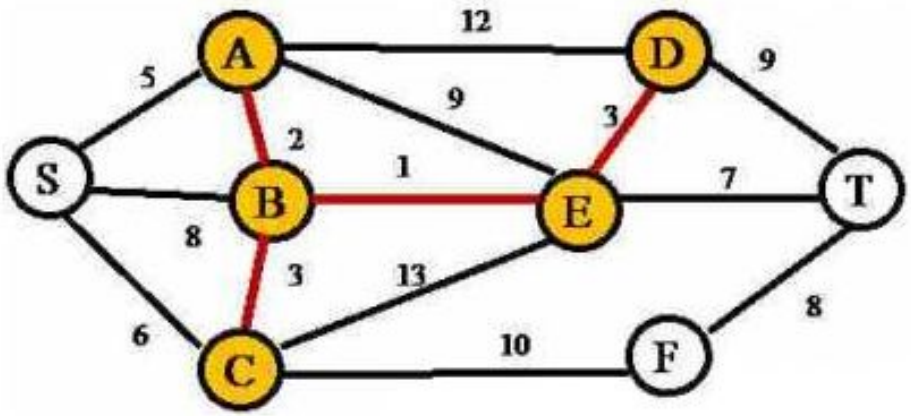
Алгоритм Прима



Находится в остове

*		A	B	C	D	E	F	T	S
*	A	0	~~2~~	0	12	9	0	0	5
*	B	~~2~~	0	~~3~~	0	~~1~~	0	0	8
*	C	0	~~3~~	0	0	13	10	0	6
	D	12	0	0	0	3	0	9	0
*	E	9	~~1~~	13	3	0	0	7	0
	F	0	0	10	0	0	0	8	0
	T	0	0	0	9	7	8	0	0
	S	5	8	6	0	0	0	0	0

