

Руководство пользователя

1. HeapSort (Пирамидальная сортировка)

В данном проекте реализовано тестирование работы алгоритма пирамидальной сортировки, который описан в статической библиотеке `library.lib`. При запуске программы откроется окно, в котором будут отображены результаты тестов.

Первый тест – сортировка массива, сгенерированного случайно в заданном отрезке и с заданным размером. Пользователю будет предложено сначала ввести размер массива целым числом (Рис. 1).



Рис. 1 Демонстрация работы программы.

Далее необходимо ввести соответственно левый и правый концы отрезка целым или вещественным числом (Рис. 2).

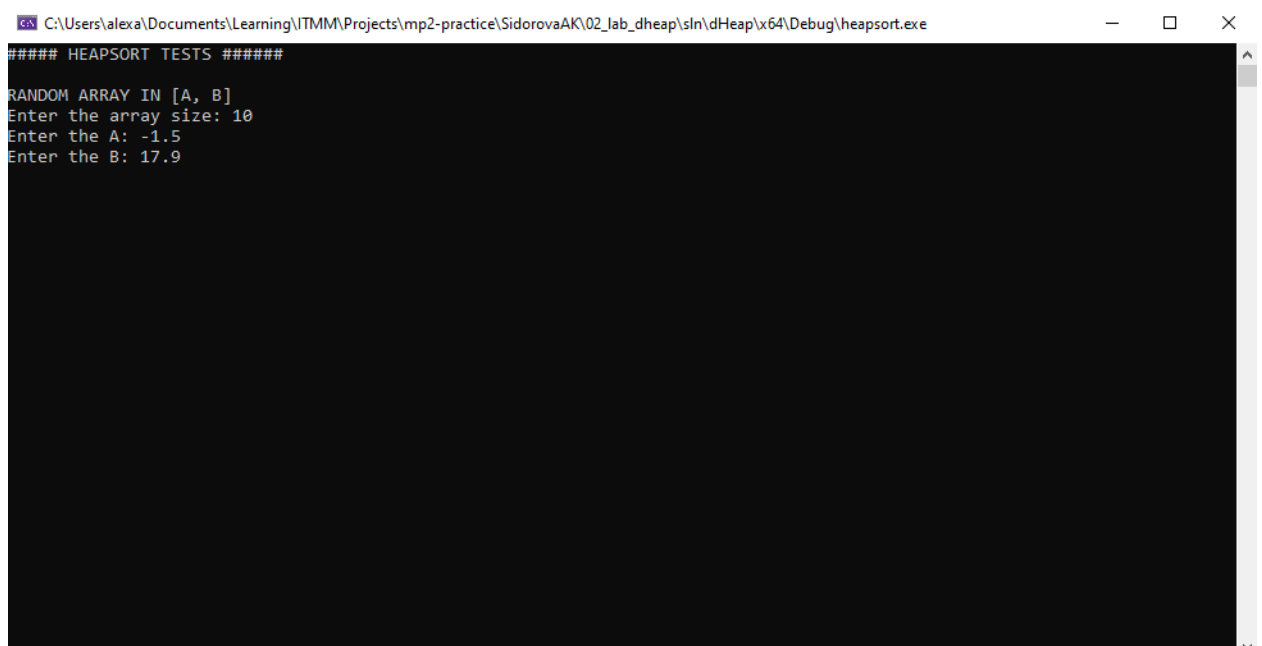


Рис. 2 Демонстрация работы программы.

Таким образом, сначала отобразится сгенерированный массив, а после сразу же отсортированный в порядке убывания (Рис. 3).



