ECE368 Project #4: Map Routing
Due July 30, 2017 11:59PM

Description:

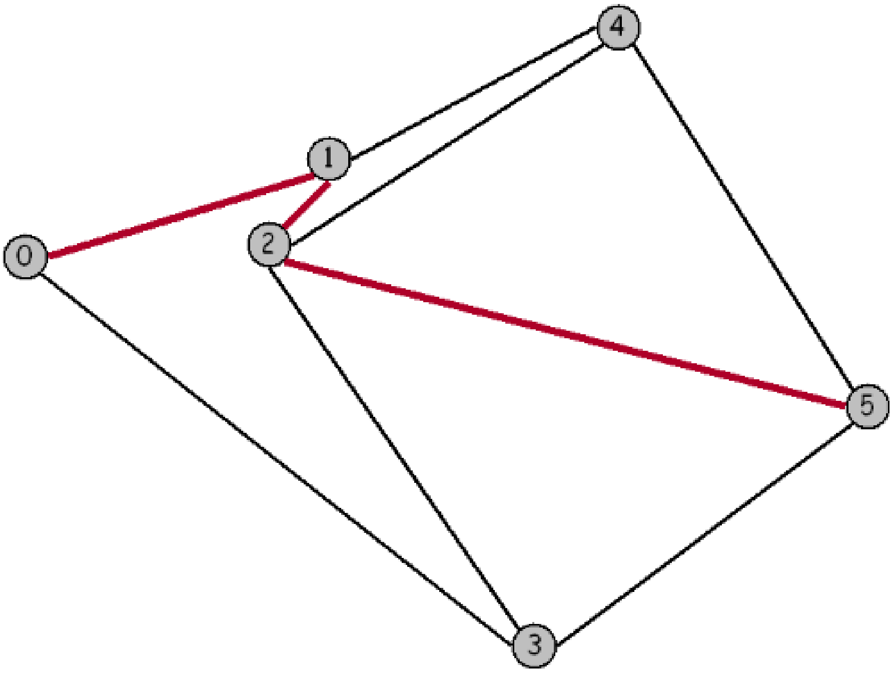
In this project, you will implement Dijkstra's shortest path algorithm for weighted undirected graphs. Variants/enhancements of this algorithm are widely used in geographic information systems including MapQuest and GPS-based car navigation systems. For a description of Dijkstra's algorithm, please see Wikipedia. (https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm)

You may either work alone or with a partner on this project. Use Piazza or talk to others in class to get a partner if desired. Only one person per team should submit the project. In your report you should have both names at the top.

Input

You will be given two input files. The first input file will represent a map, which is an **undirected** graph whose vertices are points on a plane and are connected by edges whose weights are Euclidean distances. Think of the vertices as cities and the edges as roads connected to them. Multiple nodes in a map may have identical coordinates. To represent such a map in a file, we list the number of vertices and edges on the first line, then list all the vertices (index followed by its x and y coordinates), and then list all the edges (pairs of vertices). For example, the input shown on the left below represents the map shown on the right below:

```
6 9
0 1000 2400
1 2800 3000
2 2400 2500
3 4000    0
4 4500 3800
5 6000 1500
0 1
0 3
1 2
1 4
2 4
2 3
2 5
3 5
4 5
```



The second input file contains a list of search queries with the first line containing the number of such queries (this is just to make your job of reading the file easier) and each of the following lines containing one query in the form of a source vertex and destination vertex pair. For example, the query input that is listed below contains two queries, one from node 0 to node 5, and the other from node 4 to node 3.

```
2
0 5
4 3
```

Goal: Given a map file and a query file as inputs, your goal is to compute the shortest path from each source listed in the query file to the corresponding destination using Dijkstra's algorithm. Your program takes the name of the map file and the name of the query file. Given these files, your program should then compute the shortest path for each query in the query file.