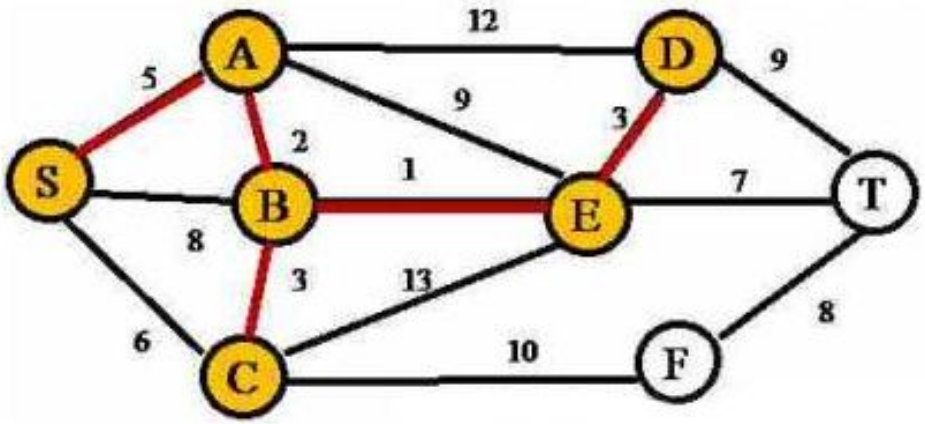
Алгоритм Прима



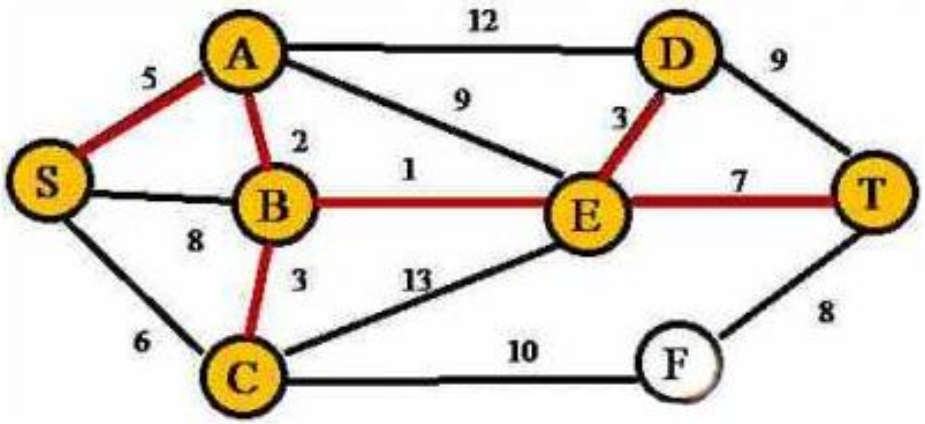
Находится в остове

*		A	B	C	D	E	F	T	S
*	A	0	~~2~~	0	12	9	0	0	5
*	B	~~2~~	0	~~3~~	0	~~1~~	0	0	8
*	C	0	~~3~~	0	0	13	10	0	6
*	D	12	0	0	0	~~3~~	0	9	0
*	E	9	~~1~~	13	~~3~~	0	0	7	0
	F	0	0	10	0	0	0	8	0
	T	0	0	0	9	7	8	0	0
	S	5	8	6	0	0	0	0	0

Алгоритм Прима



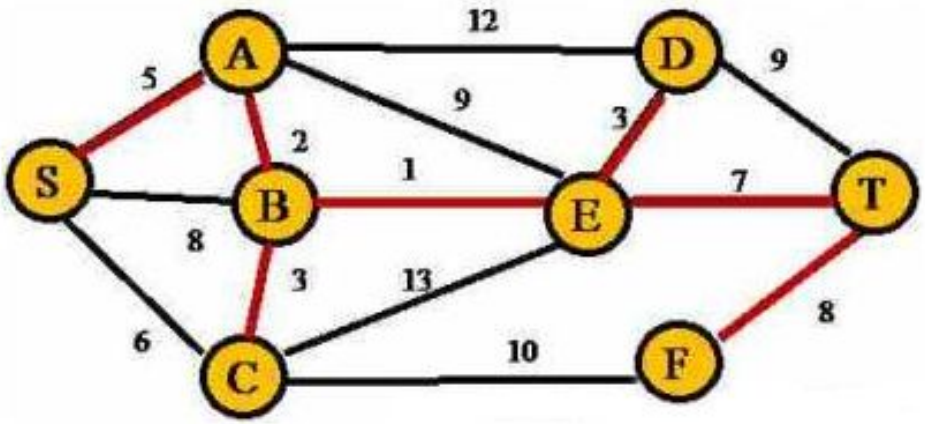
Алгоритм Прима



Находится в остове

*		A	B	C	D	E	F	T	S
*	A	0	~~2~~	0	12	9	0	0	~~5~~
*	B	~~2~~	0	~~3~~	0	~~1~~	0	0	8
*	C	0	~~3~~	0	0	13	10	0	6
*	D	12	0	0	0	~~3~~	0	9	0
*	E	9	~~1~~	13	~~3~~	0	0	~~7~~	0
	F	0	0	10	0	0	0	8	0
*	T	0	0	0	9	~~7~~	8	0	0
*	S	~~5~~	8	6	0	0	0	0	0

Алгоритм Прима



Находится в остове

*		A	B	C	D	E	F	T	S
*	A	0	~~2~~	0	12	9	0	0	~~5~~
*	B	~~2~~	0	~~3~~	0	~~1~~	0	0	8
*	C	0	~~3~~	0	0	13	10	0	6
*	D	12	0	0	0	~~3~~	0	9	0
*	E	9	~~1~~	13	~~3~~	0	0	~~7~~	0
*	F	0	0	10	0	0	0	~~8~~	0
*	T	0	0	0	9	~~7~~	~~8~~	0	0
*	S	~~5~~	8	6	0	0	0	0	0

Пример алгоритма Прима

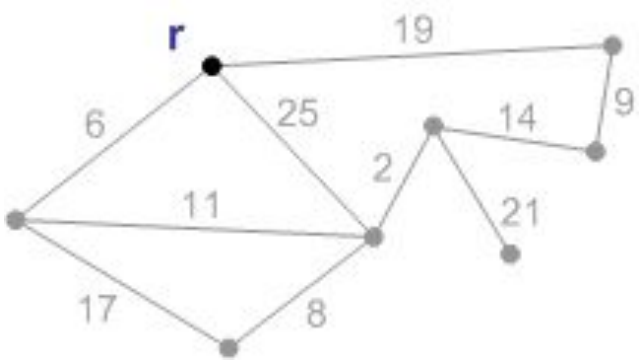


Рис.1.

