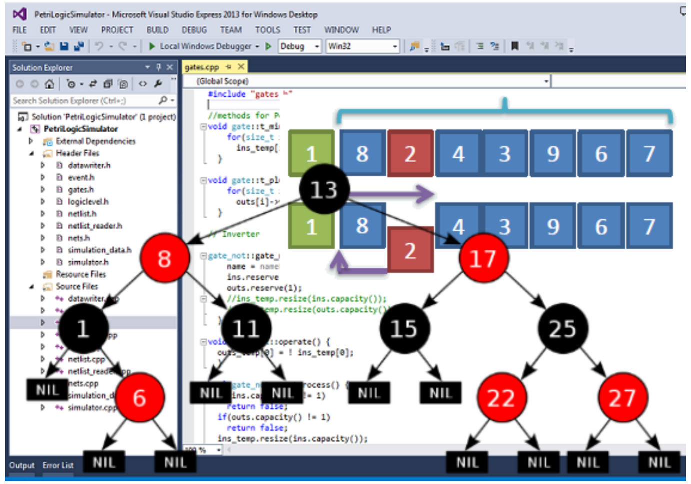
Национальный исследовательский университет «МИЭТ»

Институт интегральной электроники (группы ЭН-24-25, каф. ПКИМС)

Теория алгоритмов



Лекция 4

Графы. Способы представления и основные алгоритмы обработки

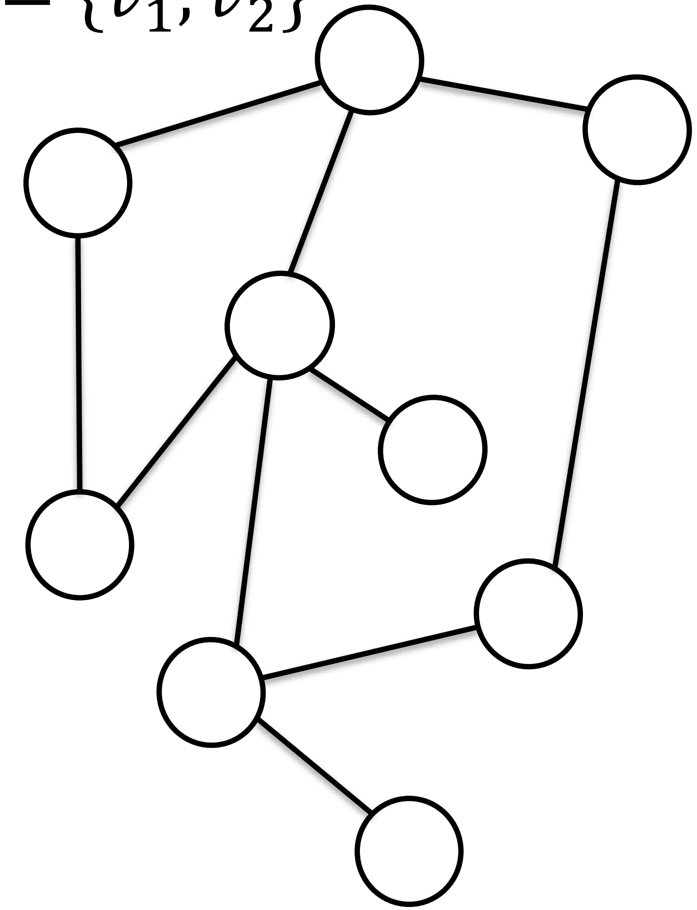
Что такое граф?

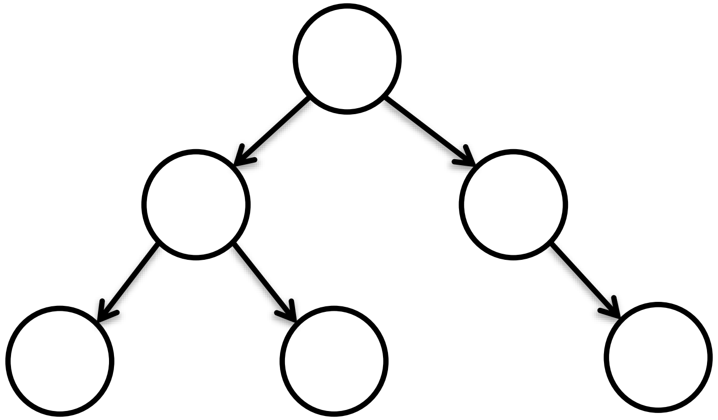
$$G : (V, E)$$

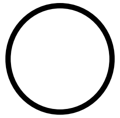
$$v_i \in V, \qquad V \neq \emptyset$$

$$e_i \in E \ : \ e = \{v_1, v_2\}$$

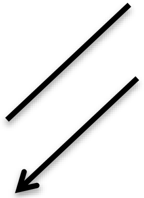
Порядок графа – число вершин графа
Степень вершины – число рёбер, инцидентных вершине





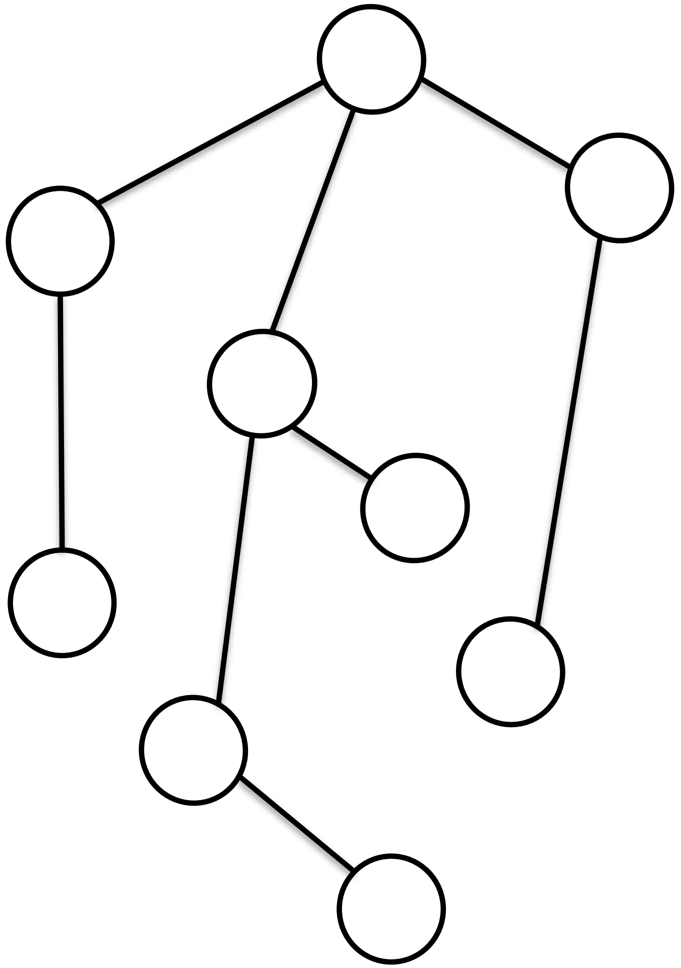


Вершина графа

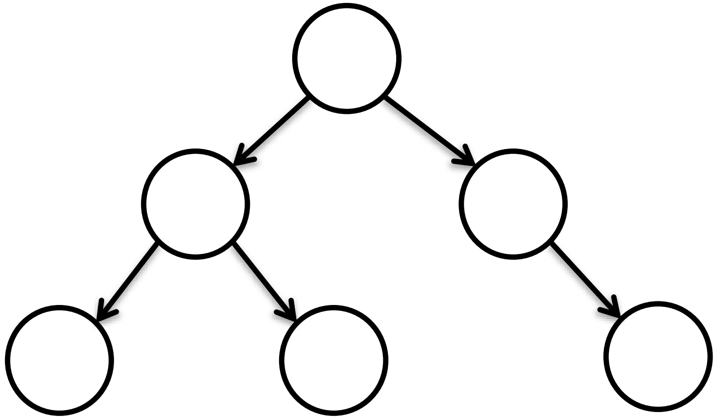


Ребро графа

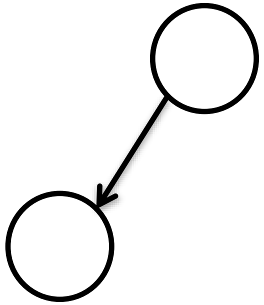
Теория графов: основные определения



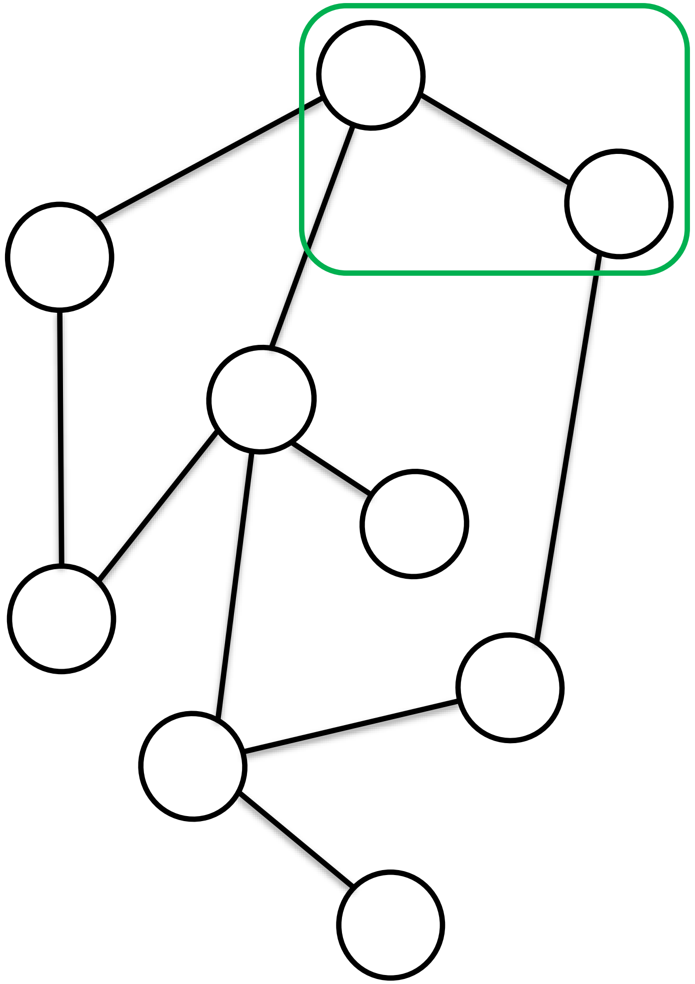
Неориентированный граф

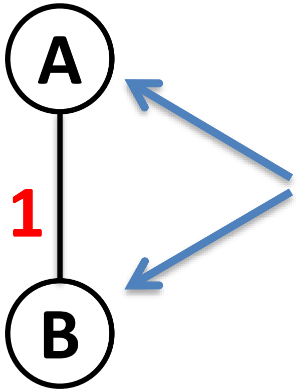


Ориентированный граф
(орграф)

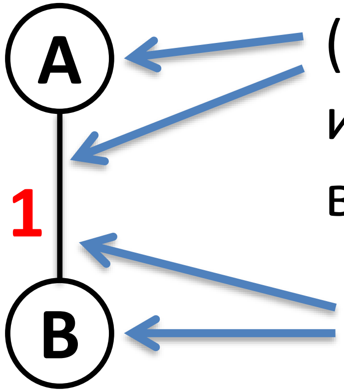


Теория графов: основные определения (2)





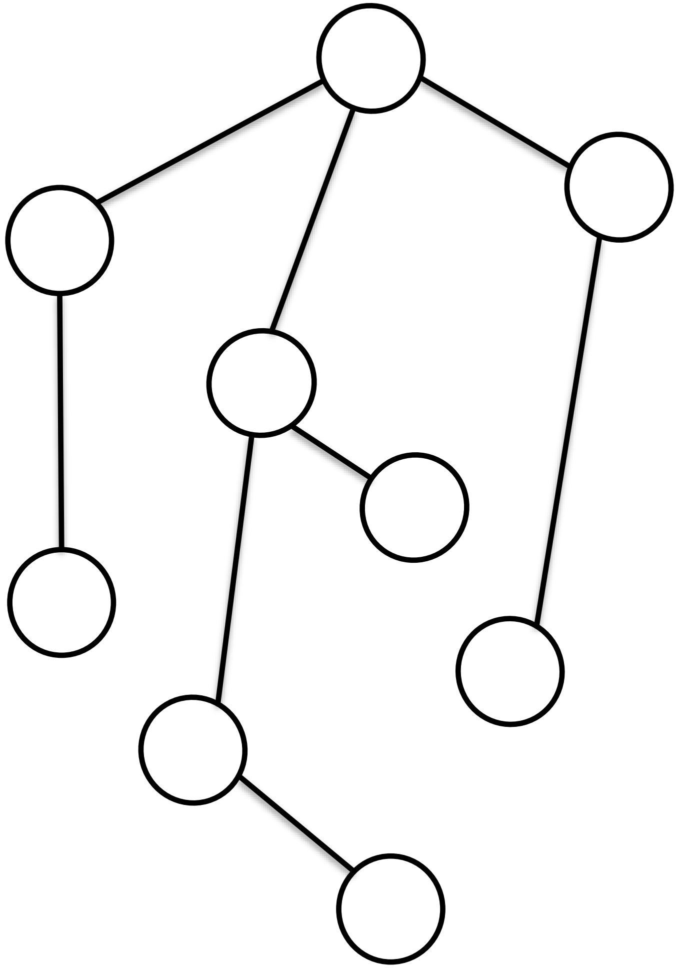
Вершины А и В - смежные вершины



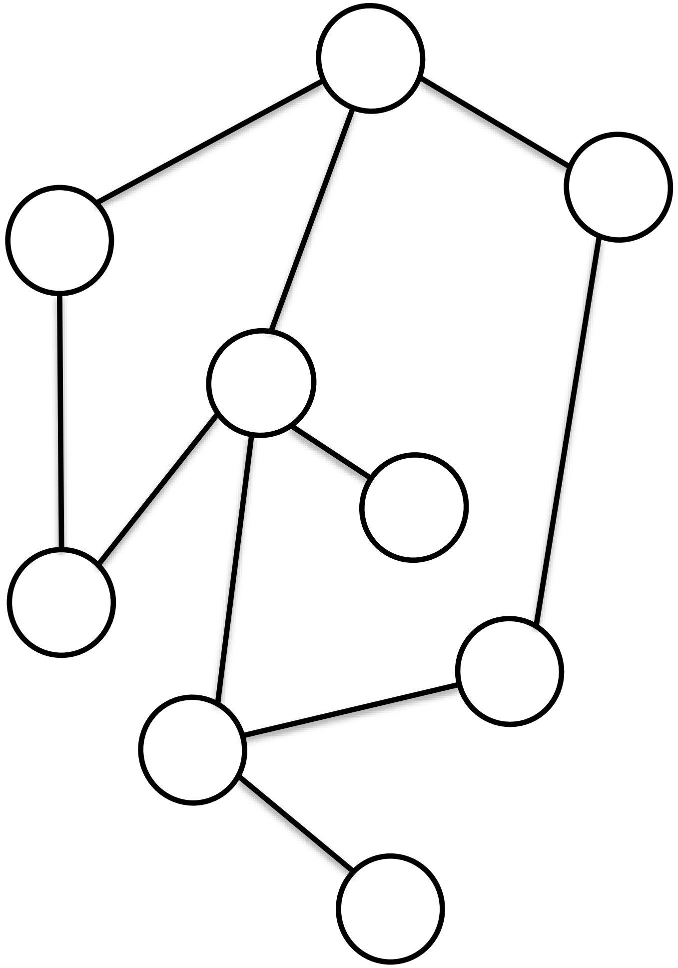
Вершина А инцидентна ребру 1 (ребро 1 инцидентно вершине А)

Вершина В инцидентна ребру 1 (ребро 1 инцидентно вершине В)

Теория графов: основные определения (3)

