

**КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ**

Кратчайший путь

Старший преподаватель
кафедры теоретической кибернетики
Хадиев Р.М.

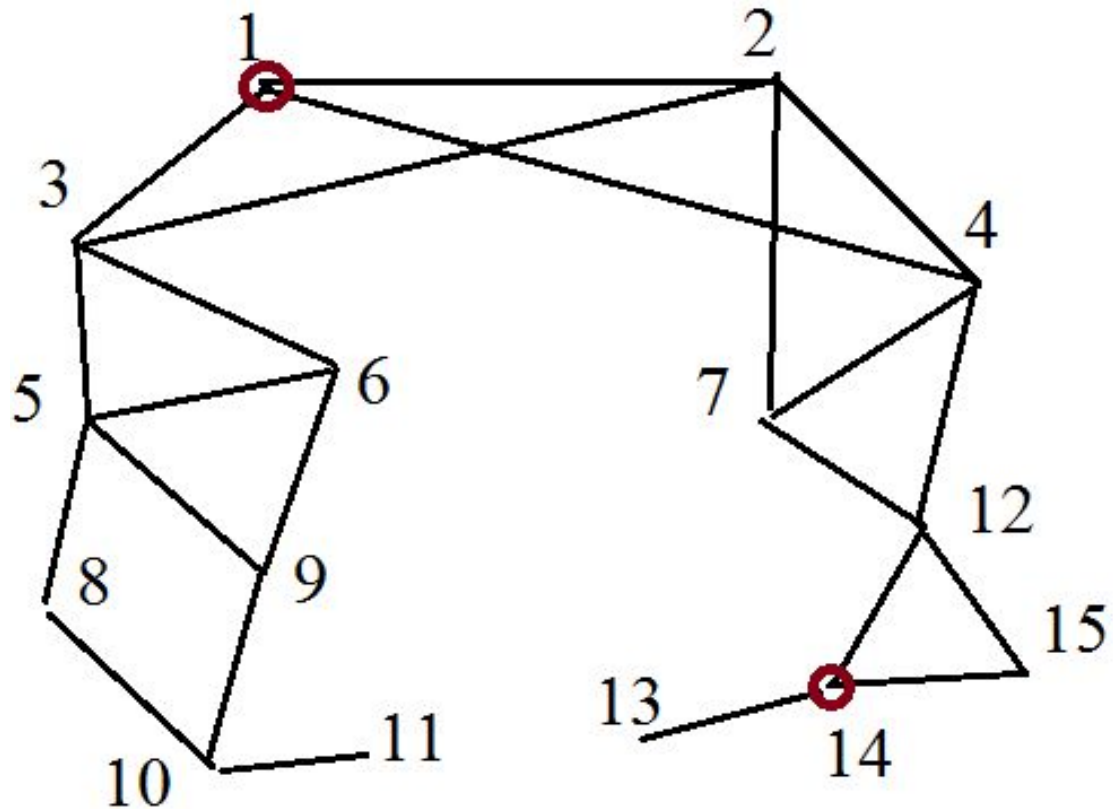
Кратчайший путь

В

неориентированном

графе без весов

Задан граф
с начальной 1-ой и конечной 14-ой



Найти кратчайший путь

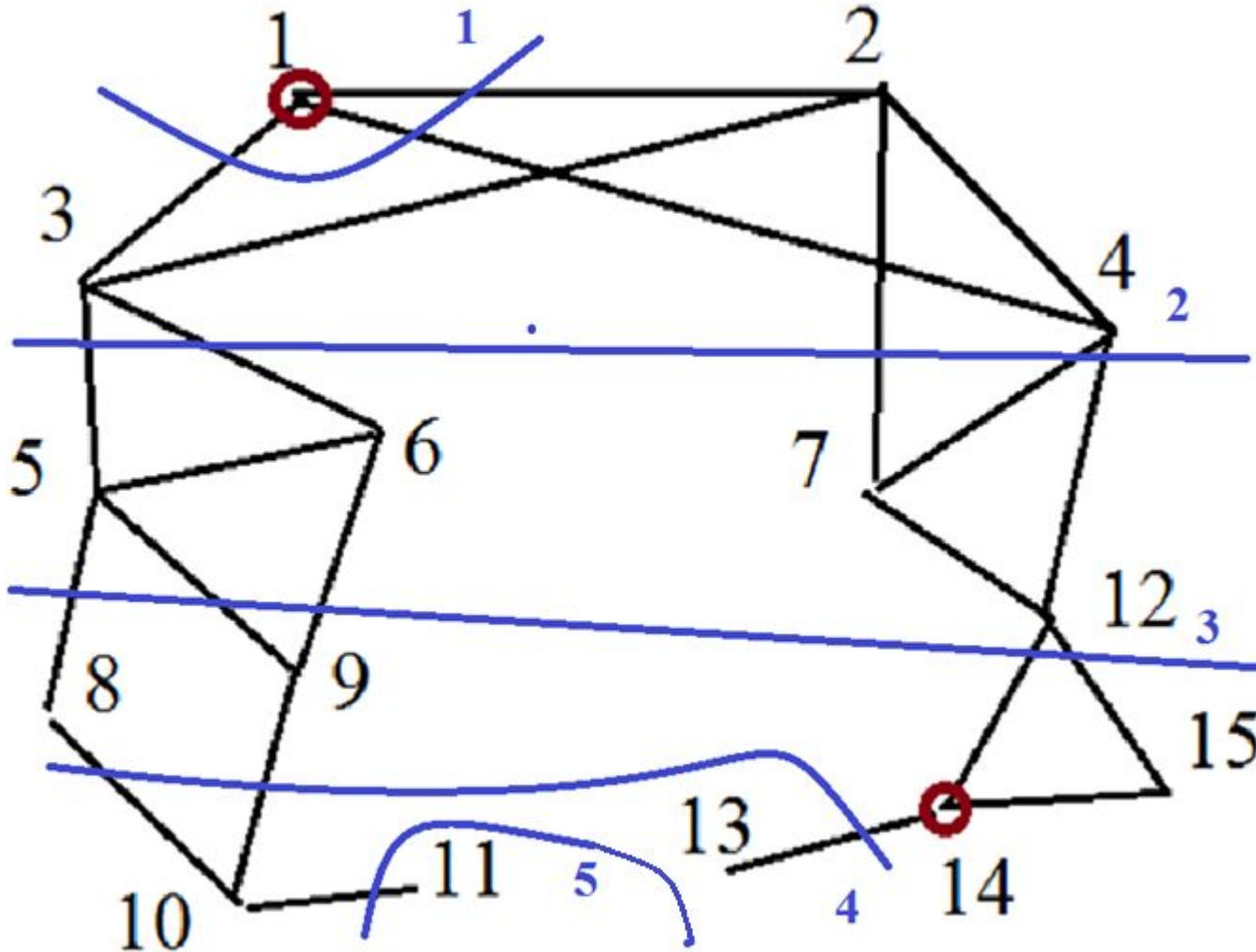
Матричная форма графа

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
2	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0
3	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0
4	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0
5	0	0	1	0	0	1	0	1	1	0	0	0	0	0	0
6	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0
7	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0
10	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	1	0	0	1	0	0	0	0	0	0	1	1
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1
15	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

Ввод данных

```
int main() {  
    int G[100][100], // граф транспортной сети  
        l,j,n, // n – число вершин  
        n_p,k_p; // начало и конец пути  
    cin >> n >> n_p >> k_p;  
    for (i=1; i<=n; i++)  
        for (j=1; j<=n; j++)  
            cin >> G[i][j];  
}
```

