CMPU2002 Assignment Report
Student Number: C21394933

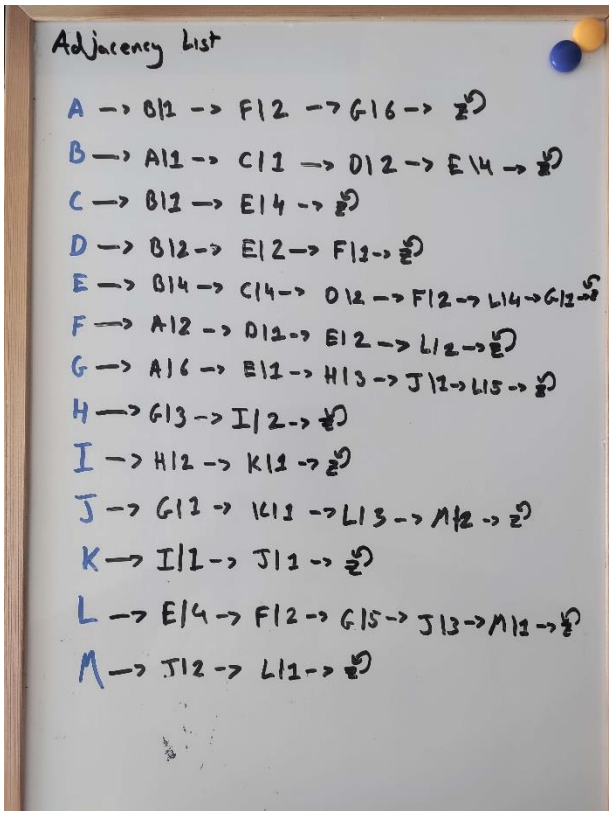
Introduction.

In computer science and graph theory, there are several algorithms that are widely used to solve different problems related to graphs. These algorithms can help us find the shortest paths, minimum spanning trees and perform various searches on a graph. In this assignment I will discuss 5 of these algorithms: MST Prim, MST Kruskal, STP Dijkstra, BF Search and DF Search through images and output.

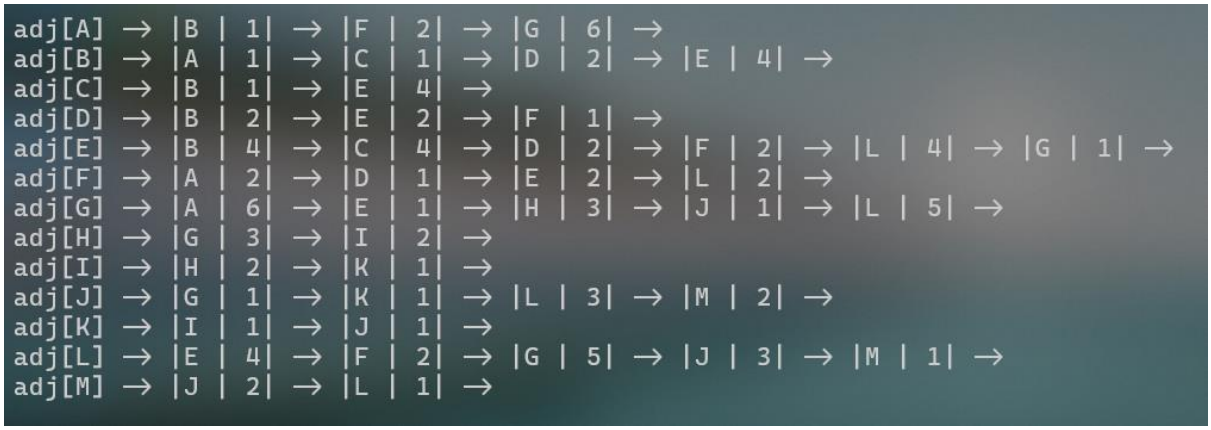
Time Complexity of each algo:
MST Prim with Adjacency lists: O(nlog(n))
MST Kruskals: O(E log (V))
SPT Dijkstra's: O((V + E) log2(V))
DF Search: O(V + E)
BF Search: O(E + V)

In summary, each of these algorithms has its unique strengths and weaknesses when it comes to searching and routing in a graph. Understanding their time and space complexity can help us choose the most efficient algorithm for a given problem. In the following sections, we will provide a detailed analysis of each algorithm's working principle, time complexity, and space complexity.

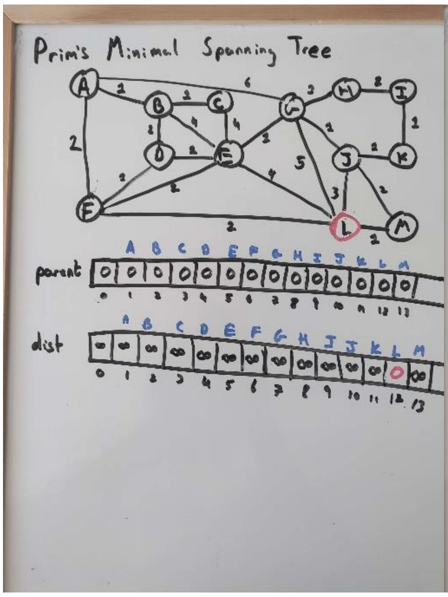
Adjacency List Diagram.

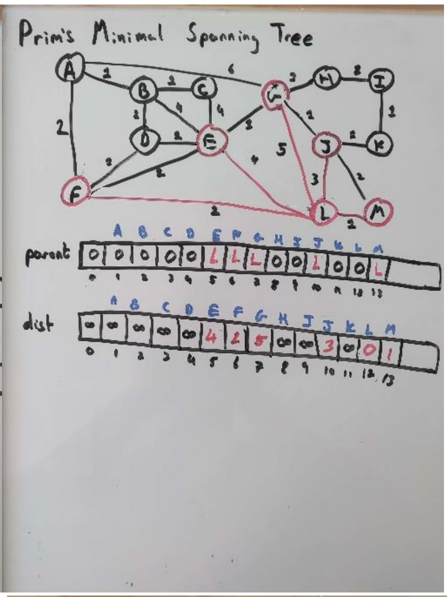


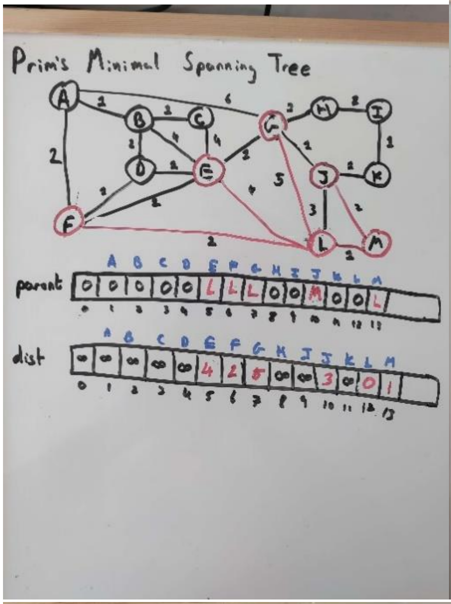
CMPU2002 Assignment Report
Student Number: C21394933

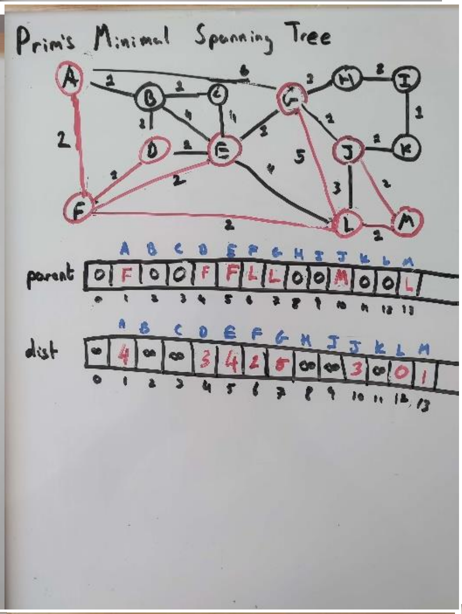


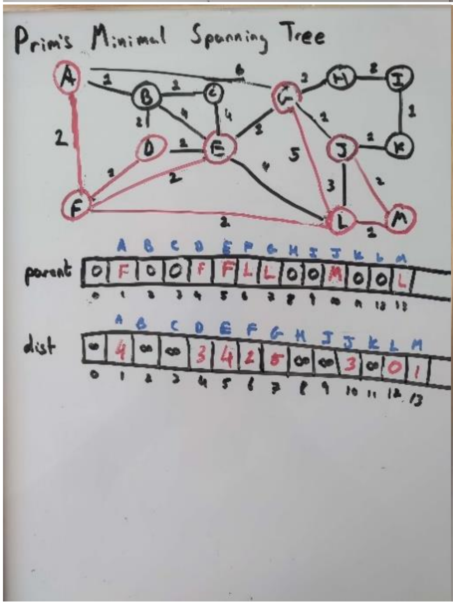
Minimal Spanning Tree using Prim's algo: Step by Step.