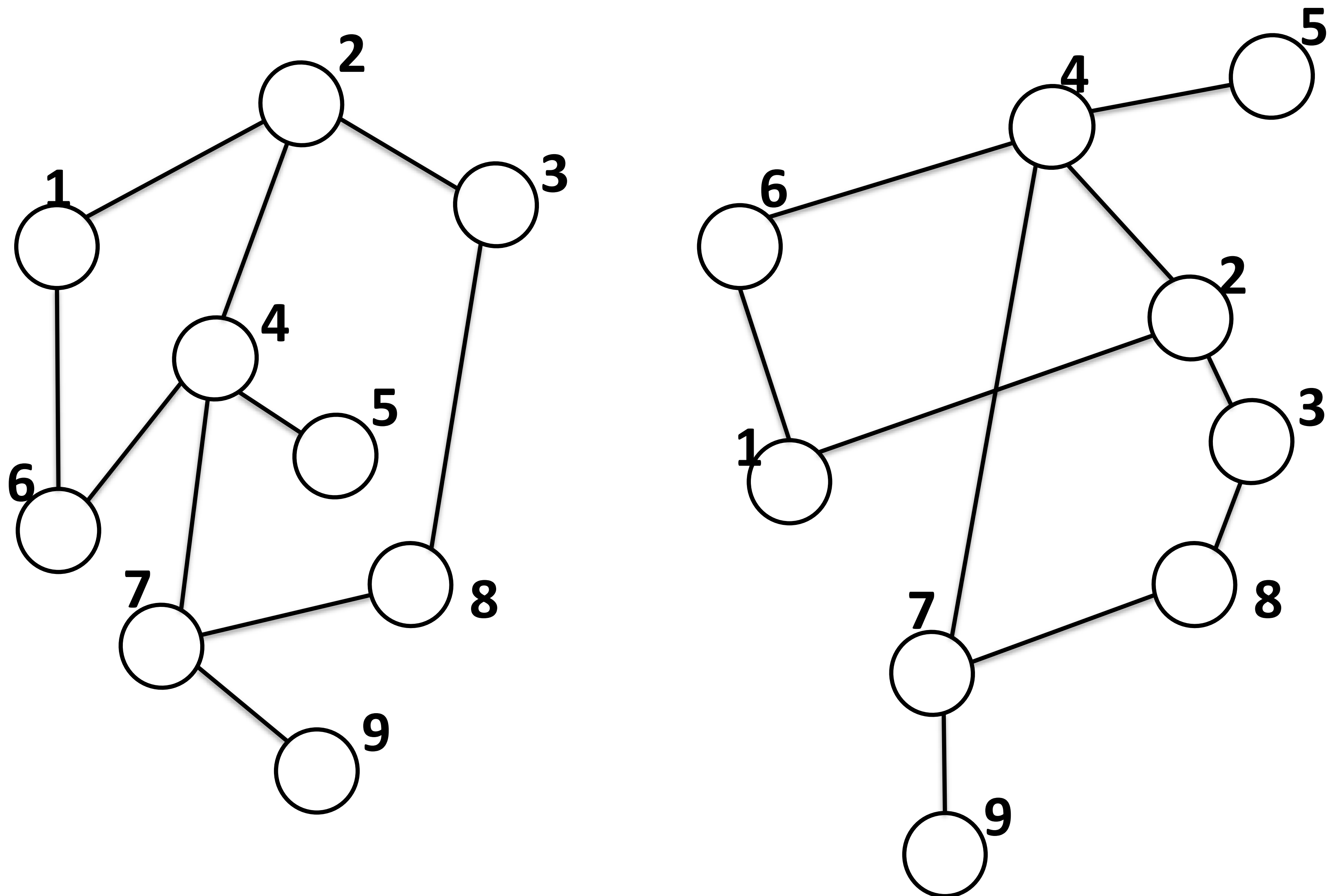


Не содержит циклов
(дерево)

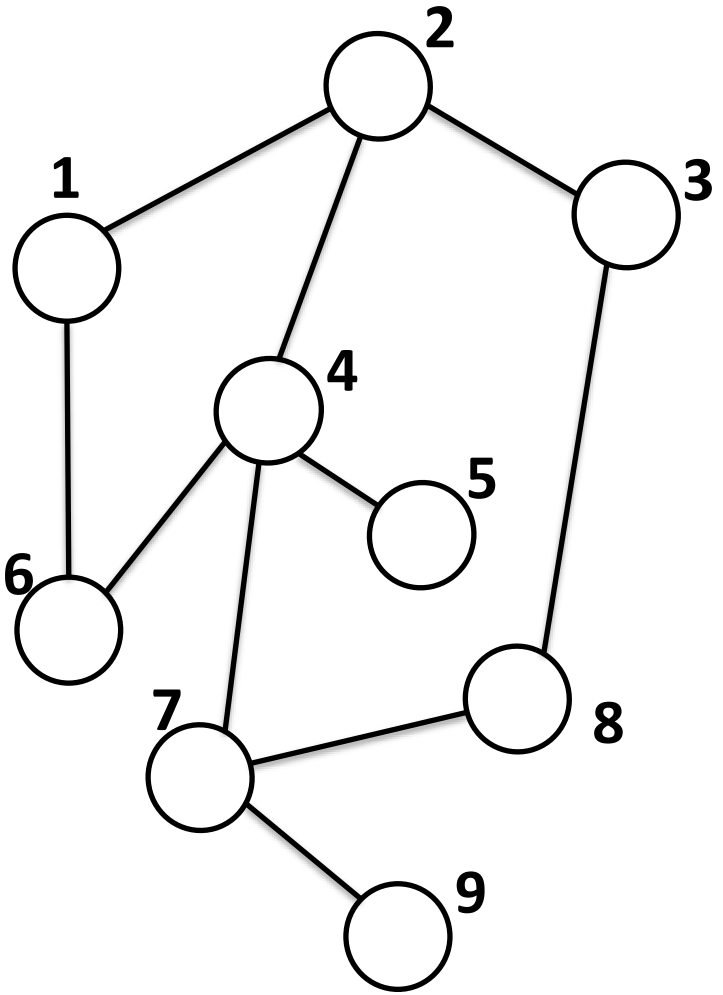


Содержит циклы

Изоморфизм графов



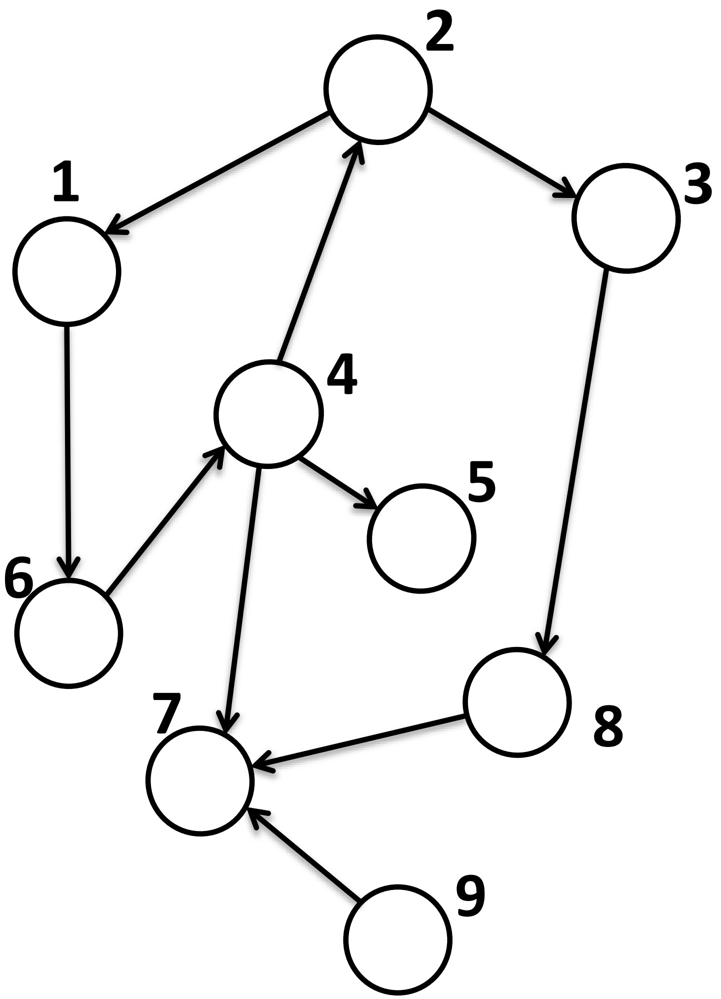
Способы представления графа: матрица смежности (1)



Матрица смежности

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

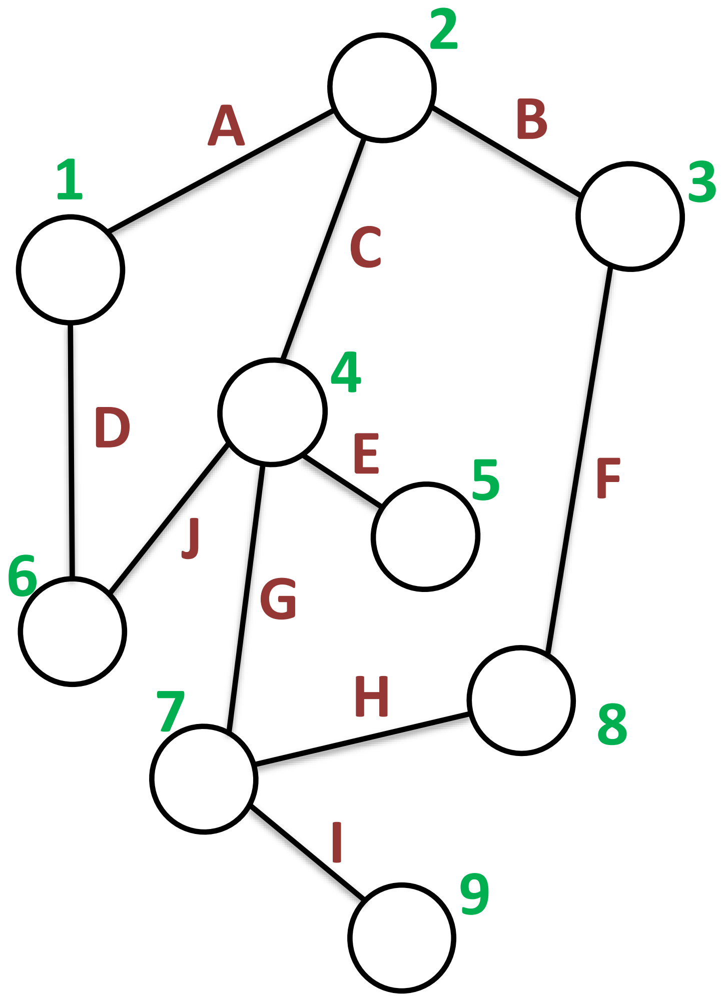
Способы представления графа: матрица смежности (2)



Матрица смежности для орграфа

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

Способы представления графа: матрица инцидентности



Матрица инцидентности

	A	B	C	D	E	F	G	H	I	J
1	1			1						
2	1	1	1							
3		1				1				
4			1		1		1			1
5					1					
6				1						1
7							1	1	1	
8						1		1		
9									1	

Представление графов в виде списков и массивов (1)

Вариант 1 - списки

```
struct GraphNode {
  std::vector<GraphNode *> nodes;

  std::string              name;
  size_t                   index;

  double                   value;
  double                   parameter1,
                           parameter2;
};
```

Представление графов в виде списков и массивов (1)

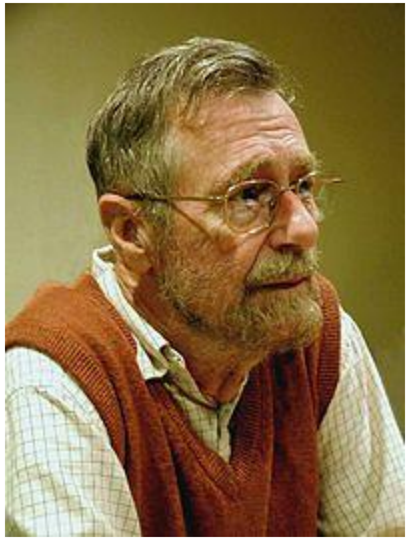
Вариант 2 - массивы

```
struct Branch {
  std::string name;
  size_t      index;
  ...
};
```

```
struct Node {
  std::vector<int> my_branches;
  std::string      name;
  size_t           index;
  ...
};
```

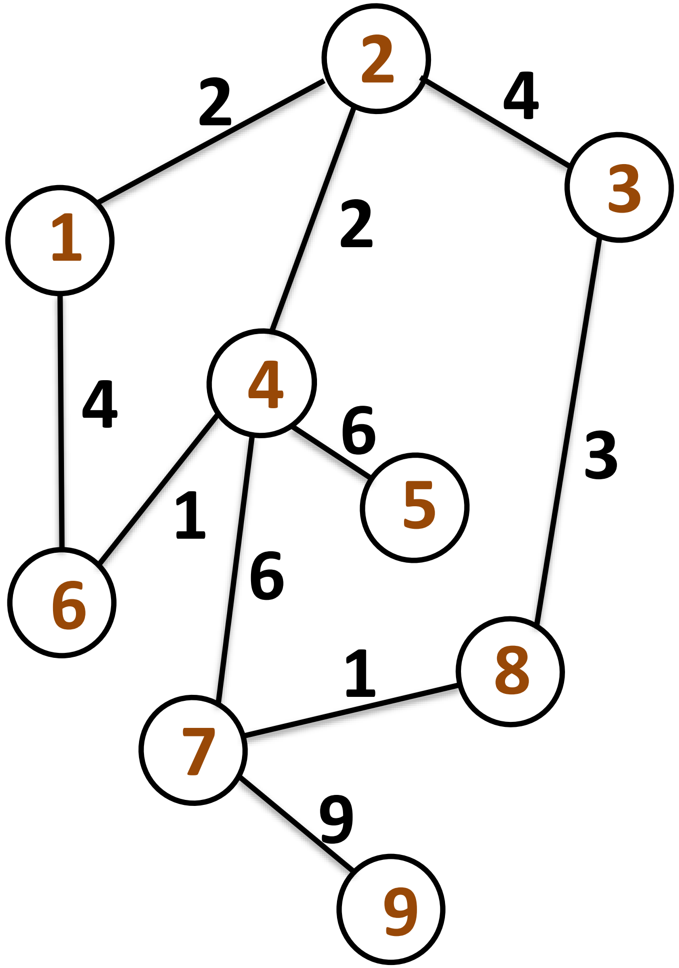
```
std::vector<Node   *> nodes;
std::vector<Branch *> branches;
```

Алгоритм Дейкстры: обход элементов графа «в ширину» (1)



Алгоритм Дейкстры – алгоритм на графах, находящий кратчайшее расстояние от одной из вершин до всех остальных

Эдсгер Вибе Дейкстра



Для демонстрации алгоритма необходимо задать рёбрам весовые коэффициенты.

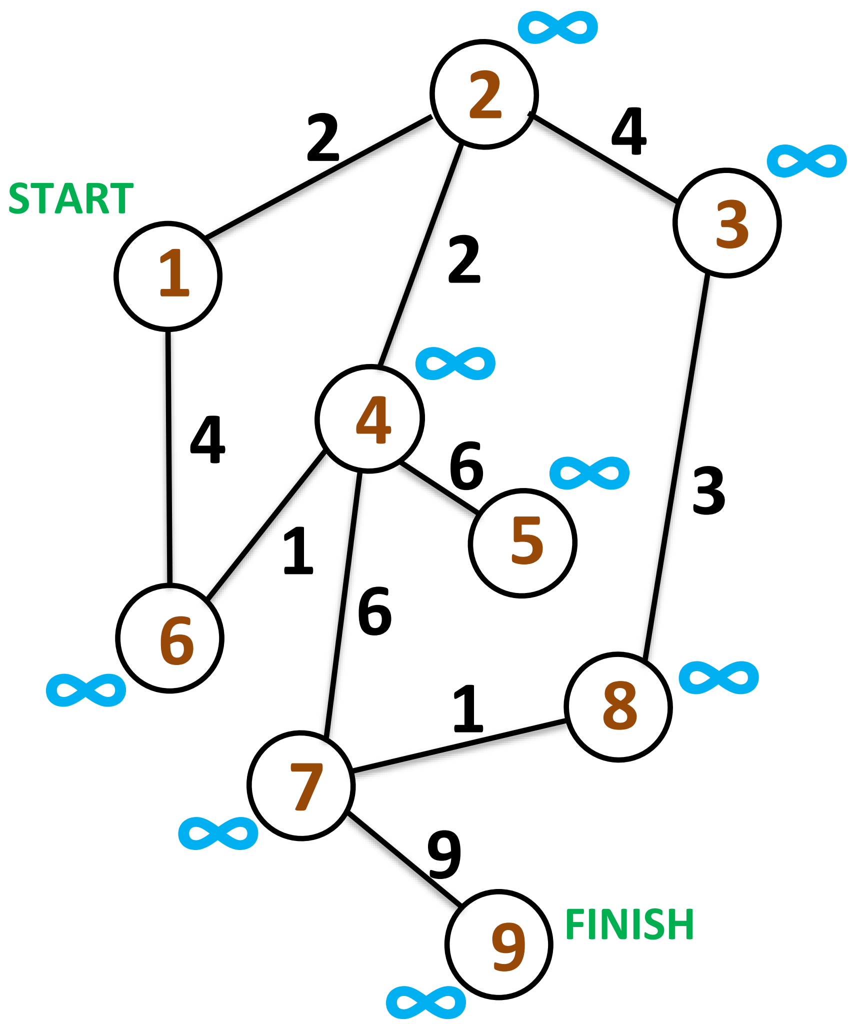
Такой граф называется взвешенным.

