Homework 4: Graphs & Single Source Shortest Path Problems

Due Date Upload your submission on or before 10:00PM, 12/8/2016. 24 hours rule apply. Also, you need to follow the instructions given in blackboard site to upload your signature file.

Objective

In this programming homework you will implement a *simple version* of a data structure for graphs, and, use it to solve the *single source shortest path problem* where all the edges have non negative weights.

By completing this homework, you should gain a better understanding on the use of an *adjacency list* to represent a graph. In addition, you should see how a graph algorithm utilizes the adjacency list together with a priority queue to solve the single source shortest path problem via a greedy strategy. Hopefully you will be able to further improve this data structure for more general applications.

Introduction To complete this homework, you may like to review our class discussions on graph data structures and Dijkstra's Algorithm. You may find the materials in Chapter 22, Chapter 24 (Section 3) in Cormen's text helpful when working on the homework.

Graph You are expected to develop a *Graph* class which uses the *adjacency list* data structure to represent the graph internally. In our case, the set of vertices of the graph $G = (V, E)$ is $V = \{1, \ldots, n\}$ and an edge e in the graph from vertex i to vertex j is denoted by (i, j). Note that, in case the graph is an undirected graph, we will use the same notation (i, j) to represent the edge which connects i and j.

Basic Requirements The following are the basic requirements for the *Graph* class:

1. Use adjacency list to represent the underlying Graph.
2. Has member functions that allow the adding and removal of edges and vertices.
3. Can take in external data in csv file format and construct a graph base on the external data.
4. Can take in external data in csv file format and construct a graph with edge weights base on the external data.
5. Has a name function which assign names to the vertices. In this homework, the name of vertex i is either an integer i or a character.
6. Has member function(s) which will be able to draw the graph (by returning a `dot` file)

7 It must use the Dijkstra's algorithm to compute the shortest paths from a single source to all other vertices.

8. Has member function(s) which will be able to compute the shortest distance to each vertex from a source vertex.
9. Has member function(s) which will be able to display the shortest path from vertex i to vertex j either to the screen or as a colored path in the graph (by returning a dot file).

10 (Priority Queue) Dijkstra's Algorithm utilizes a priority queue to support its greedy strategy. You may reuse your implementation in HW2 or choose to implement the priority queue via other reasonable means.

Data We will provide csv files to represent graphs (both weighted or unweighted; directed/undirected) either a majority of our test cases. The number of vertices of the graphs in each of our test cases has at most 20 vertices.

Test Cases You are asked to create the *Graph* data structure(s) as `C++ classes`, conduct simple computational experiments (via the `C++` language). These experiments/test cases will help you to

- *Verify your implementation* of the *Graph* data structure
- *Verify your implementation and the correctness* of the *Dijkstra's* algorithm.

Submission

Your submission will contain a collection of `C++` header files (`.h` files), implementation files (`.cpp` files) and the main program (named as `main.cpp`). They are expected to have adequate comments.

The remaining details regarding this homework submission will be posted by our TA/Graders.