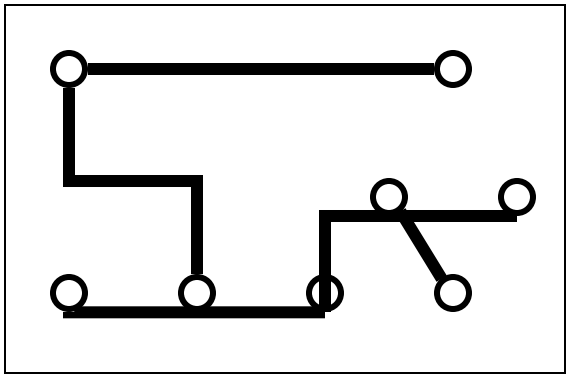
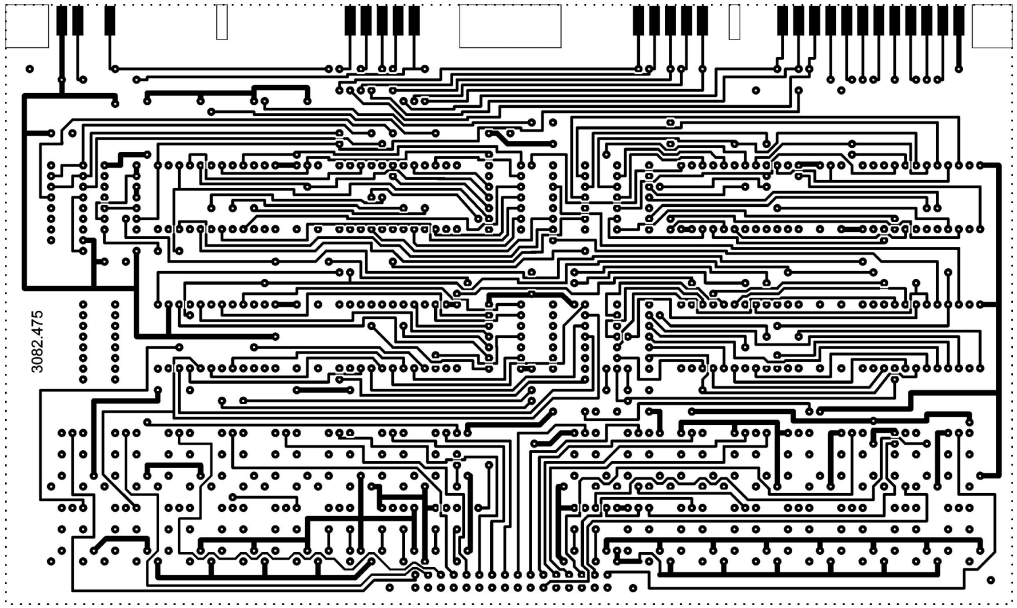
Тема 10. Алгоритмы на графах

Задачи на связность

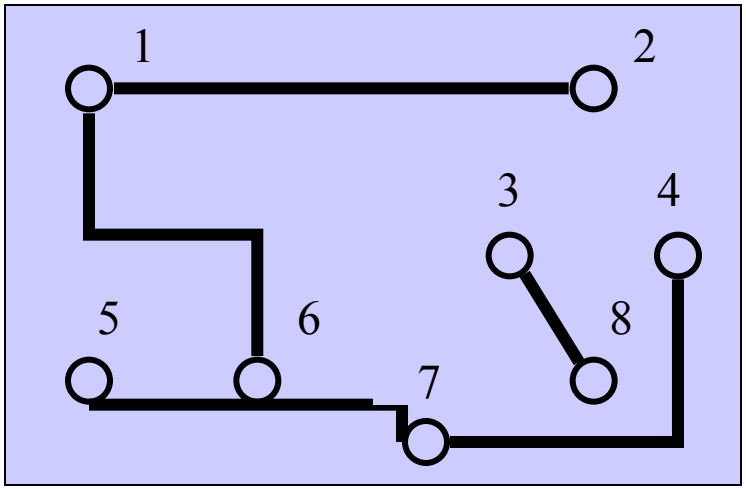
Пример 1. Контроль качества печатных плат





В процессе изготовления печатных плат металлизированные отверстия соединяются проводниками. При контроле продукции проверяется наличие или отсутствие соединения между различными точками.