Алгоритм Дейкстры: обход элементов графа «в ширину» (2)

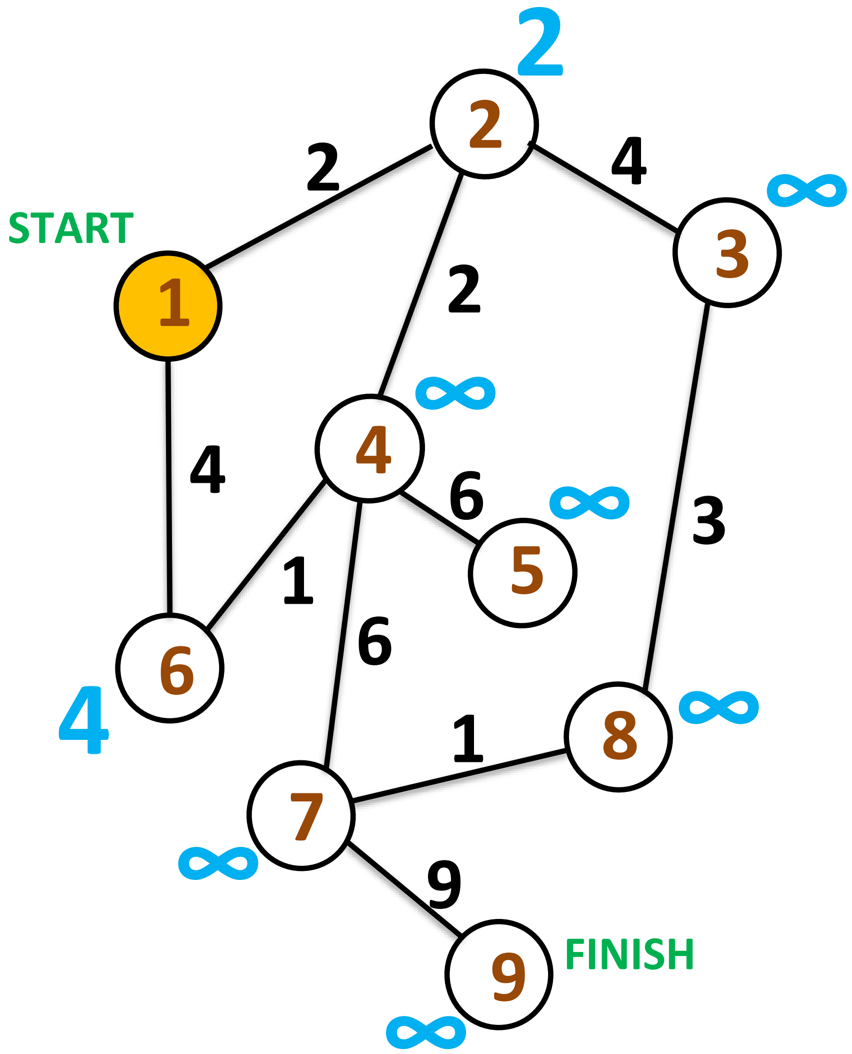
Шаги алгоритма Дейкстры «в ширину»:

1. задать для всех вершин графа значения расстояний, равные бесконечности;
2. выбрать начальную и конечную вершины графа (откуда и куда ищем кратчайшее расстояние);
3. начиная с выбранной вершины вести лексикографический обход смежных вершин;
4. для каждой смежной вершины проставить значение минимального получающегося суммарного расстояния;
5. повторять шаги 3,4, пока есть непосещённые вершины.

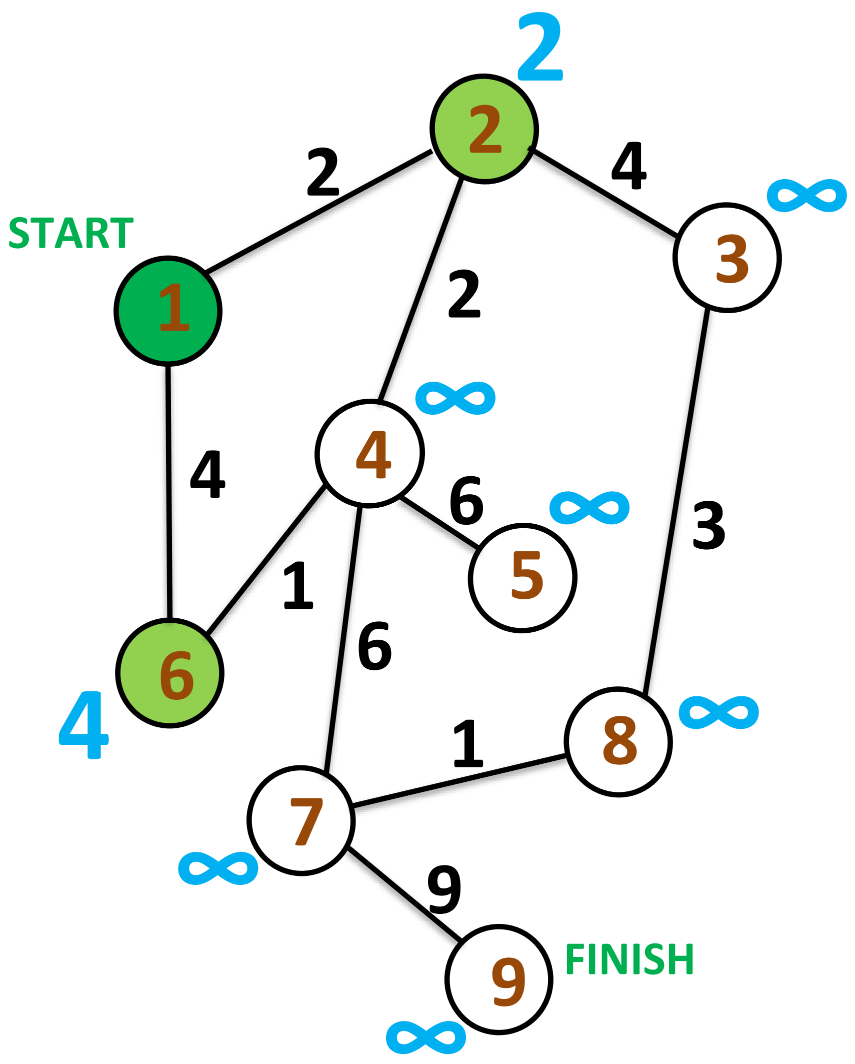
Алгоритм Дейкстры (1)



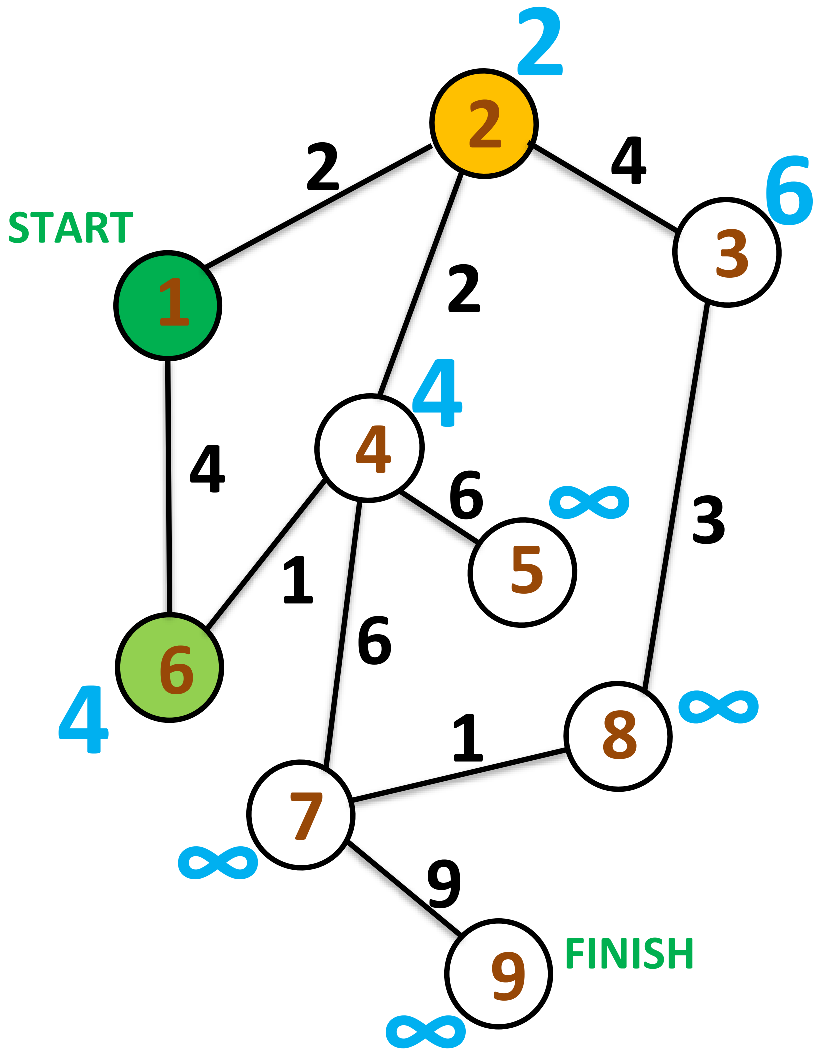
Алгоритм Дейкстры (2)



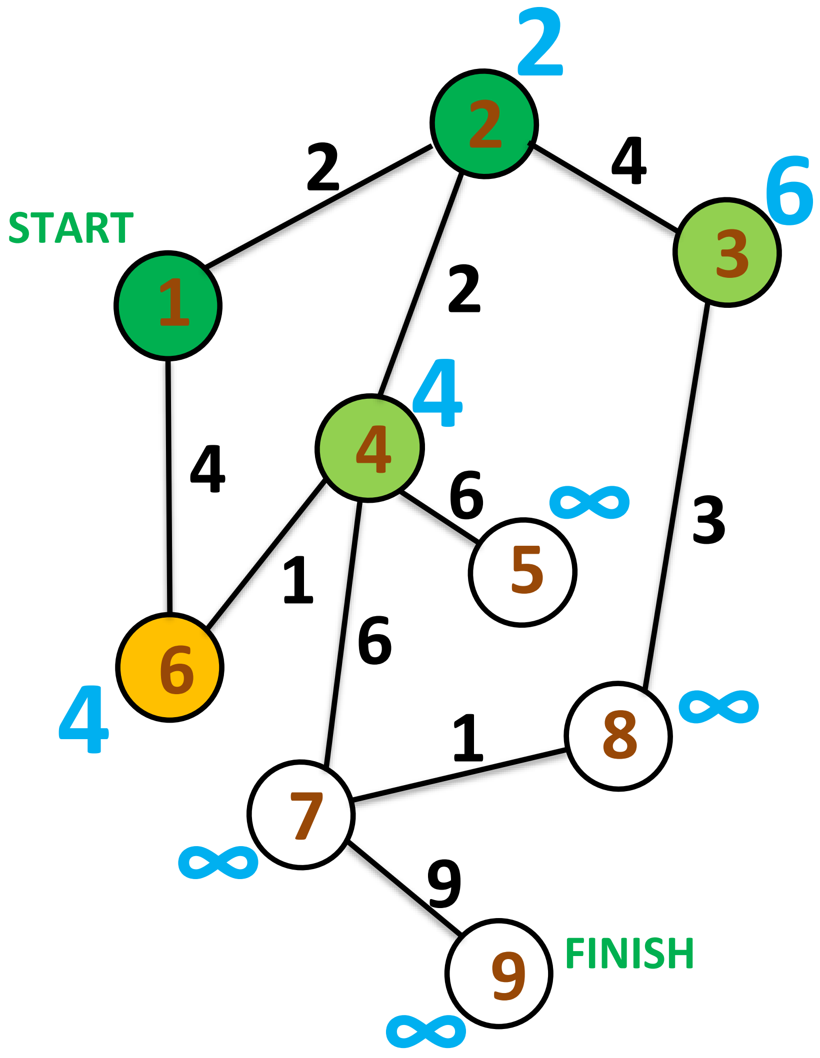
Алгоритм Дейкстры (3)



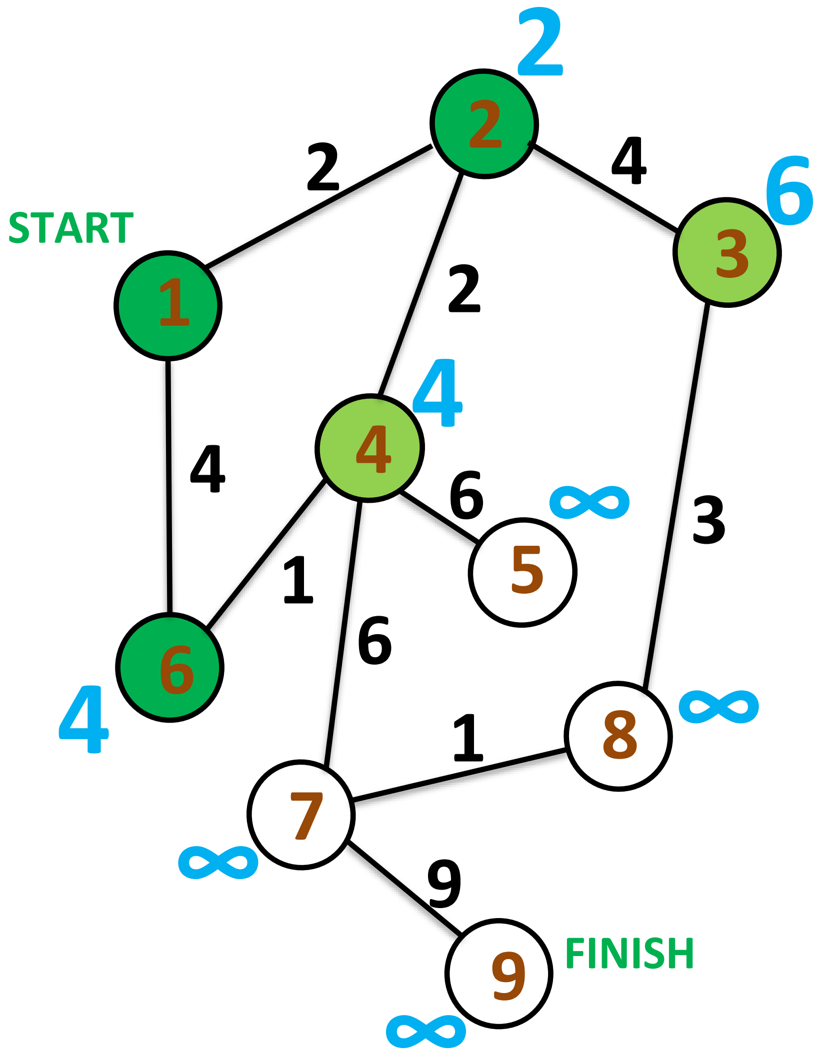
Алгоритм Дейкстры (4)



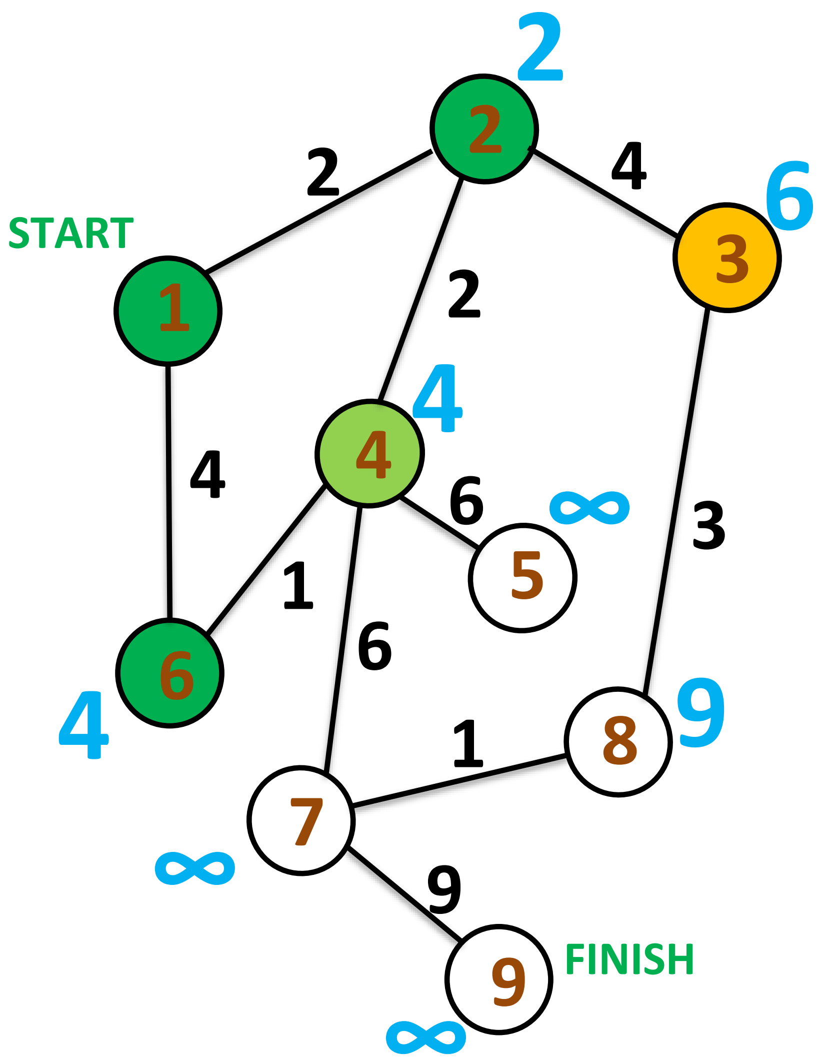
Алгоритм Дейкстры (5)



