CSCE 221 Cover Page
Programming Assignment # 6

FirstName: Eric LastName: Gonzalez

UIN: 322002089

User Name: egoftcp

E-mail address: egoftcp@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, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources	
People	
Web pages (provide URL)	stackoverflow.com
Printed material	
Other Sources	CSCE 221 slides

I certify that I have listed all the sources that I used to develop the solutions/code 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 (signature) Eric Gonzalez Date 05/04/2015

Program Description

This program implements a graph structure in Part 1 by building the graph, inserting edges into it, and getting the weight of the edge between two vertices. Part 2 implements Kruskal’s algorithm to create a Minimum Spanning Tree. Input files are provided and outputs are in the form of a visual Adjacency Matrix and Minimum Spanning Tree for the graph.

Purpose of the Assignment

The purpose of this assignment was to test our understanding of the graph data structure and Kruskal's algorithm. We used these to implement a Minimum Spanning Tree.

Data Structures Description

The buildGraph() function first resizes the vertices and edges of the graph, then takes in n vertices implemented as DListNodes of <Vertex> type. It then uses a for loop to form a new node with the current i value and pushes said node into vector AdjacencyList. Finally, it sets the trailer to itself.

The insertEdge() function inserts a new edge into AdjacencyList. It sets the next node to be a new node with a vertex value of j, then sets the new trailer to be the newly created node. It then makes a new edge and pushes it into the EdgeList vector. Finally, it sets the edge of the vertex value equal to the newly created edge.

The getWeight() function returns the weight of the edge between two vertices. It uses a while loop and two if statements to check if the index value of the current spot in the AdjacencyList is equal to end value j. If so, it accesses the weight of that edge and returns it, assuming it is not a null edge; otherwise, it returns 0.

The sortEdge() function implements the built-in std::sort() function as a lambda function to sort all edges based on weight. It iterates through the entire tree and compares the weights of two edges a and b to see if a's weight is less than b's, sorting all edges in ascending order based on weight.

The MSTAlgo() function creates a disjoint set of vertices VertSet and sorts it using sortEdge(). It compares all edges in EdgeList and if the FindSet() value of the first is not equal to the FindSet() value of the second, then it adds an edge to the vector MST representation of the Minimal Spanning Tree, increments the total weight of the edges by the current edge weight between two vertices i and j, and adds i and j to VertSet with Union().

Runtime Analysis

buildGraph() = O(n)

insertEdge() = O(1)

getWeight() = O(max(adj(v))) [adj(v) = adjacent vectors of v]

sortEdge() = O(n log n)

MSTAlgo() = O(m log n) [m = edges, n = vertices]

Instructions to Compile and Run

Run the following in the program directory:

