Algorithms and Data Structures Assignment Report

Name: Christopher Noblett
Student number: C22454222

1. Introduction/explanation
2. Adjacency lists diagram
3. Step by step construction of the MST and contents of arrays using Prim's algorithm
4. Step by step construction of the SPT and contents of arrays using Dijkstra's algorithm
5. Step by step construction of the MST, union-find partition and set representations using Kruskal's algorithm
6. Diagram showing the MST superimposed on the graph
7. Diagram showing the SPT superimposed on the graph
8. Screen captures showing all my programs executing
9. Discussion/analysis/reflection

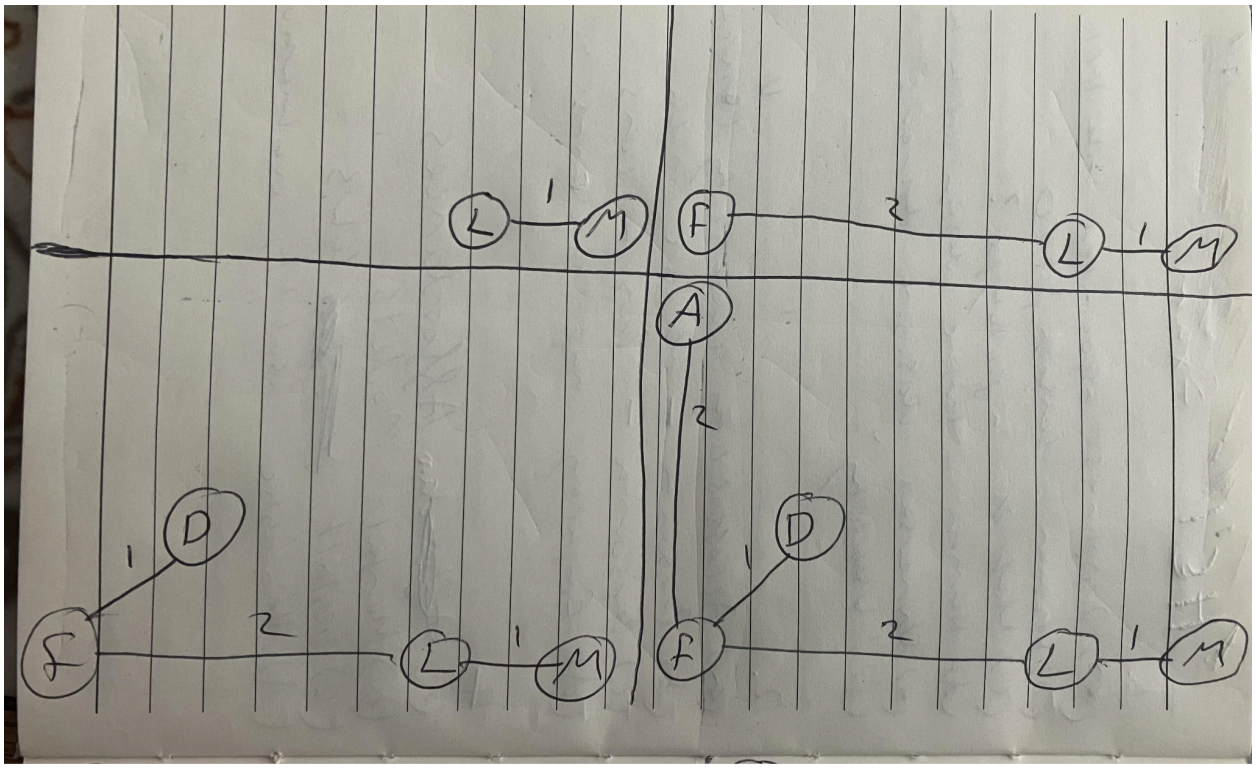
1. **Introduction/Explanation**

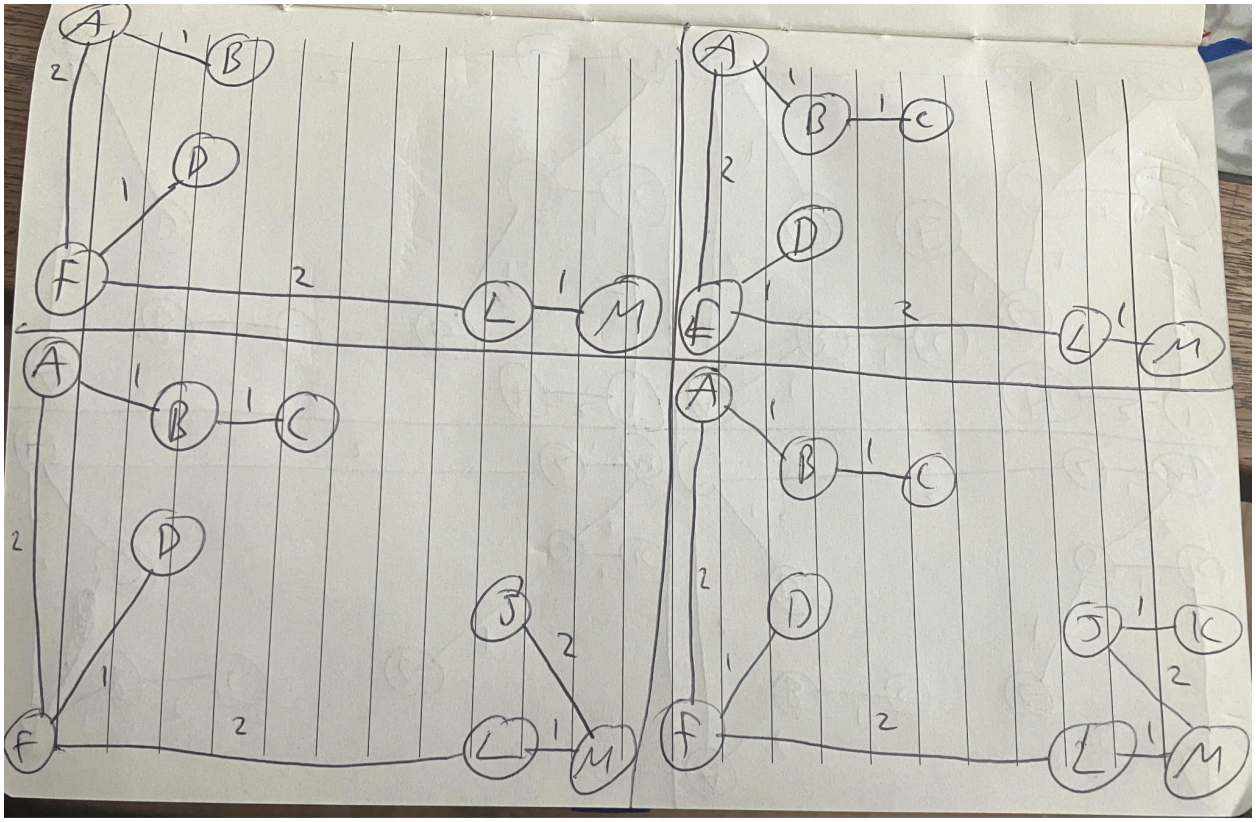
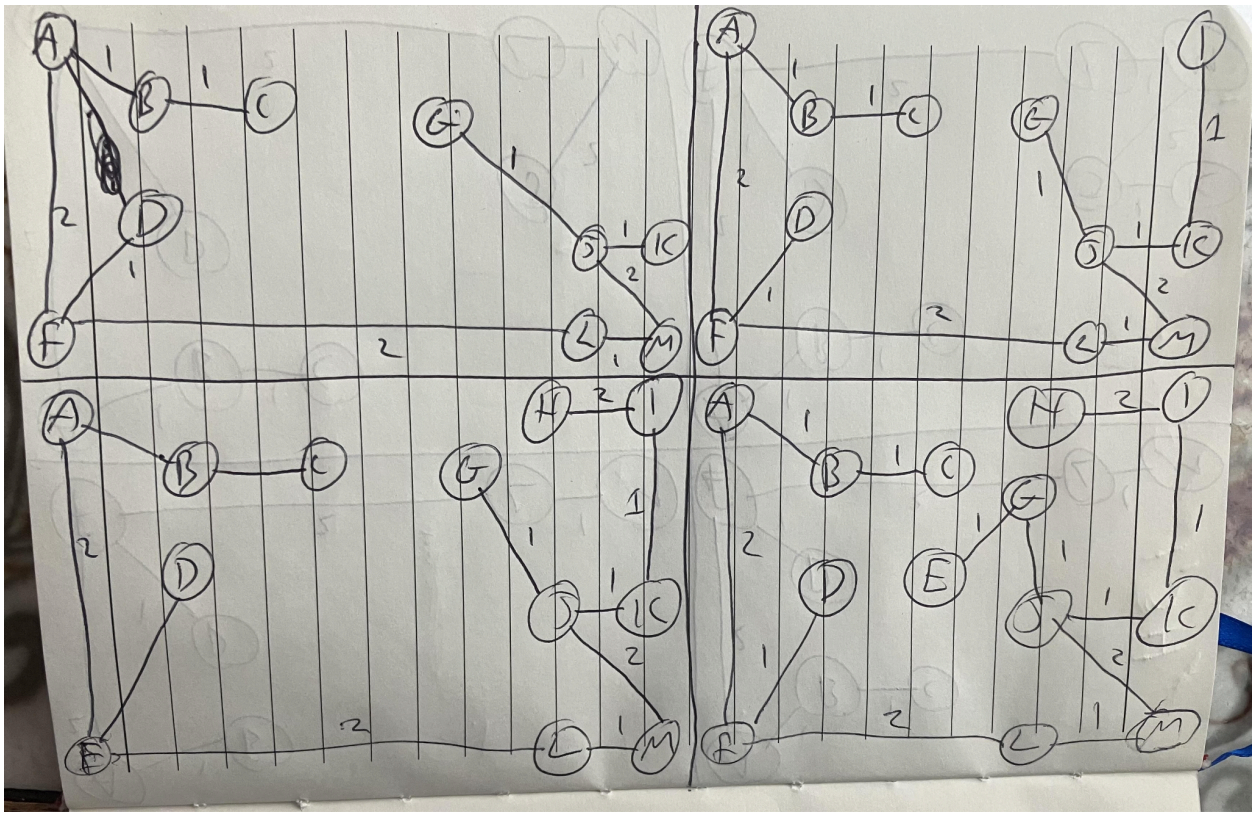
   The provided Java files offer implementations of essential graph algorithms, each serving a distinct purpose in graph analysis and traversal. I worked on some of the code with a few classmates which was allowed . "GraphLists.java " demonstrates the applications of Prim's Minimum Spanning Tree (MST), Depth-First Search (DFS), and Breadth-First