

Program #4 – Fire Response

Due midnight, Friday December 11

The purpose of this assignment is to understand and implement Dijkstra’s algorithm.

1 Introduction

The town of New Los Vertices has a number of one-way streets connecting its various neighborhoods. Unfortunately, the town also has an arsonist on the loose. Your job is to write a program that helps the fire department deal with the many fires that arise each night.

As in the previous assignment, the input to your program comes in two parts: first there is a map of New Los Vertices indicating the streets and travel times between neighborhoods. You may assume that there are at most 99 neighborhoods, and the neighborhoods will be identified by (different) integers from 1 to the number of neighborhoods. Neighborhood 1 is where the fire station is located. Your program should read in this map and store it as a *directed* graph in adjacency list form. Following the map will be a list of fire locations. After reading in each fire location, your program should find and print out a quickest route from the starting neighborhood to the destination (note that each directed edge in the graph will have its own travel time).

The suggested implementation is to use Dijkstra’s algorithm once to store a shortest path tree for the graph rooted at the fire station (neighborhood 1). This shortest path tree can then be used to print out the needed paths.

2 Technical details

Each neighborhood (vertex) is identified by a single integer. Neighborhoods will be numbered consecutively starting with 1. The first line of the map contains an integer n giving the number of neighborhoods in the map. The next lines each represent a (one-way) street, listing the neighborhood that the street goes from, the neighborhood that the street goes to, and the time taken by a fire engine (with siren blazing) to drive along the street (a floating point number of minutes). After the map will be a dummy line containing “0 0 0.0” – this dummy line marks the end of the map and does not represent a street. At this point you should print out the adjacency list representation of the map so we can check that your program is reading in the graph correctly.

After printing out the graph, you should run Dijkstra’s algorithm to compute a shortest path tree rooted at neighborhood 1 (the fire station).

Following the dummy line, there will be a series of fire locations. Each fire location is a line with a single integer identifying the neighborhood containing the fire. For each of these fire locations you should:

1. print out an estimated response time based on the distance computed by Dijkstra’s algorithm
2. use these “parent” neighborhoods to print out the quickest route from the fire station to the destination.

The last line in the file is another dummy line, containing “0”

Thus one map of a 5-neighborhood district would be input like the following:

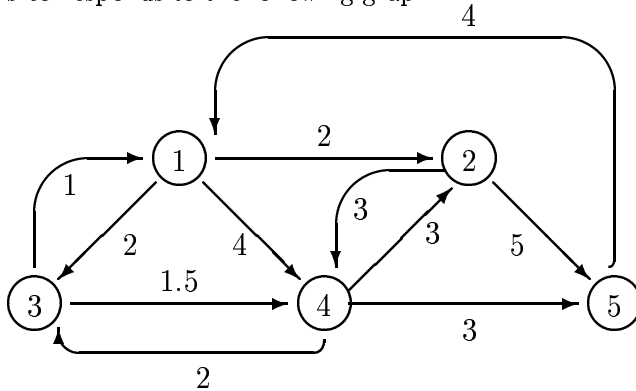
```
4
1 3 2.0
3 1 1.0
3 4 1.5
4 3 2.0
1 4 4.0
1 2 2.0
4 2 3.0
2 4 3.0
```

```

5 1 4.0
4 5 3.0
2 5 5.0
0 0 0

```

This corresponds to the following graph



The adjacency list representation of the graph might be printed as:

Adjacency list, format is "from: to @cost,"

```

1: 3 @2.0, 4 @4.0, 2 @2.0,
2: 4 @3.0, 5 @5.0,
3: 1 @1.0, 4 @1.5,
4: 3 @2.0, 2 @3.0, 5 @3.0,
5: 1 @4.0,

```

Notice that each edge is printed together with its cost. Printing just one digit past the decimal point is sufficient.

Once it has read in the graph and output the adjacency list your program will run Dijkstra's algorithm and then accept fire reports. (Note that you will only have to run Dijkstra's algorithm once for this assignment.) Each fire report is just a single neighborhood number on its own line. The program should print a shortest path that goes from the firestation (neighborhood 1) to given neighborhood.

For example, using the above graph, the fire report "5" would cause something like:

```

Fire in Neighborhood 5! Estimated transit time: 6.5 minutes.
Route:  fire station (1) to 3 to 4 to 5.

```

to be printed.

The entire input could then look like:

```
4
1 3 2.0
3 1 1.0
3 4 1.5
4 3 2.0
1 4 4.0
1 2 2.0
4 2 3.0
2 4 3.0
5 1 4.0
4 5 3.0
2 5 5.0
0 0 0
5
2
3
0
```

You may assume that all travel times are positive and there is a path from neighborhood 1 to every other vertex (although some of the other neighborhoods can be “dead ends”).

Re-read the implementation strategy for the previous program, the structure of your program for this assignment should be very similar.

Extra Credit: It is better to have a firestation in the middle of the city than on the outskirts. If each neighborhood is equally likely to have a fire, best location for the firestation is a neighborhood v where the sum: $\sum_{u \in V}$ path cost v to u is minimized. For 5 points of extra credit you can compute a neighborhood minimizing this sum and output something like:

```
A more central location for the firestation would be in
neighborhood 3, where the total cost to go to each other
neighborhood is 11.0.
```

after reading in the graph (I think these numbers are correct for the sample graph above). This extra credit is difficult, and only worth 5 programming points, so studying for the final might be a better use of your time.

Warning: we cannot accept late programs for this assignment.