ECE250: Lab Project 4

Due Date: Saturday, April 1, 2017 – 11:00PM

1. Project Description

The goal of this project is to implement the Dijkstra's algorithm for finding the shortest path between two nodes in a graph. Forty percent of the grade will be attributed to run-time relative to other students in your class. The class average and standard deviation will be calculated, and those achieving the class average will get 80% (provided that their codes pass all test cases). Other grades will be based on 10 marks per standard deviation in either direction.

2. How to Test Your Program

We use drivers and tester classes for automated marking, and provide them for you to use while you build your solution. We also provide you with basic test cases, which can serve as a sample for you to create more comprehensive test cases. You can find the testing files on the course website.

3. How to Submit Your Program

Once you have completed your solution, and tested it comprehensively, you need to build a compressed file, in tar.gz format, with should contain the file:

- Weighted_graph.h

Build your tar file using the UNIX tar command as given below:

- *tar –cvzf xxxxxxxx_pn.tar.gz Weighted_graph.h*

where *xxxxxxxx* is your UW user id (i.e., jsmith), and *n* is the project number which is 4 for this project. All characters in the file name must be lower case. Submit your tar.gz file using LEARN.

4. Class Specifications

The Weighted_graph.h class allows the user to create and destroy an undirected weighted graph. You will be able to find the shortest distance between two connected vertices. The vertices are numbered 0 through $n - 1$ where n is the argument to the constructor.

Member Variables

You may define whatever member variables as you wish.

Constructor

Weighted_graph (int n = 50)

Construct an undirected graph with n vertices (by default, 50). If $n \leq 0$, use $n = 1$.

Destructor

~ Weighted_graph ()

Clean up any allocated memory.

Accessors

This class has four accessors:

- *int degree(int n) const* - Returns the degree of the vertex *n*. Throw an illegal argument exception if the argument does not correspond to an existing vertex. (**O**(1))
- *int edge_count() const*- Returns the number of edges in the graph. (**O**(1))
- *double adjacent(int m, int n) const* - Returns the weight of the edge connecting vertices *m* and *n*. If the vertices are the same, return 0. If the vertices are not adjacent, return infinity. Throw an illegal argument exception if the arguments do not correspond to existing vertices.
- *double distance(int m, int n)* - Return the shortest distance between vertices *m* and *n*. Throw an illegal argument exception if the arguments do not correspond to existing vertices. The distance between a vertex and itself is 0.0. The distance between vertices that are not connected is infinity.

Mutators

This class has one mutator:

- *void insert(int m, int n, double w)* - If the weight $w \leq 0$, throw an illegal argument exception. If the weight $w > 0$, add an edge between vertices *m* and *n*. If an edge already exists, replace the weight of the edge with the new weight. If the vertices do not exist or are equal, throw an illegal argument exception. (**O**(1))