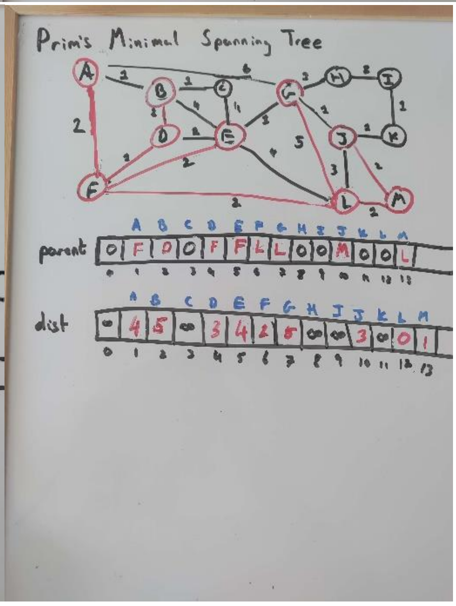


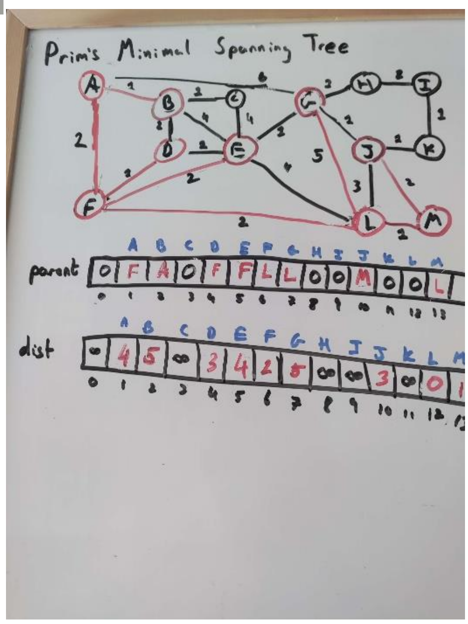




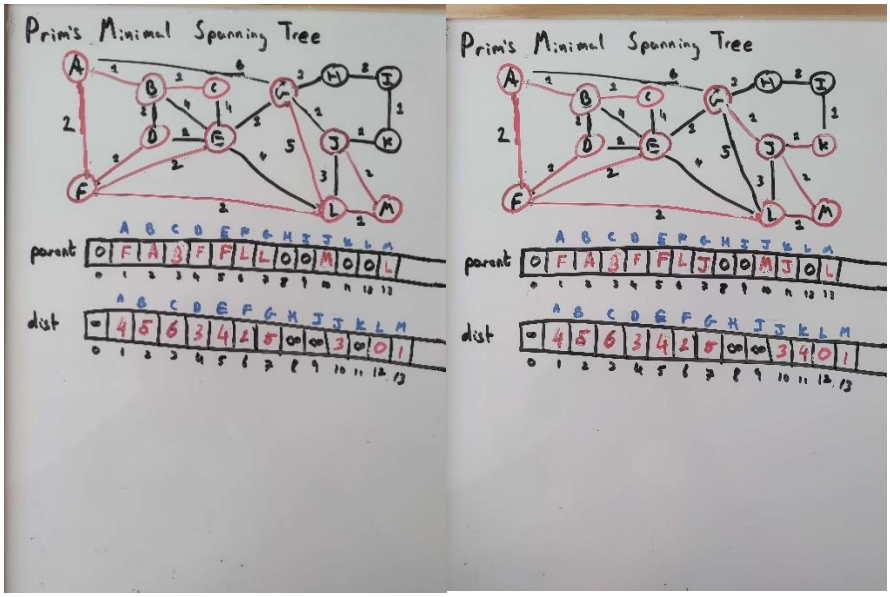


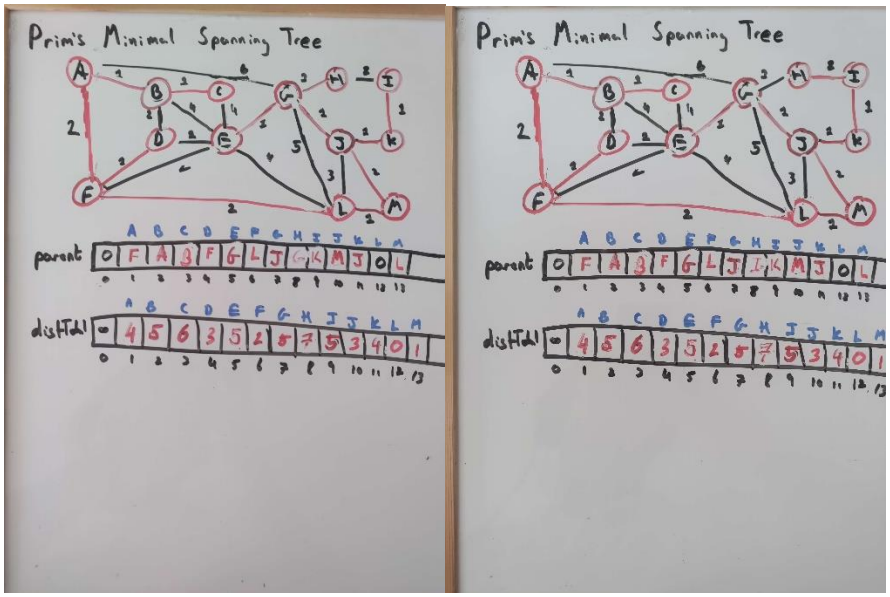




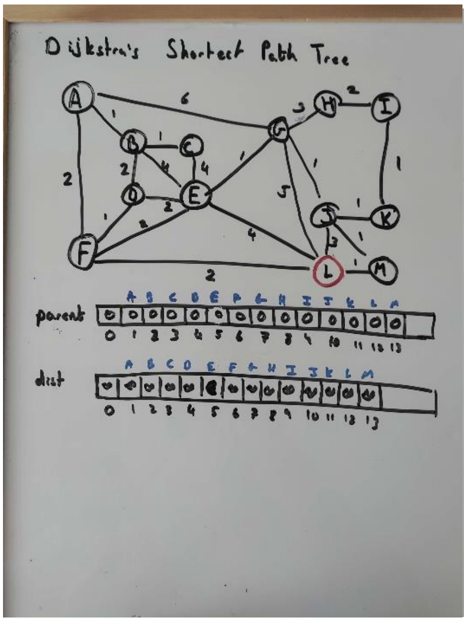


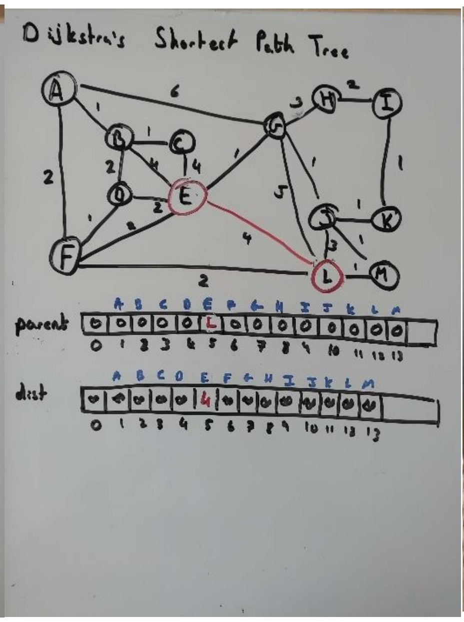
CMPU2002 Assignment Report
Student Number: C21394933