Алгоритм Уоршалла



C_{ij}	1	2	3	4	5	6	7	8
1		1				1		
2	1							
3								1
4							1	
5							1	
6	1							
7				1	1			
8			1					

$T = C$

Цикл i

Цикл j

Цикл k

$T_{jk} = T_{jk}$ or $(T_{ji}$ and $T_{ik})$

T_{ij}	...
...	

T - матрица транзитивных замыканий.

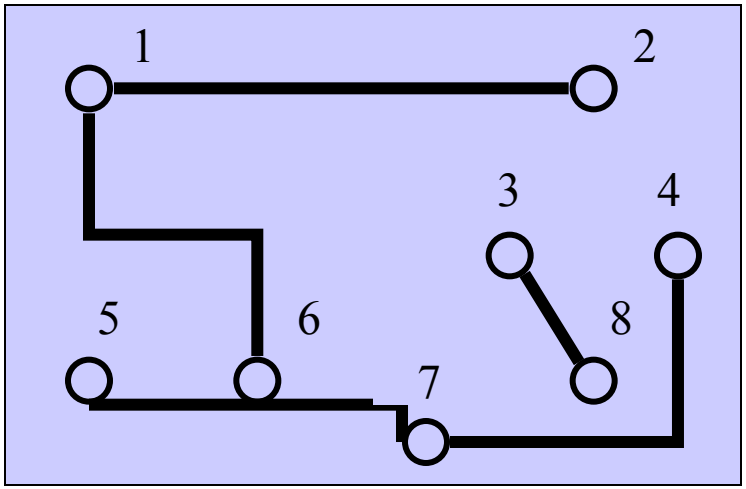
Алгоритм Уоршалла

```
int C[8][8] = {{-1, 1, 0, 0, 0, 1, 0, 0}, ...};
int T[8][8];
int n = 8;

for(int i = 0; i < n; i++)
   for(int j = 0; j < n; j++)
      T[i][j] = C[i][j];

for(int i = 0; i < n; i++)
   for(int j = 0; j < n; j++)
      if(j != i)
         for(int k = 0; k < n; k++)
            if(k != j)
               T[j][k] = T[j][k] || T[j][i] && T[i][k];
```

Алгоритм Уоршалла

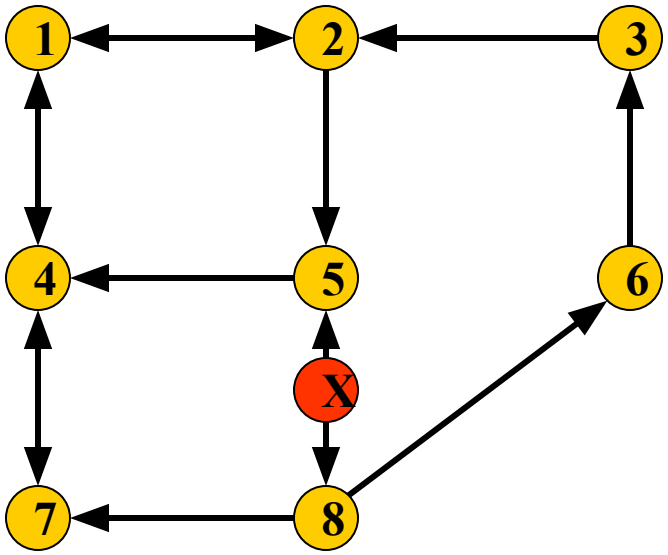


T_{ij}	1	2	3	4	5	6	7	8
1		1				1		
2	1					1		
3								1
4					1		1	
5				1			1	
6	1	1						
7				1	1			
8			1					

```
bool TestConnection(int Node1, int Node2)
{
    return(T[Node1-1][Node2-1] == 1);
}
```