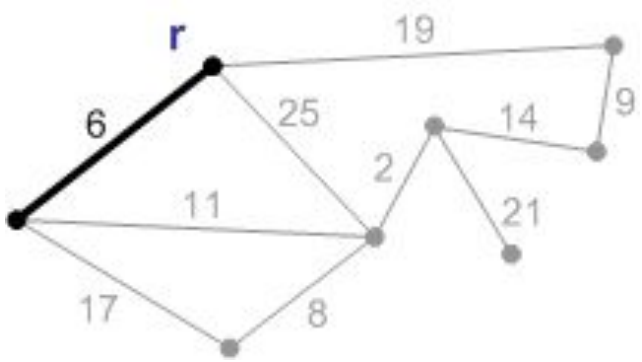


Рис.2.

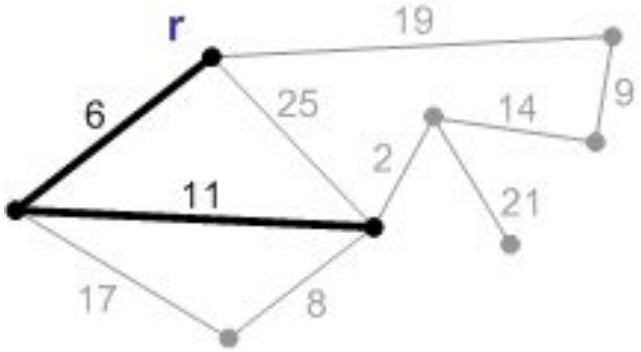


Рис.3.

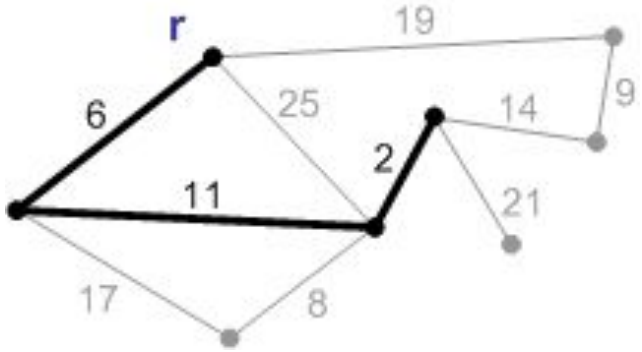


Рис.4.

Пример алгоритма Прима

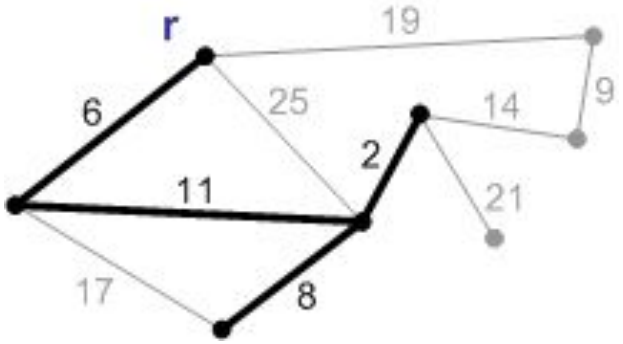


Рис.5.