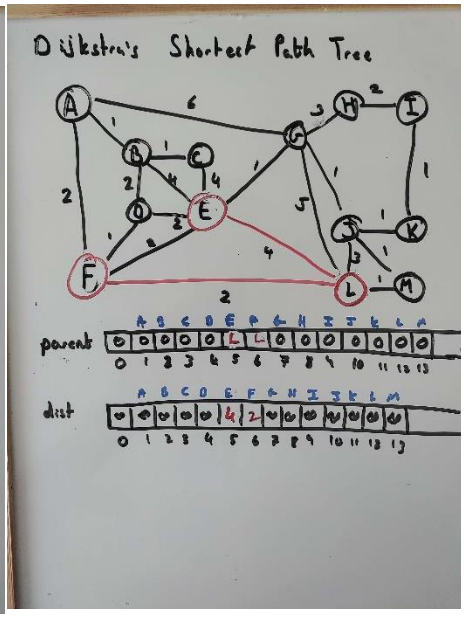


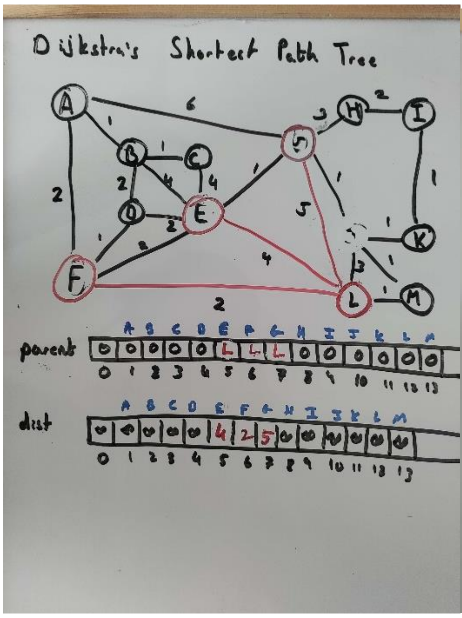


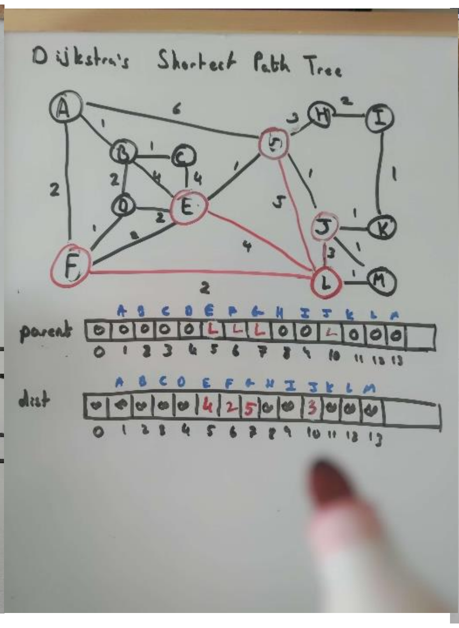
Shortest Path using Dijkstra's algo: Step by Step.

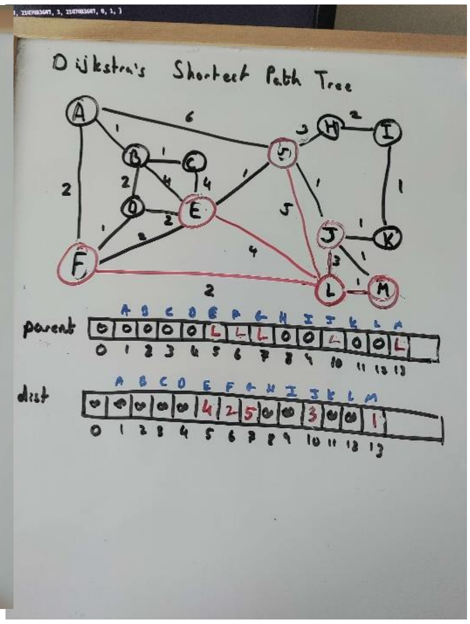




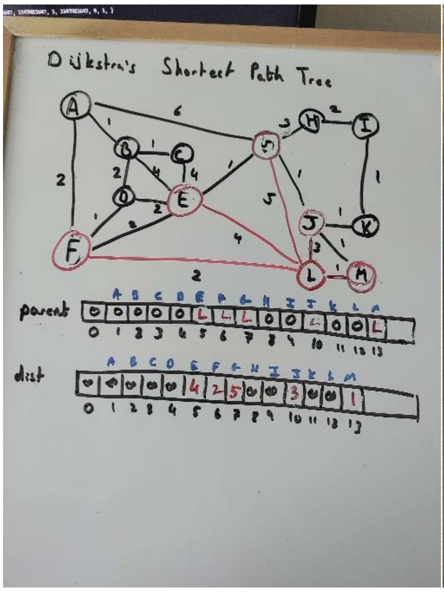


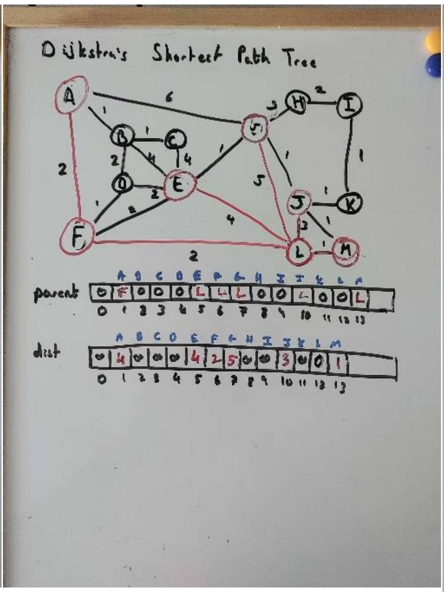


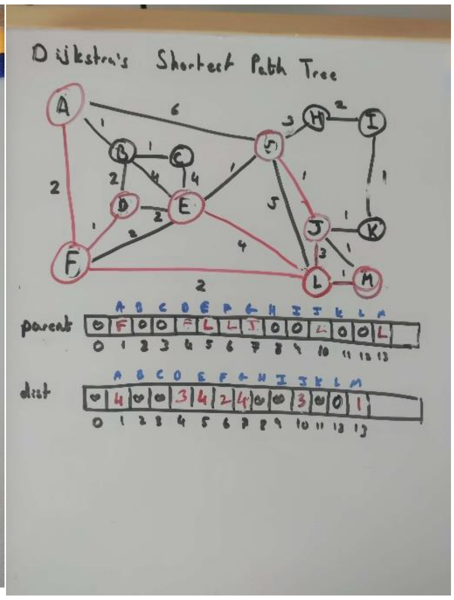


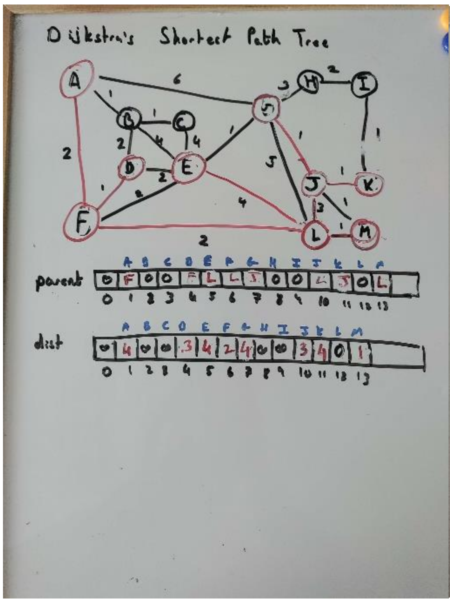


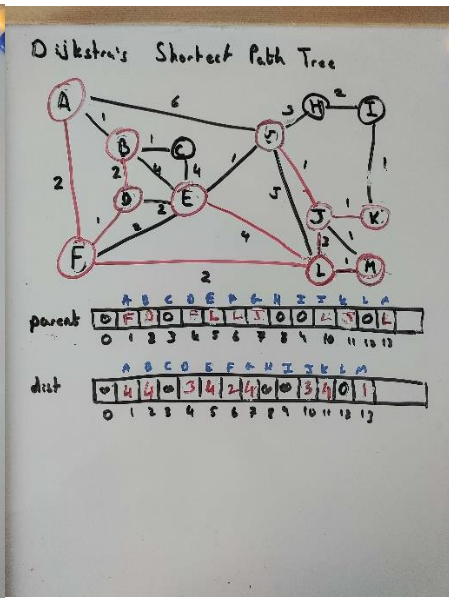
CMPU2002 Assignment Report
Student Number: C21394933

