Jay Offerdahl
EECS 560
Lab 11 Report

Lab 11 Report: Minimum Spanning Trees

My program is structured as follows: First, I read in from the text file. I assume the text file shows n vertices, which corresponds to n^2 values. I read in the values the same way I did for lab 6.

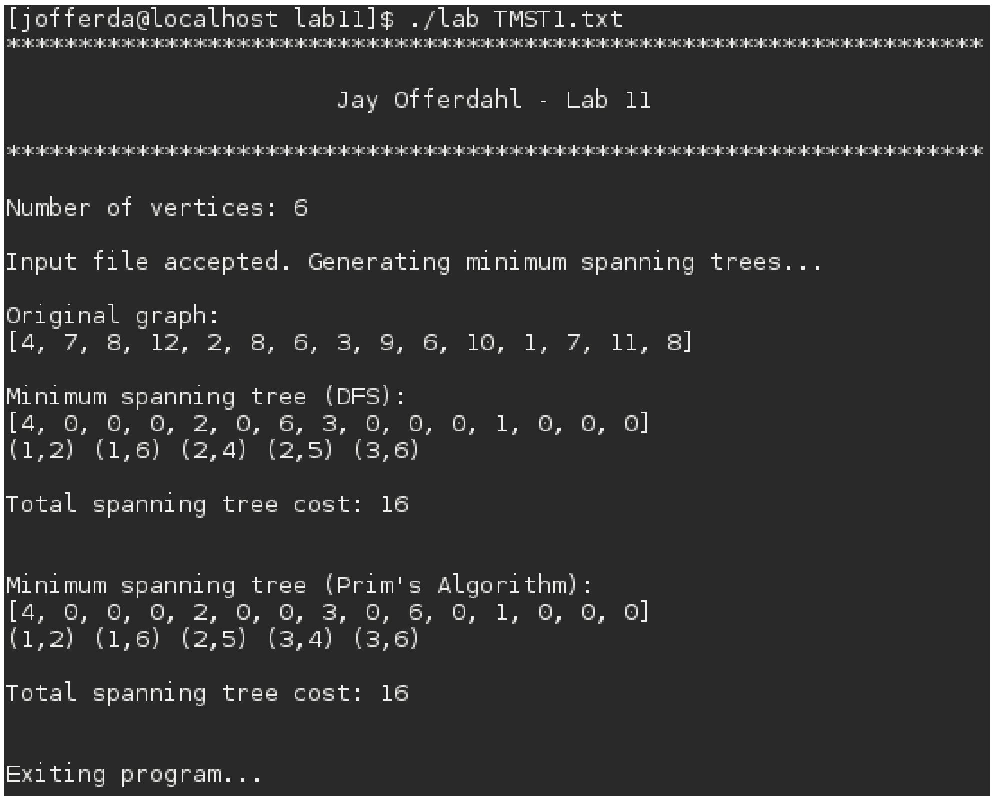
Next, I generate the minimum spanning tree using a depth first search. I modified my algorithm from lab 6 to determine when a cycle was found. When this happened, I stored necessary cycle information into a struct I created called CycleHelper. This allowed me to easily track the cycle information through the recursion. As soon as a cycle was found, the recursion would terminate (after going back up through all the layers and performing necessary operations). I then checked my cycle helper object outside of the dfs function call to see if a cycle was found. If so, I removed the largest edge that was found on the way out of the recursion. This method produces a minimum spanning tree over the connected graph.

Next, I generate another minimum spanning tree using Prim’s algorithm. This involved looping while there was a vertex that was unknown. I kept track of the “current vertex”, and upon entering the loop, I marked this vertex as known. After this, I looped through the current vertex’s connections, and if they didn’t connect to a known vertex and they were less than the lowest cost edge to that vertex, I updated the d_v table, as well as the p_v table. After I updated the tables, I found the lowest cost path to any unknown vertex. I then proceeded to remove this path and then iterate over the vertex that path lead to, which would effectively set that vertex as known in the next loop. It’s worth mentioning that I set lowestSoFar to 100,000, which I believed would be the “infinity” value as described in the text. When my program checked for the lowest so far value after updating the tables, a result of 100,000 would terminate the program, as the minimum spanning tree had been completed.

After both methods, I printed out my results. Please note that for TMST1.txt, the outputs differ, however, the costs of the minimum spanning trees are the same, which shows the difference in how each algorithm computes the minimum spanning trees. I’ve included my output for several different test files below.