For example, given the files map6.txt and query2.txt, we expect your program (**shortestpath**) to produce the following output:

```
$./proj4 map6.txt query2.txt
6273
0 1 2 5
5245
4 5 3
INF
6 7
```

In this example, your program prints out four lines, where:
Line 1 and 2 contain result for the 1st query in the input file `query2.txt`

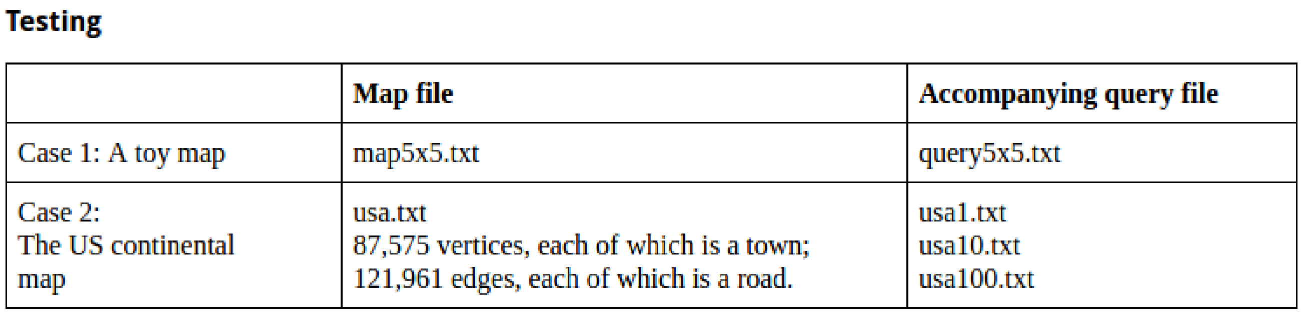
Line 1: the shortest distance from node 0 to node 5, which is 6273
Line 2: all nodes on the shortest path from node 0 to node 5, delimited by a space character

Line 3 and 4 contain result for the 2nd query in the input file `query2.txt`

Line 3: the shortest distance from node 4 to node 3, which is 5245
Line 4: all nodes on the shortest path from node 4 to node 3, delimited by a space character

Line 5 and 6 contain result for the 3rd query in the input file `query2.txt`

Line 5 & 6: node 6 and 7 are disconnected and their distance is infinity.



Testing

	Map file	**Accompanying query file**
Case 1: A toy map	map5x5.txt	query5x5.txt
Case 2: The US continental map	usa.txt 87,575 vertices, each of which is a town; 121,961 edges, each of which is a road.	usa1.txt usa10.txt usa100.txt

A sample output of Case 1 (path may not be unique):

```
7800
6 7 8 13 18 23
```

Grading:

Correctness (7.5 points):

- Program doesn't crash on any inputs, up to the max number of 100,000 vertices with corresponding edges.
- Gives the correct output.

Performance(1.5 point):

- Speed (.75 points): Is the speed within 150% of the provided binary.
  - If the provided binary takes 10 seconds, yours must run in 15 seconds, this will be tested on the largest test case
- Memory usage (.75 points): Is memory usage within 150% of the provided binary.
  - If the provided binary uses 1,000 bytes on the program, yours can take a maximum of 1,500.
  - Will be tested using Valgrind total heap usage. This will be tested on the largest test case.

Report(1 points):

- Format (PDF), clarity, quality, complexity analysis.

BONUS:

Submitting by July 27, 2017 11:59PM and scheduling an appointment with the Instructor to explain how you did the project.

- UP TO 2 points. Must be able to answer algorithm, code, conceptual questions on the project and algorithm.

Final Words:

The Euclidean distance between vertex A with coordinates (x,y) and vertex B with coordinates (p,q) is given by:

$$\sqrt{(p-x)^2+(q-y)^2}$$

You are allowed to store the edge weights (Euclidean distances) as integers. You may also assume that the x and y coordinates of any vertex in the map file will be less than or equal to 10,000 and that the number of vertices in the map file will be less than or equal to 100,000.