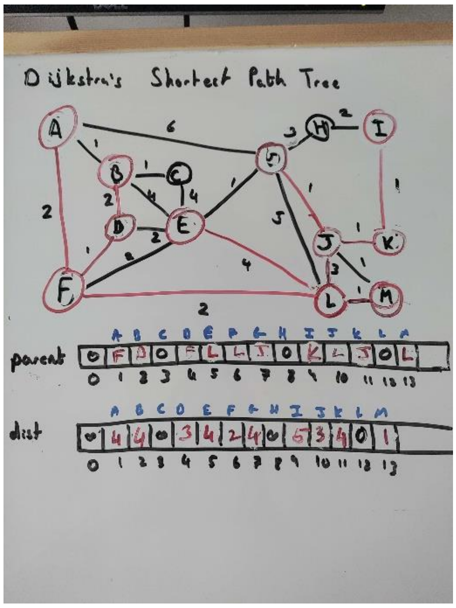


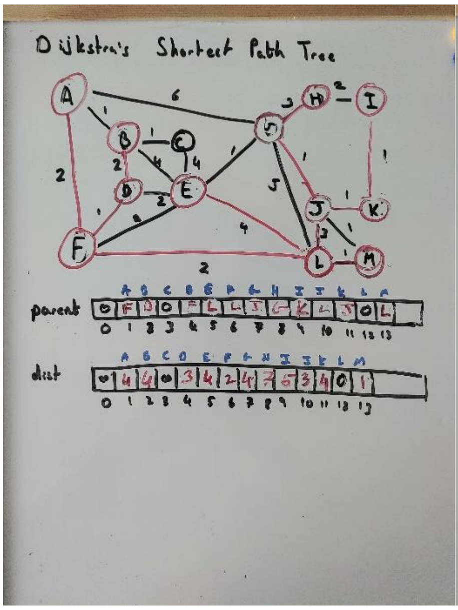


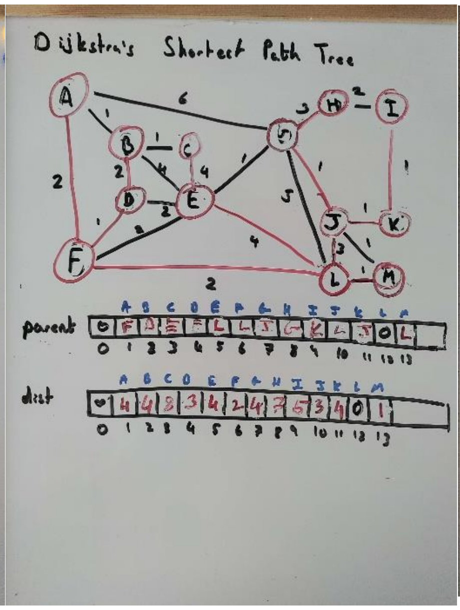


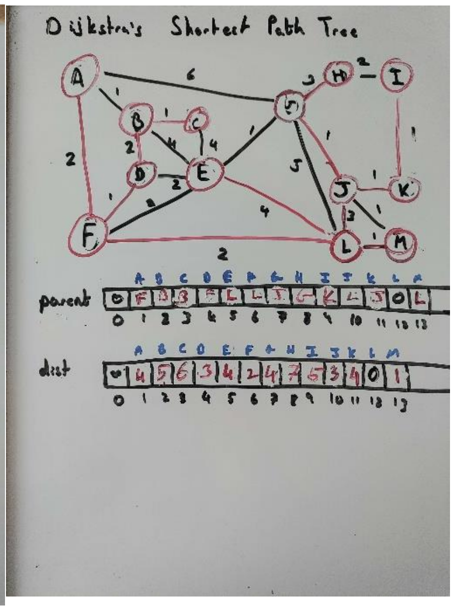






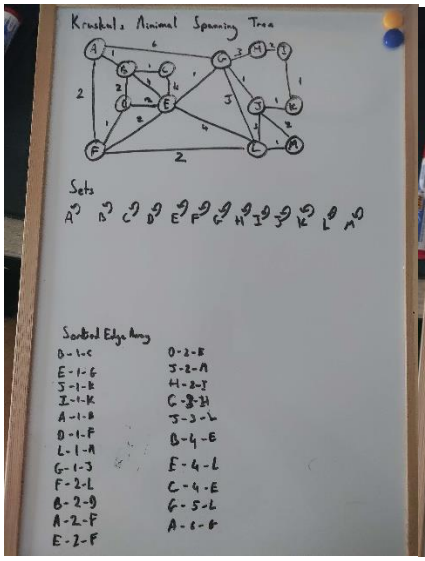


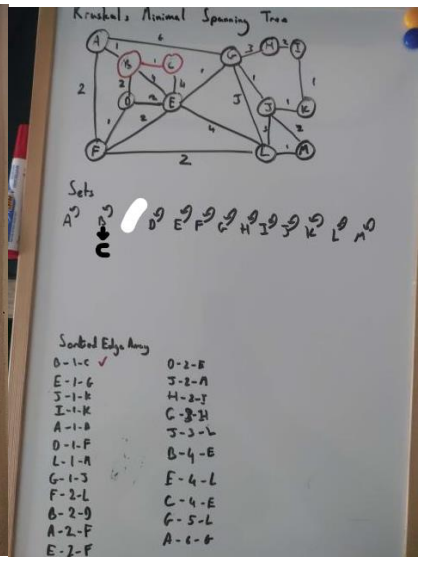


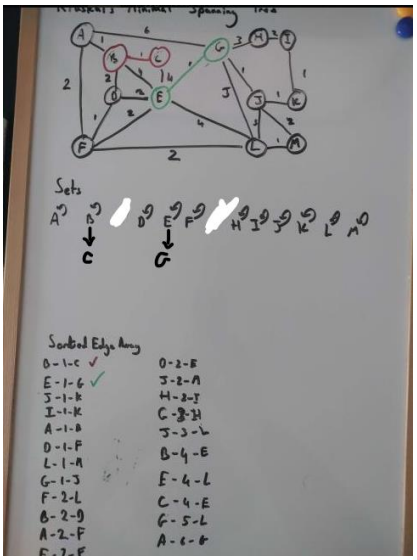


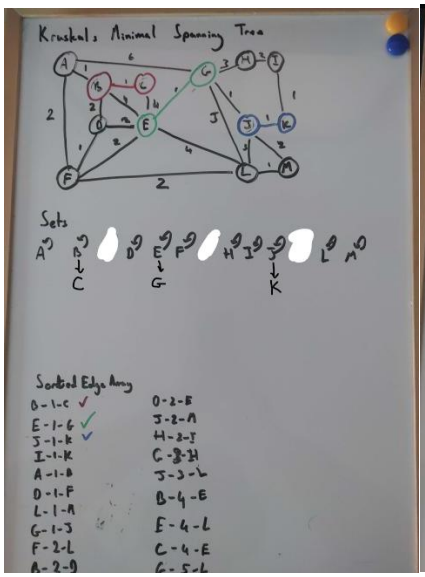
CMPU2002 Assignment Report
Student Number: C21394933

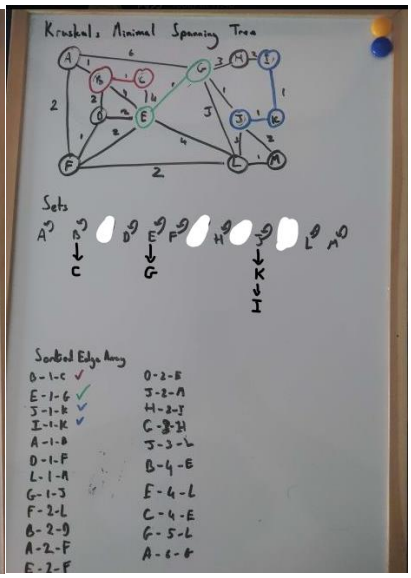
Minimal Spanning Tree for Kruskal & Union-find: Step by Step.