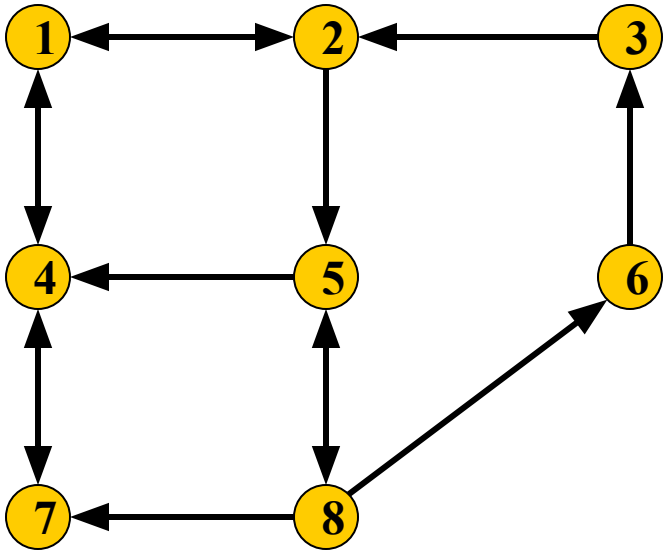
Задачи на связность

Пример 2. Транспортная сеть



Транспортная сеть представлена ориентированным графом, в котором узлы соответствуют населенным пунктам, точкам пересечения улиц или дорог, а дуги - дорогам, связывающим узлы. Требуется проверить возможность путешествия из пункта A в пункт B.

Алгоритм Уоршалла



C_{ij}	1	2	3	4	5	6	7	8
1		1		1				
2	1				1			
3		1						
4	1						1	
5				1				1
6			1					
7				1				
8					1	1	1	

$T = C$

Цикл i

Цикл j

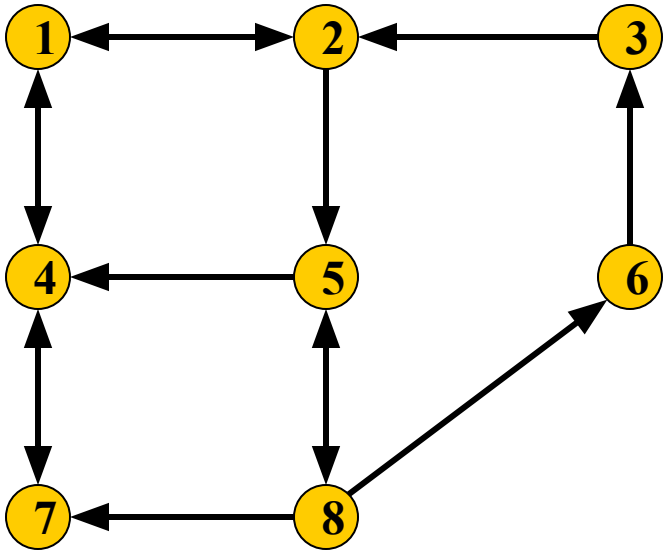
Цикл k

$T_{jk} = T_{jk}\ or\ (T_{ji}\ and\ T_{ik})$

T_{ij}	...
...	

T - матрица транзитивных замыканий.

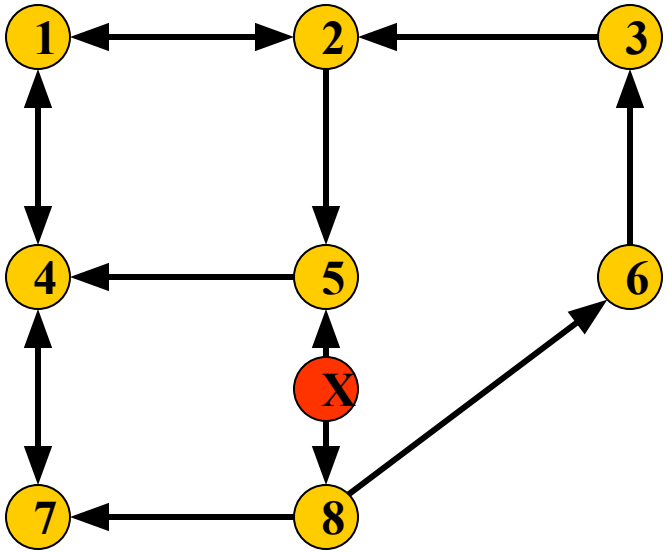
Алгоритм Уоршалла



T_{ij}	1	2	3	4	5	6	7	8
1		1	1	1	1	1	1	1
2	1		1	1	1	1	1	1
3	1	1		1	1	1	1	1
4	1	1	1		1	1	1	1
5	1	1	1	1		1	1	1
6	1	1	1	1	1		1	1
7	1	1	1	1	1	1		1
8	1	1	1	1	1	1	1	

```
bool CanTravel(int A, int B)
{
    return(T[A-1][B-1] == 1);
}
```

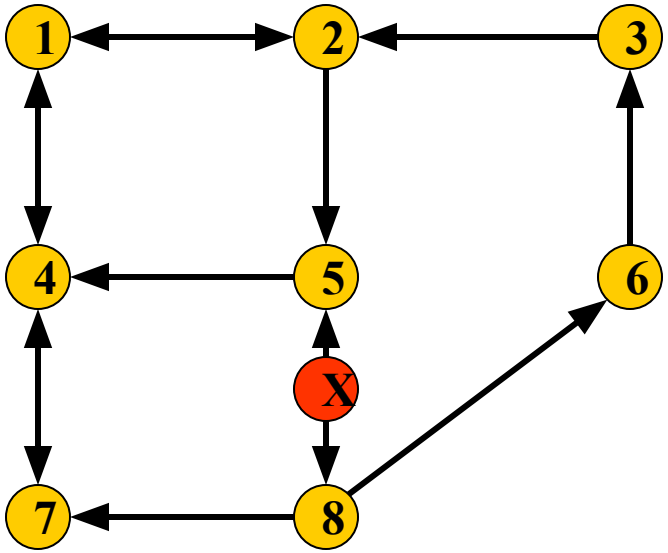
Алгоритм Уоршалла



C_{ij}	1	2	3	4	5	6	7	8
1		1		1				
2	1				1			
3		1						
4	1						1	
5				1				
6			1					
7				1				
8						1	1	

T_{ij}	...
...	

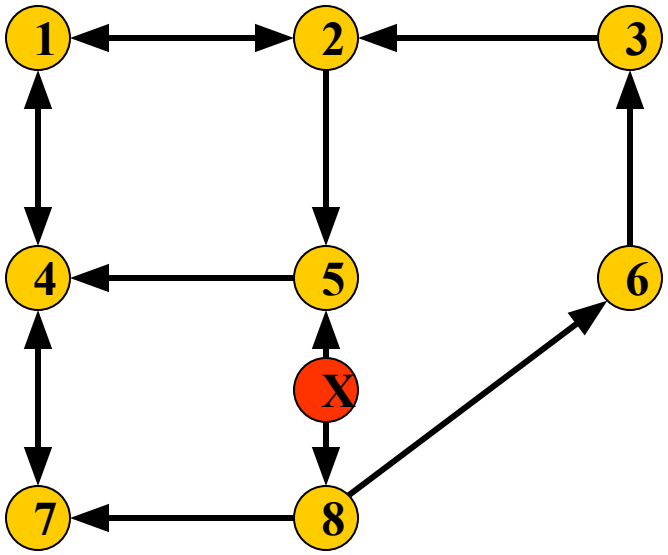
Алгоритм Уоршалла



T_{ij}	1	2	3	4	5	6	7	8
1		1		1	1		1	
2	1			1	1		1	
3	1	1		1	1		1	
4	1	1			1		1	
5	1	1		1			1	
6	1	1	1	1	1		1	
7	1	1		1	1			
8	1	1	1	1	1	1	1	

