Approach Document Lab 5

Lab 5 – Joyce Berdkan

# Assignment Objective

Implement two functions: one to read a graph from a file and the other to write a graph to a file.

# Assignment Requirements

Your system should have the ability to

* Implement the two functions
* Test the functions by implementing a complete MST program that reads an undirected graph from a file, constructs the MST, and then writes to a second file the graph representing the MST.

# Approach

* I will first read the instructions in good detail and review any concept that I am unsure about prior to the start of my program.
* Subsequently, I will create a new Visual Studios program and implement the code given in the book.
* I will determine whether I need to add any extra files.
* After concluding my understanding the assignment, I will begin the implementation for the following Graph.h functions:
* Void Init(int n);
* Int n();
* Int e();
* Int first();
* Int next();
* Void setEdge(int v1, int v2, int wgt);
* Void deleteEdge(int v1, int v2);
* Bool isEdge(int I, int j);
* Int weight(int v1, int v2);
* Int getMark(int v);
* Void setMark(int v, int Val);
* I will likely need to add header files with implementation of Graph called Graphm and create the main file.
* I will determine if I need to add any private variables
* Once I complete the implementation of the functions in Graph.h, I will determine the implementation of Graphm.
* I will add the book.h that was given in Source Files from text.
* I will add comments where an explanation may be required.

# Build Log

11/24/2022 - I reviewed the lab instructions and other documents today and have I've already written my strategy document. I entered the specified I uploaded the data file and loaded the files that I borrowed from the C++ Source Files into Visual Studios.

11/28/2022 - Today, I worked on the Graph.h implementation. I created the Graph.h header file and started working on the functions' implementation. The ADT assumes that the number of vertices is fixed when the graph is created, but that edges can be added and removed. It also supports a mark array to aid graph traversal algorithms. I learned that all functions needed are in the book. I then implemented all the functions in Graphm.h.

11/29/2022 – Today I fixed the minor issues as I implemented the functions in Graphm.h were some of the functions shouldn’t be in virtual. I looked over the book and oversaw which functions I can implement in main and see which ones should be used Dijkstra’s algorithm when using a MST or the Prim’s algorithm. I chose Prim’s algorithm since it shows the searches Prim’s algorithm do for the distance matrix for the next closest vertex. For each vertex I, when I is processed by Prim’s algorithm, an edge going to I is added to the MST that we are building. Array V[I] stores the previously visited vertex that is closest to Vertex I. This information lets us know which edge goes into the MST when Vertex I is processed. It also contains calls to AddEdgetoMST to indicate which edges are actually added to the MST.

12/01/2022 - I started on the main source file. I used a unique pointer as my smart pointer with my length being equated to 6. I added seven new functions to main.cpp because they were all being called by the smart pointer. I copied the code minVertex() and Prim() functions from the book. I altered with those functions so that the smart pointer would be effective. In my void process(), I called the file to be read and printed out. I created four different text files for the original matrix, the MST 1 starting at vertex 1, the MST 2 starting at vertex 4 and MST 3 starting at vertex 2.

Screenshots are in the page below:

