

Algorithms Assignment Report

DFS Traversal & Prim's MST Algorithms

AMIR AKBARI (C18442284)

DT228/2

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Amir Akbari

Amir Akbari

12/05/2020

Table of Content

Introduction	3
Kruskal & MST	3
Diagram	5
Contents:	5
Step by Step construction of MST	7
Dist and Parent array printout:	7
Conclusion.....	9
Learning outcomes:.....	13

Introduction

This program uses Prim's Algorithm to return Minimum Spanning Tree weight. This program would use heaps and linked list to create and calculate the graph matrix through a text file. It would read the file and then will create the linked list for displaying the vertices and edges. At the end will go through the heap and find the MST and output the total number of weight of the MST.

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.¹

Prim's Algorithm is Vertex based when compared to Kruskal's Algorithm which is Edge based. In Prim's Algorithm we build the spanning tree from a given Vertex, adding the smallest edge to the Minimum Spanning Tree.

This program uses the Eager Implementation of Prim's Algorithm as we keep updating the heap if the distance from a Vertex to the Minimum Spanning Tree has changed.

Kruskal & MST

Prim's Algorithm is a greedy algorithm that finds a Minimum Spanning Tree for a connected weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every Vertex, where the total weight of all the edges in the tree is minimized.¹ Prim's Algorithm is Vertex based when compared to Kruskal's Algorithm which is Edge based. In Prim's Algorithm we build the spanning tree from a given Vertex, adding the smallest edge to the Minimum Spanning Tree. This program uses the Eager Implementation of Prim's Algorithm as we keep updating the heap if the distance from a Vertex to the Minimum Spanning Tree has changed. Time Complexity for Adjacency List: $O(E + V \log V)$ ¹

We will be looking at Graphs and how to work with them. A Graph has multiple edges and vertices, where the numbers of edges are usually greater than the number of vertices.² Each vertex is connected to another by the edges in between each one. A combination of any number of each would result in a graph. There are different types of graphs and different ways to approach them. There are incomplete graphs and complete graphs. The difference between the two is a complete graph has all the vertices connected to each other one way or another, which leaves no vertex hanging on its own. An incomplete graph is different in the sense that there is at least one vertex that is only connected to one other vertex itself.

¹ <https://iq.opengenus.org/prim-minimum-spanning-tree-algorithm/>
(Last Accessed 12/05/2020)

² https://en.wikipedia.org/wiki/Minimum_spanning_tree
(Last Accessed 12/05/2020)

We will create a subset of one graph in this report. This is known as a tree. Specifically, we will be looking at the smallest possible tree to be made from the graph. This is known as the minimum spanning tree and takes into the account making a tree that connects all the vertices with the smallest weight possible. To achieve this, we will implement and document through two algorithms. One is known as the Kruskal algorithm while the other is the Prim Algorithm. These approaches do similar approach as they both create a minimum spanning tree. However, the approaches are different.

Prim takes in a starting point and works its way around the tree by connecting all the vertices. It checks all the possible routes and takes the shortest route. Let's say we have vertex A. A is connected to B, C and D. The distance between A and these vertices as follows are: to B is 5, C is 3 and D is 7. What Prim will do is go to C as that is the shortest path. They continue to connect until all the vertices are connected creating the minimum spanning tree.

Kruskal is different. Instead of reading vertices, Kruskal reads the edges in between each vertex and connects them based off the smallest edges. Taking the above example and extending the information so B connects to C with weight of 7, to D with a weight of 2 and C connects to D with a weight of 10. Kruskal connects B to D as it is the smallest, then A to C as it is the second smallest. If both was the same size it would not matter which one was picked first as them both will be connected in the minimum spanning tree.