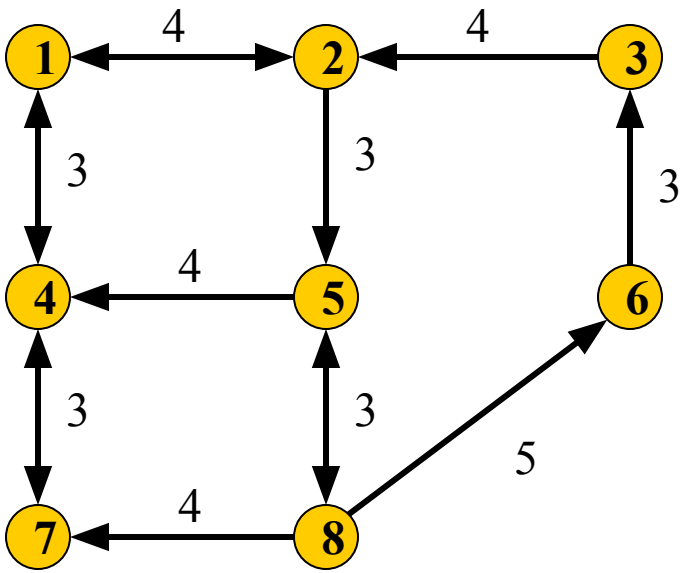
Задачи на поиск минимального пути

Пример. Транспортная сеть



Транспортная сеть представлена ориентированным графом, в котором узлы соответствуют населенным пунктам, точкам пересечения улиц или дорог, а дуги - дорогам, связывающим узлы. Требуется найти минимальный путь из пункта A в пункт B.

Алгоритм Флойда



C_{ij}	1	2	3	4	5	6	7	8
1		4	∞	3	∞	∞	∞	∞
2	4		∞	∞	3	∞	∞	∞
3	∞	4		∞	∞	∞	∞	∞
4	3	∞	∞		∞	∞	3	∞
5	∞	∞	∞	4		∞	∞	3
6	∞	∞	3	∞	∞		∞	∞
7	∞	∞	∞	3	∞	∞		∞
8	∞	∞	∞	∞	3	5	4	

$T = C$

Цикл i

Цикл j

Цикл k

$$T_{jk} = \min(T_{jk},\ T_{ji} + T_{ik})$$

H_{ij}	...
...	

T_{ij}	...
...	

T - матрица минимальных расстояний,

H - матрица путей.

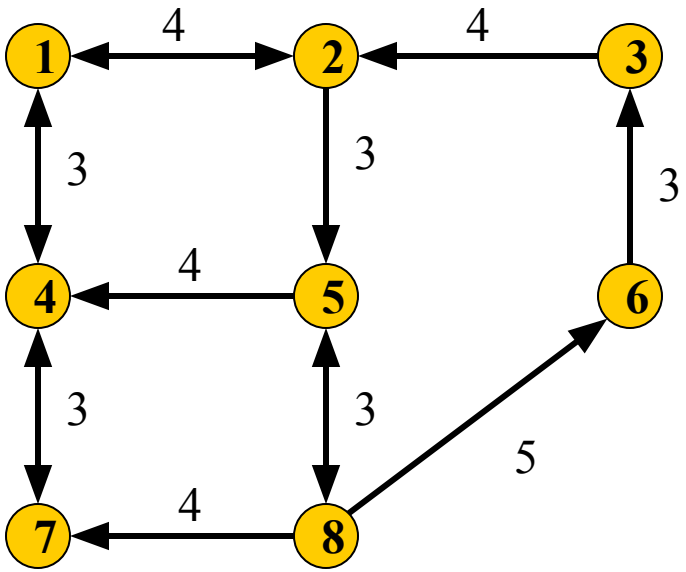
Алгоритм Флойда

```
const int INF 1000000
int C[8][8] = {{INF, 4, INF, 3, INF, INF, INF, INF}, ...};
int T[8][8];
int H[8][8];
const int n = 8;
for(int i = 0; i < n; i++)
   for(int j = 0; j < n; j++)
   {
      T[i][j] = C[i][j];
      if(C[i][j] == INF)
         H[i][j] = -1;
      else
         H[i][j] = j;
   }
```

Алгоритм Флойда

```
for(int i = 0; i < n; i++)
   for(int j = 0; j < n; j++)
      if(j != i)
         for(int k = 0; k < n; k++)
            if(k != j)
               if(T[j][i]+T[i][k] < T[j][k])
               {
                  H[j][k] = H[j][i];
                  T[j][k] = T[j][i]+T[i][k];
               }
```

