CSCE 221 Cover Page
Programming Assignment #6

First Name Shaeeta Last Name Sharar UIN 822006676

User Name ssharar E-mail address ssharar@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero. According to the University Regulations, Section 42, scholastic dishonesty are including: acquiring answers from any unauthorized source, working with another person when not specifically permitted, observing the work of other students during any exam, providing answers when not specifically authorized to do so, informing any person of the contents of an exam prior to the exam, and failing to credit sources used. Disciplinary actions range from grade penalties to expulsion read more: Aggie Honor System Office

Type of sources			
People	Peer Teacher Lab		
Web pages (provide URL)			
Printed material	Textbook: Data Structure and Algorithims in C++		
Other Sources	Slides: Analyzing Algorithims	Piazza	Graph ADT ppt

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Your Name Shaeeta Sharar Date 05/05/2015

Report
Shaeeta Sharar

Program Description

The programming assingment called for implementing a graph structure. Using the provided supplemental code as well as prewritten Disjoint Set code from a past assignment. Part one required implementing basic build graph functions, including inserting edges and getting the weight. Part two required implementing a sorting edge function adn building a minimum spanning tree using Kruskal's algorithm.

Data Structures and Algorithm Description

The data structure that was focused on was the graph structure. We implemented this structure by using the adjaceny list method. There were two vectors, the Adjaceny List which stores linked lists and Edge List which stores edges. The linked list was implemented using nodes. The graph class itself used these two vectors to store verticies and edges and form the graph.

Algorithm Description

Algorithims:

Build Graph: O(n) - This function runs through the number of verticies of a graph and implements a new vertex into the adjacency list. This is done by creating a for loop which iterates until it reaches "n" the number of verticies. A new node of type vertex is made and then pushed back into the adjaceny list.

Insert Edge: O(1) - This function creates a new edge, makes new verticies and inserts them into the respective adjacency list indexes, then sets their edges to the new edge created. In the implementation, a new edge pointer was created and then pushed back into the edge list. Two verticies were created and pushed back into the respective adjacency list indexes (ie if the edge was from 0 to 1, the vertex of 0 was pushed back into the list for 1 and vice versa). Then the edges of these verticies were set using a set edge function created by me (of O(1)). Because this implementation creates two vertexes at once, an additional check of weights was added to the input to ensure that the edge list was not doubling edge inputs.

Get Weight: O(n) - This function traverses through the adjacency list at provided index and gets the weight of the edge. This is implemented by having a temporary node and using that to move through the list at index provided. While the node is not null, the function checks to see if an element exists. If it does but doesn't have an edge, then it returns 0. Otherwise the function returns the weight.

Sort Edge: O(1) - This function is simply the standard library sort. An additional global function which returns the comparisons of the weights of two verticies was implemented as well.

MST: O(n) - This is Kruskals algorithm. First this function creates a new disjoint set. It runs through the adjacency list and makes a set for each vertex. It then sorts the edges. Next, it runs through the edge list and if the sets of u and v are not equal, it pushes back the edge into MST and does a union of the two find sets. The function returns the value of the MST which is done in a seperate function, Value_Checker (O(n)). Value checker runs through the MST vector-1 and adds the values then returns that.

Program Organization and Description of Classes

The linked list was the main class. It contained the usual member functions such as accessors, insertion and deletion functions. The node class contained the element and pointers to the other nodes. The linked list class was a sort of shell for the node class. Building on this, the Disjoint Set class used a link list and a vector of nodes to store its values. Likewise, the graph contianed vectors of type liked list and of type edge. Edges and verticies were defind as classes in graph.h though they only contained simple contructors and set functions.

Instructions to Compile and Run your Program and Input and Output Specifications

To compile go into the directory which contains all the files. Once in the directory, simply use the "make clean", "make all" and "./main nameoftestfile.mat" commands to run as specified in the makefile. Using the g++ comands in the comand line could also work. When running, make sure to include the name of the graph file after the ./main as specified above.

Logical Exceptions

In the algorithims written, there are checks to ensure that that there are no null pointers being accessed and causing errors. Exceptions in the graph function are used to ensure that there are no negative columns or rows, edge numbers are correct, and the like.

C++ object oriented or generic programming features

Templates were used in the implementation of disjoint set as well as linked lists. Classes were used and often depended on each other as in the case of graph

which was made up of many smaller classes including edge, vertex, vector, and linked list. These classes inherited things from each other.

Tests