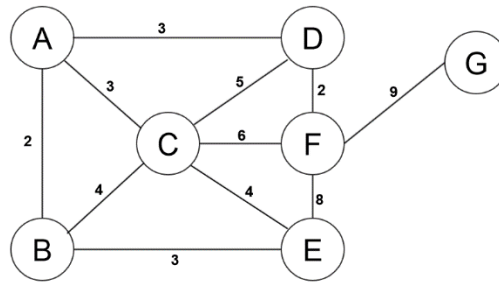
Both will be assisted by sorting methods to assist in where they are positioned and the combining of them. For example, Kruskal will be assisted by Union by rank, Set Representation and a heap in this report, where Prim will be assisted by the heap, distance and keeping track of its parent as well as its heap position. These mechanisms will make each running of the algorithm within a java program more efficient and smooth out the layout. Although both heaps are different in their content, they are both priority heaps in their layout.

Kruskal's Algorithm sort's the edges according to their edge weights, it can be done with Merge-Sort or Quick-Sort although for this assignment the Union Find Data Structure is used. Kruskal's algorithm can be shown to run in $O(E \log V)$ time. The implementation for this algorithm is that we build the spanning tree separately adding the smallest edge to the spanning tree if there is no cycle. As mentioned before Prim's Algorithm is Vertex based and Kruskal's algorithm is Edge based.³

³ https://github.com/michaelrinos/Minimum_Spanning_Tree (Last Accessed 12/05/2020)

Diagram

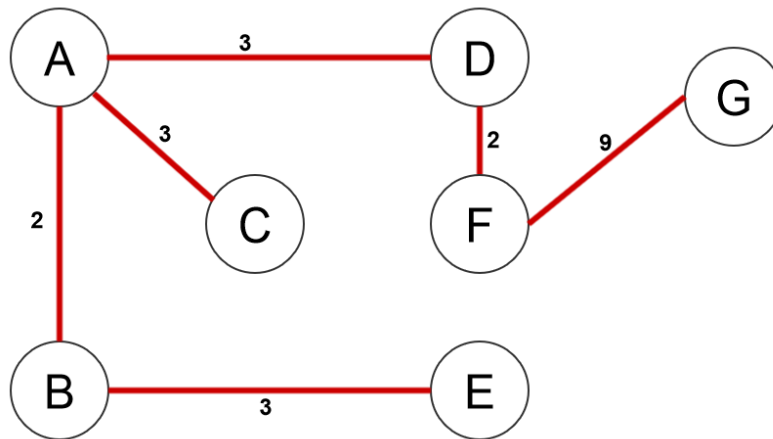
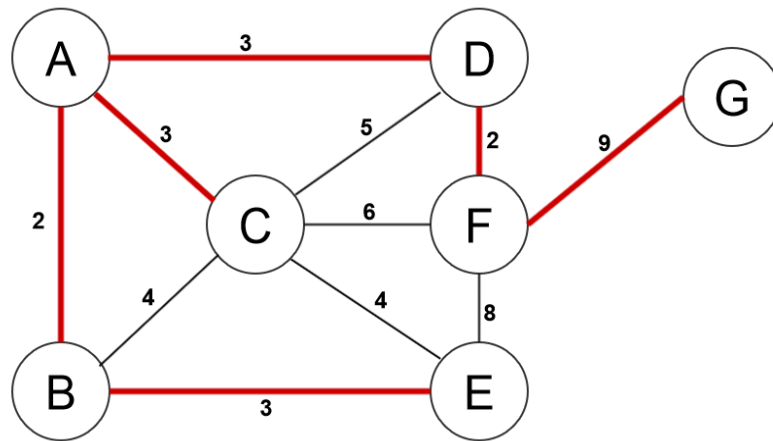
Here is my diagram which I created for testing:



Contents:

7	11
1	2 2
1	3 3
1	4 3
2	3 4
2	5 3
3	4 5
3	5 1
3	6 6
4	6 7
5	6 8
6	7 9

This is the graph showing the MST circuit or route highlighted in red after Prim's algorithm has been provided to show the MST of the graph, assuming we start on Node 'A'.



Adding the weight of all the edges that the traverse route taken for the minimal spanning tree we can obtain the weight of the graph.

In this case it is...

$$2 + 3 + 3 + 3 + 2 + 9 = 22$$

So the weight of the minimal spanning tree starting Node A is 2

The traverse route is represented as...