Search (BFS). This file allows users to input a graph definition from a file and applies these algorithms to explore and analyze the graph structure efficiently. Conversely, "SPT.java" focuses on Dijkstra's algorithm for finding the shortest path tree in weighted graphs. By inputting a graph definition file and a source vertex, users can compute and visualize the shortest paths from the source to all other vertices in the graph. Finally, "KruskalTrees.java" introduces Kruskal's algorithm for determining the Minimum Spanning Tree (MST) of a graph. It prompts users to provide a file containing the graph definition, constructs the MST using Kruskal's algorithm, and displays the edges comprising the resulting tree. Together, these Java files offer a comprehensive toolkit for analyzing, traversing, and understanding graph structures efficiently.

2. **Adjacency lists diagram**

```
Adjacency lists data structure:

Vertex      Edge Weight

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

3. **Step by step construction of the MST using Prim's algorithm**



Inserting: L

Removing: L

Visited: L

Inserting: M

Inserting: J

Inserting: G

Inserting: F

Inserting: E

Heap Contents:

Node: M, Distance: 1

Node: F, Distance: 2

Node: G, Distance: 5

Node: J, Distance: 3

Node: E, Distance: 4

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> @

B -> @

C -> @

D -> @

E -> L

F -> L

G -> L

H -> @

I -> @

J -> L

K -> @

L -> L

M -> L

Dist Array:

A -> 2147483647

B -> 2147483647

C -> 2147483647

D -> 2147483647

E -> 4

F -> 2

G -> 5

H -> 2147483647

I -> 2147483647

J -> 3

K -> 2147483647

L -> 0

M -> 1

Removing: M

Visited: M

Heap Contents:

Node: F, Distance: 2

Node: E, Distance: 4

Node: G, Distance: 5

Node: J, Distance: 2

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> @

B -> @

C -> @

D -> @

E -> L

F -> L

G -> L

H -> @

I -> @

J -> M

K -> @

L -> L

M -> L

Dist Array:

A -> 2147483647

B -> 2147483647

C -> 2147483647

D -> 2147483647

E -> 4

F -> 2

G -> 5

H -> 2147483647

I -> 2147483647

J -> 2

K -> 2147483647

L -> 0

M -> -1

Removing: F

Visited: F

Inserting: D

Inserting: A

Heap Contents:

Node: D, Distance: 1

Node: J, Distance: 2

Node: G, Distance: 5

Node: E, Distance: 2

Node: A, Distance: 2