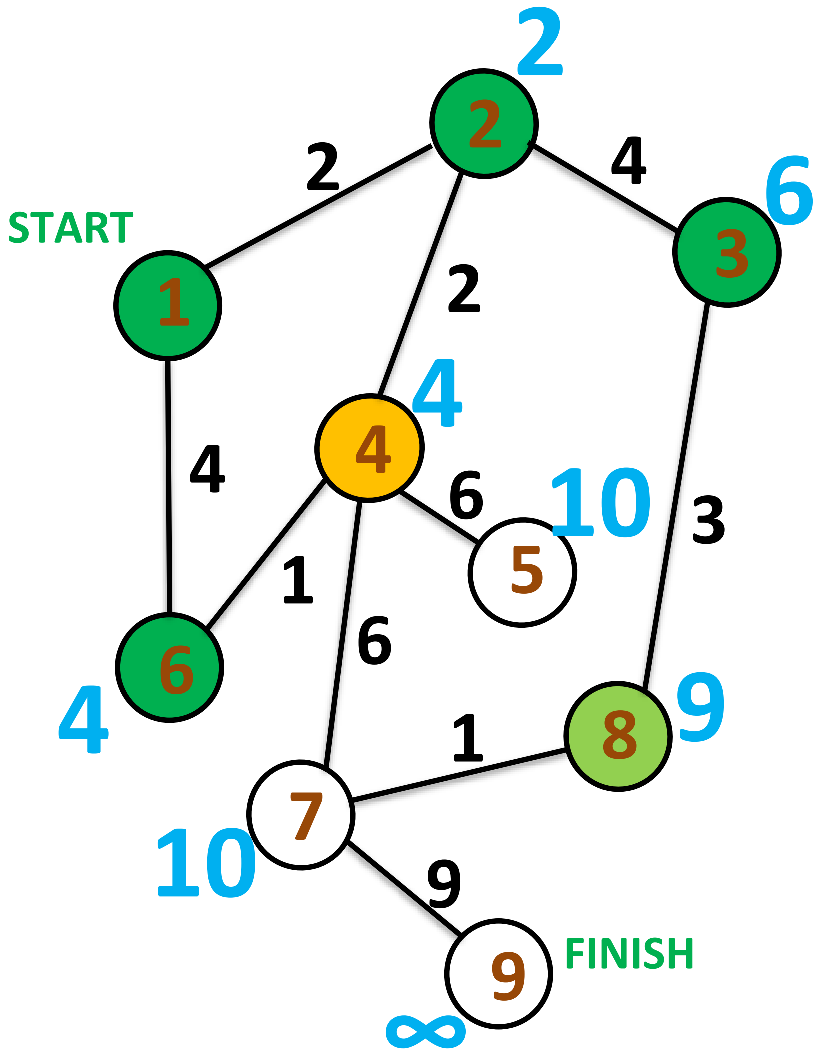
Алгоритм Дейкстры (6)



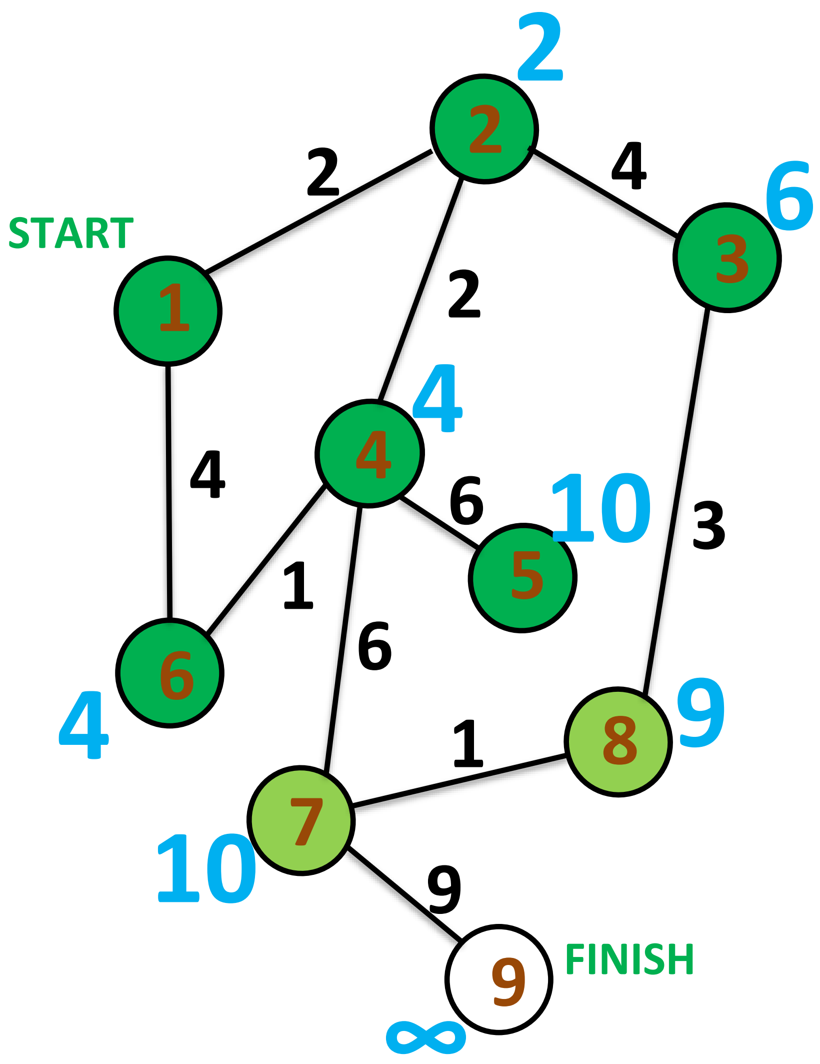
Алгоритм Дейкстры (7)



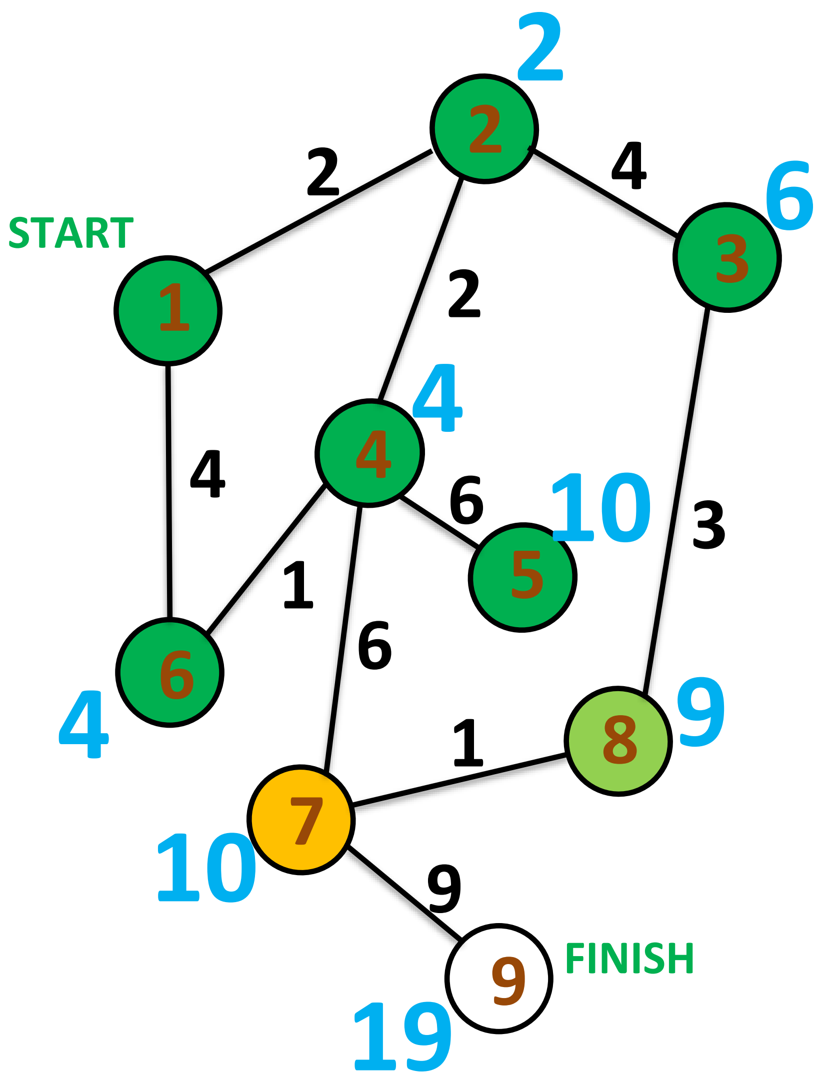
Алгоритм Дейкстры (8)



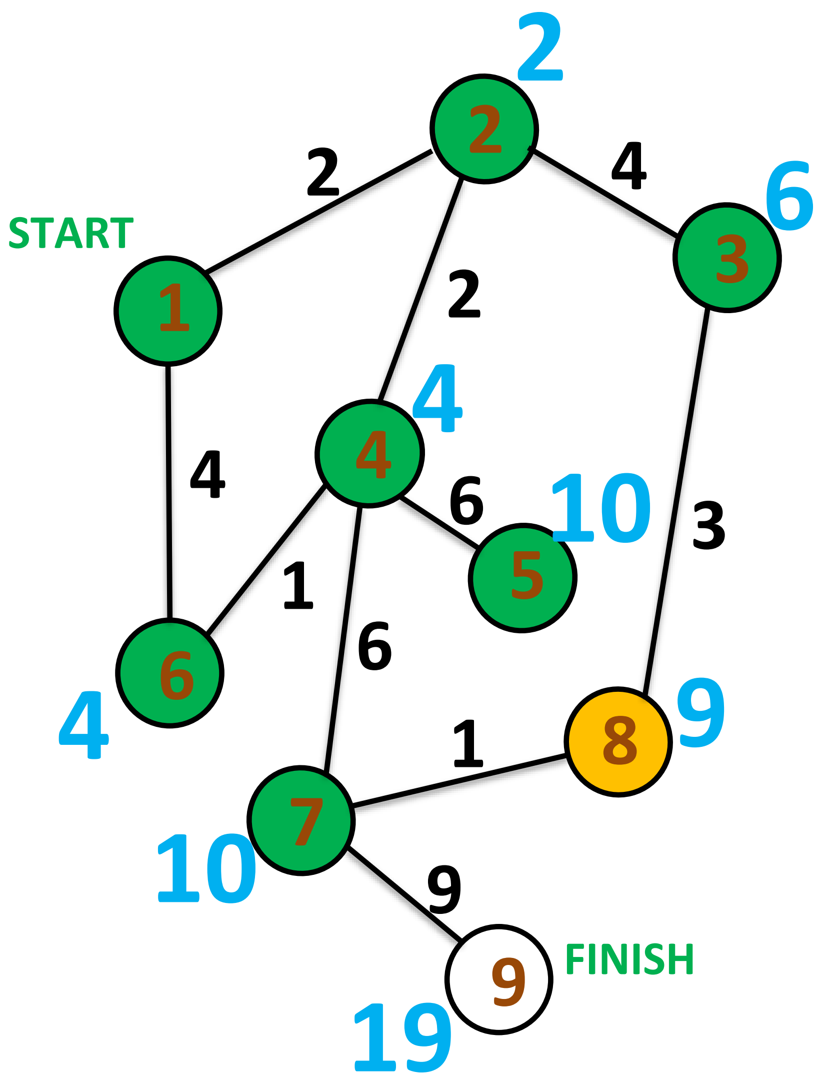
Алгоритм Дейкстры (9)



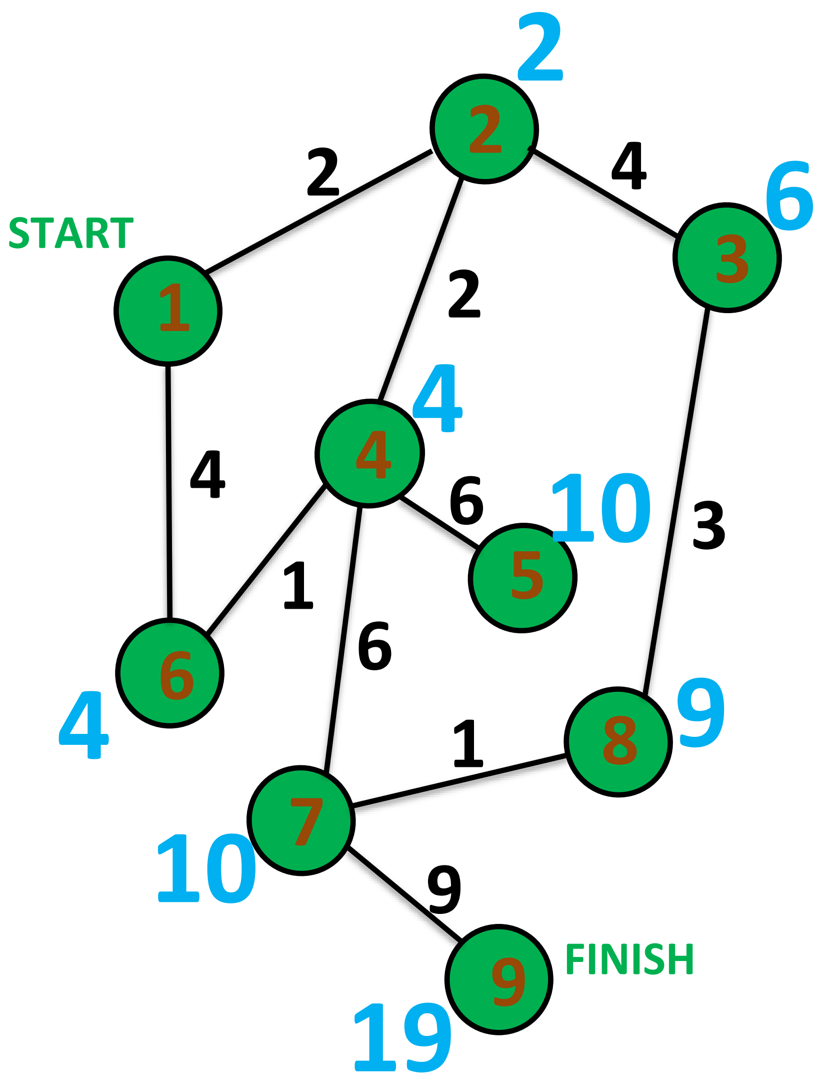
Алгоритм Дейкстры (10)



Алгоритм Дейкстры (11)

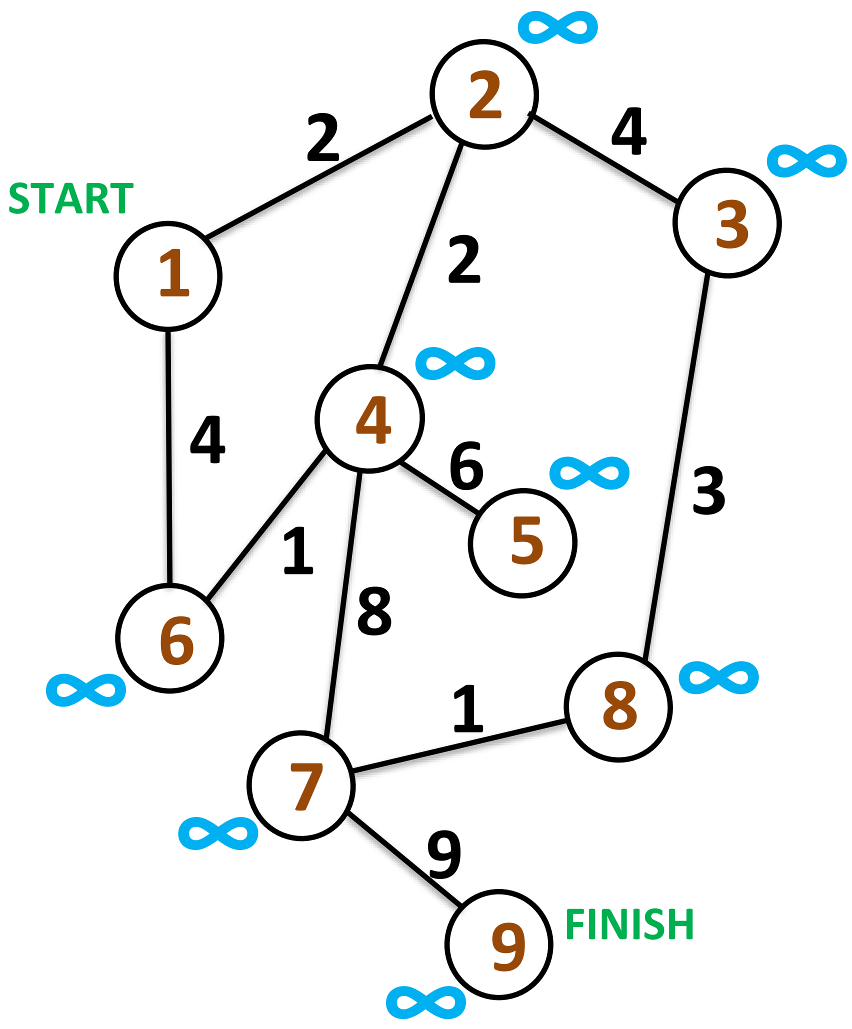


Алгоритм Дейкстры (12)

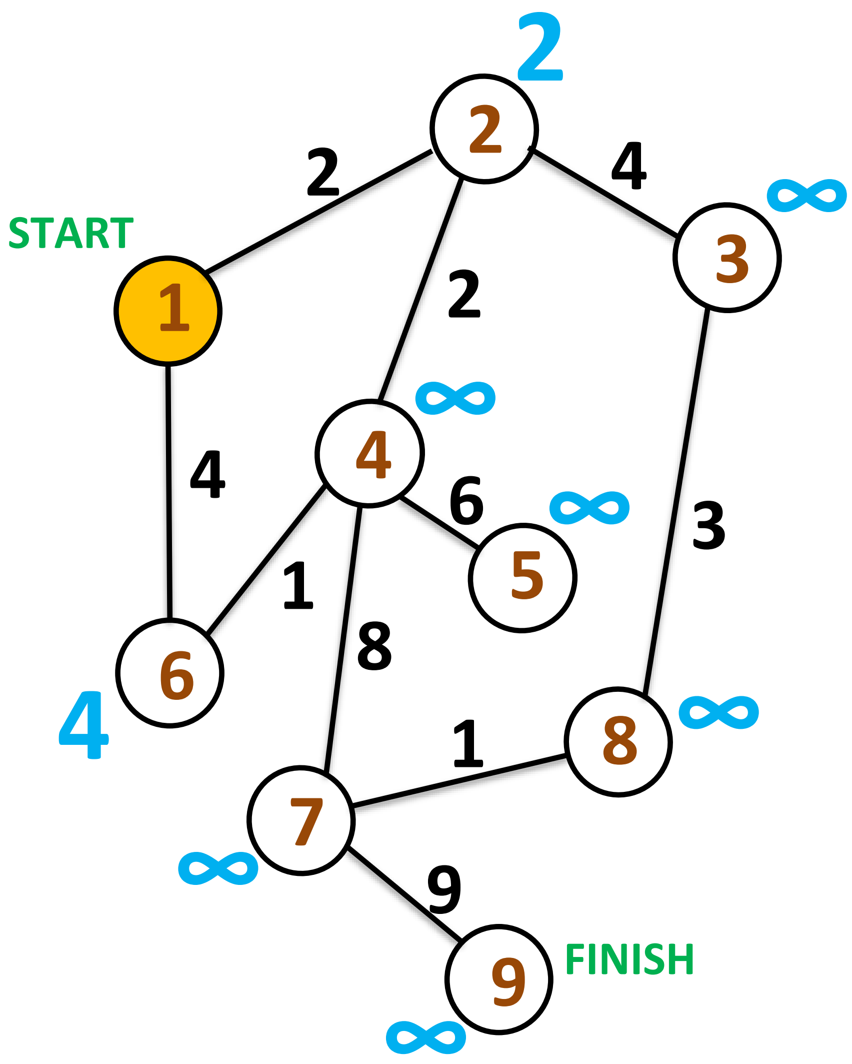


Путь из точки **START** в точку **FINISH** существует, самый короткий путь занимает 19 единиц.