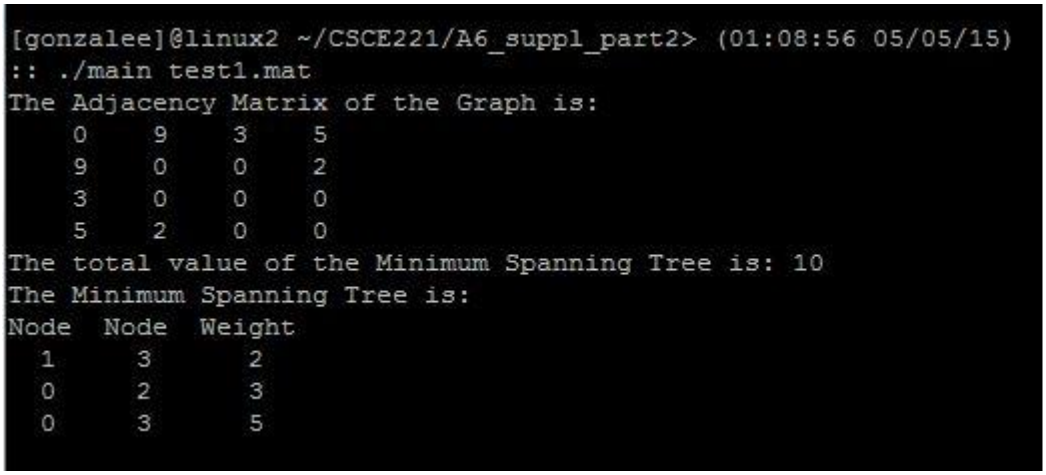
make clean

make

./main test1.mat OR test2.mat

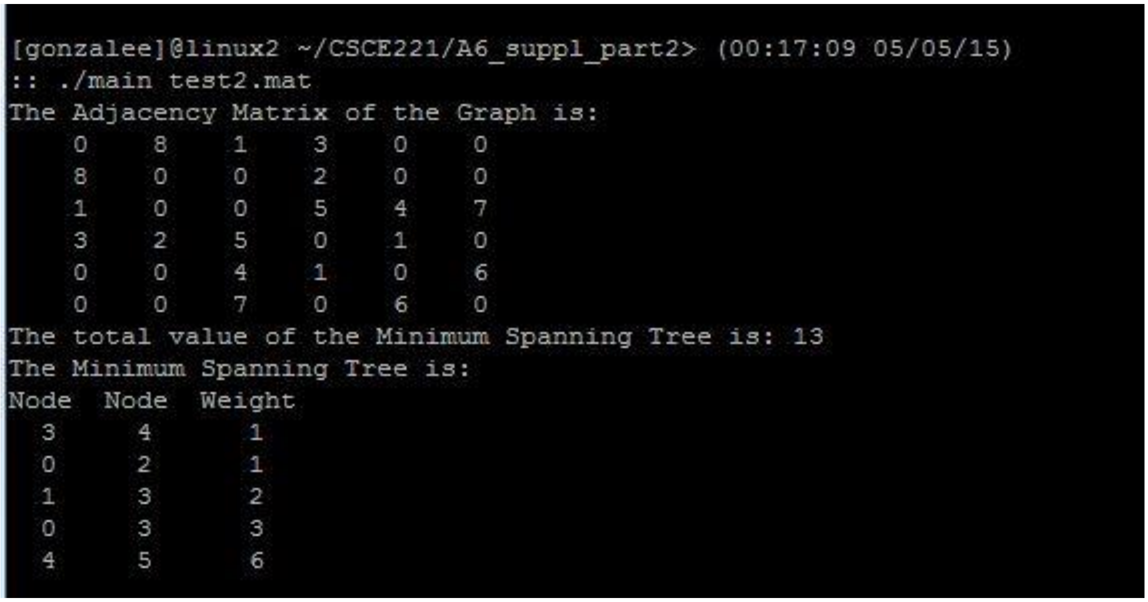
Testing Results

Test 1:



```
[gonzalee]@linux2 ~/CSCE221/A6_suppl_part2> (01:08:56 05/05/15)
:: ./main test1.mat
The Adjacency Matrix of the Graph is:
    0    9    3    5
    9    0    0    2
    3    0    0    0
    5    2    0    0
The total value of the Minimum Spanning Tree is: 10
The Minimum Spanning Tree is:
Node  Node  Weight
  1     3      2
  0     2      3
  0     3      5
```

Test 2:



```
[gonzalee]@linux2 ~/CSCE221/A6_suppl_part2> (00:17:09 05/05/15)
:: ./main test2.mat
The Adjacency Matrix of the Graph is:
    0    8    1    3    0    0
    8    0    0    2    0    0
    1    0    0    5    4    7
    3    2    5    0    1    0
    0    0    4    1    0    6
    0    0    7    0    6    0
The total value of the Minimum Spanning Tree is: 13
The Minimum Spanning Tree is:
Node  Node  Weight
  3     4      1
  0     2      1
  1     3      2
  0     3      3
  4     5      6
```