Lab 5 – Minimum Cost Spanning Tree

Cameron Burkholder

I used a smart pointer to manage the main graph object in main.cpp.

All graph files can be opened with Nodepad.

# 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)

# Monday 11-23-2020

## Goals

* Address to-do list from previous page
  + Write function to read graph from file
  + Write function to write graph to file
  + Write function to print graph to console
  + Write function for mcst
  + Write main

## Notes

* I have created three methods in my adjacency matrix graph class: read, write, and print. Read and write take a string as an argument that represents a file to either read or write from/to. Print will print the graph to the console.
* Read will be a private function used by a special form of the constructor
* I’m gonna use a smart pointer to store the graph in main.cpp
* Now that everything is set up, I’m gonna write a function to generate a mcst from a given vertex that will return a pointer to the mcst using Prim’s Algorithm from the textbook.
* In my implementation of Prim’s Algorithm, I’ve used the author’s source code and helper functions as a guide. I had to look at them a lot before I understood what was going on. The way he wrote out the algorithm seemed much simpler than the code I was reading, and I still can’t match every bit one to one but I have enough understanding of what it is doing to modify it in order to accomplish my goals. I did have to edit the addEdgetoMST part because for whatever reason, in my code it was only marking the edges directionally. I accounted for this by simply mirroring the setEdge code.
* I additionally had a problem where if I did any of the vertices’ MST by itself, it worked fine, but if I tried to add more than one to main it would crash. I didn’t realize at first, but this was because I was never marking all the nodes as unvisited again after one of the mcst attempts ran.

## Results

* I’ve built the read, write, and print functions and all of them work as expected.
* I built the MCST function, which took a lot longer than expected, and got it to work in accordance with the lab directions. It seems as though all three starting vertices give the same mcst, and I’m not sure if that is because of the graph that I’m using or a flaw in the logic I implemented. Because my implementation heavily relies on the author’s, I’m going to hope it’s the former.
* After reviewing the instructions, I had to make a slight modification to my MCST function so that it returns a pointer to the MCST created, which can then be used to print it to a file.