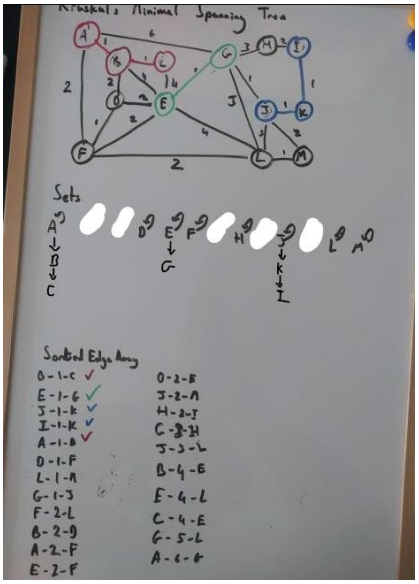




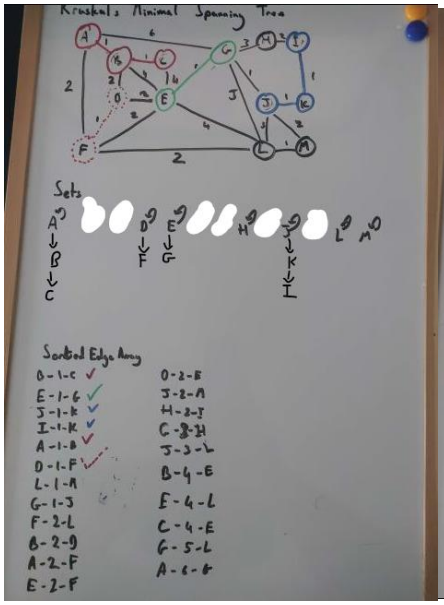


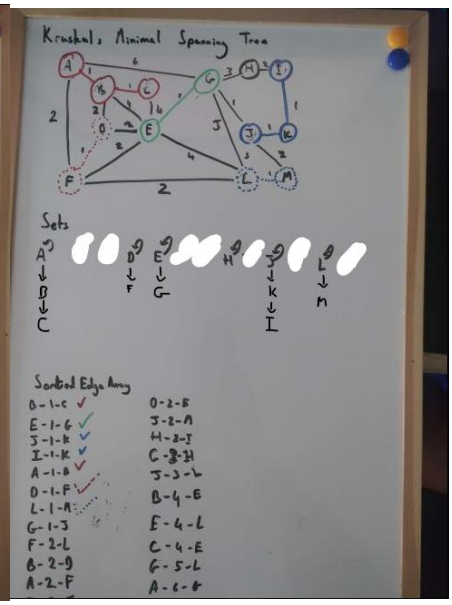


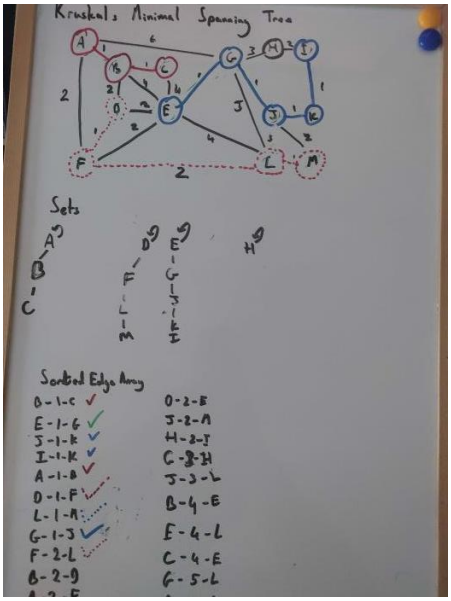


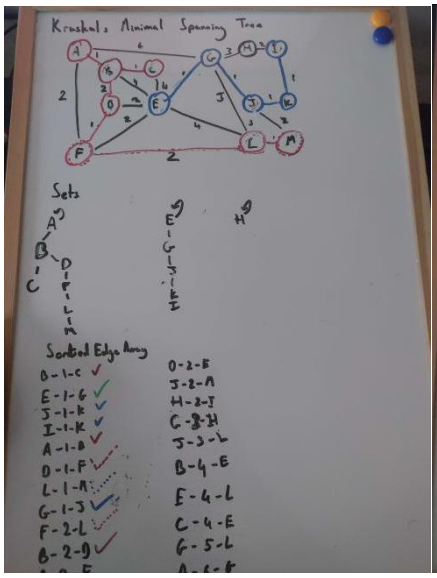


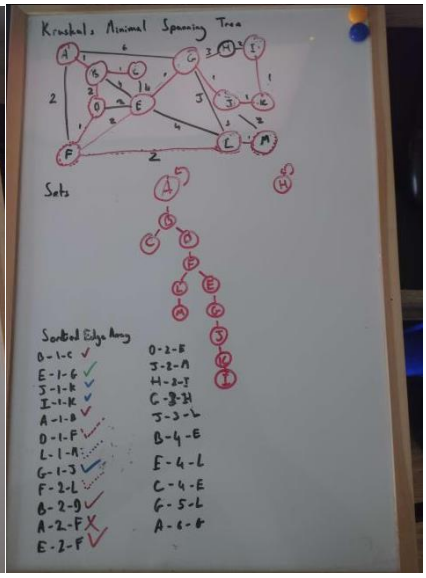
CMPU2002 Assignment Report
Student Number: C21394933