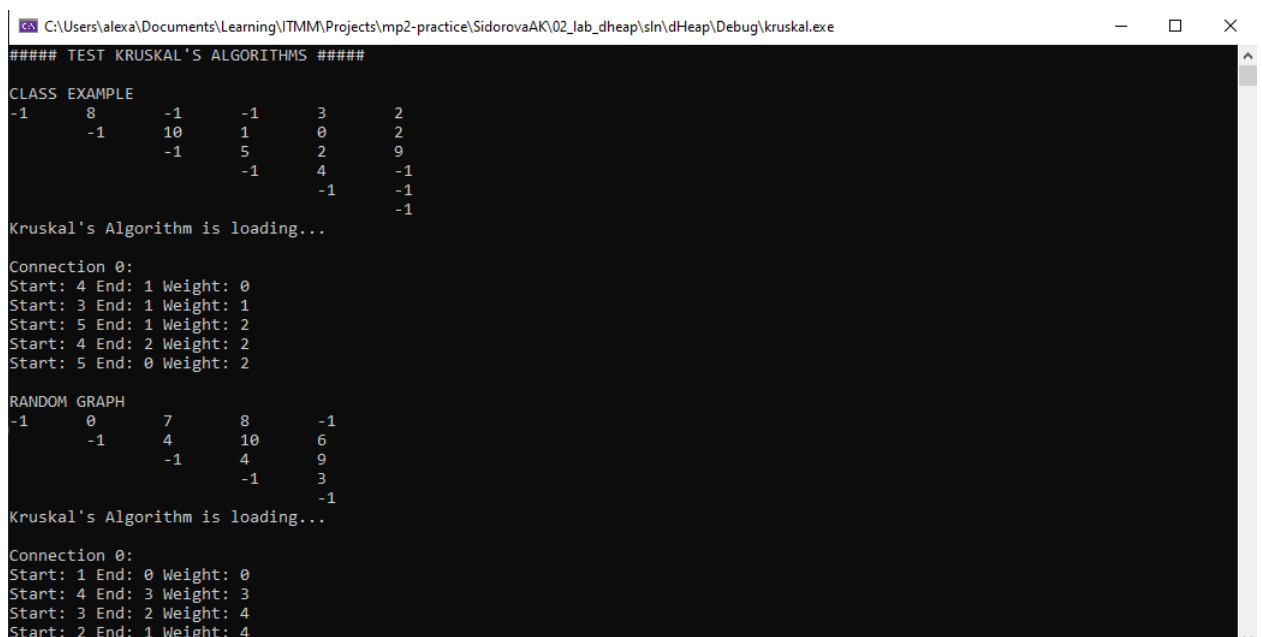
```
##### HEAPSORT TESTS #####  
  
RANDOM ARRAY IN [A, B]  
Enter the array size: 10  
Enter the A: -1.5  
Enter the B: 17.9  
14.2 17.6 11.8 0.9 17.4 4.1 7.9 14.9 8.4 11.4  
Sorting...  
17.6 17.4 14.9 14.2 11.8 11.4 8.4 7.9 4.1 0.9  
  
EMPTY ARRAY  
-842150451  
Sorting...  
[ERROR] Array is empty!  
  
INCORRECT SIZE -100 OF RANDOM ARRAY IN [-20; 20]  
10 -14 18 -12 16 15 0 12 7 -17  
Size = -100  
Sorting...  
[ERROR] Incorrect size!  
  
C:\Users\alexa\Documents\Learning\ITMM\Projects\mp2-practice\SidorovaAK\02_lab_dheap\sln\dHeap\x64\Debug\heapsort.exe (п  
роцесс 5276) завершает работу с кодом 0.  
Чтобы закрыть это окно, нажмите любую клавишу...
```

Рис. 3 Демонстрация работы программы.

Кроме того, сразу же будут выведены результаты остальных двух тестов: попытка отсортировать пустой массив и попытка отсортировать массив с указанным отрицательным размером. Но будут выведены ошибки, поскольку отсортировать нечего, а размер не может быть отрицательным.

2. Kruskal (Алгоритм Краскала)

Данный проект содержит в себе файл с тестами работы алгоритма Краскала. При запуске программы пользователь увидит тест с графом, который был разобран в аудитории на паре, чтобы сравнить результаты работы программы и результаты, полученные в аудитории (Рис. 4).



```
C:\Users\alexa\Documents\Learning\ITMM\Projects\mp2-practice\SidorovaAK\02_lab_dheap\sln\dHeap\Debug\kruskal.exe  
##### TEST KRUSKAL'S ALGORITHM #####  
  
CLASS EXAMPLE  
-1      8      -1      -1      3      2  
      -1      10      1      0      2  
      -1      5      2      9  
      -1      4      -1  
      -1      -1  
      -1  
  
Kruskal's Algorithm is loading...  
  
Connection 0:  
Start: 4 End: 1 Weight: 0  
Start: 3 End: 1 Weight: 1  
Start: 5 End: 1 Weight: 2  
Start: 4 End: 2 Weight: 2  
Start: 5 End: 0 Weight: 2  
  
RANDOM GRAPH  
-1      0      7      8      -1  
      -1      4      10      6  
      -1      4      9  
      -1      3  
      -1  
  
Kruskal's Algorithm is loading...  
  
Connection 0:  
Start: 1 End: 0 Weight: 0  
Start: 4 End: 3 Weight: 3  
Start: 3 End: 2 Weight: 4  
Start: 2 End: 1 Weight: 4
```