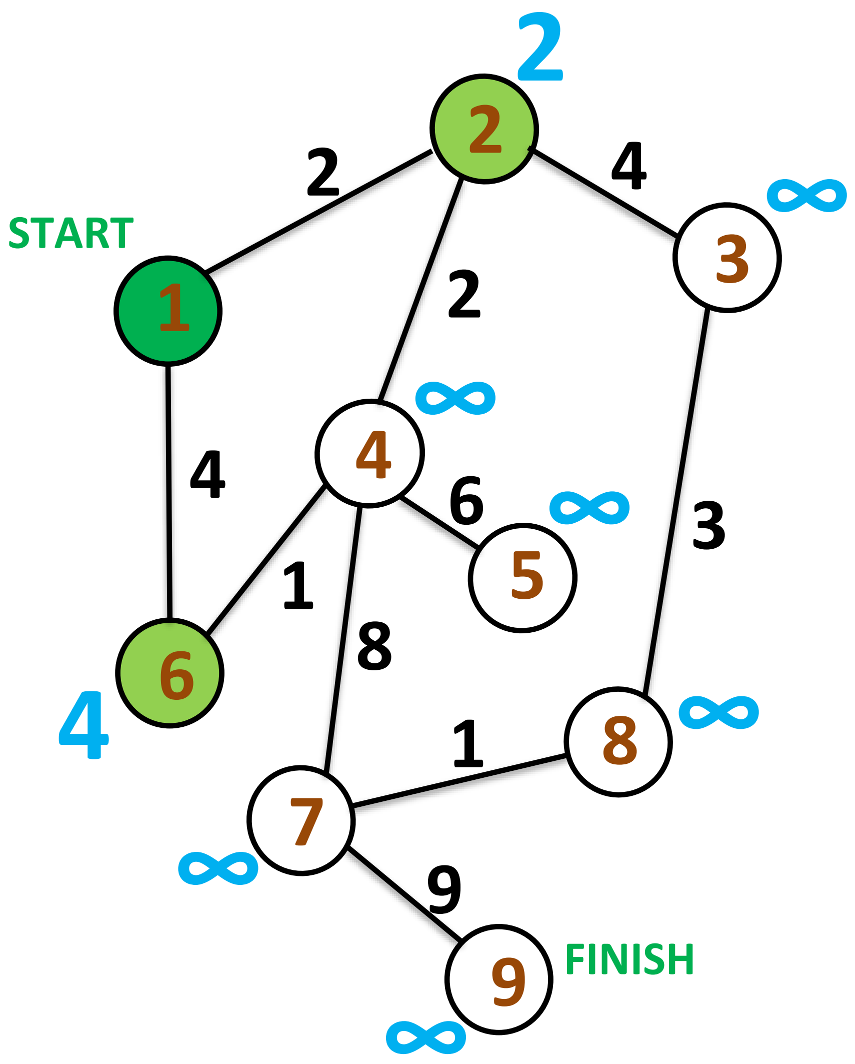
Алгоритм Дейкстры (1)



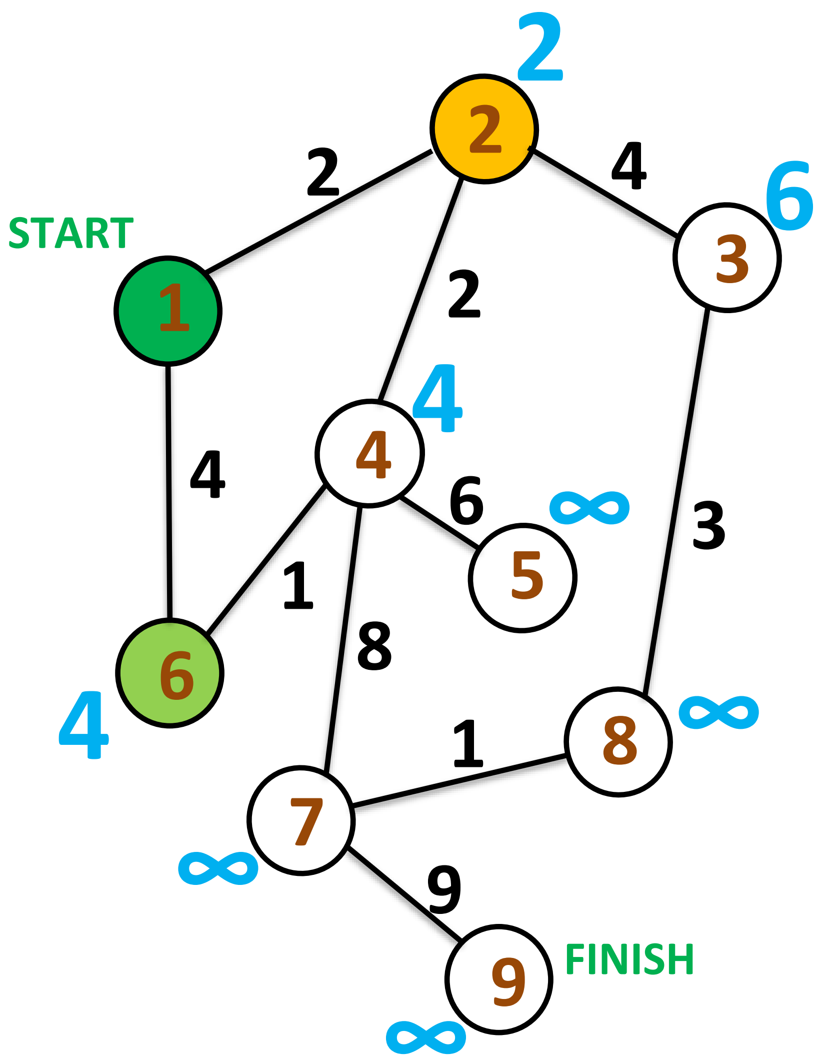
Алгоритм Дейкстры (2)



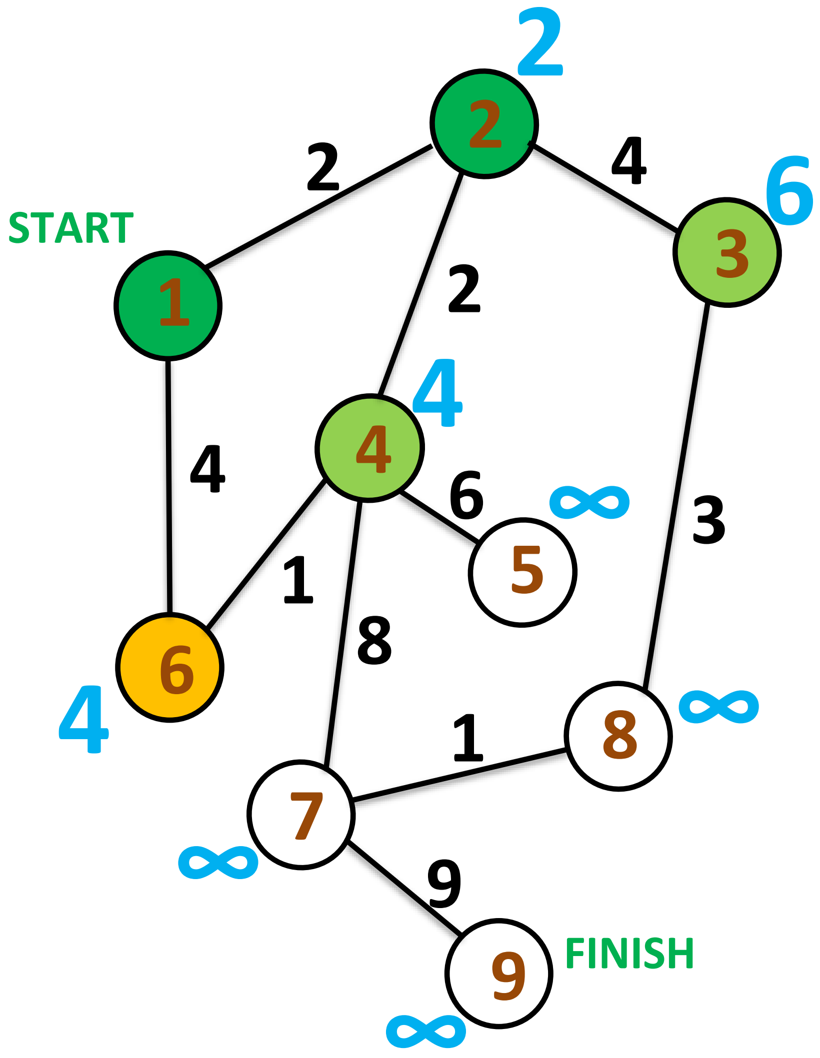
Алгоритм Дейкстры (3)



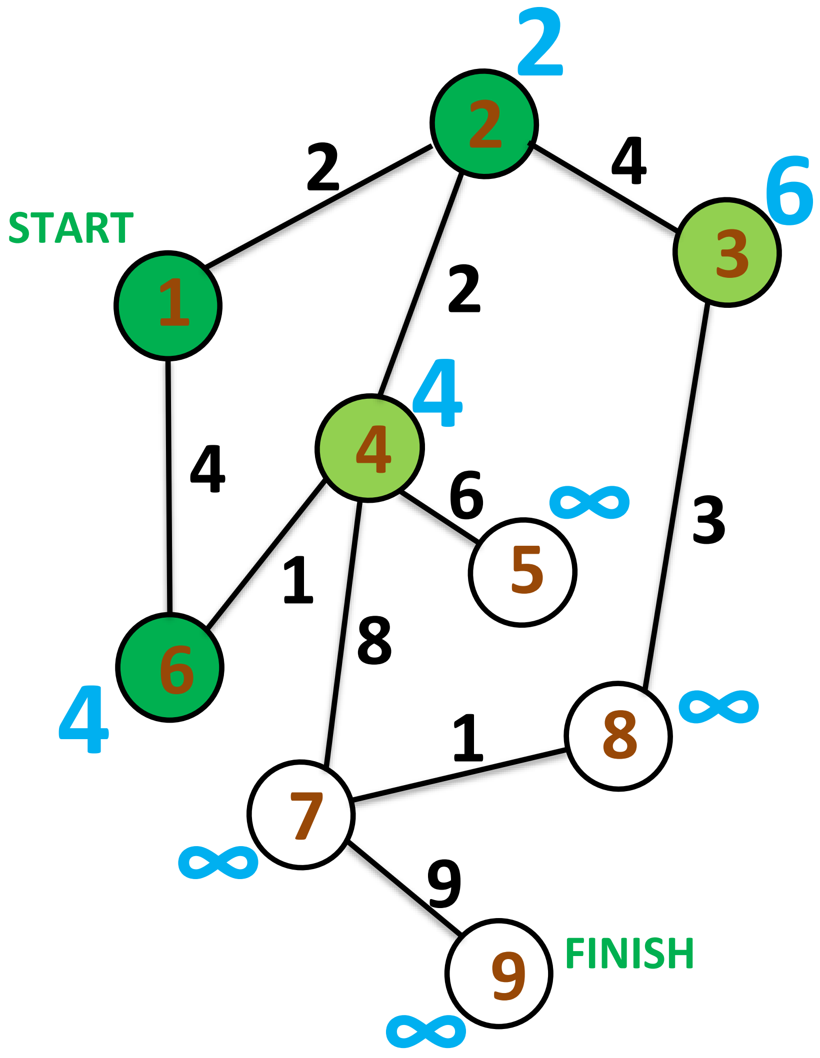
Алгоритм Дейкстры (4)



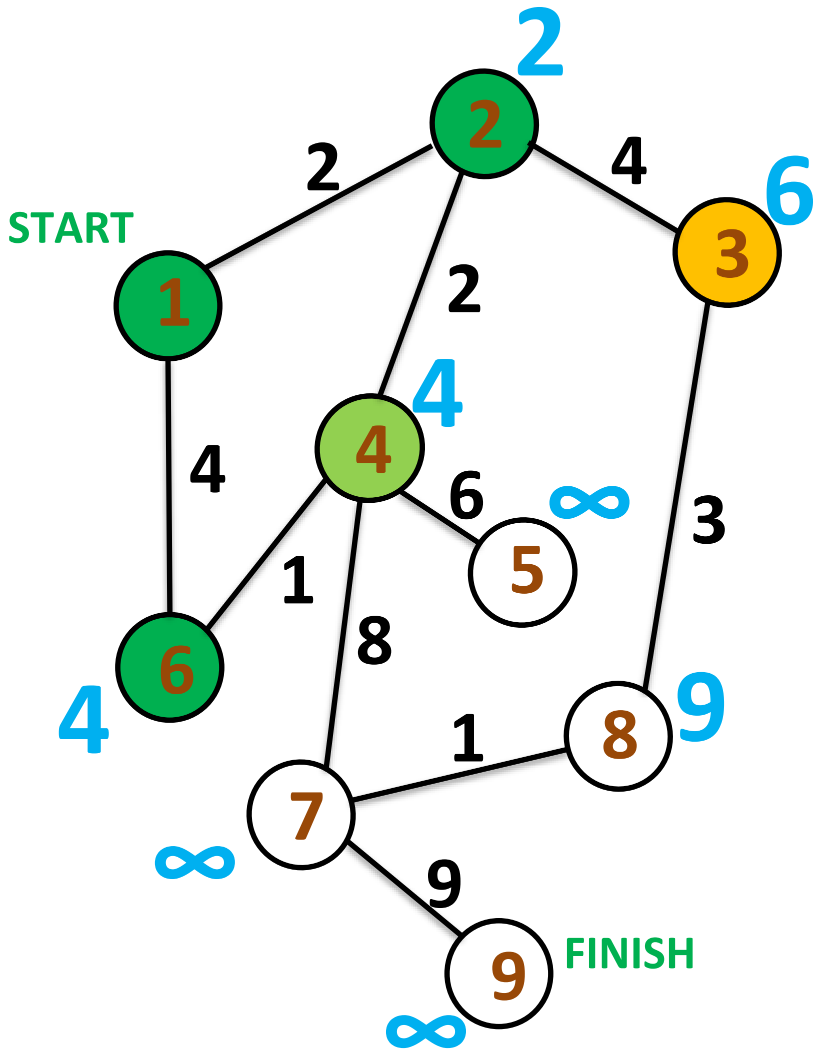
Алгоритм Дейкстры (5)