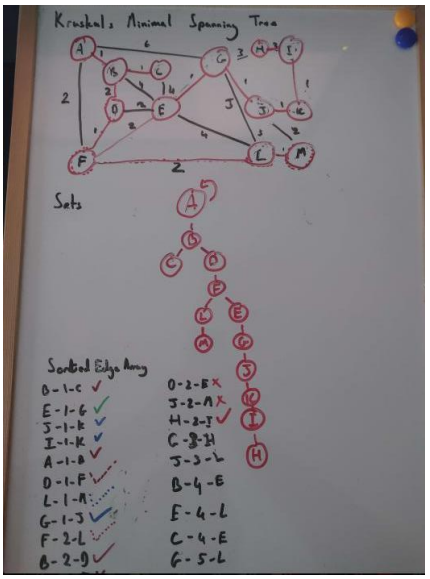




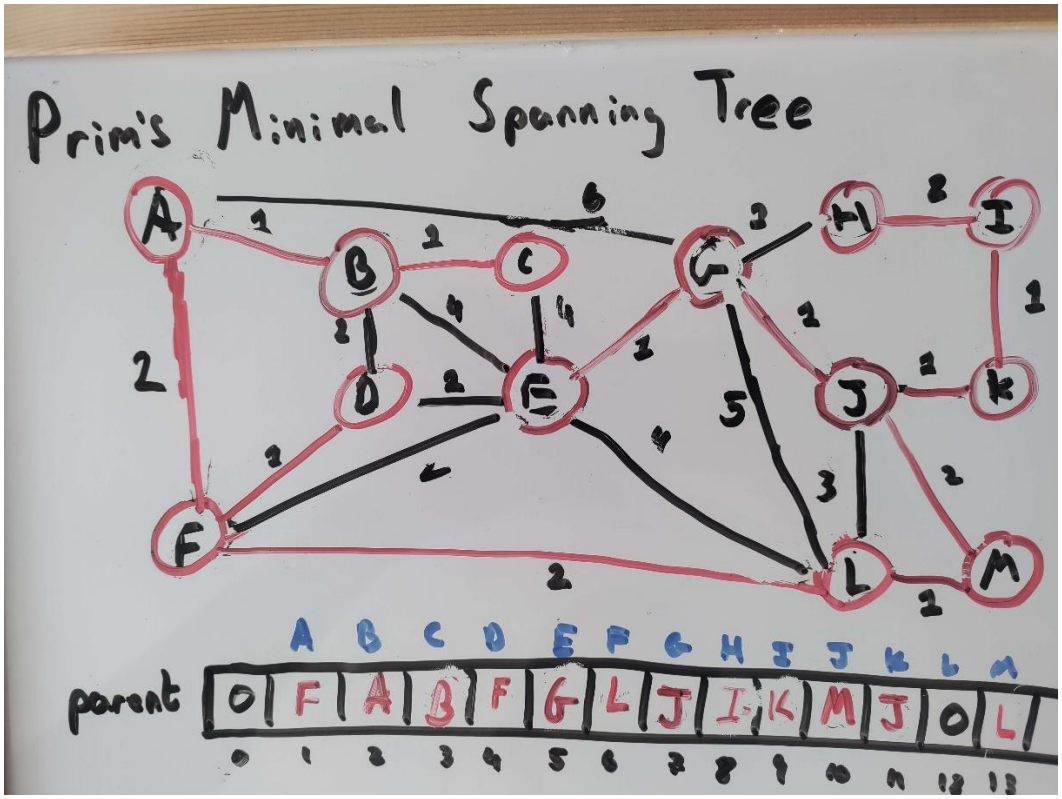


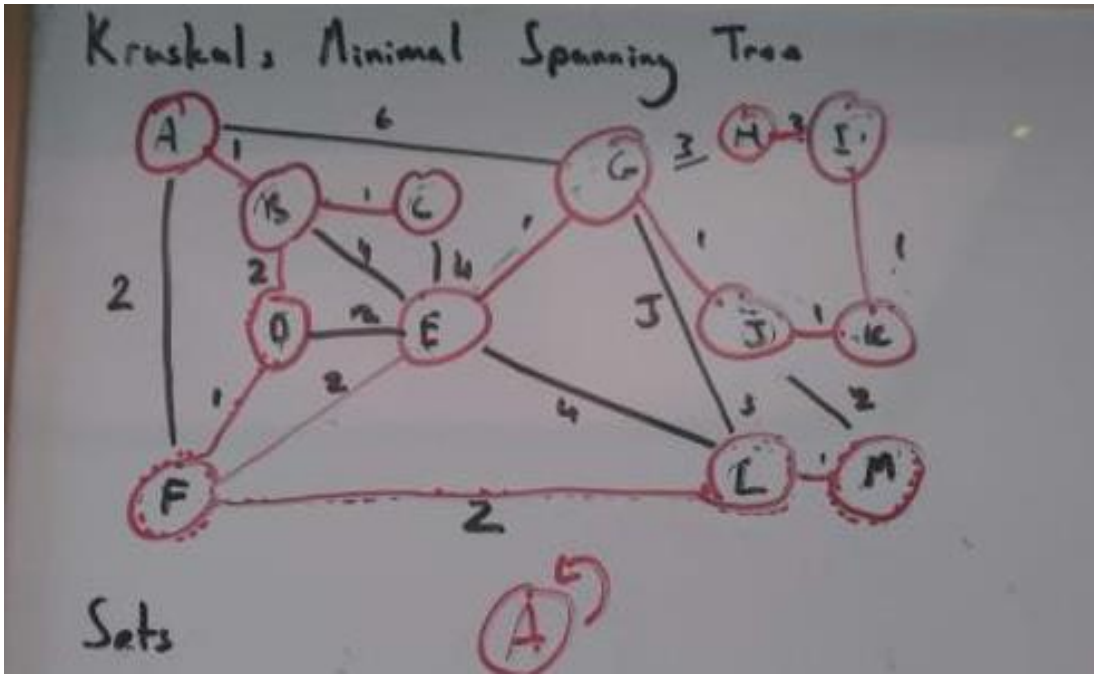




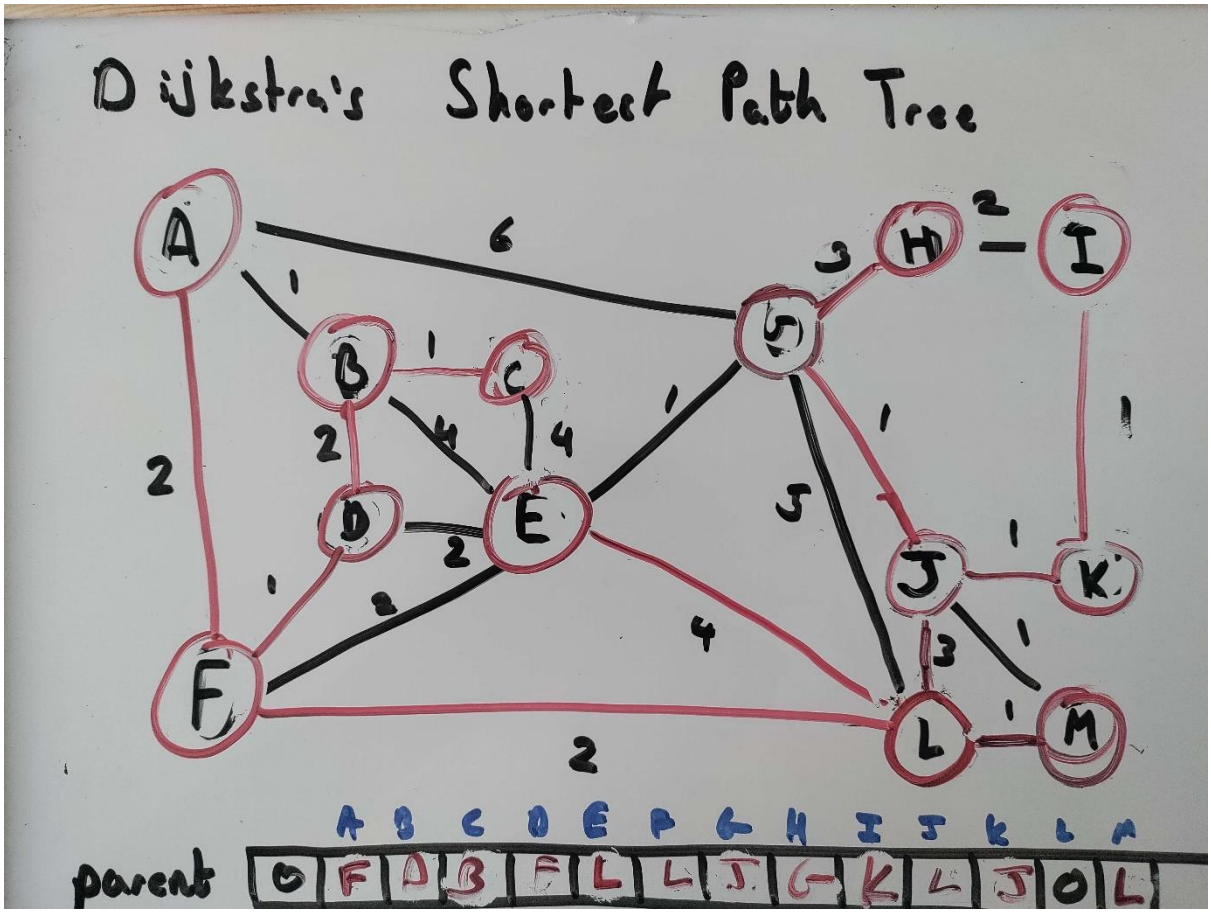


MST Superimposed on Graph Diagram.





SST Superimposed on Graph Diagram.



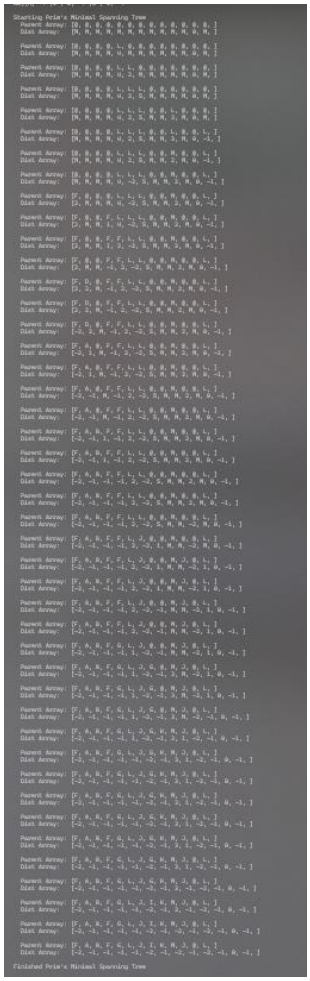
Screen captures of program execution.