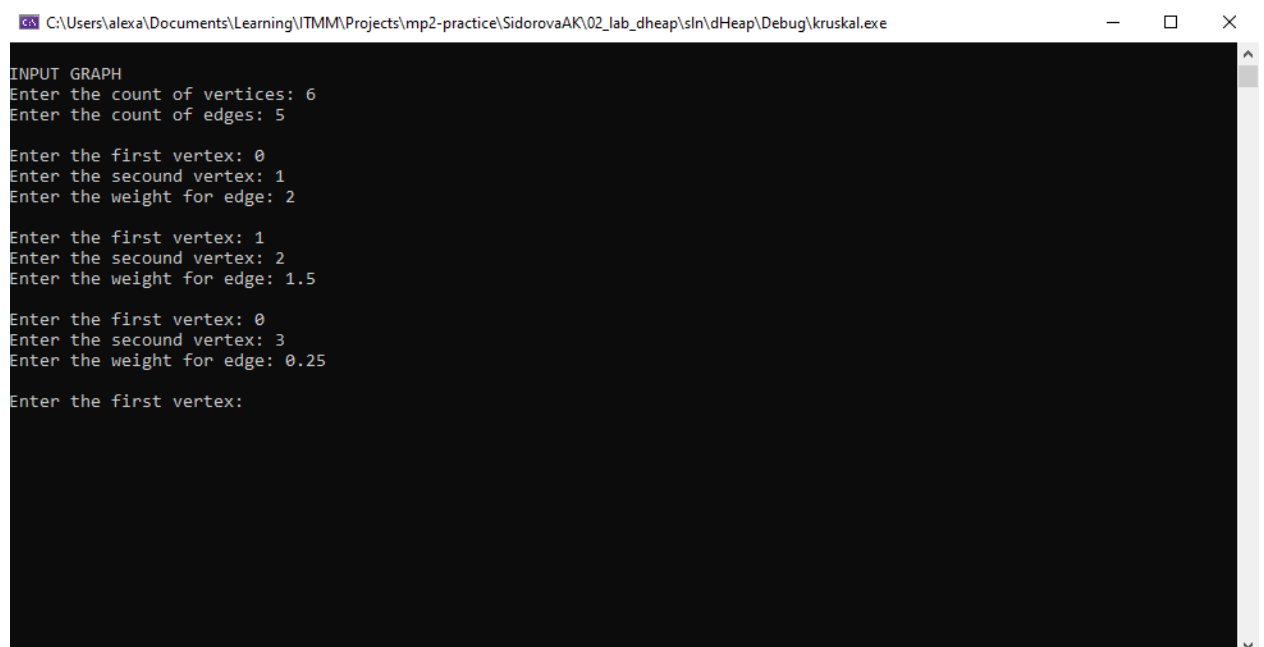
Рис. 4 Демонстрация работы программы

Сначала выводится верхне-треугольная матрица смежности графа, ведь ее достаточно, чтобы определить смежные ли вершины и какой вес у определенного ребра. Стоит отметить, что «-1» - определяет отсутствие ребра, а любое другое неотрицательное число – вес ребра между данными вершинами (номер столбца и строки).

Затем для каждого класса смежности («Connection #») выводится список ребер в формате: стартовая вершина, конечная вершина, вес между данными вершинами. Таким образом, для примера из аудитории получается 1 класс смежности с 5 ребрами наименьшего размера, не образующие цикл.

Далее на том же рисунке можно увидеть второй тест для графа, который сгенерирован случайно, и результаты теста по этому графу.