1 задача – определение числа вершин в кратчайшего пути до вершин графа



Длина пути

1 – 1

2,3,4 – 2

5,6,12 – 3

8,9,14,15 –

4

10,13 – 5

11 – 6

Определение длины кратчайшего

пути

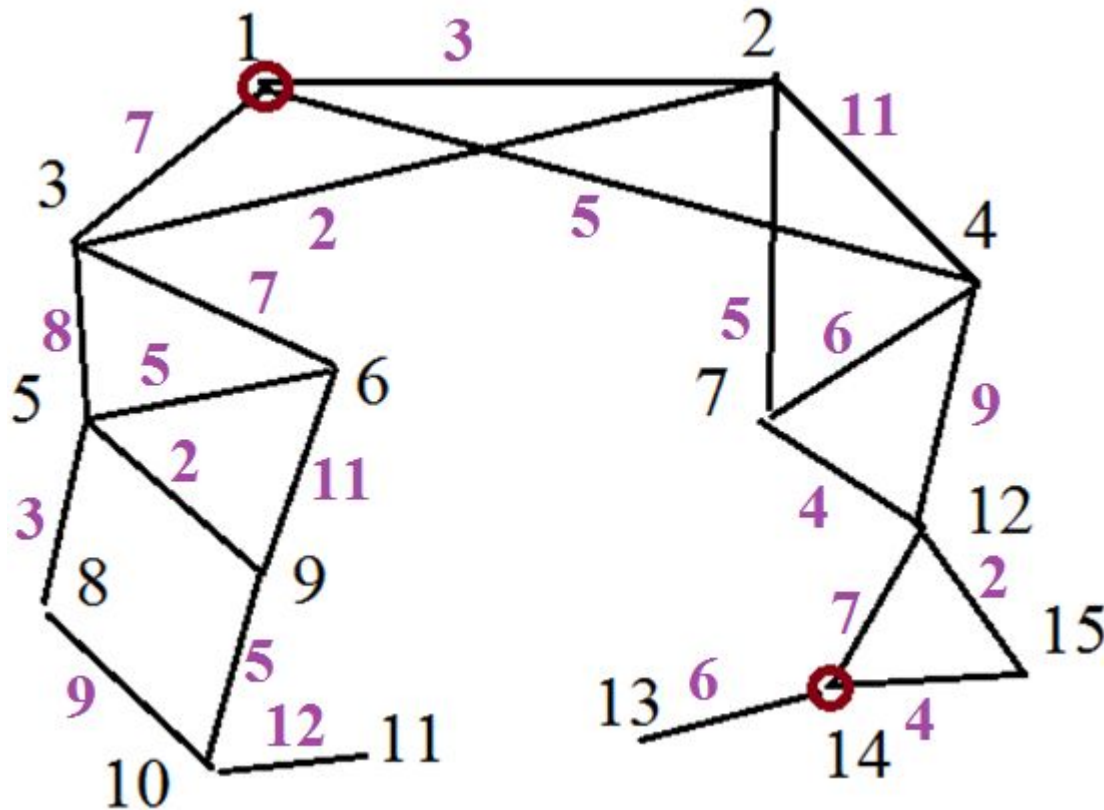
```
int r[100]={0}, // 0 – расстояние не определено
    ob[100], // обработанные вершины
    a=1, // вершина из ob , которая обрабатывается
    p=2; // пустое место для записи новых вершин
r[n_p]=1; // кратчайший путь в n_p – 1
ob[1]=n_p; //
while a<p do {
    for (i=0; i<n; i++) // ищем связанные с ob[a]
        if (G[i][ob[a]]==1 & r[i]==0) { //необработанные вершины
            r[i]=r[ob[a]]+1;
            ob[++p]=i;
        }
    a++;
}
```

2 задача - Анализ вектора расстояний

```
if (r[k_p]==0) {cout << "нет пути"; return 0;}
int jul[100], // кратчайший путь
    m=k_p; // новая найденная вершина в
пути
while (n_p!=m) {
    jul[r[m]]=m;
    for (i=0;G[i][m]==0 || r[i]>=r[m]; i++);
    m=i;
}
Jul[1]=n_p;
for (i=1; i<=r[k_p]; i++) cout << jul[i]<< " ";
```


