Lab 5 – Minimum Cost Spanning Tree

Cameron Burkholder

# Friday 11-20-2020

## Objectives

* Read and understand the lab
* Review textbook material on graphs
* Design format for storing graphs in files
* Plan my implementation and collect the appropriate source files from the textbook
* Build out some of the base classes

## Notes

* The goal of this lab is to build a function graph class and a format for storing graphs in a file.
* I am going to implement the graph class using the adjacency matrix implementation.
* Today, I want to build the graph class, and file format.
* I’m going to start by building the GraphADT class from the textbook (pg. 387)
* Now that the GraphADT class has been built, I’m going to use the author’s adjacency matrix implementation (pg 389). I did make a few changes for naming conventions. Namely, changing numEdge and numVertext to numEdges and numVertices. Also, I use v1 and v2 where the author uses i and j.
* Now that the graph class has been built, I want to outline the file format I’m going to use. What I’ve come up with is simple
  + Line 1 will simply be “\_GRAPH\_”. This is to verify the format is correct so far.
  + Line 2 will be “NUM\_VERTICES=n” where n is the number of vertices. This will let the program know how many vertices to read in.
  + The next n lines will consist of n integers separated by spaces.

## Results

* I’ve built the graph class that will be used based on the author’s implementation.
* I’ve designed the file format to use
* TODO:
  + Add a function to initialize the graph from a file
  + Add a function to write the graph to a file
  + Implement smart pointers somewhere
  + Write a function that creates a minimum cost spanning tree (find algorithms…)
  + Write main that tests reading graph from file, creating three mcst from three different vertices, and prints the graph to a file (as well as console)