Lab 13: Shortest Path

Lecture date 5/26/2015

Due date 5/30/2015 11:59 PM

In this lab, we will implement Dijkstra’s algorithm for finding the shortest path. When a graph structure (i.e. a set of nodes and edges) is given, your program prints the shortest path as a result of running Dijkstra’s algorithm. You may want to use a priority queue to find the node with the smallest distance from the source node.

1. Input

Read a set of vertices from the first line and a set of edges from the second line of the given input file. Each line is described below. You may assume that your node is represented by any integer.

* Vertices are given in the first line. Each vertex is separated by a space.
* Edges are given in the second line. Each edge is represented by a pair of vertices and its weight. For example, “1-3-4” represents an edge from the vertex 1 to 3 with the weight value of 4.

An exemplary input file is given below with the corresponding graph for your reference.



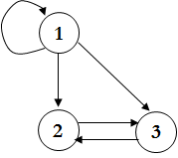
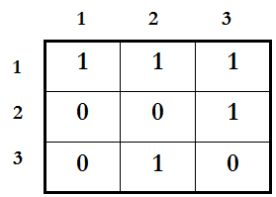
Input.txt

|  |
| --- |
| 1 2 3 4 5 6 7  1-2-2 1-4-1 2-5-10 2-4-3 3-1-4 3-6-5 4-3-2 4-6-8 4-7-4 4-5-2 5-7-6 7-6-1 |

2. Data structure

(1) Data structure for the graph

You can use an adjacency matrix to store your graph information as we discussed in class. An example is shown below.

(2) Data structure for nodes in priority queue

struct Node {

int vertex;

int priority;

}

3. Program description

* name : p11.c
* input : an input file name is given as a command line argument. See the example in “1. input” .
* output : the shortest path (for the given graph in input) in the standard output

Submit to the course website ([https://portal.hanyang.ac.kr](https://portal.hanyang.ac.kr/)) your source code and a written report. Your report should include the description of your own implementation.