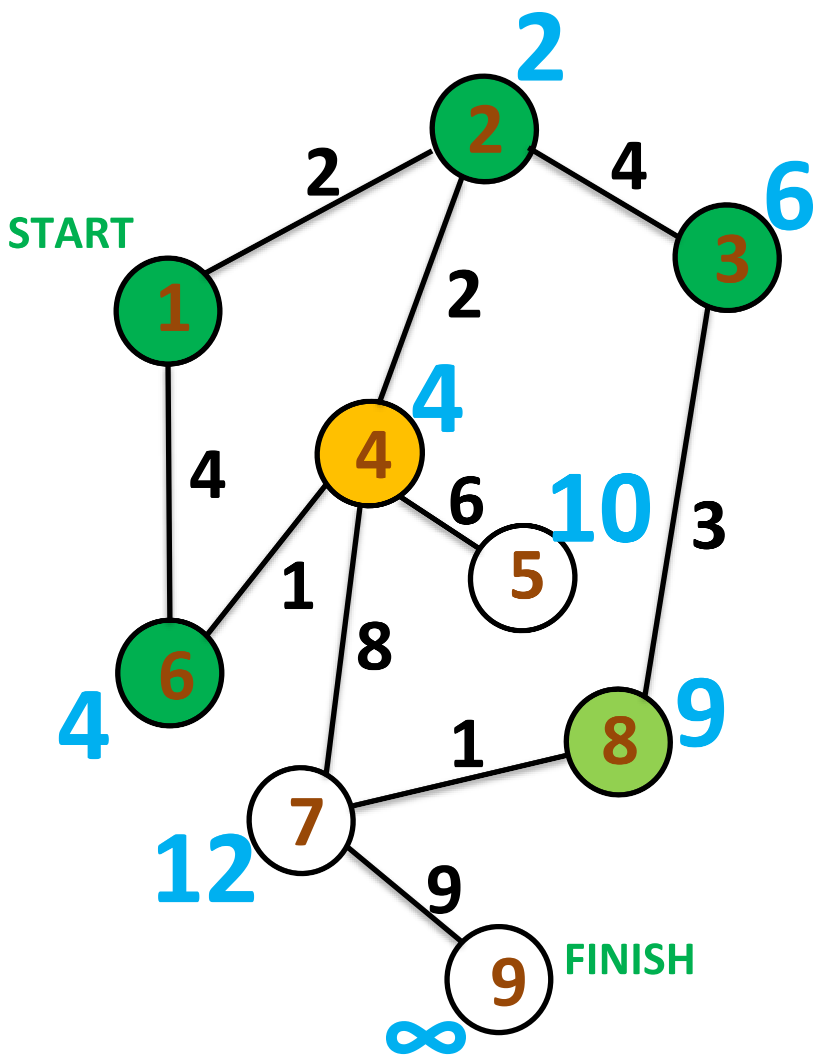
Алгоритм Дейкстры (6)



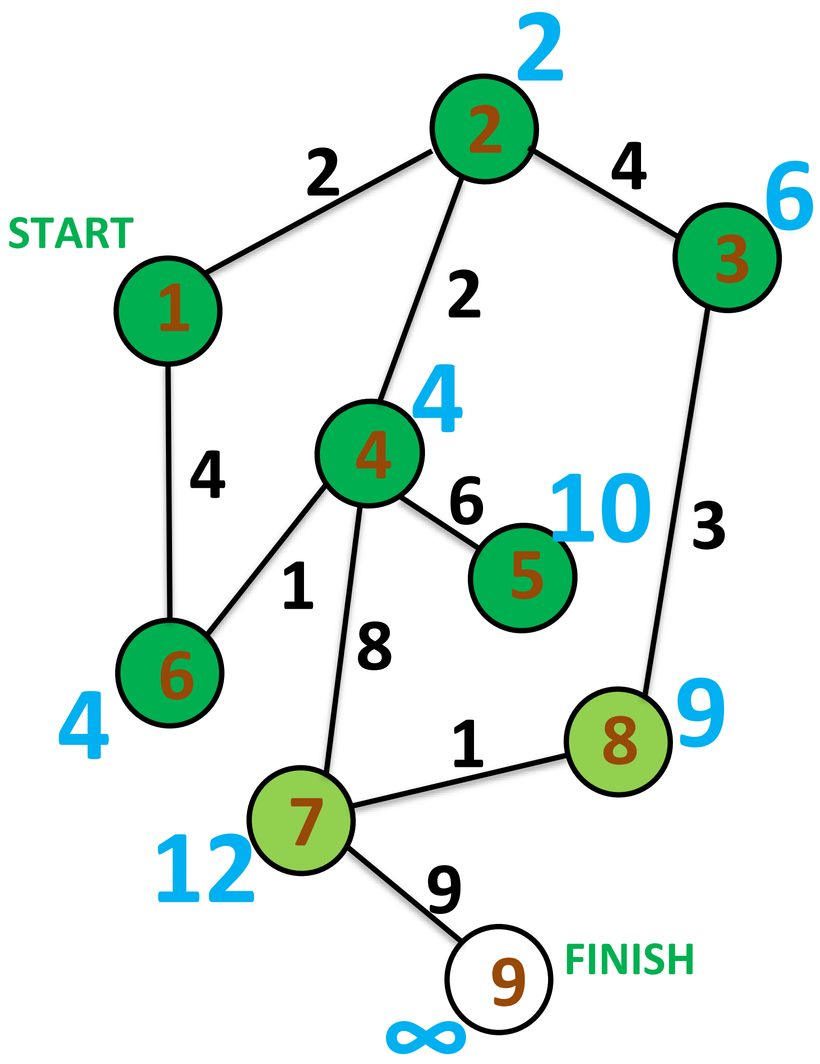
Алгоритм Дейкстры (7)



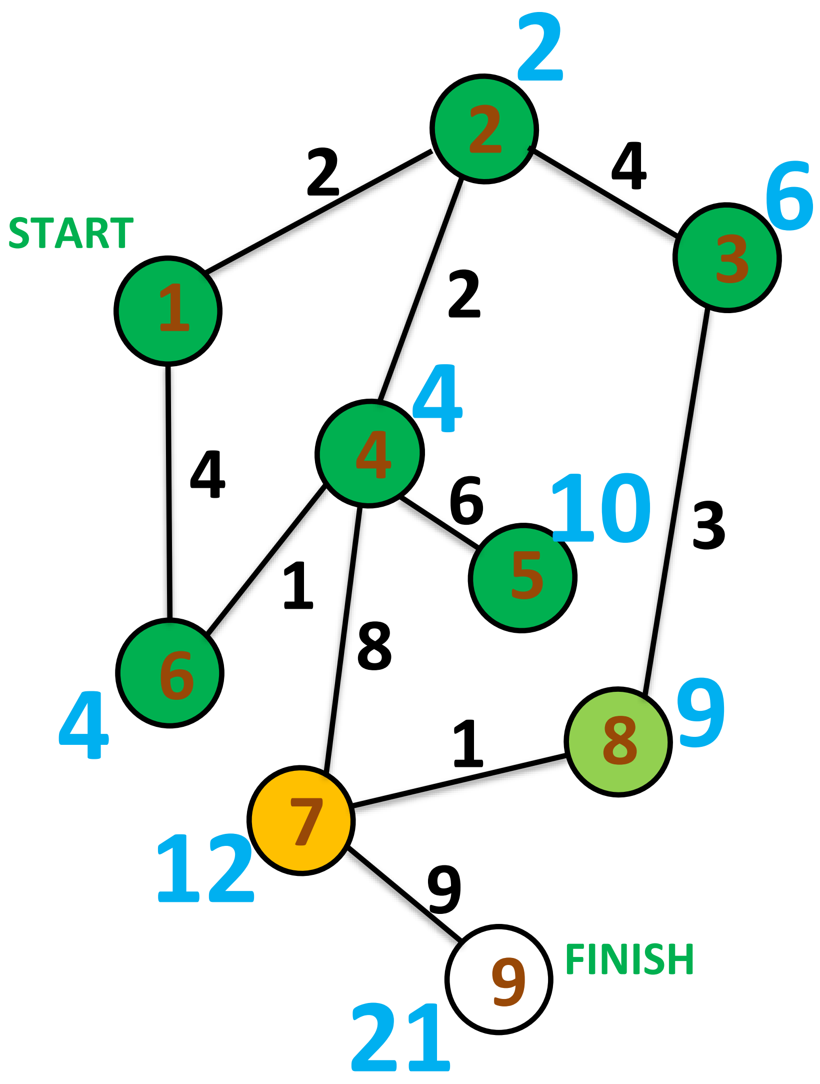
Алгоритм Дейкстры (8)



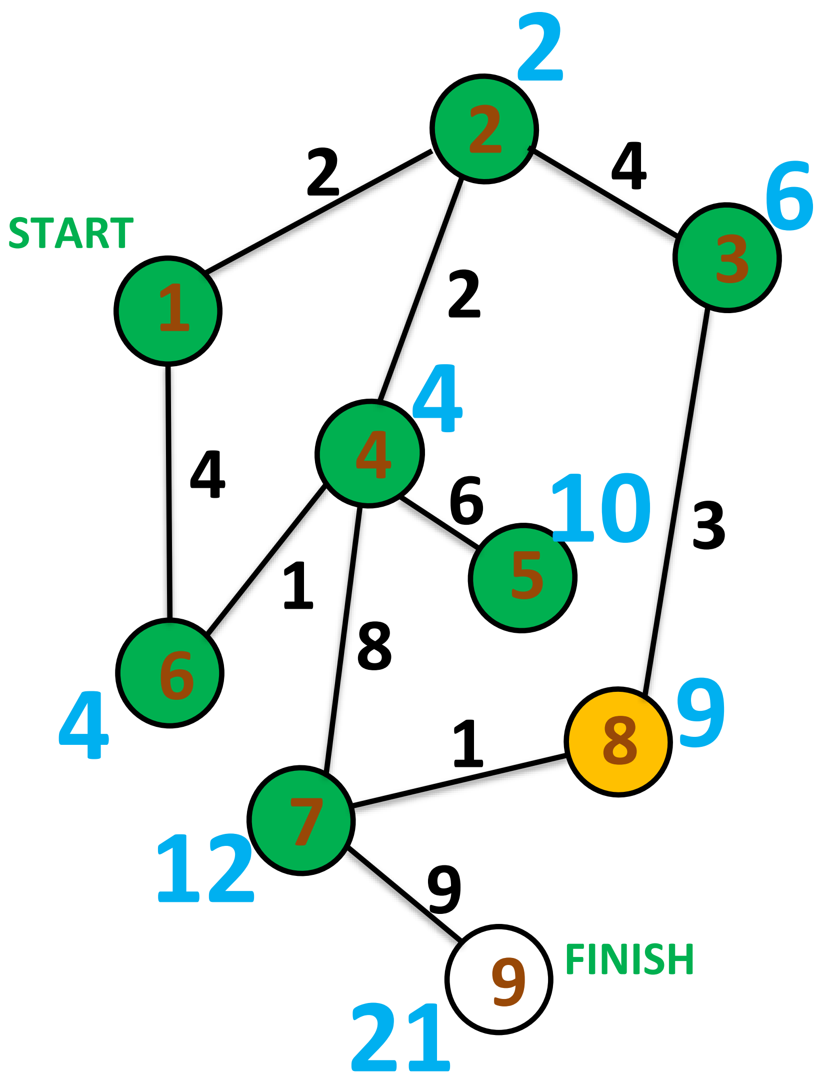
Алгоритм Дейкстры (9)



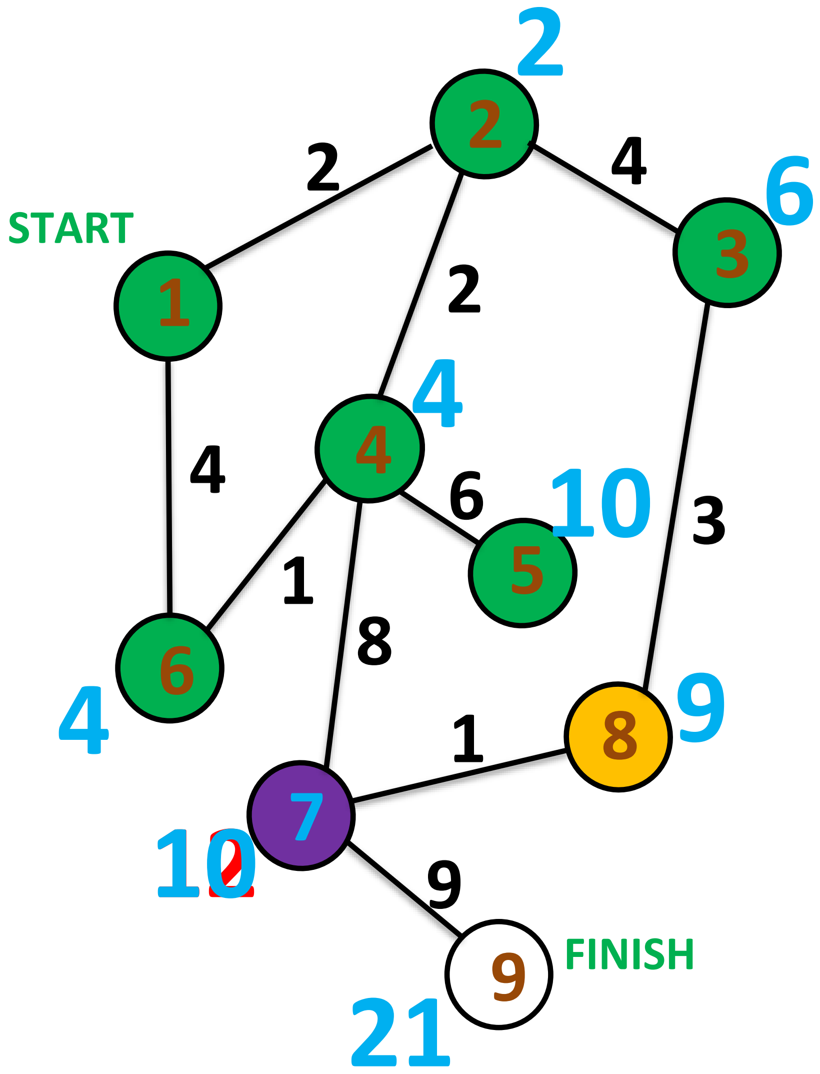
Алгоритм Дейкстры (10)



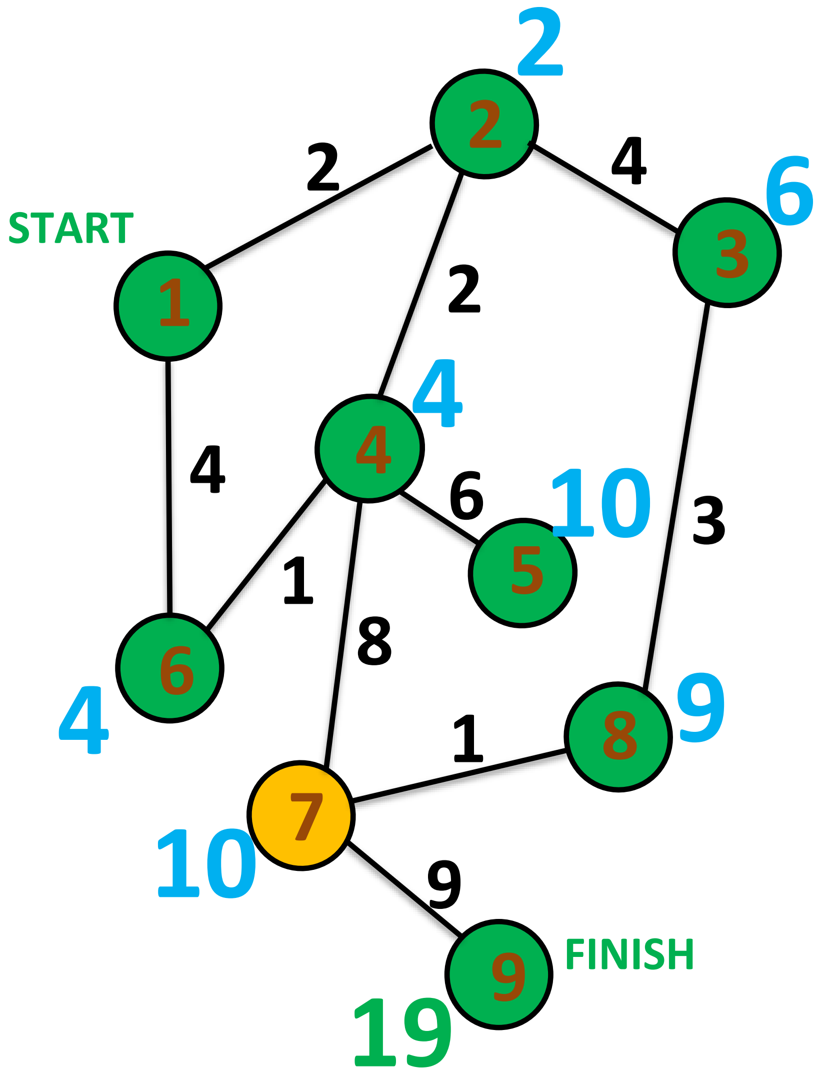
Алгоритм Дейкстры (11)



