ECE368 Fall 2014: project3

ECE368 Project #3: Map Routing

Milestone 1: Due Nov 25, 11.59pm
Milestone 2: Due Dec 02, 11.59pm
Final Submission: Due Dec 11, 11.59pm

Note

- This project is to be completed on your own. You need to submit deliverables for two milestones and the final results.
- Please use Shay from the very beginning to code, debug, and submit your program
- If you prefer a local editor, you can map your Shay home folder locally as a network drive, see here.

Description

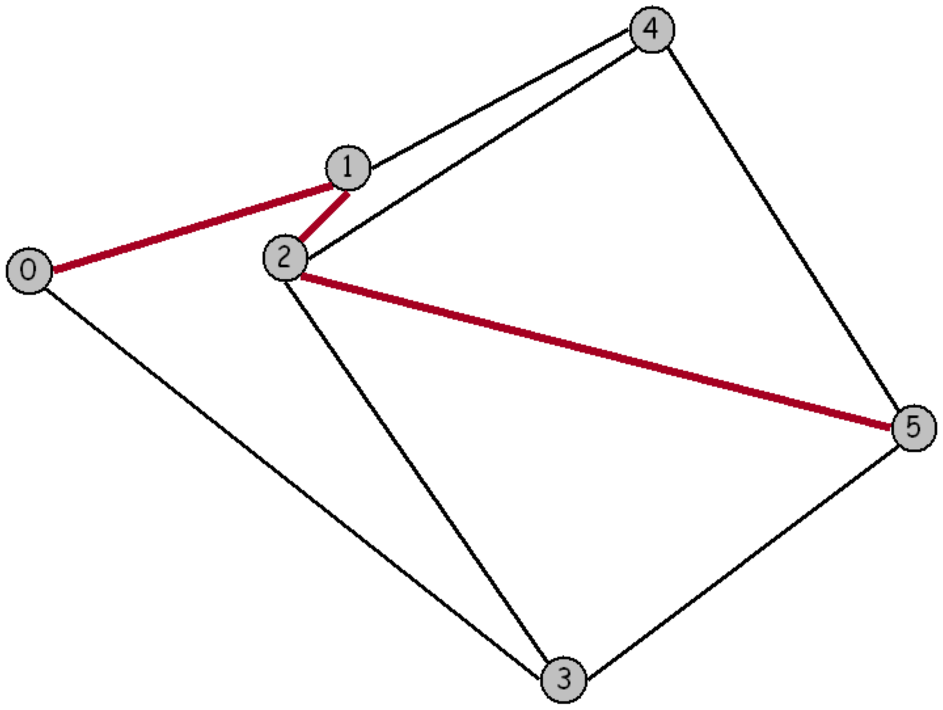
Summary

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.

Input

You will be given two input files. The first input file will represent a map, which is a 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. 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 take 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:

```
$ ./shortestpath map6.txt query2.txt
6273
0  1  2  5
5245
4  5  3
```

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 2: the shortest distance from node 4 to node 3, which is 5245
    Line 2: all nodes on the shortest path from node 4 to node 3, delimited by a space character

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

All test files

Update (12/8/2014) A sample output of Case 1 (path may not be unique):

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

Deliverables

Milestone 1

Due Nov 25, 11.59pm

Write a few paragraphs (around 400 words in total) to explain your overall approach to the problem. In particular, how are you going to parse the input files? What data structures are you going to use to represent graphs? Use your own words to briefly describe the Dijkstra algorithm.

To submit:

```
  turnin -c ee368 -p proj3-ms1 milestone1.pdf
```

Milestone 2

Due Dec 2, 11.59pm

Submit the source file of a program (**adjacent**) that takes in a map file and prints the adjacent list representation. For instance:

```
$ ./adjacent map6.txt
0: 1 3
1: 0 2 4
2: 1 3 4 5
3: 0 2 5
4: 1 2 5
5: 2 3 4
```

Each line starts with the index of a vertex, followed by the list of vertices that are adjacent to the first vertex.
To submit:

```
  turnin -c ee368 -p proj3-ms2 adjacent.c
```

Final

Due Dec 11, 11.59pm

The project requires the submission (electronically) of the C-code (source and include files) and a report detailing the results. A template is provided for the report. You should start with this template, fill in the various sections, and turn in a PDF version. Each report should not be longer than one page. To submit:

```
  turnin -c ee368 -p proj3 file1.c file2.c ... file1.h file2.h ... report.pdf
```

where `file1.c`, `file2.c`, *etc.*, are the names of the files that contain your code. Make sure that you submit all your source files and also list them in the report. You will be responsible for any files that you forget to submit.

Grading

Your submission will be evaluated based on the correctness of your program, readability of your code, as well as the quality of your report. Your program will be compiled using the following command (if you use any special libraries, *e.g.,* math.h, which need additional compilation flags, explicitly mention those in your report):

```
gcc -Werror -Wall file1.c file2.c ... -o proj3
```

Final words

The Euclidean distance between vertex A with co-ordinates (x,y) and vertex B with co-ordinates (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 co-ordinates 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.