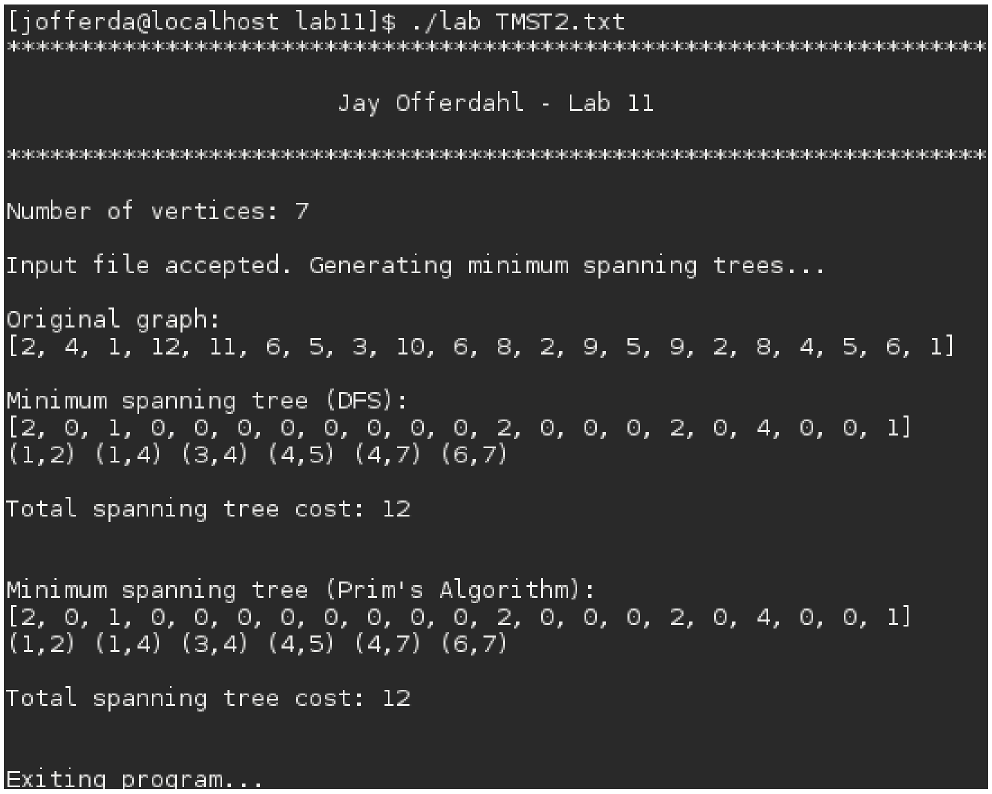
I have included all test text files with my submission, should you want verification on my output. Thanks, and see you next week.

Jay Offerdahl
EECS 560
Lab 11 Report



```
[jofferda@localhost lab11]$ ./lab TMST1.txt
*******************************************************************

                       Jay Offerdahl - Lab 11

*******************************************************************

Number of vertices: 6

Input file accepted. Generating minimum spanning trees...

Original graph:
[4, 7, 8, 12, 2, 8, 6, 3, 9, 6, 10, 1, 7, 11, 8]

Minimum spanning tree (DFS):
[4, 0, 0, 0, 2, 0, 6, 3, 0, 0, 0, 1, 0, 0, 0]
(1,2) (1,6) (2,4) (2,5) (3,6)

Total spanning tree cost: 16


Minimum spanning tree (Prim's Algorithm):
[4, 0, 0, 0, 2, 0, 0, 3, 0, 6, 0, 1, 0, 0, 0]
(1,2) (1,6) (2,5) (3,4) (3,6)

Total spanning tree cost: 16


Exiting program...
```