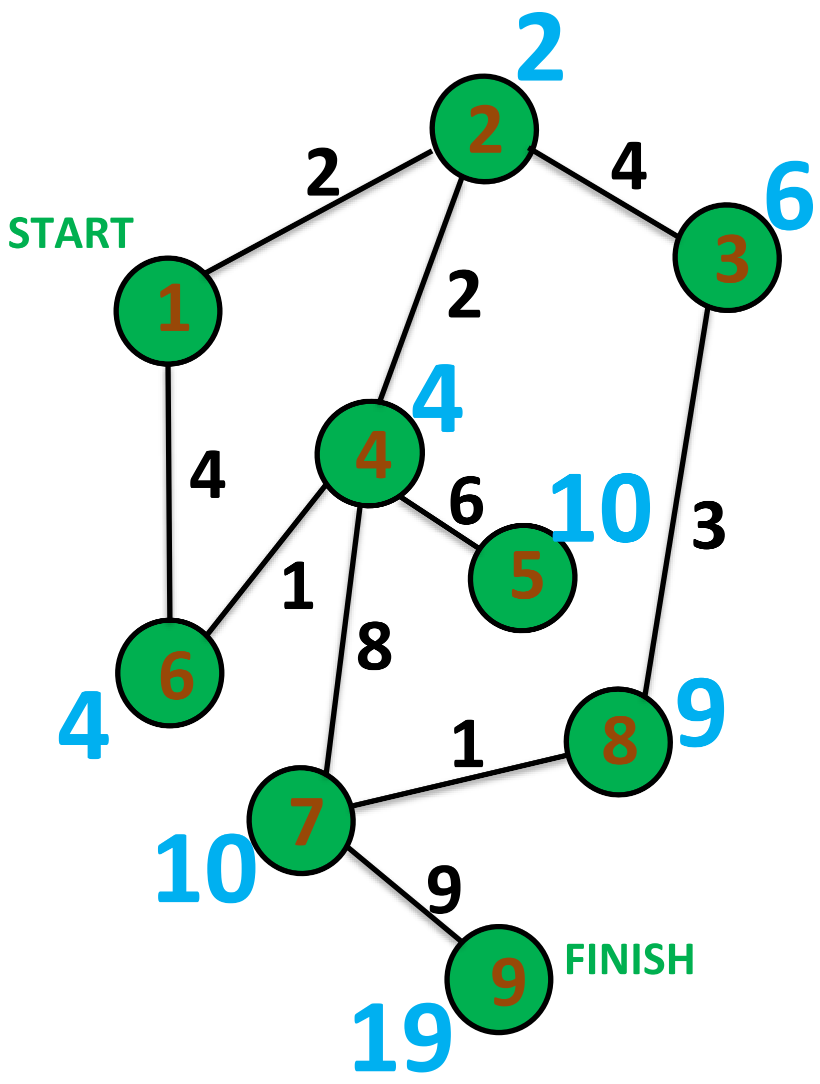
Алгоритм Дейкстры (12)



Алгоритм Дейкстры (13)



Алгоритм Дейкстры (14)

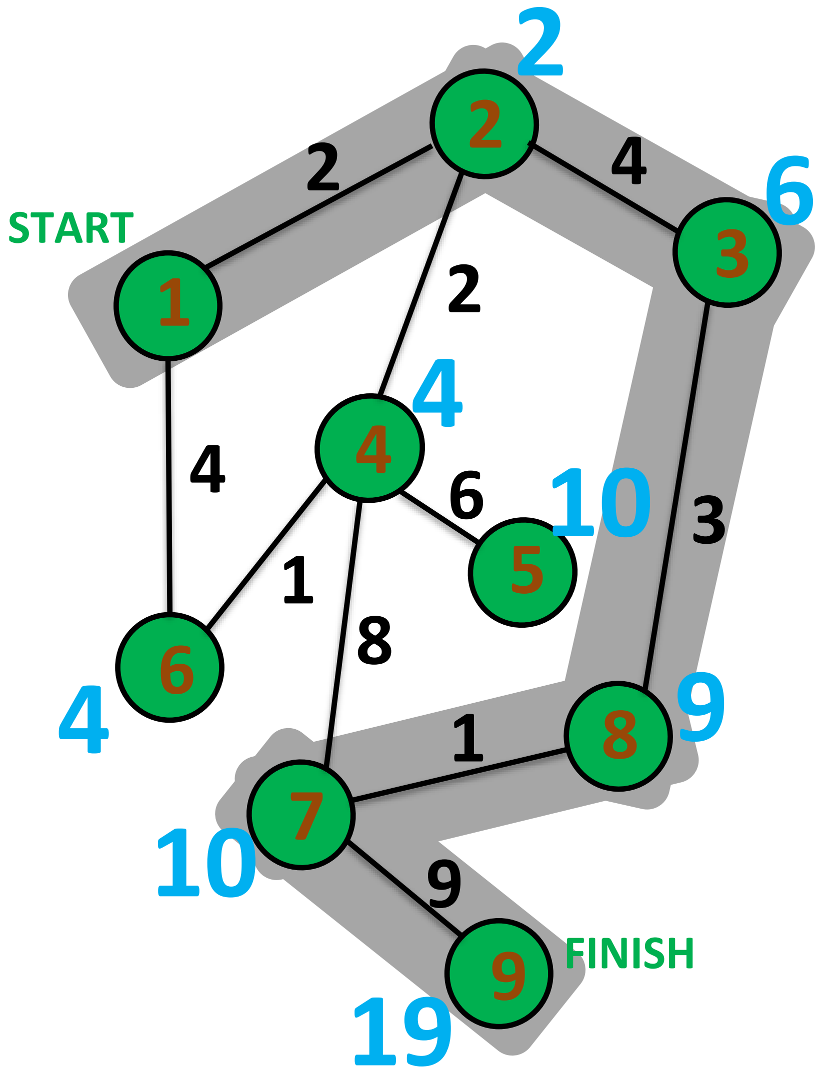


Путь из точки **START**
в точку **FINISH** существует,
самый короткий путь
занимает 19 единиц.

Алгоритм Дейкстры: поиск обратного пути (1)

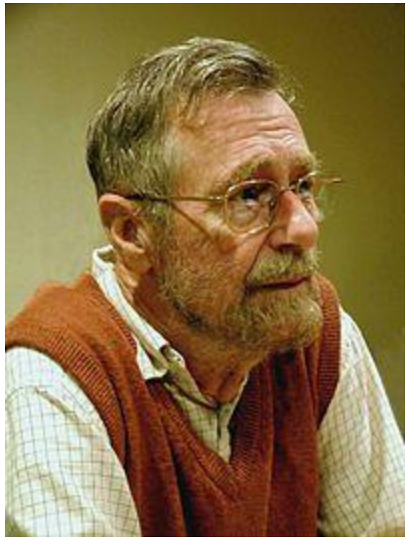
1. каждый узел хранит указатель на вершину, от которой мы пришли к текущей вершине;
2. каждый узел хранит номер (индекс) вершины, от которой мы пришли к текущей вершине;
3. каждый узел хранит массив вершин, по которым мы прошли по пути до этой вершины.

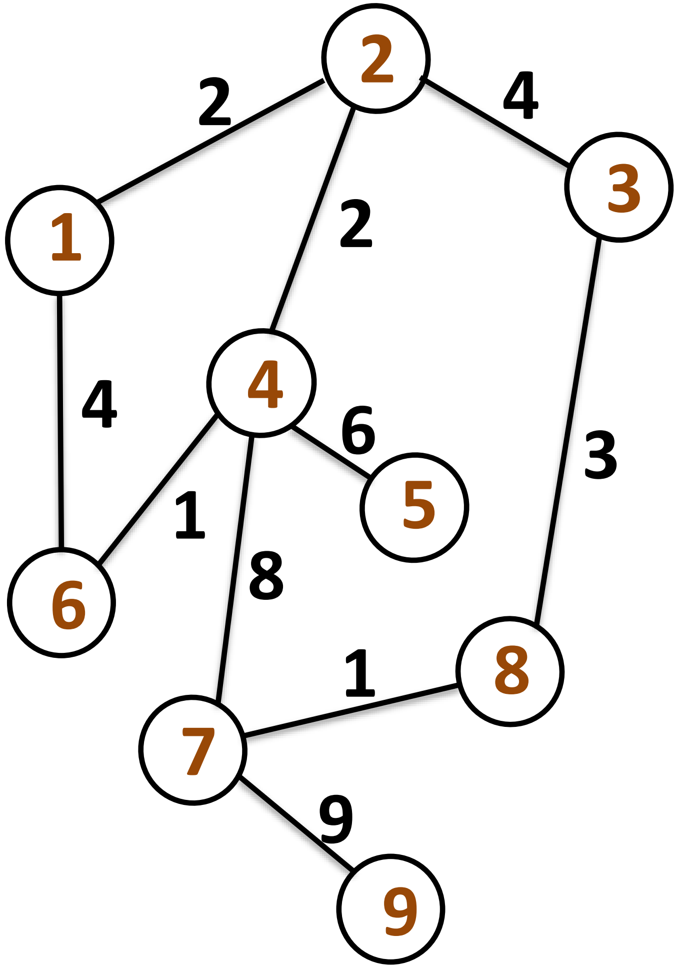
Алгоритм Дейкстры: поиск обратного пути (2)



Алгоритм Дейкстры: обход графа «в глубину»

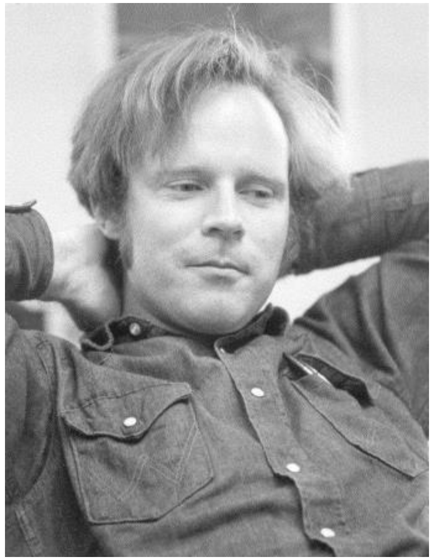
Эдсгер Вибе Дейкстра



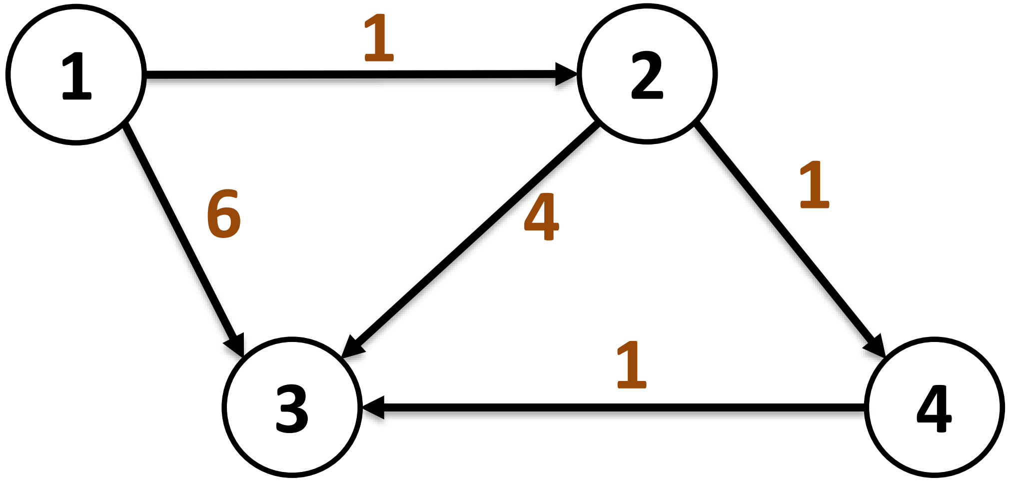


Реализация «в глубину» традиционно отличается наличием рекурсии

Алгоритм Флойда: обход орграфа «в ширину» (1)



Роберт Флойд



Шаг 1: заполняем весами матрицу смежности

	1	2	3	4
1		1	6	∞
2	∞		4	1
3	∞	∞		∞
4	∞	∞	1	

Алгоритм Флойда: обход орграфа «в ширину» (2)

	1	2	3	4
1		1	6	∞
2	∞		4	1
3	∞	∞		∞
4	∞	∞	1	

Шаг 2: запускается итерационный процесс по k от 1 до N, на каждой итерации которого элементы матрицы обновляются согласно формуле:

$$A_k[i,j] = \min(A_{k-1}[i,j], A_{k-1}[i,k] + A_{k-1}[k,j])$$

Алгоритм Флойда: пример (1)

A_0

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_1

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

Первая итерация, k = 1:

$$A_1[i,j] = \min(A_0[i,j], A_0[i,1] + A_0[1,j])$$

min (1 , 0 + 1)

Алгоритм Флойда: пример (2)

A_0

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_1

	1	2	3	4
1	0	**1**	**6**	
2		0		
3			0	
4				0

Первая итерация, k = 1:

$$A_1[i,j] = \min(A_0[i,j], A_0[i,1] + A_0[1,j])$$

min (6 , 0 + 6)

Алгоритм Флойда: пример (3)

A_0

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_1

	1	2	3	4
1	0	1	6	∞
2		0		
3			0	
4				0

Первая итерация, k = 1:

$$A_1[i,j] = \min(A_0[i,j], A_0[i,1] + A_0[1,j])$$

$$\min(\infty, 0 + \infty)$$

Алгоритм Флойда: пример (4)

A_0

	1 (j)	2	3	4
1	0	1	6	∞
i 2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_1

	1	2	3	4
1	0	1	6	∞
2	∞	0		
3			0	
4				0

Первая итерация, k = 1:

$$A_1[i,j] = \min(A_0[i,j], A_0[i,1] + A_0[1,j])$$

$$\min(\infty, \infty + 0)$$

Алгоритм Флойда: пример (5)

A_0

	1	2	3 (j)	4
1	0	1	6	∞
i 2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_1

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	
3			0	
4				0

Первая итерация, k = 1:

$$A_1[i,j] = \min(A_0[i,j], A_0[i,1] + A_0[1,j])$$

min (4 , ∞ + 6)

Алгоритм Флойда: пример (6)

Результат прохода по матрице A при k = 1

A_0

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_1

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

$$A_1[i,j] = \min(A_0[i,j], A_0[i,1] + A_0[1,j])$$

Алгоритм Флойда: пример (7)

A_1

	1	2 j	3	4
i 1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_2

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

Вторая итерация, k = 2:

$$A_2[i,j] = \min(A_1[i,j], A_1[i,2] + A_1[2,j])$$

min (1 , 0 + ∞)

Алгоритм Флойда: пример (8)

A_1

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_2

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

Первая итерация, k = 2:

$$A_2[i,j] = \min(A_1[i,j], A_1[i,2] + A_1[2,j])$$

min (6 , 1 + 4)

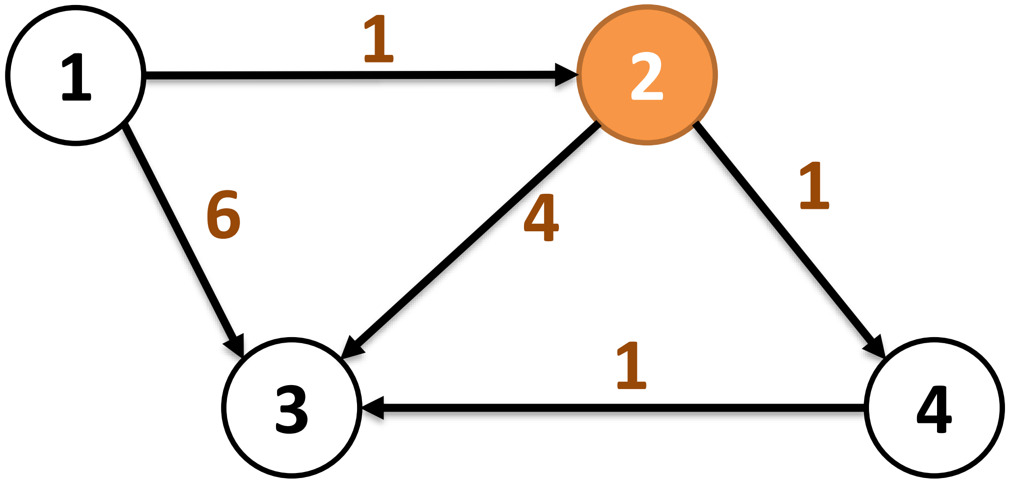
Алгоритм Флойда: пример (9)

A_1

	1	2	3	4
1	0		6	
2		0		
3			0	
4				0

A_2

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



Алгоритм Флойда: пример (10)

A_1

i \ j	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_2

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

Первая итерация, k = 2:

$$A_2[i,j] = \min(A_1[i,j], A_1[i,2] + A_1[2,j])$$

min (∞ , 1 + 1)

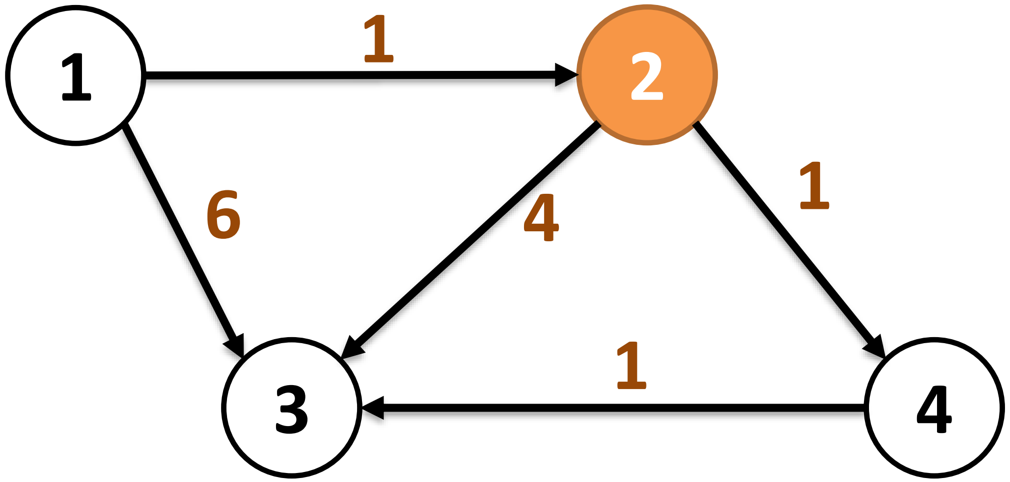
Алгоритм Флойда: пример (11)

