MST

Homework #12

By:

Pratyusha Thundena

CS 303L – L1A

November 27, 2018

### Problem Specification

The objectives included implementing weighted graph and Prim’s Algorithm for minimum spanning trees (MST).

### Program Design

This program required the following: BufferedReader class, File class, IOException class, FileReader class, util.\* class, tinyDG.txt, mediumDG.txt, largeDG.txt, and XtraLargeDG.txt.

The following steps were required to develop the program:

1. implement a program containing weighted graph class
2. create a driver program that reads input file mediumDG.txt and displays the adjacency list
3. implement Prim’s algorithm on mediumDG.txt
4. add code to read from the other input files: tinyDG.txt, largeDG.txt, and XtraLargeDG.txt in driver program
5. run Prim’s algorithm and find the minimum spanning tree (MST) for each txt file
6. print the adjacency list for all text file and the time it took to find MST

The following constructors and methods were defined within the class:

1. WGraph ()

Basic method that reads weighted graph information in WGraph.java.

b) reader ()

Constructor that reads string line in input files in WGraph.java.

c) st ()

Constructor that makes sure that the string is read properly in WGraph.java.

d) Integer (), Double ()

Constructors that read the numbers properly in text files in WGraph.java.

e) st ()

Constructor that allows string to be broken into tokens.

f) printAdjList ()

Basic method that prints the adjacency list in WGraph.java.

g) MST ()

Basic method that finds MST in WGraph.java.

h) printMST ()

Basic method that runs prints MST in WGraph.java.

i) e[o] (), queue ()

Constructors that instantiate objects in WGraph.java.

j) driver ()

Driver method that reads input files, prints adjacency list, and displays MST in Driver.java.

k) Filename (), FR1(), BR (), graph ()

Constructors used to instantiate objects in Driver.java.

l) TimeKeeper ()

Basic method that keeps track of total time to find MST in Driver.java.

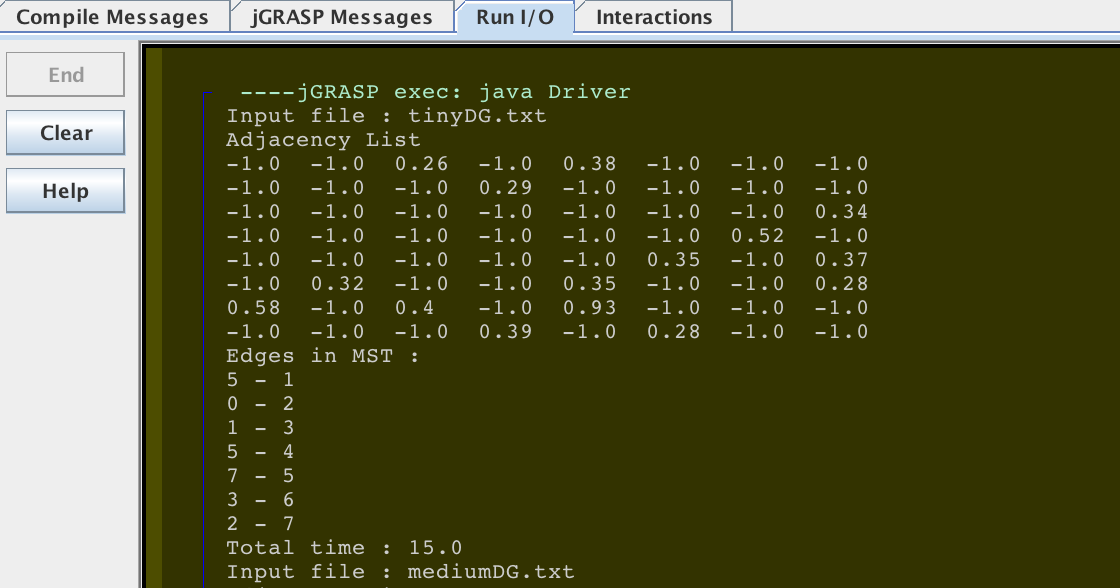
The println method of the System.out object displays the inputs and results for the driver program.

### Testing Plan

The test plan on mediumDG.txt involved the following: implementing weighted graph and Prim’s algorithm, displaying adjacency list, and recording the time it took to find MST. Then testing the algorithm on the other text files and displaying the results and time.

### Results

### Figure 1: Output Illustration



### Figure 2: Output Table

|  |  |
| --- | --- |
| Text File | Time (milliseconds) |
| tinyDG.txt | 15.0 |
| mediumDG.txt | 45.0 |
| largeDG.txt | 67.0 |
| XtraLargeDG.txt | 300.0 |

### Analysis and Conclusions

A minimum spanning tree (MST) would be a spanning tree of minimum weight in an undirected connected weighted graph. The Prim’s algorithm would be a “greedy” algorithm that begins with an empty spanning tee. In Prim’s algorithm, a cut was found and then the minimum weight edge from the cut was picked in the program. The result was displayed in Figure 1 and Figure 2. The time it took to find the MST increased linearly with the amount of data in the text files. So, it took the longest time to find MST for XtraLargeDG.txt file. The time required for one call to EXTRACT-MIN(Q) was equal to O (log V). The while loop in line 6 was executed V times. Hence, the total time required for EXTRACT-MIN(Q) was equal to O (V logV). The total time required to execute line 11 was O (E logV). The total time complexity of MST-PRIM was O ((V logV + (E logV + V)), which can be simplified to O (E logV).

### References

The parameters and input files (tinyDG.txt, mediumDG.txt, largeDG.txt, XtraLargeDG.txt) was provided in the homework assignment (by Dr. Bangalore) and Introduction to Algorithms (3rd ed.) was used to do the lab report.