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> @

C -> @

D -> F

E -> F

F -> L

G -> L

H -> @

I -> @

J -> M

K -> @

L -> L

M -> L

Dist Array:

A -> 2

B -> 2147483647

C -> 2147483647

D -> 1

E -> 2

F -> -2

G -> 5

H -> 2147483647

I -> 2147483647

J -> 2

K -> 2147483647

L -> 0

M -> -1

Removing: D

Visited: D

Inserting: B

Heap Contents:

Node: A, Distance: 2

Node: J, Distance: 2

Node: G, Distance: 5

Node: E, Distance: 2

Node: B, Distance: 2

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> D

C -> @

D -> F

E -> F

F -> L

G -> L

H -> @

I -> @

J -> M

K -> @

L -> L

M -> L

Dist Array:

A -> 2

B -> 2

C -> 2147483647

D -> -1

E -> 2

F -> -2

G -> 5

H -> 2147483647

I -> 2147483647

J -> 2

K -> 2147483647

L -> 0

M -> -1

Removing: A

Visited: A

Heap Contents:

Node: B, Distance: 1

Node: J, Distance: 2

Node: G, Distance: 5

Node: E, Distance: 2

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> @

D -> F

E -> F

F -> L

G -> L

H -> @

I -> @

J -> M

K -> @

L -> L

M -> L

Dist Array:

A -> -2

B -> 1

C -> 2147483647

D -> -1

E -> 2

F -> -2

G -> 5

H -> 2147483647

