Assignment 6: Mapping Flight Routes

Out: 12th April 2014 **Due: 21st April 2014 at 11:55pm**

Every location on earth has a latitude and longitude (i.e. GPS coordinates). The shortest distance between two points on earth is not a straight line, because the shortest path has to be along the earth's curvature. Therefore the shortest flight routes are seldom in a straight line. In this assignment you will determine the shortest route for a flight between any two locations in the world. You will accomplish this by creating a graph and implementing the Dijkstra's shortest path algorithm for it.

Provided code and what it means

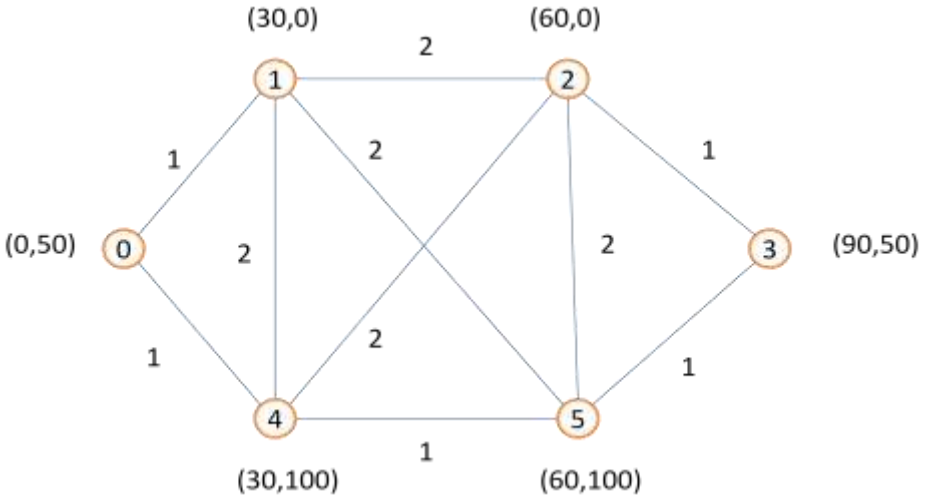
Two programs have been given to you, both of which share some common source code.

Program 1: GraphTest (use this program to debug)

An empty Graph class has been provided to you. In this assignment, all you have to do is fill its functions and add instance variables appropriately.

1. Graph vertices have an id, which is provided to you when the vertex is added using addVertex. You can assume that graph vertices are integers that begin at 0 and are contiguous.
2. Each vertex has a GPS location associated with it, which is simply a GPSLocation object (class provided).
3. You must represent connectivity as a map-of-maps, as discussed in class. **Remember that this is a directed graph, so each edge added in addEdge must be only one way.**
4. The shortest function must do the following:
    a. Find two vertices 's' and 't' that are closest to the provided start and end location. You must do this by using the getDistance function from the GPSLocation class.
    b. Using Dijkstra's shortest path algorithm, determine the shortest path between 's' and 't'. A PriorityQueue class has been provided to you to use for this.
    c. Return a path as a vector of GPSLocation objects. The path should start with the provided start location, followed by all the locations in the shortest path determined in (b) and finally the provided end location.

A small graph of 6 vertices and 20 edges has been provided to you in "smallgraph.txt", that looks like this:



A test program "GraphTest.cpp" has been provided to you. This program already reads in the provided graph file, and has hardcoded start and end locations.

After you have completed the Graph class:

1. Use the provided Makefile to build this program by typing “make GraphTest”.
2. When you run the program as:

   “./GraphTest smallgraph.txt”

   It will create a Graph object, read the graph using the provided file, compute a shortest path between the two hard-coded positions (from 0,50 to 90,50) and print the result. The result should look as follows:

```
Read a graph with 6 vertices
and 20 edges.
done
Path has 6 vertices.
(0,50)
(0,50)
(30,100)
(60,100)
(90,50)
(90,50)
```

   You can test your program by choosing alternate starting and ending locations.

Program 2: Flight Planner (use this program to see results visually)

Note:

1. **This program will work only on Apollo. It may or may not work on your own computer locally.**
2. **The accompanying graphs are quite large, so it is recommended that you use the first program to debug and use this one to ensure robustness.**

This program uses your Graph class and graphs provided to you in “small-worldgraph.txt”, “medium-worldgraph.txt” and “large-worldgraph.txt”.

After you have completed your Graph class works:

1. Build this program by typing “make FindFlightPath”. Typing simply “make” will build both programs.
2. Run this program by typing “./FindFlightPath graph-file-name”, replacing graph-file-name with any one of the above three graph files.

   After the program loads in the graph file (which may take a few seconds depending on which of the three files you choose), it will open a window with the world map.

   a. Press “G” to show a globe. You can use the left mouse button to drag this globe to rotate it.
   b. Press “M” to show the world map (flat). The left mouse button has no effect in this mode.
   c. To determine a flight path (in any mode):
      (i) Right-click anywhere on the globe or map to denote a starting position.
      (ii) Right-click anywhere else on the globe or map to denote an ending position. After this second right-click, the program will use your Graph class and provided graph to compute the shortest flight path between the two points, and draw it in magenta!

Steps (a) and (b) should work as-is, but Step (c) will work successfully only after you have completed the Graph class.

What to submit

Please submit all .cpp and .h files (including the ones provided to you), a **Makefile** and “medium-worldgraph.txt” in a single zipped file using Reggienet.