Алгоритм Флойда

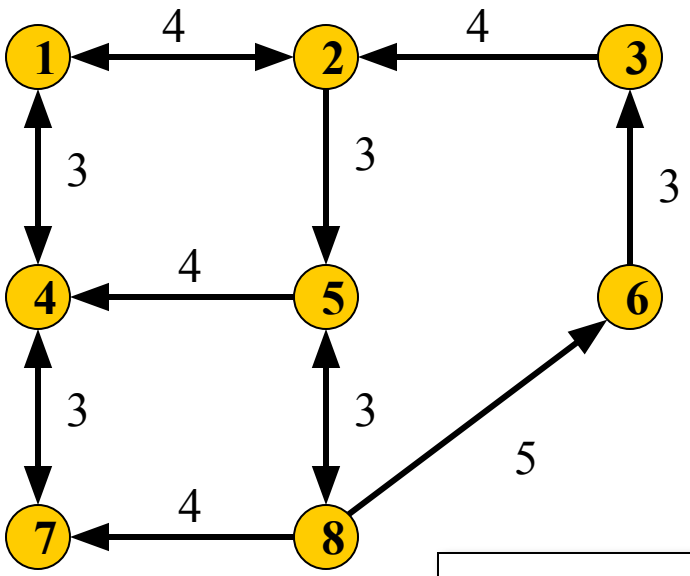



```
int Distance(int A, int B)
{
   return(T[A-1][B-1]);
}
```

T_{ij}	1	2	3	4	5	6	7	8
1		4	18	3	7	15	6	10
2	4		14	7	3	11	10	6
3	8	4		11	7	15	14	10
4	3	7	21		10	18	3	13
5	7	11	11	4		8	7	3
6	11	7	3	14	10		17	13
7	6	10	24	3	13	21		16
8	10	12	8	7	3	5	4	

H_{ij}	1	2	3	4	5	6	7	8
1		1	6	1	2	8	4	5
2	2		6	5	2	8	8	5
3	2	3		5	2	8	8	5
4	4	1	6		2	8	4	5
5	4	1	6	5		8	8	5
6	2	3	6	5	2		8	5
7	4	1	6	7	2	8		5
8	4	3	6	7	8	8	8	

