ECE 650 Assignment 3: Fall 2016
Inter-Process Communication: Load Balancing
Due: 11:59 PM 20th November 2016

Objective

For this assignment we will shift, for the moment, from systems programming to working on an application dealing with data structures and algorithms. However, when developing this application you should know that we will ultimate be deploying it within a client/server (cloud) environment which will receive requests from remote clients, and process those requests. Knowing this fact will likely affect your design. The specific application we will develop is a simple road-map system.

After this assignment, students will have a good understanding of, and ability to create, manipulate, and search over C/C++ dynamic data structures, together with a reasonable comprehension of how to compute the shortest path in a directed, weighted graph. In addition, students should have a good understanding of a C/C++ Application Program Interface, and its importance in program design.

Requirements

Consider a simple road map: it contains points of interest, roads, intersections, speed limits, etc. A map can be modeled as a weighted, directed graph. The vertices of the graph are points of interest and/or intersections between roads. Each vertex will have a type, which for now will be either a point-of-interest or an intersection. A point of interest might be "The University of Waterloo Davis Center", an address, *etc.*). Fundamentally, it is a place you are driving to/from. An intersection is any point at which two or more road segments meet. Note than an intersection may also be a point-of-interest, but a point-of-interest need not be an intersection.

Edges between vertices represent roads. The edges are directed because not all roads are two-way. Further, construction, accidents, etc., may cause one of the directions to be closed or have a significantly lower speed than the other direction. Each edge will have two weights: one weight is the specified speed limit (in kilometers per hour), while the second weight is the length of the edge (in kilometers). An edge may have additional labels, which we will call "events." An event on an edge may be a hazard (*e.g.*, a car stopped, debris on the road, *etc.*, or a closure.

Roads are defined as one or more edges that form a path (in graph theory terminology) together with a name.

The purpose of a map is to allow planning for travel from place A to place B, where A and B are vertices in the map (typically points-of-interest, not intersections). In generally, the shortest path is preferred for a trip.

For this assignment you are required to design and implement the following APIs for this mapping program (in C/C++):

```
addVertex(type, label);
addEdge(vertex1, vertex2, directional, speed, length);
edgeEvent(edge, eventType);
road(edges[]);
```

```
trip(fromVertex, toVertex);
vertex(point-of-interest);
store(filename);
retrieve(filename);
```

Since most roads are two-way, adding an edge should specify if it is directional, and if it is not it should simply add both directed edges.

The vertex API should simply return the vertex associated with a specified point-of-interest, or identify if such a point is not known.

The trip API takes two vertices and should find the shortest path between those vertices. Note that in finding the shortest path, it should take into account whether or not a road-closure event has occurred.

The store and retrieve APIs are intended to allow the storage and retrieval of whatever the current state of your map system is. You need not worry about this being sophisticated (that is a database problem), but it is necessary to be able to build a substantial map system.

Note that this assignment requires the design and implementation of an API, but not the main routine to use it. You will need to write a main routine to call and test tests functions. You may wish to specify a simple input language so that you can drive tests of your system accordingly. In particular, since code without tests is of little value and tests run manually are problematic, you are required to generate several test cases, which can be comprised of a mixture of scripting and main() routines.

Deliverables

Submit the following items to a dropbox on Learn before the deadline. We will grade the last submission when there are multiple submissions of the same item.

1. Source code together with test cases.
   Zip the entire source code together with other documents identified below and name the file `assignment3.zip`.

2. An assignment report (pdf format, named "report3.pdf") which containing design choices and results of your two programs.

3. Run experiments multiple times, and compute averages and deviations.

You may work in groups of two for this project. To facilitate this, please clearly identify both group members on your report.