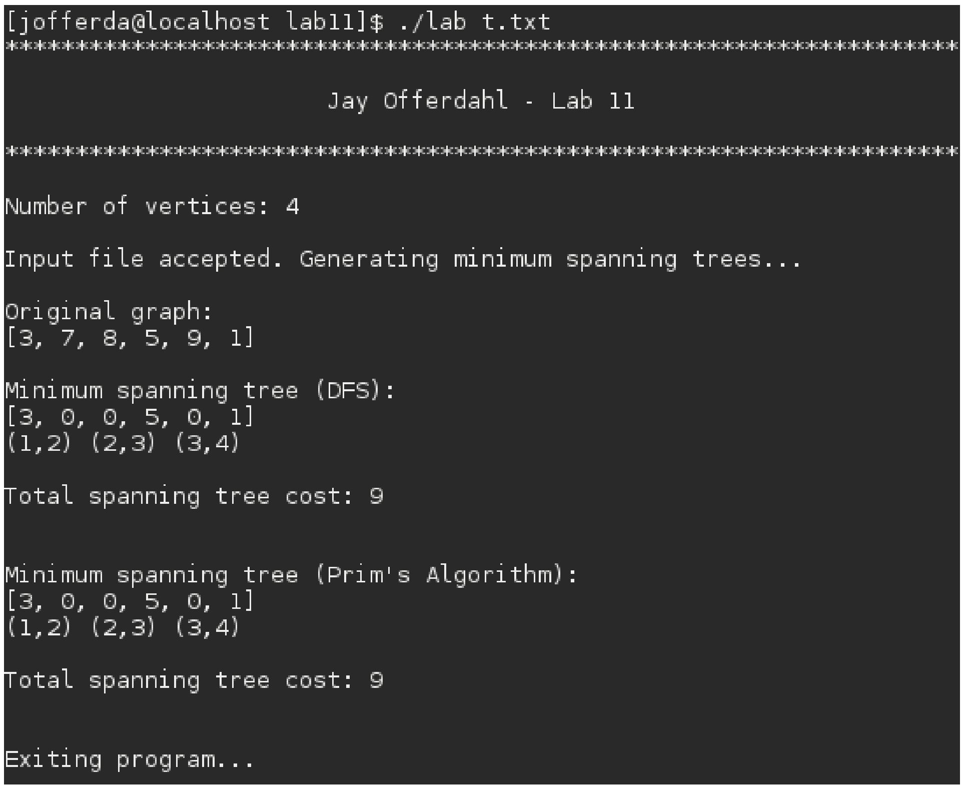
// Output from TMSTX.txt files



```
[jofferda@localhost lab11]$ ./lab TMST2.txt
*******************************************************************

                       Jay Offerdahl - Lab 11

*******************************************************************

Number of vertices: 7

Input file accepted. Generating minimum spanning trees...

Original graph:
[2, 4, 1, 12, 11, 6, 5, 3, 10, 6, 8, 2, 9, 5, 9, 2, 8, 4, 5, 6, 1]

Minimum spanning tree (DFS):
[2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 4, 0, 0, 1]
(1,2) (1,4) (3,4) (4,5) (4,7) (6,7)

Total spanning tree cost: 12


Minimum spanning tree (Prim's Algorithm):
[2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 4, 0, 0, 1]
(1,2) (1,4) (3,4) (4,5) (4,7) (6,7)

Total spanning tree cost: 12


Exiting program...
```

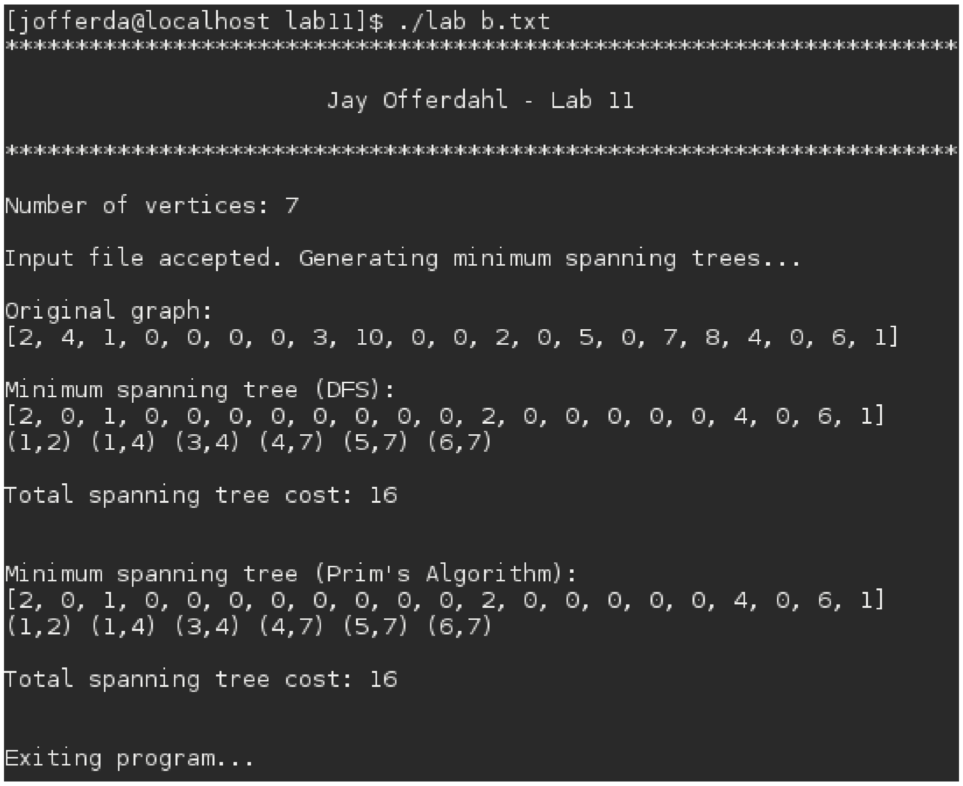
Jay Offerdahl
EECS 560
Lab 11 Report

// Output from the connected graph on the lab assignment page



```
[jofferda@localhost lab11]$ ./lab t.txt
*********************************************************************

                       Jay Offerdahl - Lab 11

*********************************************************************

Number of vertices: 4

Input file accepted. Generating minimum spanning trees...

Original graph:
[3, 7, 8, 5, 9, 1]

Minimum spanning tree (DFS):
[3, 0, 0, 5, 0, 1]
(1,2) (2,3) (3,4)

Total spanning tree cost: 9


Minimum spanning tree (Prim's Algorithm):
[3, 0, 0, 5, 0, 1]
(1,2) (2,3) (3,4)

Total spanning tree cost: 9


Exiting program...
```