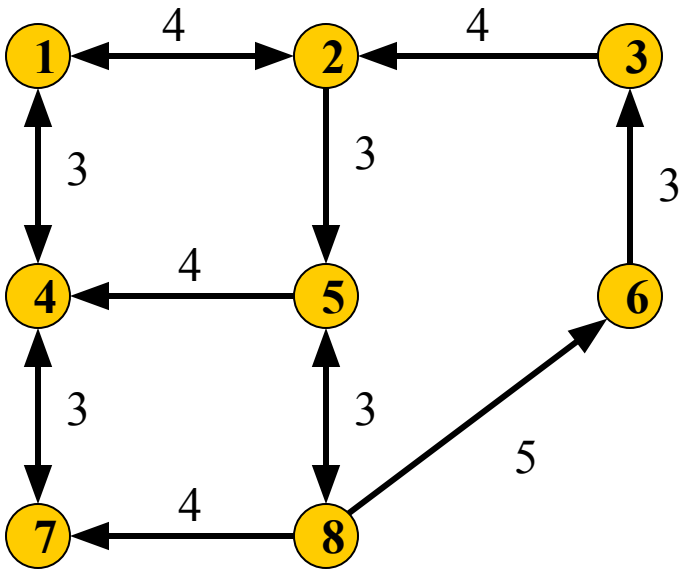
Алгоритм Флойда

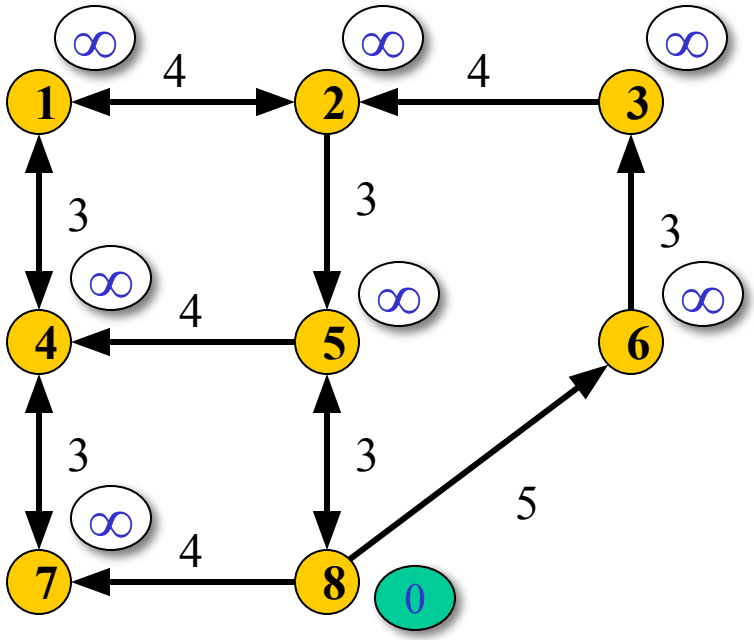


H_{ij}	1	2	3	4	5	6	7	8
1		1	6	1	2	8	4	5
2	2		6	5	2	8	8	5
3	2	3		5	2	8	8	5
4	4	1	6		2	8	4	5
5	4	1	6	5		8	8	5
6	2	3	6	5	2		8	5
7	4	1	6	7	2	8		5
8	4	3	6	7	8	8	8	

```
void Path(int A, int B, int P[], int &N)
{
   N = 0;
   P[N++] = H[A-1][B-1];
   while(P[N-1] != A)
      P[N++] = H[A-1][P[N-1]];
}
```

