CS6140: Advanced Programming Lab

Assignment 9: Finding the Number of Shortest Paths Modifying Dijkstra's to your Convenience.

Lead TA: Pradip Hatwar.

October 30, 2014.

Goal of the assignment

You will be given an undirected graph with positive integer weights; lets call it $G = (V, E)$. Moreover, one of G's vertices will be a designated source vertex s, and another vertex t will be a designated sink vertex. Your task is twofold.

1. You are to report the number of shortest paths from s to t along with the shortest path length.
2. Furthermore, you are to output the graph in dot language. Any vertex or edge that participates in some shortest path from s to t must be coloured blue, while the rest are in black. Furthermore, each blue vertex v must be annotated with the number of shortest path from s to v.

Use Dijkstra's single souce shortest path algorithm and modify it to find number of shortest paths. Also make sure that your implementation of Dijkstra's algorithm uses a binary heap. Thus the running time of your algorithm should be $O((m + n) \log n)$ where n and m are respectively the number of vertices and edges in G.

Input File Format

The input file has the following format which must be followed strictly. (To help in visualisation, (both) the input (and the output) file(s) are in dot language; you can visualise the graphs using graphviz.)

```
graph <graph_name> {
//nodes
<node_1>;
<node_2>;
:
:
```

```
:
<node_n>;
//edges
<node_i> -- <node_j> [label=‘‘<positive integer weight of edge>"];
:
:
:
}
```

The first line specifies the name of the graph and the second line is always “//nodes”. The n nodes must be listed subsequently, one per line. Implicitly assume that the first node is the source and the last node is the destination. To delineate the start of the edges, we add a line with just “//edges” in it, and then, we start listing edges — again one per line — in the prescribed format. Note that you have the read each line parse it to extract the required information such as the names of the two vertices and the weight of the edge. For example:

```
graph SampleGraph {
//nodes
hello;
world;
how;
are;
you;
NOT;
//edges
hello -- world [label="45"];
world -- how [label="45"];
are -- how  [label="34"];
NOT -- world  [label="673"];
are -- hello  [label="23"];
are -- NOT  [label="234"];
}
```

Assume that the first vertex listed in this order is the source vertex s and the last vertex listed in this order is the destination vertex t.

Command Line

```
<executable name> <input dot file> <output dot file>
```

Output File Format

The output file format is quite open ended. It must be a valid dot format file with a “.dot” extension. Here is a (very basic) valid output file. (The corresponding graph is shown in Figure 1. You are allowed to (and encouraged to) make your output more elaborate and interesting as long as the required information is contained in it.

```
graph SampleGraphWithAnnotations {
     size="1,4";
     one [label="one (source)",color=blue];
     two [label="two (1 sp)",color=blue];
     three [label="three (1 sp)",color=blue];
     four [label="four (2 sp)",color=blue];
     five [label="five"];
     six [label="six (2 sp)",color=blue];
     seven [label="seven (2 sp)",color=blue];
     eight [label="eight (2 sp)",color=blue];
     nine [label="nine (2 sp)",color=blue];
     ten [label="ten (destination) (4 sp)",color=blue];

one -- two [label="45",color=blue];
one -- three [label="7",color=blue];
two -- four [label="32",color=blue];
three -- four [label="70",color=blue];
four -- five [label="48"];
four -- six [label="23",color=blue];
five -- seven [label="12"];
six -- seven [label="27",color=blue];
seven -- eight [label="56",color=blue];
seven -- nine [label="65",color=blue];
eight -- ten [label="44",color=blue];
nine -- ten [label="35",color=blue];
 }
```

Any vertex of edge that participates in some shortest path from s to t must be coloured blue, while the rest are in black. Furthermore, each blue vertex v must be annotated with the number of shortest path from s to v.

Output to the Console

In addition to the output file, your program must also print the length of the shortest path and the number of shortest paths (in that order and with a space between them). Nothing else should be printed in the final version that is submitted. To be precise, the format of the command line and console output should be:

```
\$<executable> <input dot file> <output dot file>
<length of the shortest path> <Number of shortest paths>
```

For example:

```
\$<executable> <input dot file> <output dot file>
227 4
```

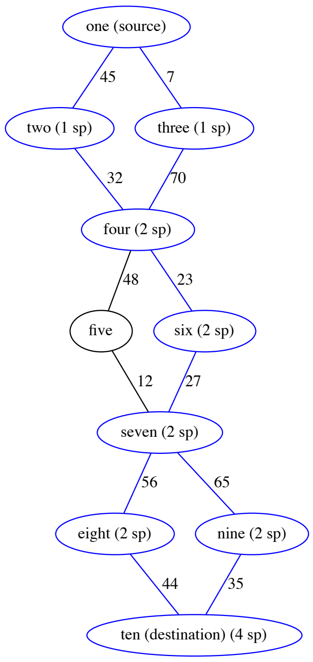

Figure 1: Note how the number of shortest paths are annotated in this graph. Plus, notice the colour of those edges and nodes that participate in some shortest path.

In the above example (which corresponds to the example Figure 1) the shortest path from *one* (source s) to *nine* (destination t) is of length 232 and there are four paths of that length from source to destination.

Submission Guidelines

Implement all your classes and functions (including main) in a single file called *roll-number.cpp*. Tar all your sources and make a zip file named *roll-number*.tar.gz to upload it to moodle.