A_1

	1	2	3	4
1	0			∞
2		0		
3			0	
4				0

A_2

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



Алгоритм Флойда: пример (12)

Результат прохода по матрице A при k = 2

A_1

	1	2	3	4
1	0	1	6	∞
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_2

	1	2	3	4
1	0	**1**	**5**	**2**
2	**∞**	0	**4**	**1**
3	**∞**	**∞**	0	**∞**
4	**∞**	**∞**	**1**	0

$$A_2[i,j] = \min(A_1[i,j], A_1[i,2] + A_1[2,j])$$

Алгоритм Флойда: пример (13)

Результат прохода по матрице А при k = 3

A_2

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

A_3

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

$$A_3[i,j] = \min(A_2[i,j], A_2[i,3] + A_2[3,j])$$

Алгоритм Флойда: пример (14)

A_3

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

j

i

A_4

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

Первая итерация, k = 4:

$$A_4[i,j] = \min(A_3[i,j], A_3[i,4] + A_3[4,j])$$

min (1 , ∞ + ∞)

Алгоритм Флойда: пример (15)

A_3

j

i	1	2	3	4
1	0	1	5	2
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_4

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

Первая итерация, k = 4:

$$A_4[i,j] = \min(\underbrace{A_3[i,j]}, \underbrace{A_3[i,4]} + \underbrace{A_3[4,j]})$$

min (5 , 2 + 1)

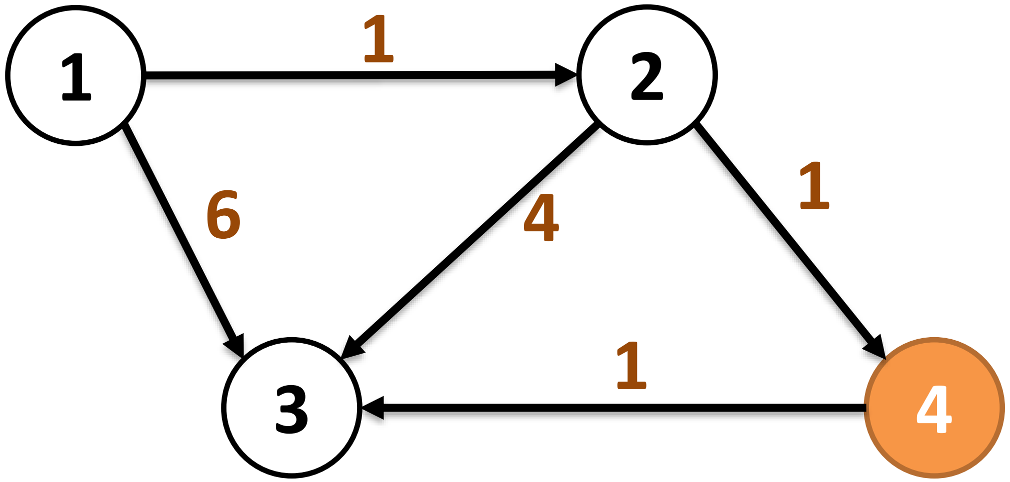
Алгоритм Флойда: пример (16)

A_1

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

A_2

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



Алгоритм Флойда: пример (17)

A_3

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

A_4

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

Первая итерация, k = 4:

$$A_4[i,j] = \min(A_3[i,j], A_3[i,4] + A_3[4,j])$$

min (2 , 2 + 0)

Алгоритм Флойда: пример (18)

A_3

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

j

i

A_4

	1	2	3	4
1	0	**1**	**3**	**2**
2	**∞**	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

Первая итерация, k = 4:

$$A_4[i,j] = \min(A_3[i,j], A_3[i,4] + A_3[4,j])$$

$$\min(\infty, 1 + \infty)$$

Алгоритм Флойда: пример (19)

A_3

j

i

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

A_4

	1	2	3	4
1	0	**1**	**3**	**2**
2	**∞**	0	**2**	1
3	∞	∞	0	∞
4	∞	∞	1	0

Первая итерация, k = 4:

$$A_4[i,j] = \min(A_3[i,j], A_3[i,4] + A_3[4,j])$$

min (4 , 1 + 1)

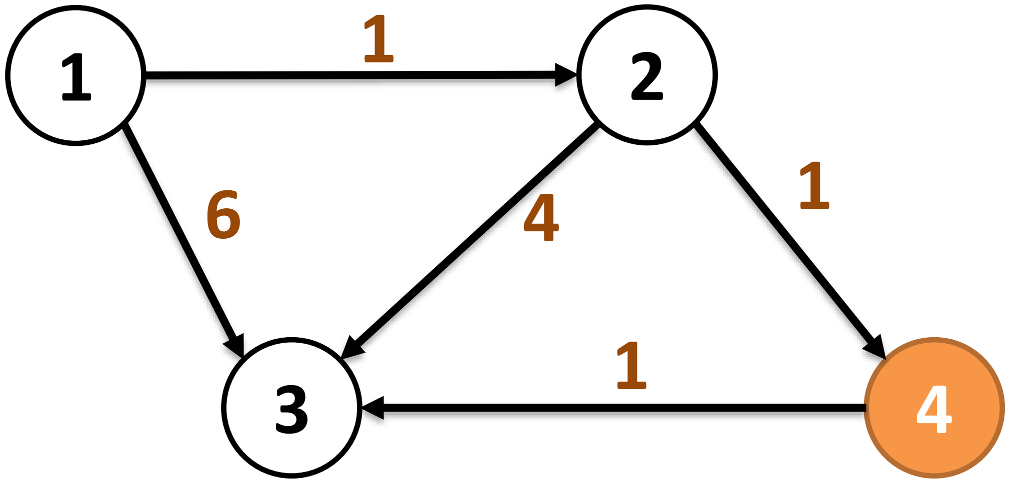
Алгоритм Флойда: пример (20)

A_1

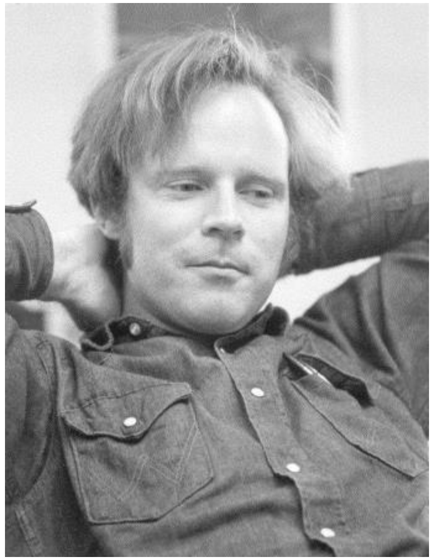
	1	2	3	4
1	0	1	5	2
2	∞	0	4	1
3	∞	∞	0	∞
4	∞	∞	1	0

A_2

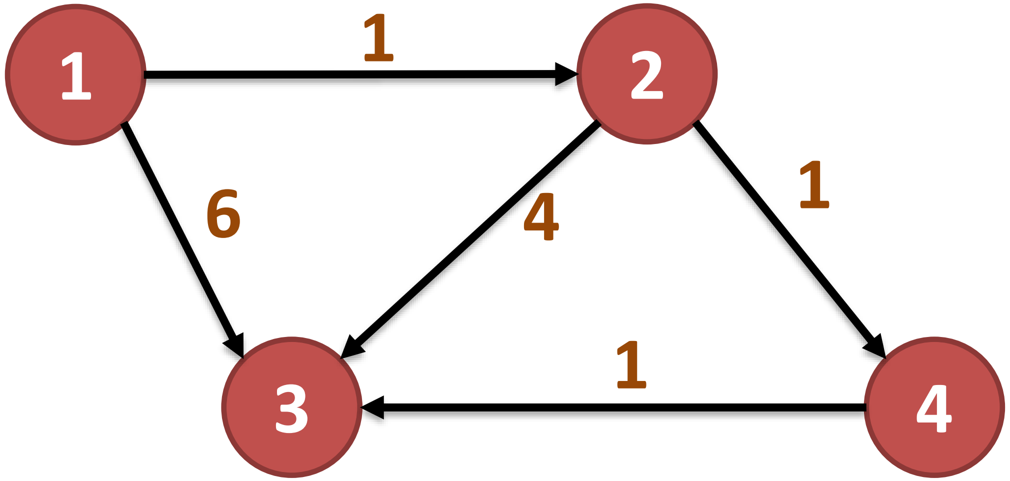
	1	2	3	4
1	0			
2		0	2	
3			0	
4				0



Алгоритм Флойда: обход орграфа «в ширину»



Роберт Флойд



Особенность алгоритма:

после работы алгоритма есть информация о том, за сколько можно дойти **в любую** вершину **из любой** вершины.

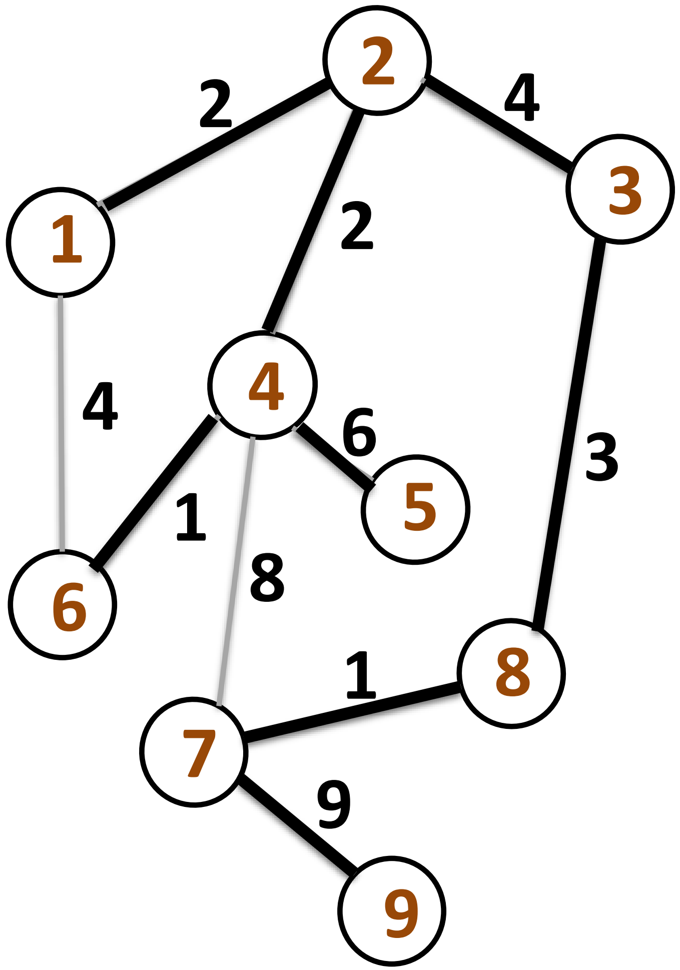
	1	2	3	4
1		1	3	2
2	∞		2	1
3	∞	∞		∞
4	∞	∞	1	

Построение минимального остовного дерева (1)

Алгоритм Краскала позволяет построить минимальное остовное дерево для неориентированного взвешенного графа:

1. выписать таблицу рёбер с их весами;
2. отсортировать таблицу рёбер с их весами по возрастанию;
3. выбрать ребро минимального веса, добавить его к дереву;
4. выбрать минимальное ребро, не дающее цикла в дереве, добавлять его к дереву;
5. повторять пункт 4, пока не дошли до конца списка рёбер.

Построение минимального остовного дерева (3)



Рёбра графа с весами:

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

Отсортированные рёбра графа по весам:

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

Начинаем формировать дерево:

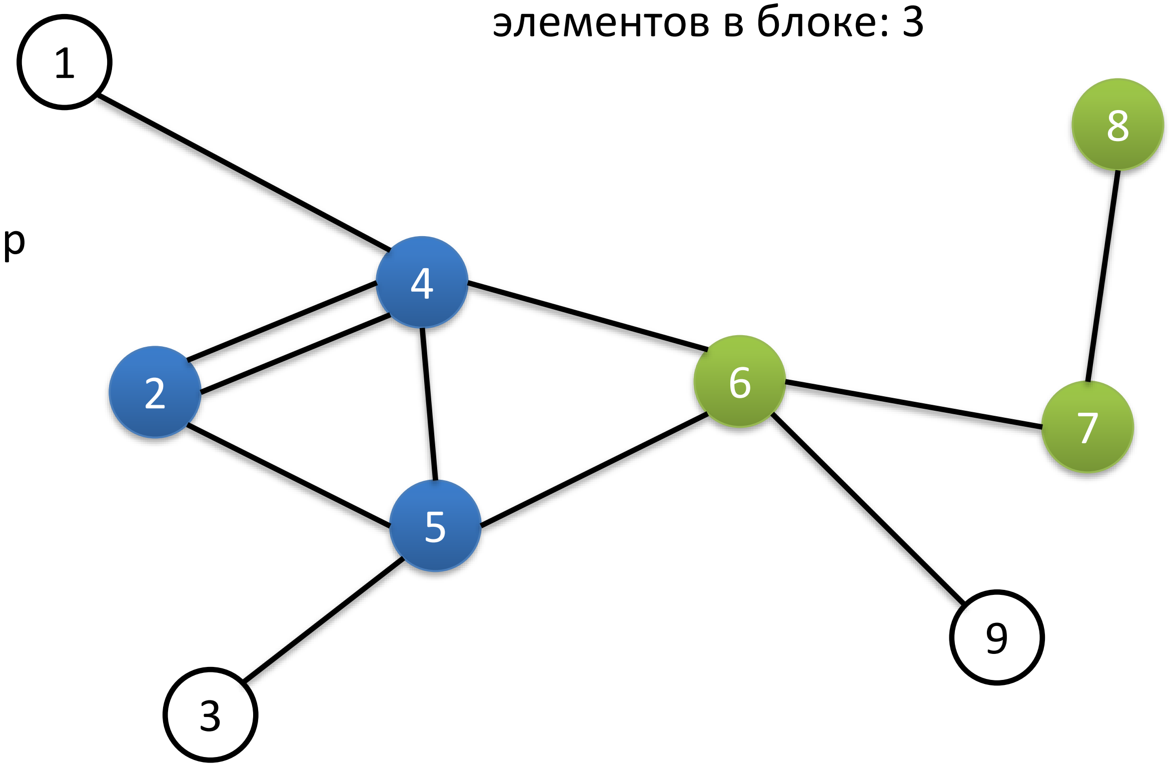
4-6	7-8	1-2	2-4	3-8	1-6	2-3	4-5	4-7	7-9
1	1	2	2	3	4	4	6	8	9

Декомпозиция: базовый алгоритм кластеризации

Критерии декомпозиции:

1. внутренняя связанность блоков должна быть максимальной
2. есть ограничение на размер блоков

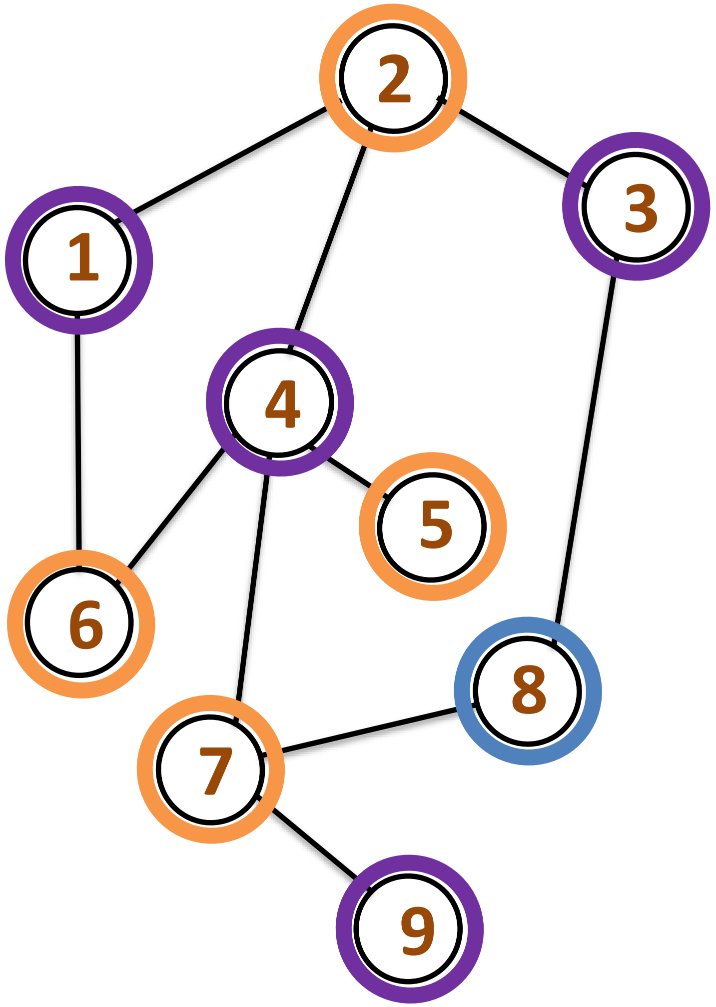
Ограничение на размер элементов в блоке: 3



Шаг 1.
Находим лексикографически элемент с максимальным количеством связей

Шаг 2.
Находим лексикографически элемент, максимально связанный с формируемым блоком

Алгоритм раскраски графа



В один цвет раскрашиваются все несмежные вершины слева направо:

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

Когда нельзя применить цвет к вершинам (мы рассмотрели все вершины), мы выбираем новый цвет и начинаем заново