Кратчайший путь в неориентированном графе с весами

Задан граф
с начальной 1-ой и конечной 14-ой



Найти кратчайший путь

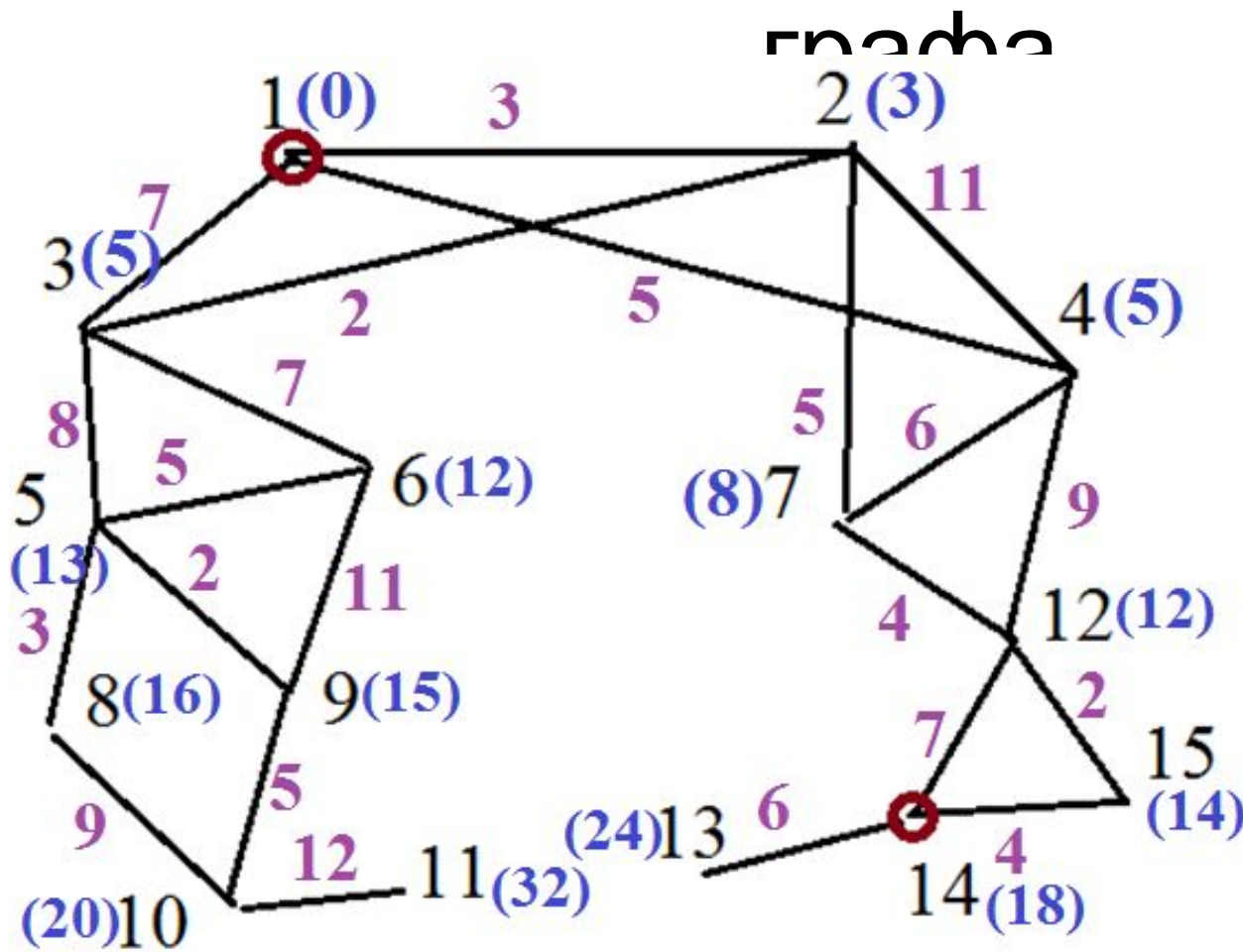
Матричная форма графа

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	3	7	5	0	0	0	0	0	0	0	0	0	0	0
2	3	0	2	11	0	0	5	0	0	0	0	0	0	0	0
3	7	2	0	0	8	7	0	0	0	0	0	0	0	0	0
4	5	11	0	0	0	0	6	0	0	0	0	9	0	0	0
5	0	0	8	0	0	5	0	3	2	0	0	0	0	0	0
6	0	0	7	0	5	0	0	0	11	0	0	0	0	0	0
7	0	5	0	6	0	0	0	0	0	0	0	0	12	0	0
8	0	0	0	0	3	0	0	0	0	9	0	0	0	0	0
9	0	0	0	0	2	11	0	0	0	5	0	0	0	0	0
10	0	0	0	0	0	0	0	9	5	0	12	0	0	0	0
11	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0
12	0	0	0	9	0	0	12	0	0	0	0	0	0	7	2
13	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0
14	0	0	0	0	0	0	0	0	0	0	0	7	6	0	4
15	0	0	0	0	0	0	0	0	0	0	0	2	0	4	0

Ввод данных

```
int main() {  
    int G[100][100], // граф транспортной сети  
        l,j,n, // n – число вершин  
        n_p,k_p; // начало и конец пути  
    cin >> n >> n_p >> k_p;  
    for (i=1; i<=n; i++)  
        for (j=1; j<=n; j++)  
            cin >> G[i][j];  
}
```

1 задача – определение длины кратчайшего пути до вершин графа



Длина пути

1 – 0	9 – 15
2 – 3	10 – 20
3 – 5	11 – 32
4 – 5	12 – 12
5 – 13	13 – 24
6 – 12	14 – 18
7 – 8	15 – 14
8 – 16	

Определение длины кратчайшего пути

```
int r[100]={-1}, // -1 – расстояние не определено
r[n_p]=0; // кратчайший путь в n_p – 0
for (int k=0; k<n; k++)
  for (i=0; i<n; i++)
    for (j=0; j<n; j++)
      if (G[i][j]>0 & r[i]>=0)
        if (r[j]==-1 | r[j]>r[i]+G[[i][j]) r[j]=r[i]+(G[i][j]);
```

2 задача - Анализ вектора расстояний

```
if (r[k_p]==-1) {cout << "нет пути"; return 0;}
int jul[100], // кратчайший путь
    m=k_p; // новая найденная вершина в
пути
while (n_p!=m) {
    jul[r[m]]=m;
    for (i=0;G[i][m]==0 || r[i]>r[m]+G[m][i]; i++);
    m=i;
}
Jul[1]=n_p;
for (i=1; i<=r[k_p]; i++) cout << jul[i]<< " ";
```

Кратчайший путь в ориентированном графе с весами

***Метод решения такой же как в
неориентированном
графе с весами***