ESO207: Data Structures and Algorithms

Programming Assignment *4* *Due: 14 November 2017*

Problem 1. In this problem, you will solve the single source shortest path problem for a given directed weighted graph with no negative weight edges using Dijkstra's Algorithm.

Input: The input consists of the directed weighted graph in adjacency lists format, followed by the source vertex.

A graph in the adjacency lists format is input as follows.

- The first number is n, the number of vertices, which will be an integer ≥ 1. The vertex set is assumed to be $V = \{0, 1, \ldots, n-1\}$.
- Following the number n of vertices, there are n lines, where, the ith line corresponds to the adjacency list of node numbered i. Each adjacency list is a sequence of (vertex id, edge-weight) pairs, where, the vertex id lies in $\{0, 1, 2, \ldots, n\} - \{i\}$ and the edge-weight is a non-negative integer. These numbers are white space separated. Each adjacency list ends with a -1.

After the adjacency lists representation is input, line $n+2$ contains the source vertex id (between 0 and $n-1$). This concludes the input. (*Note:* The source vertex id can be any vertex between 0 to $n-1$.

Output: Print the table of shortest path distances from the source vertex to all the vertices. This table should contain n lines. Each line should contain vertex-id, distance pair, successively, for vertex-id $= 0, 1, 2, \ldots, n-1$. The distance attribute is the shortest path distance from the given source vertex to vertex-id.

Sample input and output for the graph in Figure 1 is given below.

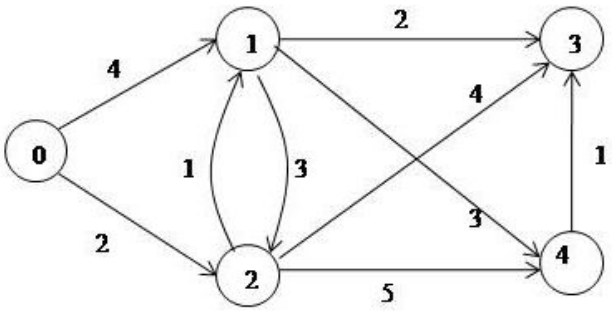

Figure 1: Example graph for sample input and output of Dijkstra's algorithm

Sample Input. The following sample presents the input for the graph shown in the Figure and asks to find the shortest path to any vertex from source vertex 0.

```
5
1 4 2 2  -1
2 3 3 2 4 3 -1
3 4 1 1 4 5 -1
-1
3 1 -1
0
```

The sample output should be the following.

```
0 0
1 3
2 2
3 5
4 6
```

Notes. You must use Dijkstra's algorithm along with min priority queue to solve the problem. Do not use Bellman-Ford (or Floyd-Warshall) to solve the problem. Write all the code on your own, including those of the data structures like priority queue. Do not use "off the shelf" code from any source. However, you may look at text book code to get ideas.