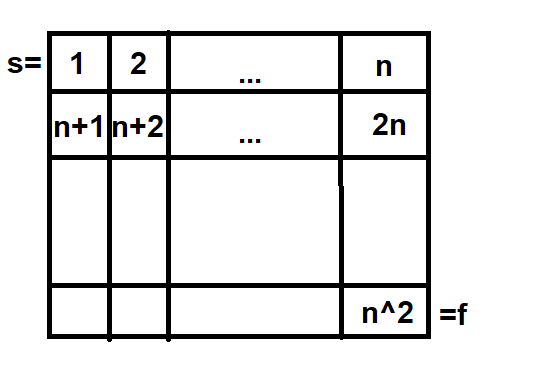
Работа 3  
Описание работы программы. Программа получает на вход файл, в котором специальным орбразом описан лабиринт (граф). На выходе программа печатает в консоль длину пути и его состав.  
Файл который идет на вход.

Лабиринт будем задавать с помощью двух параметров: размер(n) и массив дуг с длинами. По параметру n строим матрицу n\*n, и нумеруем клеточки этой матрицы таким образом



Затем на основе нумерации задаем дуги и их длины. Для обозначения дуги используем "i-j" (без кавычек), а затем через пробел (должен быть только один пробел) пишем натуральное число которое не превышает диапазона значений для типа unsigned int.

Пример: s-f 100, 1-2 2.

Замечание: значения вершин i, j должны принадлежать диапазону от 1, ..., n^2. Вершины с номерами 1 и n^2 обозначаются s и f, что соответственно означает start и finish.

Шаблон:

Размер лабиринта: 1 < n < 21

Массив дуг с длинами:

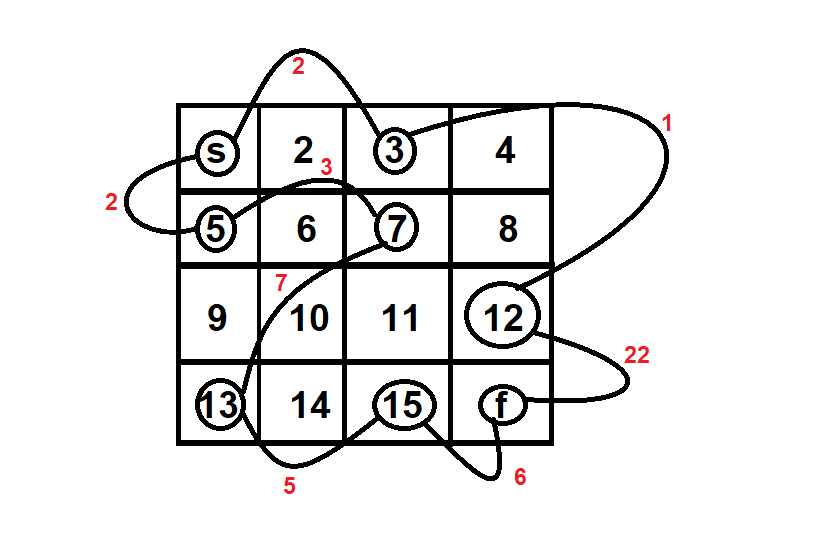
s-2 100

...

2-f 20

Пример:

Рассмотрим лабиринт представленный на картинке:



Для него файл с входными данными будет выглядеть следующим образом:

*4*

*s-3 2*

*3-12 1*

*s-5 2*

*5-7 3*

*7-13 7*

*13-15 5*

*15-f 6*

*12-f 22*

Вывод в консоль полученного результата

На первой строке длина кратчайшего пути из s-f если он есть.  
На второй строке состав пути от f к s

Пример:

Для графа на картинке имеем:

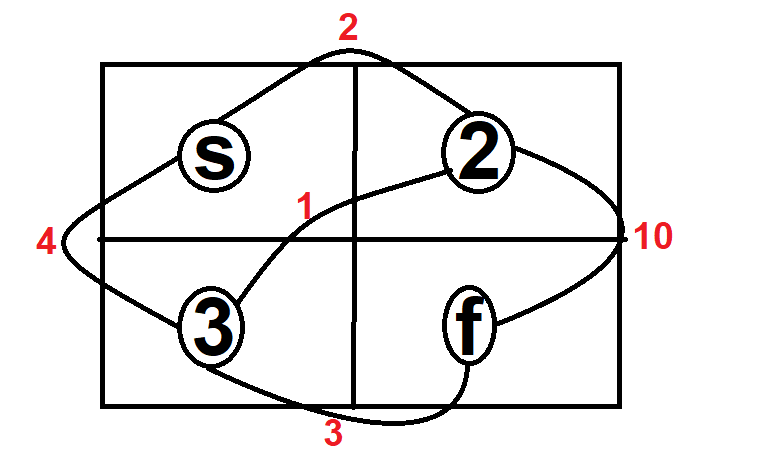
*23*

*Path: f-15-13-7-5-s*

**Описание программы.**

Для поиска кратчайшего пути в графе используем алгорит Дейкстры. Это сделано исходя из того, что алгоритм Дейкстры корректен на классе неотрицательно взвешенных орграфов (графов).  
  
Еще несколько примеров:

1)



*2*

*s-2 2*

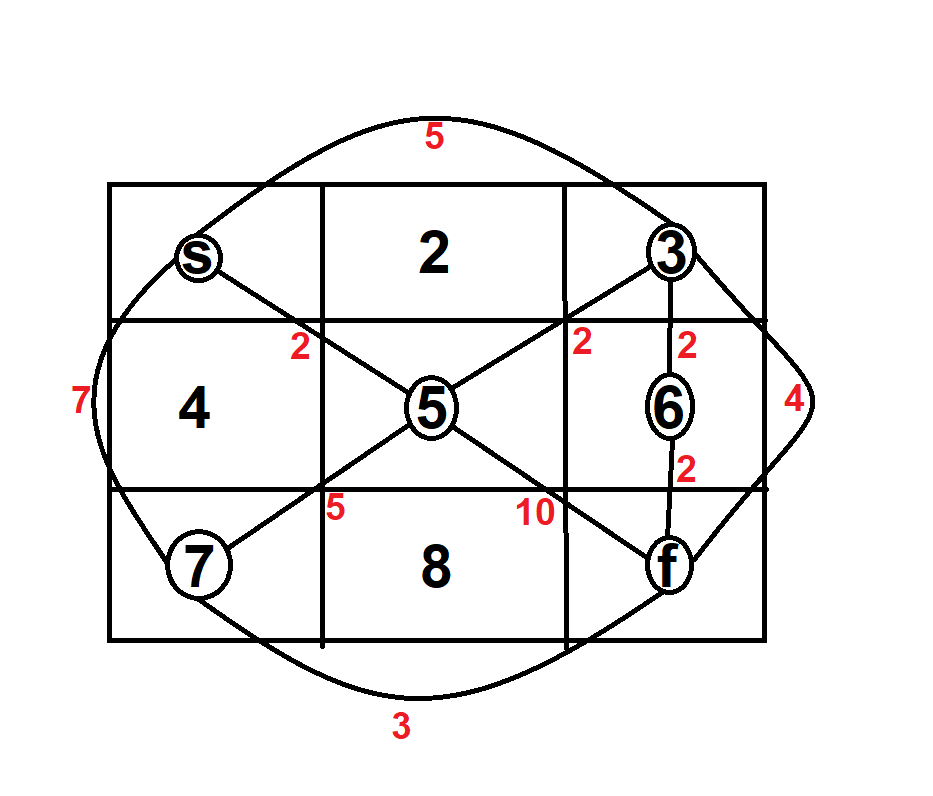
*s-3 4*

*2-3 1*

*3-f 3*

*2-f 10*

2)



*3*

*s-3 5*

*s-5 2*

*s-7 7*

*3-6 2*

*3-f 4*

*3-5 2*

*5-7 5*

*5-f 10*

*6-f 2*

*7-f 3*

Соответсвенно ответы для обоих примеров

1)6

Path: f-3-2-s

2)8

Path: f-3-5-s

В этом примере также есть путь длины 8 : f-6-3-5-s. Однако этот путь длинее на одну вершину.