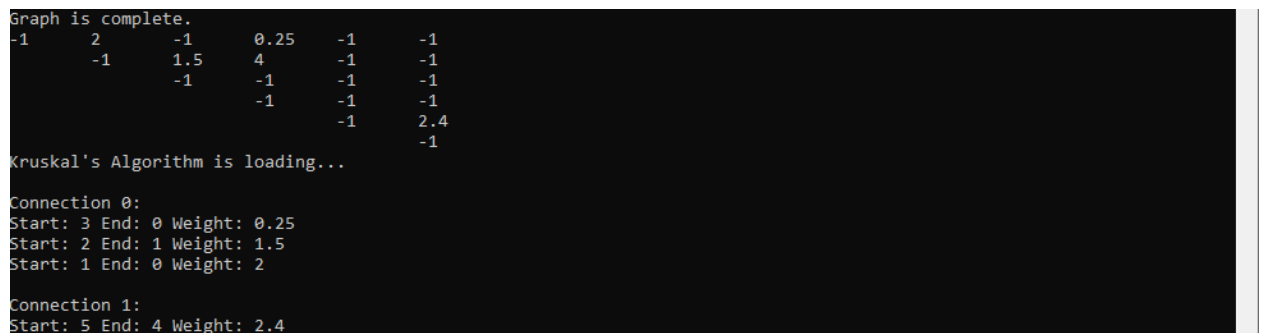
Кроме того, в конце пользователю будет предложено ввести свой граф. Сначала необходимо ввести положительное целое число вершин, затем положительное целое число ребер. После чего будет предложено ввести информацию о ребрах в формате: стартовая вершина, конечная вершина, вес. Стоит отметить, что нумерация вершин идет с нуля, а вес должен быть неотрицательным (Рис. 5).



```
INPUT GRAPH
Enter the count of vertices: 6
Enter the count of edges: 5
Enter the first vertex: 0
Enter the second vertex: 1
Enter the weight for edge: 2
Enter the first vertex: 1
Enter the second vertex: 2
Enter the weight for edge: 1.5
Enter the first vertex: 0
Enter the second vertex: 3
Enter the weight for edge: 0.25
Enter the first vertex:
```

Рис. 5 Демонстрация работы программы.

В итоге, по окончании заполнения информации о графе будет выведен граф в виде верхне-треугольной матрицы смежности, а затем результаты тестирования (Рис. 6). В нашем случае получилось 2 класса смежности и, следовательно, по каждому классу получили свое дерево (каркас).

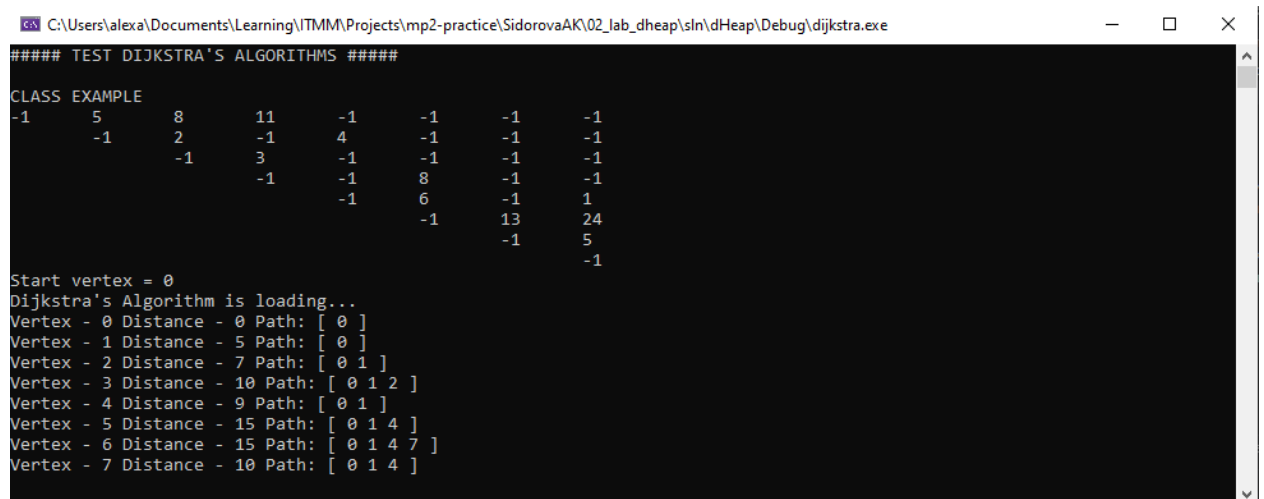


```
Graph is complete.
-1 2 -1 0.25 -1 -1
-1 -1 1.5 4 -1 -1
-1 -1 -1 -1 -1
-1 -1 -1 -1
-1 -1 2.4
-1
Kruskal's Algorithm is loading...
Connection 0:
Start: 3 End: 0 Weight: 0.25
Start: 2 End: 1 Weight: 1.5
Start: 1 End: 0 Weight: 2
Connection 1:
Start: 5 End: 4 Weight: 2.4
```

