ECE250: Lab Project 4

Due Date: April 3, 2020 @11:00 pm

1. Project Description

The goal of this project is to write a C++ implementation for the **Minimum Spanning Tree (MST)** on a weighted undirected graph, using **Kruskal's Algorithm**. We consider the *n* nodes in the graph to be numbered from *0* to *n – 1*. This means a graph with 4 nodes has nodes named *0*, *1*, *2* and *3*. Each edge has a weight (a positive number of double type) associated with it. You can represent the graph as an adjacency matrix or an adjacency list.

In order to build the minimum spanning tree T, the Kruskal's algorithm adds one edge to the T (initialized with an empty graph) in each step. To make sure that this procedure does not form loops in the tree, we need to keep track of the connected components of T. In your implementation you will write a *Disjoint sets* class to assist you with this task. *Disjoint sets* is a well-known data structure for grouping *n* elements (nodes) into a collection of disjoint sets (connected components). You can read more information on disjoint sets from the corresponding course slides (week 12) as well as Chapter 21 of CLRS book. It is recommend to write the *disjoint sets* using linked lists.

2. Program Design

Write a short description of your design. You will submit this document along with your C++ solution files for marking. This document must include your design decisions. Please refer to the course website for "Programming Guidelines" and the expected content for your design document.

3. Project Requirements

Write a test program (named **msttest.cpp**) that will read commands from standard input and write the output to standard output. The program will respond to the commands described in this section.

Command	**Parameters**	**Description**	**Output**
n	*m*	*m* is the number of nodes. **Note**: This is always the first command and never is called again.	**success** **failure*** If m < 0.
i	*u;v;w*	Inserts an edge between nodes *u* and *v* with weight *w* (a double type).	**success** If the insertion was successful. If there is already an edge connecting *u* and *v*, this command will update the weight for the edge. **failure*** If *u* or *v* are outside the valid range, or *w* <= 0.

Command	Parameters	Description	Output
d	*u;v*	Deletes the edge between nodes *u* and *v*.	**success** If the deletion was successful. **failure** If there is no edge connecting *u* and *v*. **failure*** If *u* or *v* are outside the valid range.
degree	*u*	Returns the degree of vertex *u*.	**degree of** *u* **is** *d_u* If node *u* is a valid node. **failure*** If *u* is outside the valid range.
edge_count		Returns the total number of edges in the graph.	**edge count is** *n_edges* For an empty graph, the edge count is 0.
clear		Removes all the edges from the graph.	**success**
mst		Calculates the minimum spanning tree and returns its total weight.	**mst** weight **not connected** If the graph is not connected.

* Failure must be detected by handling an exception. That is, the tree function will "throw" an exception and the driver will "catch" the exception, creating the output "failure" for these cases. You can define an exception class called "illegal_argument" to be used for these cases.

The time complexity of Kruskal's algorithm depends on the implementation of the disjoint-set data structure. Implement **mst** command with the best possible running time based on your disjoint-set data structure. You can refer to Chapters 21 and 23.2 from CLRS for the running time that you need to add to your design document.

- **Test Files**

The course website contains example input files for the corresponding output files. The files are named *test01.in*, *test02.in* and so on with the output files named *test01.out, test02.out* and so on.

4. How to Submit Your Program

Once you have completed your solution and tested it comprehensively on your computer, you

have to transfer your files to the *eceUbuntu server* and test there since we perform the automated testing using this environment. Once you finish testing in the *eceUbuntu server*, you will create a compressed file (tar.gz) that should contain:

- A typed document (maximum three pages) describing your design. A document beyond 3 pages will not be marked. Submit this document in PDF format. The name of this file should be:

  xxxxxxxx _design_ p***n***.pdf in which ***xxxxxxxx*** is your UW user id (*e.g.*, jsmith) and ***n*** is the project number that is 4 (four) for this submission.

- A test program (**msttest.cpp)** that reads the commands and writes the output.
- Required header files and classes (ending in *.h .hpp .cpp*).
- A make file (named Makefile), with instructions on how to compile your solution and create an executable file named **mstdriver**

The name of your compressed file should be ***xxxxxxxx***_p***n***.tar.gz, where ***xxxxxxxx*** is your UW user id (*e.g.*, jsmith) and ***n*** is the project number that is 4 (four) for this submission.