Prims/Dijkstra's/DF/BR

```
PS D:\User\Ernest John Decina\Desktop\Test> java GraphAssignment
═══════════════════════════════════════════

Algorithms & Data Structures Assignment
Student Number: C21394933
═══════════════════════════════════════════



Enter file name:
wGraph1.txt
Enter starting vertex as number:
12
Parts[] = 13 22
Reading edges from text file
Edge A--(1)--B
Edge A--(2)--F
Edge A--(6)--G
Edge B--(1)--C
Edge B--(2)--D
Edge B--(4)--E
Edge C--(4)--E
Edge D--(2)--E
Edge D--(1)--F
Edge E--(2)--F
Edge E--(4)--L
Edge E--(1)--G
Edge F--(2)--L
Edge G--(3)--H
Edge G--(1)--J
Edge G--(5)--L
Edge H--(2)--I
Edge I--(1)--K
Edge J--(1)--K
Edge J--(3)--L
Edge J--(2)--M
Edge L--(1)--M
```

```
adj[A] → |B | 1| → |F | 2| → |G | 6| →
adj[B] → |A | 1| → |C | 1| → |D | 2| → |E | 4| →
adj[C] → |B | 1| → |E | 4| →
adj[D] → |B | 2| → |E | 2| → |F | 1| →
adj[E] → |B | 4| → |C | 4| → |D | 2| → |F | 2| → |L | 4| → |G | 1| →
adj[F] → |A | 2| → |D | 1| → |E | 2| → |L | 2| →
adj[G] → |A | 6| → |E | 1| → |H | 3| → |J | 1| → |L | 5| →
adj[H] → |G | 3| → |I | 2| →
adj[I] → |H | 2| → |K | 1| →
adj[J] → |G | 1| → |K | 1| → |L | 3| → |M | 2| →
adj[K] → |I | 1| → |J | 1| →
adj[L] → |E | 4| → |F | 2| → |G | 5| → |J | 3| → |M | 1| →
adj[M] → |J | 2| → |L | 1| →
```

CMPU2002 Assignment Report
Student Number: C21394933



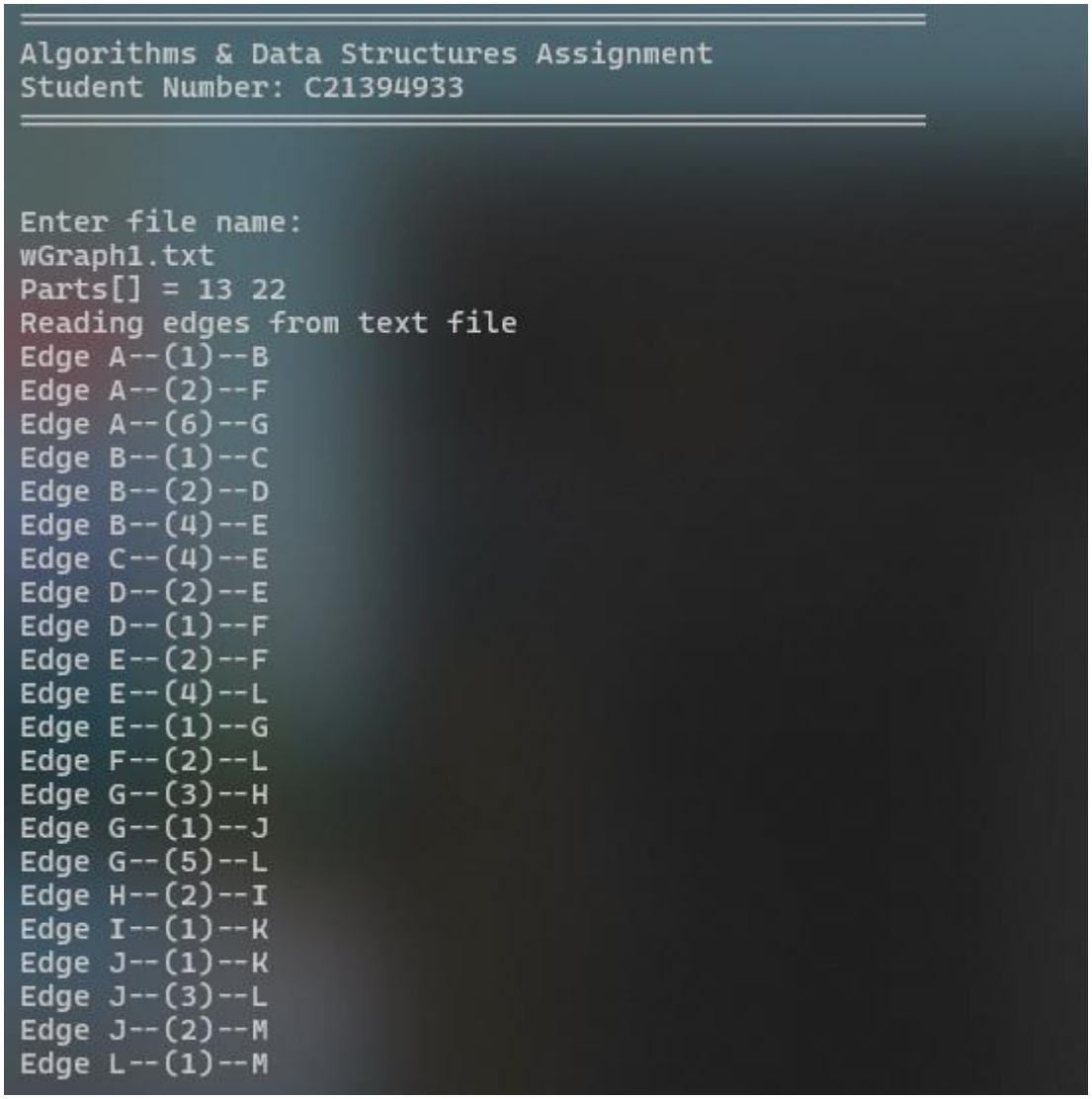
```
Weight of MST = 16


Minimum Spanning tree parent array is:
A → F
B → A
C → B
D → F
E → G
F → L
G → J
H → I
I → K
J → M
K → J
L → @
M → L
```