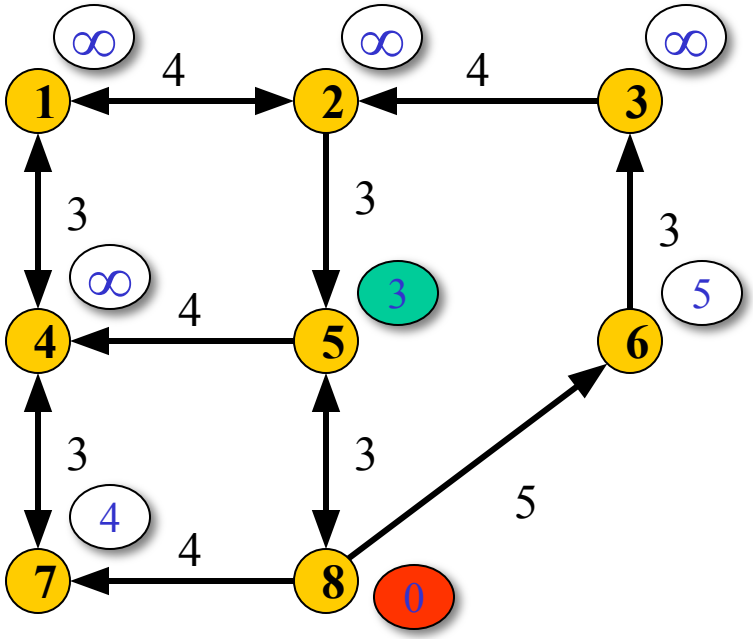
Алгоритм Дейкстры

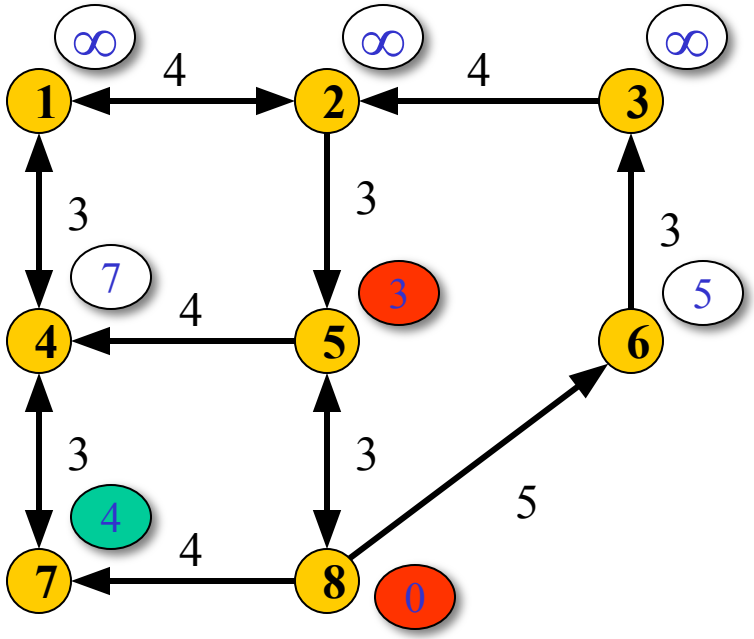




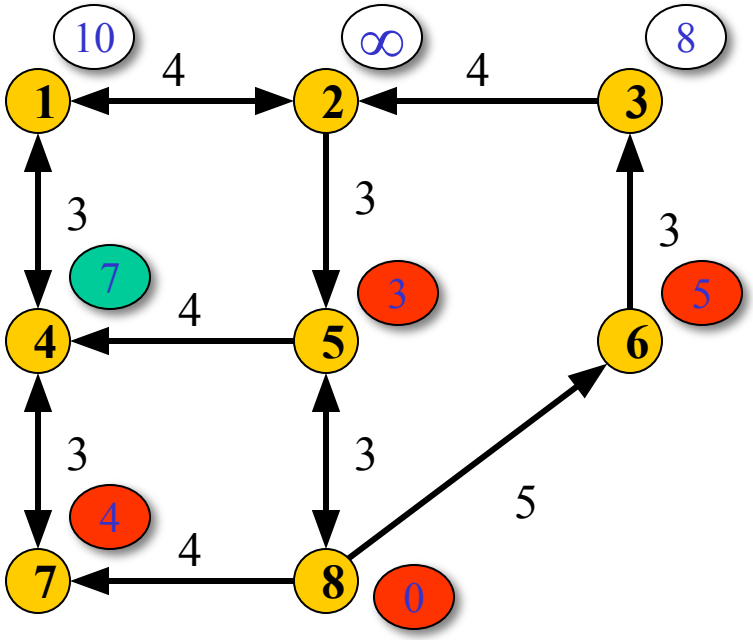
Алгоритм Дейкстры (1959 г.) позволяет найти минимальные расстояния от заданной вершины до всех остальных вершин графа. Граф не должен иметь отрицательных циклов. Применяется в технологиях маршрутизации, например в протоколах OSPF и IS-IS.

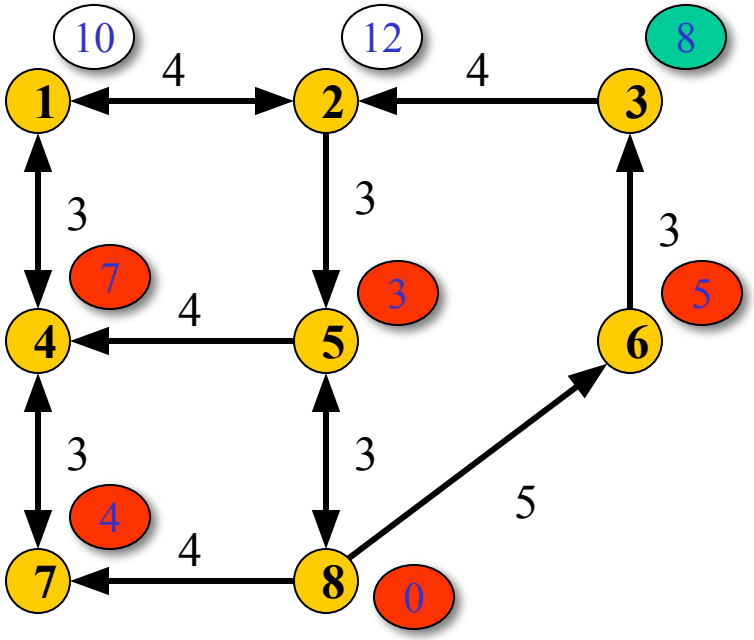
Алгоритм Дейкстры





Алгоритм Дейкстры





Алгоритм Дейкстры