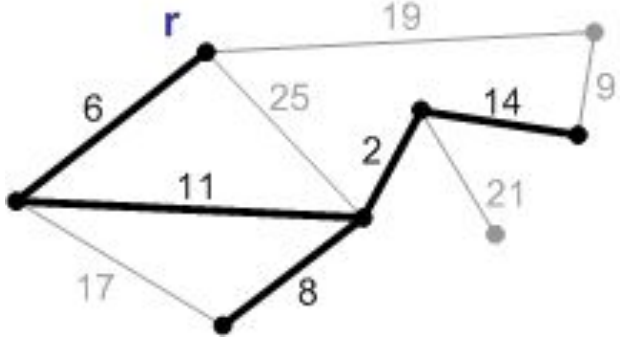


Рис.6.

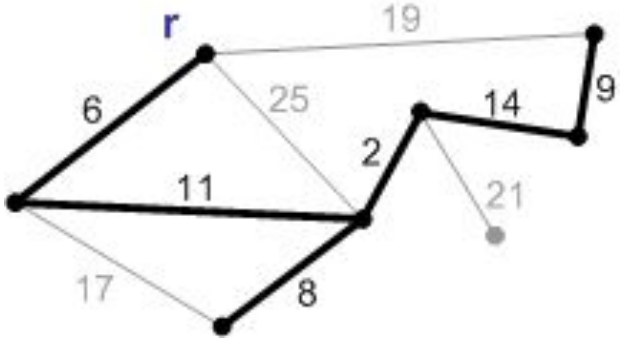


Рис.7.

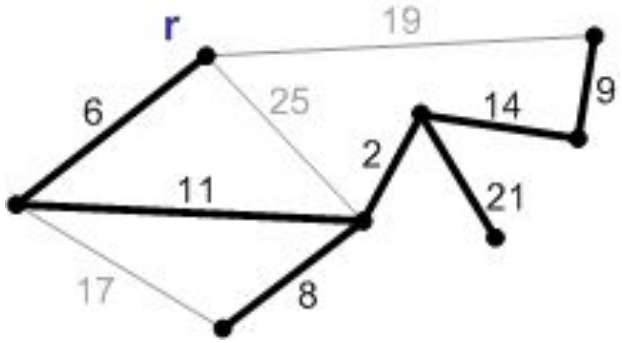
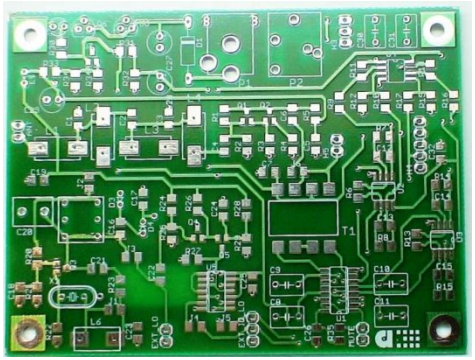


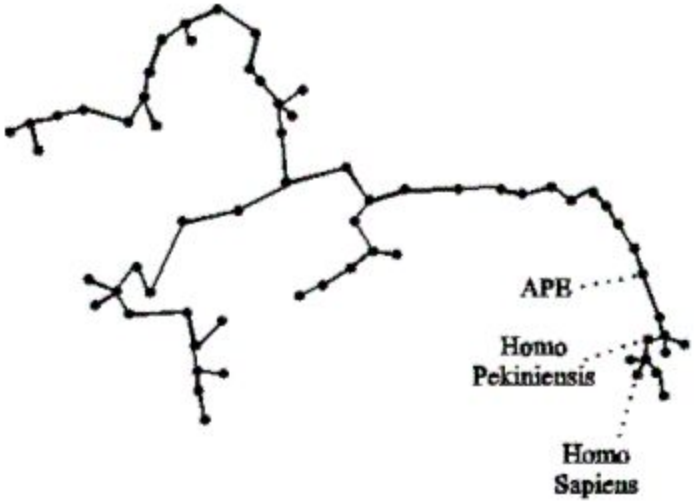
Рис.8.

Области применения задачи ПМОД

1. Разработка различного рода сетей (см.выше).
2. Производство печатных плат.



3. Визуализация многоаспектных, многомерных данных (например, для отображения их взаимосвязи)



Эволюция животных видов