CMPU2002 Assignment Report
Student Number: C21394933

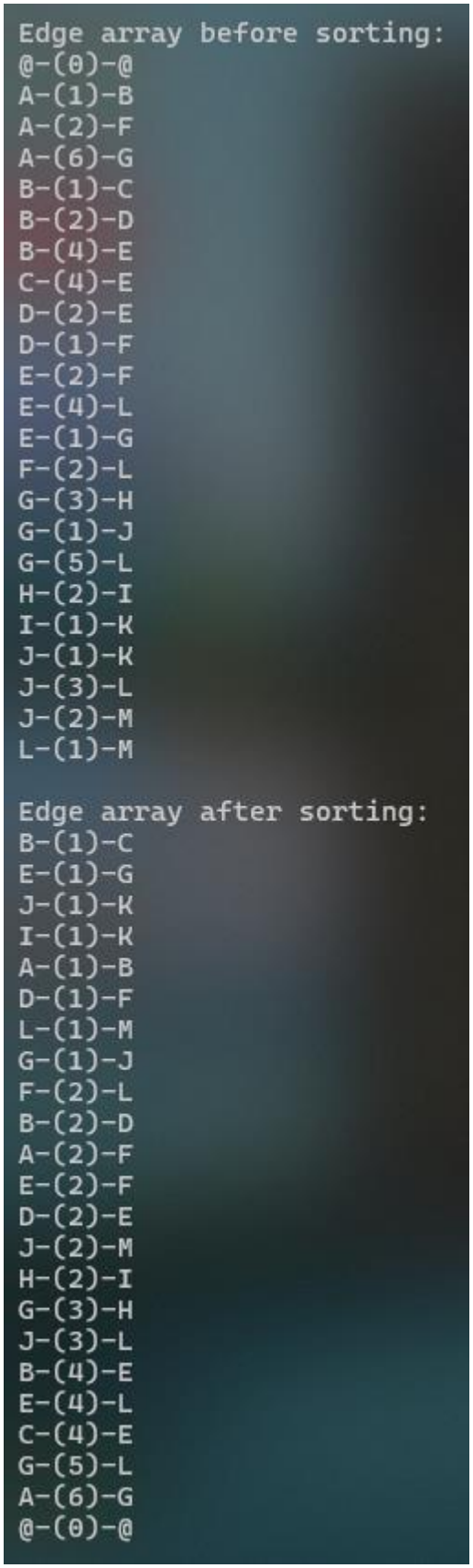
```
Shortest Path Tree values are:
[Vertex]  [Parent]  [Distance from chosen vertex]
A             F                    4
B             D                    5
C             B                    6
D             F                    3
E             L                    4
F             L                    2
G             J                    4
H             G                    7
I             K                    5
J             L                    3
K             J                    4
L             @                    0
M             L                    1
```

```
Depth First Search:
    DFS visted vertex: L From: @
    DFS visted vertex: E From: L
    DFS visted vertex: B From: E
    DFS visted vertex: A From: B
    DFS visted vertex: F From: A
    DFS visted vertex: D From: F
    DFS visted vertex: L From: F
    DFS visted vertex: G From: L
    DFS visted vertex: H From: G
    DFS visted vertex: I From: H
    DFS visted vertex: K From: I
    DFS visted vertex: J From: K
    DFS visted vertex: M From: J
    DFS visted vertex: C From: B
Depth First Search Ended.

Breadth First Search:
    BFS visted vertex: L
    BFS visted vertex: E
    BFS visted vertex: F
    BFS visted vertex: G
    BFS visted vertex: J
    BFS visted vertex: M
    BFS visted vertex: B
    BFS visted vertex: C
    BFS visted vertex: D
    BFS visted vertex: A
    BFS visted vertex: H
    BFS visted vertex: K
    BFS visted vertex: I
Breadth First Search Ended.
```

Kruskal's


Algorithms & Data Structures Assignment
Student Number: C21394933
Enter file name:
wGraph1.txt
Parts[] = 13 22
Reading edges from text file
Edge A--(1)--B
Edge A--(2)--F
Edge A--(6)--G
Edge B--(1)--C
Edge B--(2)--D
Edge B--(4)--E
Edge C--(4)--E
Edge D--(2)--E
Edge D--(1)--F
Edge E--(2)--F
Edge E--(4)--L
Edge E--(1)--G
Edge F--(2)--L
Edge G--(3)--H
Edge G--(1)--J
Edge G--(5)--L
Edge H--(2)--I
Edge I--(1)--K
Edge J--(1)--K
Edge J--(3)--L
Edge J--(2)--M
Edge L--(1)--M

CMPU2002 Assignment Report
Student Number: C21394933



```
Started Kruskals Minimal Spanning Tree:

  Tree: A→A  B→B  C→C  D→D  E→E  F→F  G→G  H→H  I→I  J→J  K→K  L→L

  Set{B C }
  Tree: A→A  B→B  C→B  D→D  E→E  F→F  G→G  H→H  I→I  J→J  K→K  L→L

  Set{B C }  Set{E G }
  Tree: A→A  B→B  C→B  D→D  E→E  F→F  G→E  H→H  I→I  J→J  K→K  L→L

  Set{B C }  Set{E G }  Set{J K }
  Tree: A→A  B→B  C→B  D→D  E→E  F→F  G→E  H→H  I→I  J→J  K→J  L→L

  Set{B C }  Set{E G }  Set{I J K }
  Tree: A→A  B→B  C→B  D→D  E→E  F→F  G→E  H→H  I→J  J→J  K→J  L→L

  Set{A B C }  Set{E G }  Set{I J K }
  Tree: A→B  B→B  C→B  D→D  E→E  F→F  G→E  H→H  I→J  J→J  K→J  L→L

  Set{A B C }  Set{D F }  Set{E G }  Set{I J K }
  Tree: A→B  B→B  C→B  D→D  E→E  F→D  G→E  H→H  I→J  J→J  K→J  L→L

  Set{L M }  Set{A B C }  Set{D F }  Set{E G }  Set{I J K }
  Tree: A→B  B→B  C→B  D→D  E→E  F→D  G→E  H→H  I→J  J→J  K→J  L→L

  Set{L M }  Set{A B C }  Set{D F }  Set{E G I J K }
  Tree: A→B  B→B  C→B  D→D  E→E  F→D  G→E  H→H  I→E  J→E  K→E  L→L

  Set{D F L M }  Set{A B C }  Set{E G I J K }
  Tree: A→B  B→B  C→B  D→D  E→E  F→D  G→E  H→H  I→E  J→E  K→E  L→D

  Set{A B C D F L M }  Set{E G I J K }
  Tree: A→D  B→D  C→D  D→D  E→E  F→D  G→E  H→H  I→E  J→E  K→E  L→D

  Set{A B C D F L M }  Set{E G I J K }
  Tree: A→D  B→D  C→D  D→D  E→E  F→D  G→E  H→H  I→E  J→E  K→E  L→D

  Set{A B C D E F G I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→H  I→E  J→E  K→E  L→E

  Set{A B C D E F G I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→H  I→E  J→E  K→E  L→E

  Set{A B C D E F G I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→H  I→E  J→E  K→E  L→E

  Set{A B C D E F G H I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

  Set{A B C D E F G H I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

  Set{A B C D E F G H I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

  Set{A B C D E F G H I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

  Set{A B C D E F G H I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

  Set{A B C D E F G H I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

  Set{A B C D E F G H I J K L M }
  Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

Finished Kruskals Minimal Spanning Tree.
```

CMPU2002 Assignment Report
Student Number: C21394933

```
Finished Kruskals Minimal Spanning Tree.
Set{A B C D E F G H I J K L M }
Tree: A→E  B→E  C→E  D→E  E→E  F→E  G→E  H→E  I→E  J→E  K→E  L→E

Minimum spanning tree build from following edges:
Edge B--1--C
Edge E--1--G
Edge J--1--K
Edge I--1--K
Edge A--1--B
Edge D--1--F
Edge L--1--M
Edge G--1--J
Edge F--2--L
Edge B--2--D
Edge E--2--F
Edge H--2--I
```

Total MST = 16

Discussion/Analysis/Reflection

After completion of the assignment the main key factors I have learnt are:

- Time complexity
    - How much it will take for an algo to complete.
- Space complexity
    - How much memory an algo will take.
- Searching
    - The different approaches to Searching for a point on a graph.
- Routing
    - The different approaches to take when trying to find a path from A to B
- Algo Improvements
    - MST Prim improved in time and space complexity using Adjacency lists
    - MST Kruskals improved in time complexity using path compression and Union by rank

Firstly, the MST Prim and MST Kruskal algorithms were particularly interesting to me as they are used to find the minimum spanning tree of a graph. I found that both algorithms are quite efficient in terms of time complexity, with MST Kruskal having a slightly better time complexity than MST Prim. However, the space complexity of both algorithms is the same, which is O(V+E). It was interesting to see how these algorithms can be used to solve a wide range of real-world problems such as network design.

Secondly, the Shortest Path (STP) Dijkstra algorithm is a widely-used algorithm in graph theory that is specifically designed to find the shortest path between two vertices in a graph with non-negative edge weights. This algorithm was originally introduced by Edsger Dijkstra in 1956 and has since been widely adopted in various applications, including the popular mapping services offered by Google Maps and Apple Maps.

Finally BF and DF search were both useful for performing various searches on a graph.

In conclusion, this assignment has provided me with a valuable learning experience and a deeper appreciation for the versatility and efficiency of these algorithms in solving various graph-related problems. I am now equipped with a better understanding of the time and space complexity of each

algorithm and how to choose the most appropriate algorithm for a given problem. I look forward to further exploring graph theory and its applications in the future.