Рис. 6 Демонстрация работы программы.

3. Dijkstra (Алгоритм Дейкстры)

В данном проекте реализованы тесты работы алгоритма Дейкстры. При запуске программы первым делом можно увидеть первый тест графа, который был разобран в аудитории на паре, и, следовательно, сравнить результаты работы программы и результаты, полученные в аудитории (Рис. 7).

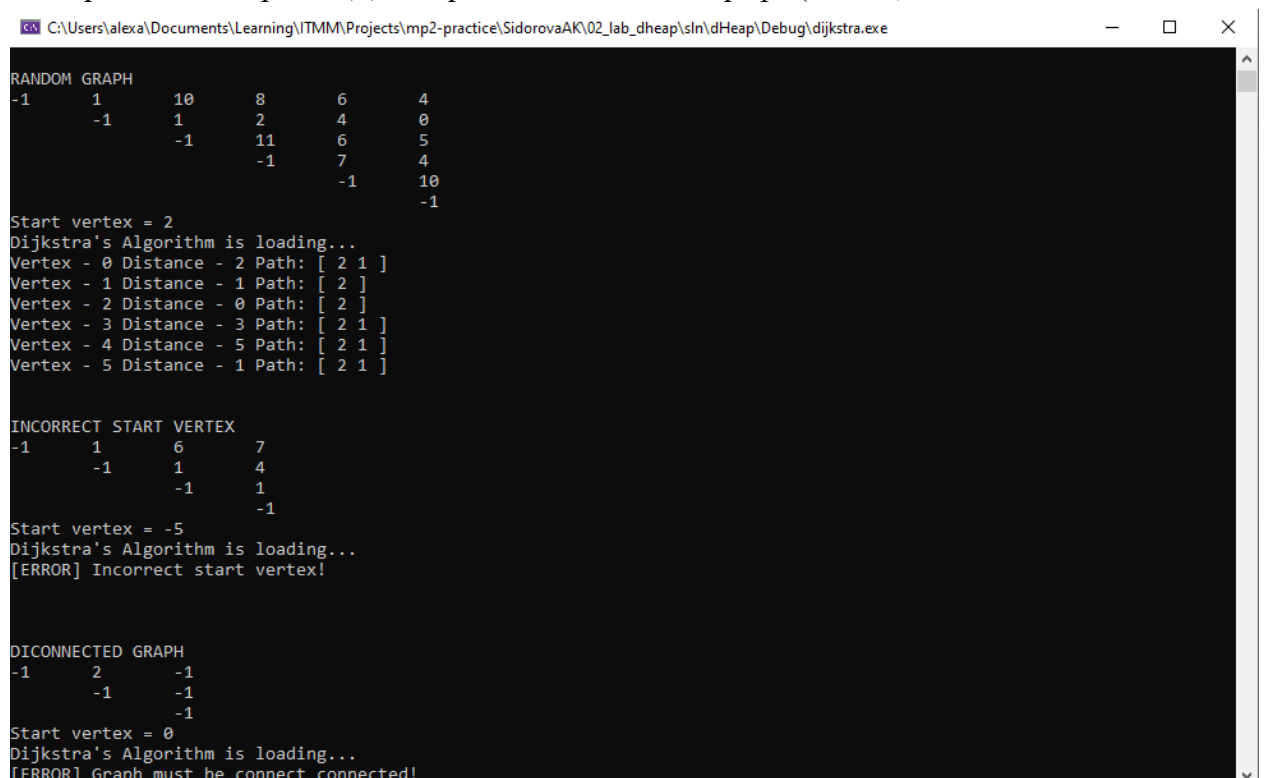


```
##### TEST DIJKSTRA'S ALGORITHMS #####
CLASS EXAMPLE
-1      5      8      11     -1     -1     -1     -1
      -1      2      -1      4     -1     -1     -1
            -1      3      -1     -1     -1     -1
                  -1     -1      8     -1     -1
                        -1      6     -1      1
                              -1     13     24
                                  -1      5
                                      -1
Start vertex = 0
Dijkstra's Algorithm is loading...
Vertex - 0 Distance - 0 Path: [ 0 ]
Vertex - 1 Distance - 5 Path: [ 0 ]
Vertex - 2 Distance - 7 Path: [ 0 1 ]
Vertex - 3 Distance - 10 Path: [ 0 1 2 ]
Vertex - 4 Distance - 9 Path: [ 0 1 ]
Vertex - 5 Distance - 15 Path: [ 0 1 4 ]
Vertex - 6 Distance - 15 Path: [ 0 1 4 7 ]
Vertex - 7 Distance - 10 Path: [ 0 1 4 ]
```

Рис. 7 Демонстрация работы программы.

Сначала будет выведена верхне-треугольная матрица смежности, затем номер стартовой вершины (нумерация идет с нуля!), а после список вершин с результатами по каждой в формате: номер вершины, кратчайшее расстояние от стартовой до данной вершины, путь от стартовой до данной, не включая ее саму (за исключением стартовой). Таким образом, можно восстановить все пути от стартовой до определенной вершины.

Затем идет тестирование случайно сгенерированного связного графа и с случайно сгенерированной стартовой вершиной, тестирование с отрицательной стартовой вершиной, тестирование алгоритма Дейкстры на несвязанной графе (Рис. 8).



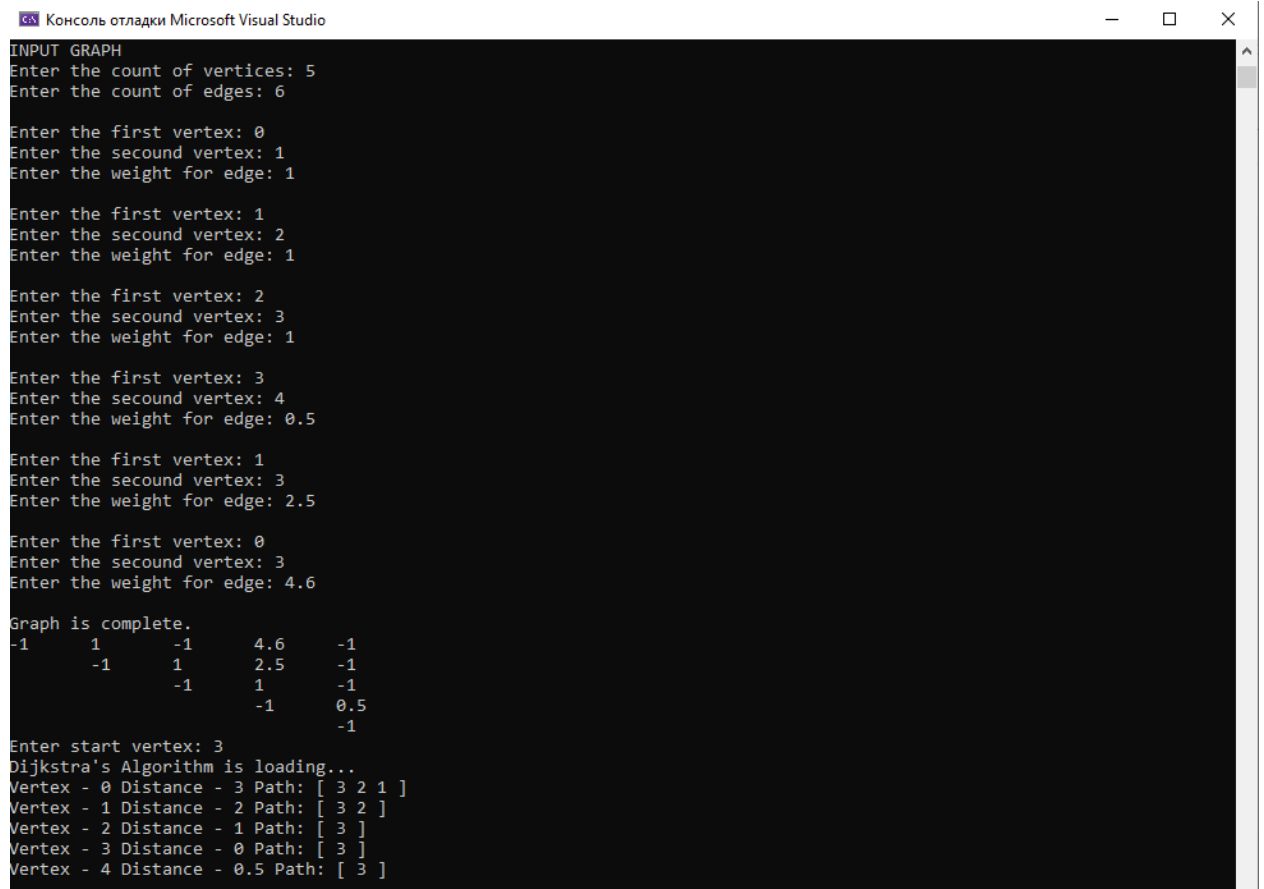
```
RANDOM GRAPH
-1      1      10      8      6      4
      -1      1      2      4      0
            -1     11      6      5
                  -1      7      4
                        -1     10
                              -1
Start vertex = 2
Dijkstra's Algorithm is loading...
Vertex - 0 Distance - 2 Path: [ 2 1 ]
Vertex - 1 Distance - 1 Path: [ 2 ]
Vertex - 2 Distance - 0 Path: [ 2 ]
Vertex - 3 Distance - 3 Path: [ 2 1 ]
Vertex - 4 Distance - 5 Path: [ 2 1 ]
Vertex - 5 Distance - 1 Path: [ 2 1 ]

INCORRECT START VERTEX
-1      1      6      7
      -1      1      4
            -1      1
                  -1
Start vertex = -5
Dijkstra's Algorithm is loading...
[ERROR] Incorrect start vertex!

DISCONNECTED GRAPH
-1      2      -1
      -1      -1
            -1
Start vertex = 0