I -> 2147483647

J -> 2

K -> 2147483647

L -> 0

M -> -1

Removing: B

Visited: B

Inserting: C

Heap Contents:

Node: C, Distance: 1

Node: E, Distance: 2

Node: G, Distance: 5

Node: J, Distance: 2

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> F

F -> L

G -> L

H -> @

I -> @

J -> M

K -> @

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> 1

D -> -1

E -> 2

F -> -2

G -> 5

H -> 2147483647

I -> 2147483647

J -> 2

K -> 2147483647

L -> 0

M -> -1

Removing: C

Visited: C

Heap Contents:

Node: J, Distance: 2

Node: E, Distance: 2

Node: G, Distance: 5

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> F

F -> L

G -> L

H -> @

I -> @

J -> M

K -> @

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> -1

D -> -1

E -> 2

F -> -2

G -> 5

H -> 2147483647

I -> 2147483647

J -> 2

K -> 2147483647

L -> 0

M -> -1

Removing: J

Visited: J

Inserting: K

Heap Contents:

Node: K, Distance: 1

Node: E, Distance: 2

Node: G, Distance: 1

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> F

F -> L

G -> J

H -> @

I -> @

J -> M

K -> J

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> -1

D -> -1

E -> 2

F -> -2

G -> 1

H -> 2147483647

I -> 2147483647

J -> -2

K -> 1

L -> 0

M -> -1

Removing: K

Visited: K

Inserting: I

Heap Contents:

Node: G, Distance: 1

Node: E, Distance: 2

Node: I, Distance: 1

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> F

F -> L

G -> J

H -> @

I -> K

J -> M

K -> J

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> -1

D -> -1

E -> 2

F -> -2

G -> 1

H -> 2147483647

I -> 1

J -> -2

K -> -1

L -> 0

M -> -1

Removing: G

Visited: G

Inserting: H

Heap Contents:

Node: I, Distance: 1

Node: E, Distance: 1

Node: H, Distance: 3

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> G

F -> L

G -> J

H -> G

I -> K

J -> M

K -> J

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> -1

D -> -1

E -> 1

F -> -2

G -> -1

H -> 3

I -> 1

J -> -2

K -> -1

L -> 0

M -> -1

Removing: I

Visited: I

Heap Contents:

Node: H, Distance: 2

Node: E, Distance: 1

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> G

F -> L

G -> J

H -> I

I -> K

J -> M

K -> J

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> -1

D -> -1

E -> 1

F -> -2

G -> -1