@ -> A

A -> B

A -> C

A -> D

B -> E

D -> F

F -> G

Step by Step construction of MST

Heap h : is a heap to find the next vertex nearest the MST so that it can be added to the MST.

Dist[] : is an int array that records the current distance of a vertex from the MST

Parent[] : is an int array that stores the MST

hPos[] : is an int array that records the position of any vertex with the heap array a[]. Vertex v is in position hPos[v] in a[].


Dist and Parent array printout:

Dist[]	Parent[]
Traverse 1	
A -> @	1 -> -1
B -> A	2 -> 2
C -> A	3 -> 3
D -> A	4 -> 3
E -> @	5 -> ∞
F -> @	6 -> ∞
G -> @	7 -> ∞
Traverse 2	
A -> @	1 -> -1
B -> A	2 -> 1
C -> A	3 -> 3
D -> A	4 -> -3
E -> @	5 -> ∞
F -> D	6 -> 7
G -> @	7 -> ∞

Traverse 3	
A -> @	1 -> -1
B -> A	2 -> -1
C -> A	3 -> 3
D -> A	4 -> -3
E -> B	5 -> 1
F -> D	6 -> -1
G -> F	7 -> 9
Traverse 4	
A -> @	1 -> -1
B -> A	2 -> -1
C -> E	3 -> 1
D -> A	4 -> -3
E -> B	5 -> -1
F -> D	6 -> -1
G -> F	7 -> 1
Traverse 5	
A -> @	1 -> -1
B -> A	2 -> -1
C -> E	3 -> 1
D -> A	4 -> -3
E -> B	5 -> -1
F -> D	6 -> -1
G -> F	7 -> -1
Traverse 6	
A -> @	1 -> -1
B -> A	2 -> -1
C -> E	3 -> -1
D -> A	4 -> -3
E -> B	5 -> -1
F -> D	6 -> -1
G -> F	7 -> -1
Traverse 7	
A -> @	1 -> -1
B -> A	2 -> -1
C -> E	3 -> -1
D -> A	4 -> -3
E -> B	5 -> -1
F -> D	6 -> -1
G -> F	7 -> -1

Conclusion

MST and DFS Output:

 Command Prompt

```
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\35386>cd C:\Users\35386\Desktop\alg\Task 1 and 3

C:\Users\35386\Desktop\alg\Task 1 and 3> javac PrimsAlgorithm.java

C:\Users\35386\Desktop\alg\Task 1 and 3> java PrimsAlgorithm

Enter .txt file name to read in: myGraph.txt

Parts[] = 7 11
Reading edges from text file
Edge A--(2)--B
Edge A--(3)--C
Edge A--(3)--D
Edge B--(4)--C
Edge B--(3)--E
Edge C--(5)--D
Edge C--(4)--E
Edge C--(6)--F
Edge D--(2)--F
Edge E--(8)--F
Edge F--(9)--G

Enter what vertex to start at (use numbers ie B = 2) : 2
Parts[] = 7 11
Reading edges from text file

Edge A--(2)--B
Edge A--(3)--C
Edge A--(3)--D
Edge B--(4)--C
Edge B--(3)--E
Edge C--(5)--D
Edge C--(4)--E
Edge C--(6)--F
Edge D--(2)--F
Edge E--(8)--F
Edge F--(9)--G
```

```

adj[A] -> |D | 3| -> |C | 3| -> |B | 2| ->
adj[B] -> |E | 3| -> |C | 4| -> |A | 2| ->
adj[C] -> |F | 6| -> |E | 4| -> |D | 5| -> |B | 4| -> |A | 3| ->
adj[D] -> |F | 2| -> |C | 5| -> |A | 3| ->
adj[E] -> |F | 8| -> |C | 4| -> |B | 3| ->
adj[F] -> |G | 9| -> |E | 8| -> |D | 2| -> |C | 6| ->
adj[G] -> |F | 9| ->