Dijkstra's Algorithm is loading...
[ERROR] Graph must be connect connected!
```

Рис. 8 Демонстрация работы программы.

И в самом конце будет предложено самому пользователю ввести свой связанный граф в таком же формате, как и в алгоритме Краскала. После отображения верхне-треугольной матрицы смежности необходимо ввести номер стартовой вершины. Затем идет работа алгоритма Дейкстры и соответственно вывод результатов его работы (Рис. 9).



```
Консоль отладки Microsoft Visual Studio
INPUT GRAPH
Enter the count of vertices: 5
Enter the count of edges: 6

Enter the first vertex: 0
Enter the secound vertex: 1
Enter the weight for edge: 1

Enter the first vertex: 1
Enter the secound vertex: 2
Enter the weight for edge: 1

Enter the first vertex: 2
Enter the secound vertex: 3
Enter the weight for edge: 1

Enter the first vertex: 3
Enter the secound vertex: 4
Enter the weight for edge: 0.5

Enter the first vertex: 1
Enter the secound vertex: 3
Enter the weight for edge: 2.5

Enter the first vertex: 0
Enter the secound vertex: 3
Enter the weight for edge: 4.6

Graph is complete.
-1      1      -1      4.6      -1
      -1      1      2.5      -1
            -1      1      -1
                  -1      0.5
                        -1

Enter start vertex: 3
Dijkstra's Algorithm is loading...
Vertex - 0 Distance - 3 Path: [ 3 2 1 ]
Vertex - 1 Distance - 2 Path: [ 3 2 ]
Vertex - 2 Distance - 1 Path: [ 3 ]
Vertex - 3 Distance - 0 Path: [ 3 ]
Vertex - 4 Distance - 0.5 Path: [ 3 ]
```

Рис. 9 Демонстрация работы программы.