Minimum Spanning Tree and Prims Algorithm

Lab 12

By

John Kiyak

CS 303

October 28 2018

### **Problem Specification**

The goal of this assignment was implement Prim’s Algorithm, a weighted graph, and minimum spanning trees.

### **Program Design**

For this program, I made a Graph class function, and a main function. The Graph class was initialized with vertices. The printMST class prints the MST with Edge - Weight formatting. The primMST method sets the key to 0 so it starts at the vertex.

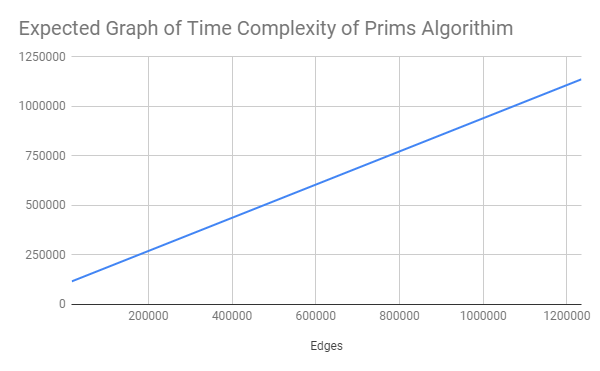
### **Testing Plan**

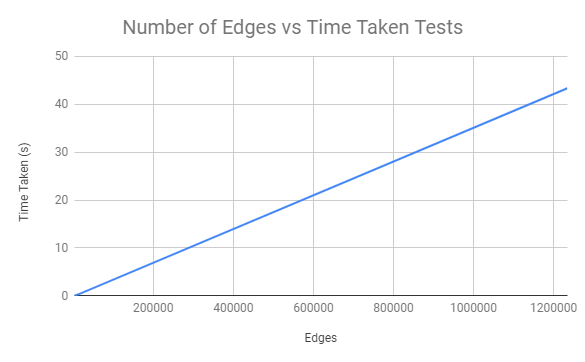
To test the program , I would make sure that the main method would be reading in the correct data from the input data, make sure the function runs without any errors, and outputs the expected output that the user should expect to see with the data constructed.

### **Test Cases**

The test cases are shown in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| Text File | “mediumDG.txt” | “largeDG.txt” | “XtraLarge.txt” |
| Vertices | 250 | 1000 | 10000 |
| Edges | 2546 | 16866 | 123462 |
| Time Complexity | O(ElogV) | O(ElogV) | O(ElogV) |
| Natural log | 14057 | 1165061 | 1137127 |
| Time Taken | .027998 seconds | .49097 seconds | 43.31325268 seconds |





So using the above data and the charts we can expect that the larger the graph is, aka more vertices and edges, the more exponential the running time of the algorithm takes. So using those numbers from the time complexity, I had a pretty good idea on how long each method of running the program would take, depending on the size of the input file.

### **Analysis and Conclusions**

Prim’s Algorithm has a time complexity of O(VlogV + ElogV), which can be simplified into O(ElogV). This pretty much means that as the number of vertices gets bigger, the edges gets bigger as well. So the bigger the tree gets, the longer it will take to process the minimum spanning tree and the algorithm itself.

### **References**

I used the GeeksforGeeks template for the algorithm.

### 