```

Total weight of MST ----->

=27

Minimum Spanning tree parent array is ----->

B -> A

@ -> B

A -> C

F -> D

B -> E

E -> F

F -> G

```

adj[A] -> |D | 3| -> |C | 3| -> |B | 2| ->
adj[B] -> |E | 3| -> |C | 4| -> |A | 2| ->
adj[C] -> |F | 6| -> |E | 4| -> |D | 5| -> |B | 4| -> |A | 3| ->
adj[D] -> |F | 2| -> |C | 5| -> |A | 3| ->
adj[E] -> |F | 8| -> |C | 4| -> |B | 3| ->
adj[F] -> |G | 9| -> |E | 8| -> |D | 2| -> |C | 6| ->
adj[G] -> |F | 9| ->

```

Depth First Search:

```

adj[A] -> |D | 3| -> |C | 3| -> |B | 2| ->
adj[B] -> |E | 3| -> |C | 4| -> |A | 2| ->
adj[C] -> |F | 6| -> |E | 4| -> |D | 5| -> |B | 4| -> |A | 3| ->
adj[D] -> |F | 2| -> |C | 5| -> |A | 3| ->
adj[E] -> |F | 8| -> |C | 4| -> |B | 3| ->
adj[F] -> |G | 9| -> |E | 8| -> |D | 2| -> |C | 6| ->
adj[G] -> |F | 9| ->

```

Depth First Search:

Depth First Graph Traversal

Starting with Vertex: A

```

Visited vertex | A | through edge | 1 | D |
Visited vertex | D | through edge | 4 | F |
Visited vertex | F | through edge | 6 | G |
Visited vertex | F | through edge | 6 | E |
Visited vertex | E | through edge | 5 | C |
Visited vertex | C | through edge | 3 | B |

```

C:\Users\35386\Desktop\alg\Task 1 and 3>

Task 2 DFS Output:

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

stSearch
Parts[] = 8 8
Reading edges from text file

Edge A--(6)--C
Edge A--(5)--E
Edge A--(4)--F
Edge B--(2)--E
Edge B--(3)--F
Edge C--(5)--D
Edge C--(4)--E
Edge E--(3)--F

adj[A] -> |F| 4| -> |E| 5| -> |C| 6| ->
adj[B] -> |F| 3| -> |E| 2| ->
adj[C] -> |E| 4| -> |D| 5| -> |A| 6| ->
adj[D] -> |C| 5| ->
adj[E] -> |F| 3| -> |C| 4| -> |B| 2| -> |A| 5| ->
adj[F] -> |E| 3| -> |B| 3| -> |A| 4| ->
adj[G] ->
adj[H] ->

Depth First Graph Traversal

Starting with Vertex: A
Visited vertex | A | through edge | 1 | F |
Visited vertex | F | through edge | 6 | E |
Visited vertex | E | through edge | 5 | C |
Visited vertex | C | through edge | 3 | D |
Visited vertex | E | through edge | 5 | B |
Visited vertex | A | through edge | 1 | F |
Visited vertex | F | through edge | 6 | E |
Visited vertex | E | through edge | 5 | C |
Visited vertex | C | through edge | 3 | D |
Visited vertex | E | through edge | 5 | B |

Number of Connected Components: 2
```

Learning outcomes:

- I have learnt that Prim's Algorithm is significantly faster in the limit when you've got a really dense graph with more edges than vertices.
- I have gained a massive understanding of how Heaps work and how they could be used in these Algorithms.
- Prim's Algorithm is better if the number of edges to vertices is high and this is in dense graphs. So both of these algorithms have their own benefits.
- Have done a lot of research on both of these Algorithms, wrote the code and performed several calculations with different types of graphs to understand what is happening.
- I have learnt how to construct graphs into a text file and how to make the program read in the graph and perform the algorithm on it so now I can make my own Boruvka's Algorithm, Dijkstra's Algorithm and so many more.
- Have gained extensive knowledge of graph traversing algorithms and its application to a computer program.
- How to construct graphs that are applicable to Prim's Algorithm and how to manually convert the graph into a text file so it's readable by the program.
- Real world applications of Prim's Algorithm.
- Have learned how to run java through command.