CPSC 350: Data Structures and Algorithms
Fall 2022
Programming Assignment 6: Spanning the Gamut
Due: December 9, 2022 at 11:59 pm

The Assignment

In this assignment you will create a program that is capable of identifying a minimum spanning tree of an undirected, weighted graph using Kruskal's algorithm.

Your program will take as a command line argument the name of a file that contains the specification for an undirected, weighted graph, G. The file will have the following format:

- The first line will be an integer, N, that represents the number of nodes in the graph. You can assume the nodes are labeled with the values [0,N-1].
- The next N lines will represent the rows of the adjacency matrix, with each line consisting of N weights, represented as doubles.

You should process the file and create a weighted graph representation using the WGraph class we implemented together in class. Once the WGraph is created, your program should compute the MST for the graph using Kruskal's algorithm. This should be done by adding a method to WGraph, *computeMST()*, that works as follows:

1. Identifies a MST for the graph. Note that a given graph may contain several MSTs, but you only need to identify one.
2. Displays the cost of the MST (the sum of all the edges in the MST) to standard output
3. Displays the adjacency matrix representation of the MST (a NxN matrix where all edges are 0 except for the edges that make up the MST) to standard output

Note that you may create any additional classes you need to support your implementation (eg. a priority queue for to use for Kruskal's algorithm), but at a minimum your solution should contain the following files:

- main.cpp
- WGraph.h
- WGraph.cpp

Rules of Engagement

- You may not use any data structures from the C++ STL or other third-party libraries. Of course, to do the file processing you may use any of the standard C++ IO classes.
- For this assignment, you must work individually.
- Develop using VSCode and make sure your code runs correctly with g++ using the course docker container.
- Feel free to use whatever textbooks or Internet sites you want to refresh your memory with C++ IO operations, just cite them in a README file turned in with your code. All

code you write, of course, must be your own. In your README please be sure to include the g++ command for compiling your code.

Due Date

This assignment is due at 11:59 pm on 12-9-2022. Submit all your commented code as a zip file to canvas. The name of the zip file should be LastName_FirstInitial_A6.zip

Grading

Grades will be based on correctness, adherence to the guidelines, and code quality (including the presence of meaningful comments). An elegant, OO solution will receive much more credit than procedural spaghetti code. I assume you are familiar with the standard style guide for C++, which you should follow. (See the course page on Canvas for a C++ style guide and Coding Documentation Requirements.)

Code that does not follow the specification EXACTLY will receive an automatic 25% deduction. Code that does not compile will receive an automatic 50% deduction.