Homework 4: Graphs & Single Source Shortest Path Problems

Due Date Upload your submission on or before 5:00PM, 5/3/2016. 24 hours rule apply. Also, you need to upload your signature file on or before 11:59PM, 5/3/2016. 24 hours rule apply.

Objective

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

Introduction Due to time limitations, we only implement a graph data structure which is adequate for solving the single source shortest path problem via the Dijkstra's algorithm. To complete this homework, you may like to review our class discussions on these topics and the materials in Chapter 22, Chapter 24 (Section 3) in Cormen's text.

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.

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 simply the integer i.
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).

We will discuss these requirements in our lecture(s).

Data We will provide csv files to represent graphs (both weighted or unweighted; directed/undirected) a majority of our test cases. The number of vertices of the graphs in most of the test cases are between 20 to 50 vertices.

Experiment You are asked to create the *Graph* data structure(s) as `C++ classes` and conduct simple computational experiments (via the `C++` language) to:

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

  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 and this data structure works together to carry out computations and will be able to further improve this data structure for more general applications.

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.

In addition, you will need to submit two text files:

i. `readme.txt`:

it should contain a concise description of how the *Graph* class works. Include

1. a description regarding your implementation of the key operations.
2. Directions for the grader to verify that the adjacency list is being implementation as a linked structures.
3. Acknowledge the reference(s) you use, if any, in this file.

ii. `output.txt`: it should contain a sample execution result in the `ubuntu/g++` environment. This file can be obtained by

```
ubuntu>g++ *.cpp -o hw4
ubuntu>./hw4 > output.txt
```

It should be in a single zip file. It should be named via the following convention:

```
<SU-EMAIL>-<FIRST-Name>-HW4.zip
```

That is, if my SU email address is `abc111@syr.edu` and my first name is Andrew, then I should name my submission as

```
abc111-andrew-HW4.zip
```

The instruction on creating the signature file will be given separately. Other relevant information regarding the submission process will be posted by our grader within our blackboard site or via our Piazza forum.