Algorithms and Data Structures (INFR 2820U)
Fall 2012
Final Project for IT skills workshop
Due Dec 6, 2012 by 11:59 p.m.

General instructions:

a. You can work in groups of up to 4 members.
b. The final project is a group project, so every member must participate. Your report needs to specify the contribution of each member

Warning: the TA will be using a new software tool that compares executable codes for similarities. All submissions will be checked, even against submissions from previous years

I- Learning Objectives of the Project

In this project, students will learn to implement graphs, update graphs, traverse graphs, and find the shortest paths in graphs, as well as team work.

II- Introduction

A data communication network such as the Internet consists of routers connected by physical (optical) cables or links. Each router can act as a source, a destination, or a forwarder of data units, called a packet. System administrators typically model a data communication network as a graph, with each router corresponding to a vertex and the link or physical connection between two routers corresponding to an edge between the vertices. The weights on the edges of the graph are typically computed based on the data transmission time, the transmission cost, and available bandwidth, and the reliability of the link. Typically each router has a complete representation of the network graph and associated information available to it.

The network topology can change dynamically based on the state of the links and the routers. For example, a link may go down when the corresponding cable is cut, and a vertex may go down when the corresponding router crashes. In addition to these transient changes, changes to a network occur when a link is added or removed.

To route packets in a network, routers use different types of routing protocols: one of the widely known routing protocols is called the Open Shortest Path First (OSPF) which uses Dijkstra's shortest path algorithm to find shortest paths between communicating routers. Border Gateway Protocol (BGP) is another protocol which is mostly used in CISCO routers. The BGP routing protocol uses the Bellman-Ford algorithm to populate the routing table. Both the algorithms calculate the shortest path to all nodes in the graph from a single source and save the information in the routing table. Each router uses its routing table to decide to which next router to forward a packet that is going a certain destination.

III- Your task

Your task for this project is to build a graph representing a network that spans cross Canada: Each major Canadian city (Vancouver, Edmonton, Calgary, Saskatoon, Regina, Winnipeg, ThunderBay, Sudbury, Kitchener, Hamilton, Toronto, Barrie, Oshawa, Peterborough, Kingston, Ottawa, Montreal, Sherbrook, Quebec, SaintJohn, Moncton, Halifax, CapeBreton) host one router labeled with the name of the city. We will simply use the distance between physical routers as the weight on the edge connecting their corresponding vertices (you can use maps.google.ca to find the distance between cities). You need to compute the best path using the criterion of minimizing the total time taken for a data packet sent from a router to reach its destination. The shortest time path minimizes the sum of the weights of the links along the path.

For this final project, your task is to write a C++ program to:

a) Build the initial graph of the network from an input file
The initial state of the graph should be read from a file called *network.txt*. the format of the file is quite simple: Each link representing an edge is listed on a line, and is specified by the names of its two vertices followed by the weight. Vertices are simply strings (cities names with no spaces) and the weights are floating point numbers. For the above example, the file will look like:
Ottawa Montreal 199
Ottawa Kingston 196
Oshawa Whitby 9
Oshawa Toronto 61
Oshawa Kingston 205
:
:

b) **Update the graph to reflect changes to the network, including**:
1. Add an edge with a given weight between two vertices (if no edge between the two vertices exists)
2. Delete an edge between two vertices
3. Mark an edge between two vertices as "down" and therefore unavailable for use. Mark an edge as 'up' again if it is marked as 'down'.

c) Find and display the shortest path from a vertex to all other vertices in the graph based on its current state
This should compute the shortest path from one vertex to all destination vertices using **Dijkstra's** algorithm or **Bellman-Ford** algorithm, based on the current state of the graph. For the display, it doesn't have to be sophisticated -- even text based display is accepted.

d) Find and display the shortest path from a vertex to one other vertex in the graph based on its current state
This should compute the shortest path from one vertex to one other destination vertices using **A*** algorithm, using the distance between the cities as the heuristic.

For the display, it doesn't have to be sophisticated -- even text based display is accepted.

e) **Quit**
Simply cause the program to exit.

f) Optional and Bonus: Compute and display the minimum spanning tree (MST) of the graph
Use **Prim's** or **Kruskal's** algorithm to find and display the minimum spanning tree of the graph, based on the current state of the graph. For the display, it doesn't have to be sophisticated -- even text based display is accepted.

Note:

You are free to design and implement the 'user interface' of your program, which prompts the user with input options, and then accepts the user's inputs.
Simple text based interface is acceptable. Graphic user interface might be considered for bonus marks.

For output, as stated in section III, text based display is acceptable. Graphic output display might be considered for bonus marks.

IV- Your report

Each group should submit one report (12 point font, no less than 5 pages excluding the cover page) describing your implementation of the algorithms. This is an argumentative Essay with technical information.
The report is due at the same time as your implementation submission (in addition to submission via Blackboard, a hard copy may also be required).

The report MUST include:

- A cover page that includes the full name of all group members
- A detailed description of the algorithms you implemented. (Detailed description means that you explain in a high level WHAT the algorithm does, then go into the specifics showing how it works each step of the way with diagrams, images, screen shots, code.
- Description of **WHO** implemented **WHAT** tools/algorithms in your game.
- **README** file describing how to run your program

V- Your submission

Each group should submit:

- The c++ program file(s).
- The input file *network.txt*
- The report.