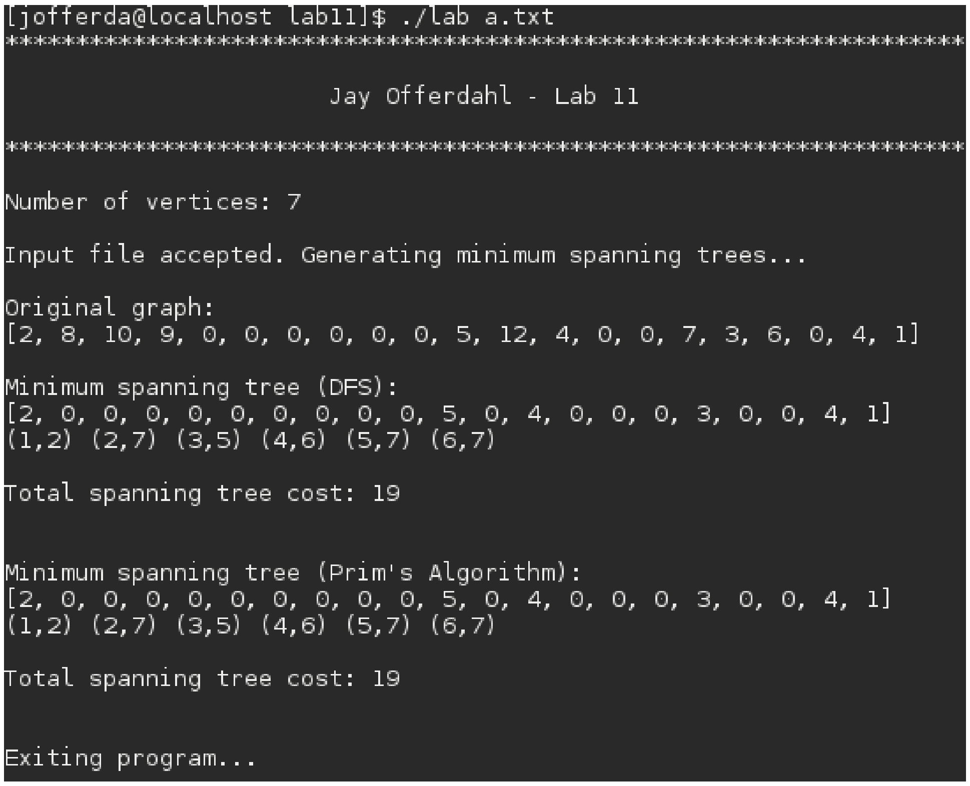
// Output from the connected graph used in the book to illustrate Prim's algorithm



```
[jofferda@localhost lab11]$ ./lab b.txt
*********************************************************************

                       Jay Offerdahl - Lab 11

*********************************************************************

Number of vertices: 7

Input file accepted. Generating minimum spanning trees...

Original graph:
[2, 4, 1, 0, 0, 0, 0, 3, 10, 0, 0, 2, 0, 5, 0, 7, 8, 4, 0, 6, 1]

Minimum spanning tree (DFS):
[2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 4, 0, 6, 1]
(1,2) (1,4) (3,4) (4,7) (5,7) (6,7)

Total spanning tree cost: 16


Minimum spanning tree (Prim's Algorithm):
[2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 4, 0, 6, 1]
(1,2) (1,4) (3,4) (4,7) (5,7) (6,7)

Total spanning tree cost: 16


Exiting program...
```

Jay Offerdahl
EECS 560
Lab 11 Report

// Output from a connected graph I made



```
[jofferda@localhost lab11]$ ./lab a.txt
**********************************************************************

                       Jay Offerdahl - Lab 11

**********************************************************************

Number of vertices: 7

Input file accepted. Generating minimum spanning trees...

Original graph:
[2, 8, 10, 9, 0, 0, 0, 0, 0, 0, 5, 12, 4, 0, 0, 7, 3, 6, 0, 4, 1]

Minimum spanning tree (DFS):
[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 4, 0, 0, 0, 3, 0, 0, 4, 1]
(1,2) (2,7) (3,5) (4,6) (5,7) (6,7)

Total spanning tree cost: 19


Minimum spanning tree (Prim's Algorithm):
[2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 4, 0, 0, 0, 3, 0, 0, 4, 1]
(1,2) (2,7) (3,5) (4,6) (5,7) (6,7)

Total spanning tree cost: 19


Exiting program...
```