

# ECE368 Project #3

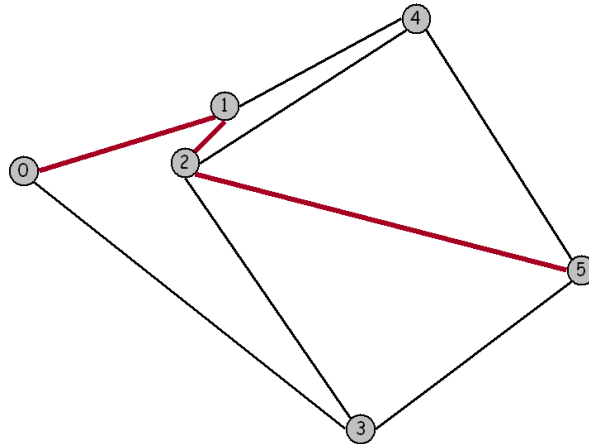
*Due on Friday, December 8, by 5:00pm*

**Modalities:** This project is to be completed by a team of two students. You can either work with the same project partner from Project 2 or decide to pick a new partner. Each team should inform me (vr@ecn.purdue.edu) of the names of the team members by Friday, December 1st.

**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 [http://en.wikipedia.org/wiki/Dijkstra's\\_algorithm](http://en.wikipedia.org/wiki/Dijkstra's_algorithm).

**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 file map6.txt represents the following map:

```
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 file query2.txt 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 should first ask the user to enter the name of the map file. The program should then ask the user to enter 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`, your program must produce the following output:

```
The shortest distance from node 0 to node 5 is 6273.
```

```
The shortest path from node 0 to node 5 is 0 -> 1 -> 2 -> 5
```

```
The shortest distance from node 4 to node 3 is 5245.
```

```
The shortest path from node 4 to node 3 is 4 -> 5 -> 3
```

**Testing:** You can use the small map file `map5x5.txt` and the accompanying query file `query5x5.txt` to test your program. Feel free to modify these files for testing if you wish. For the more adventurous, you can use the map file `usa.txt` and the query files `usa10.txt` and `usa100.txt` for testing. The map file `usa.txt` contains 87,575 vertices (towns) and 121,961 edges (roads) in the continental United States.

**Deliverables:** 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 and fill in the various sections. Only one report and electronic submission is required per team. Each report should not be longer than 2 pages. Turnin will be electronic and can be done using the following command:

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

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:

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

**Comments:** 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.