H -> 2

I -> -1

J -> -2

K -> -1

L -> 0

M -> -1

Removing: H

Visited: H

Heap Contents:

Node: E, Distance: 1

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> G

F -> L

G -> J

H -> I

I -> K

J -> M

K -> J

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> -1

D -> -1

E -> 1

F -> -2

G -> -1

H -> -2

I -> -1

J -> -2

K -> -1

L -> 0

M -> -1

Removing: E

Visited: E

Heap Contents:

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Node: @, Distance: 0

Parent Array:

A -> F

B -> A

C -> B

D -> F

E -> G

F -> L

G -> J

H -> I

I -> K

J -> M

K -> J

L -> L

M -> L

Dist Array:

A -> -2

B -> -1

C -> -1

D -> -1

E -> -1

F -> -2

G -> -1

H -> -2

I -> -1

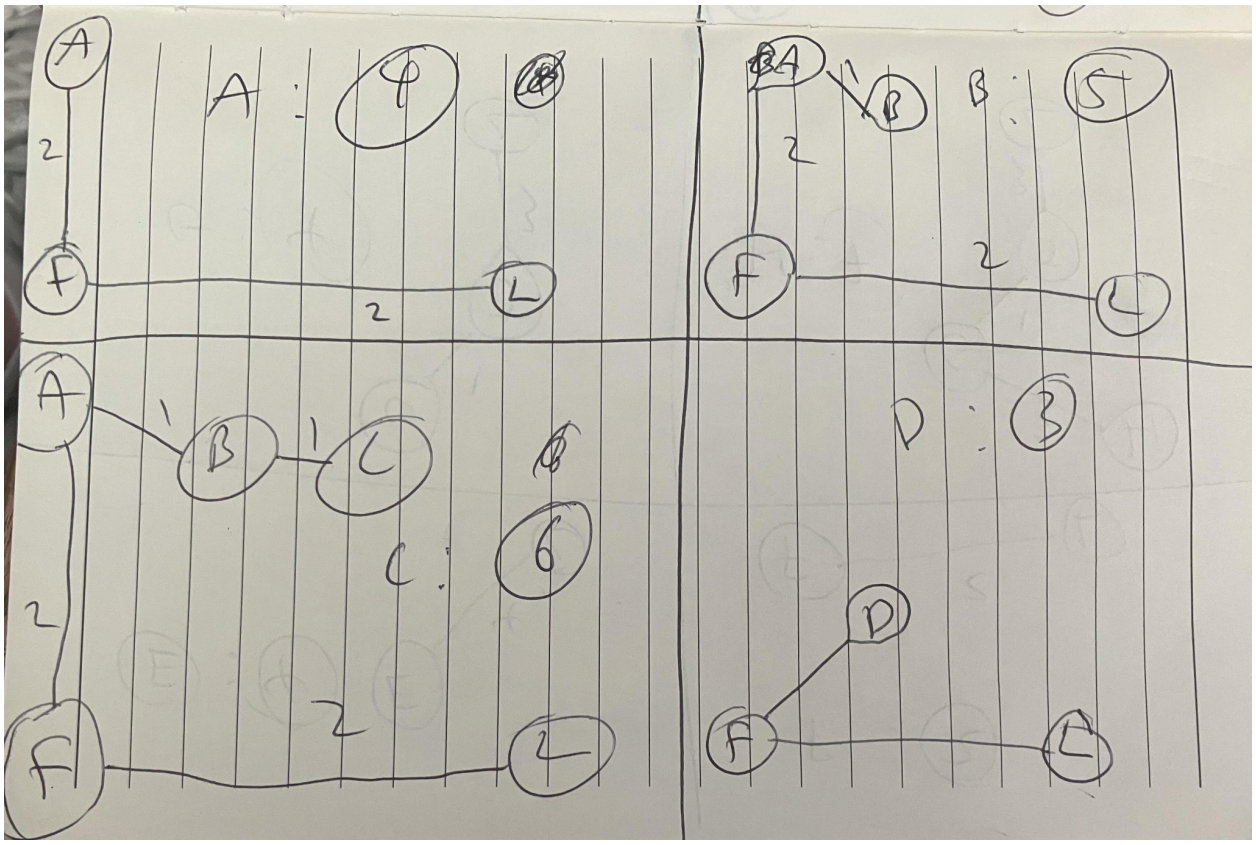
J -> -2

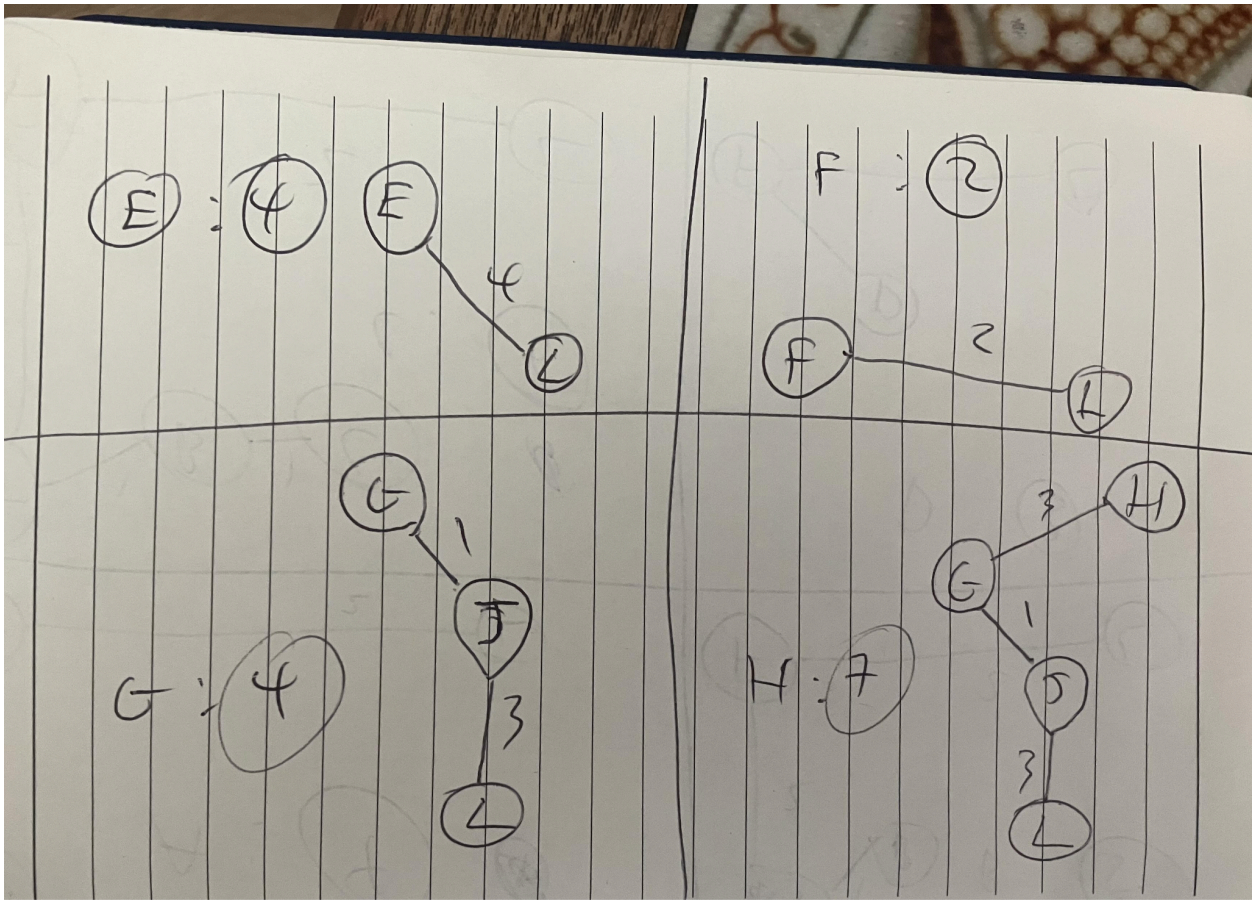
K -> -1

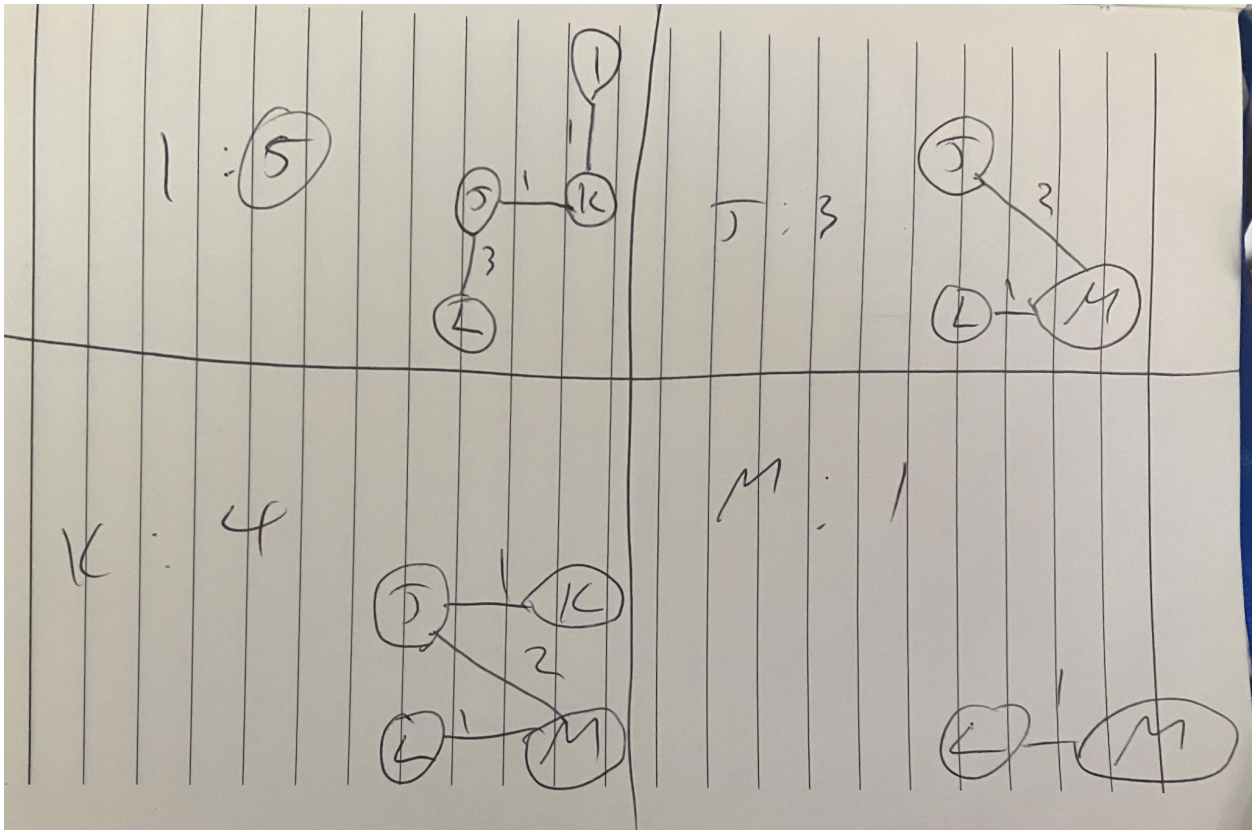
L -> 0

M -> -1

4. Step by step construction of the SPT and contents of arrays using Dijkstra's algorithm




E : 4
E
4
L
F : 2
F
2
H
G
1
5
3
L
G : 4
H : 7
H
3
G
1
5
3
L



Inserted L
Inserted E
Heap Contents:
Node: 1, Distance: 2147483647
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 2147483647
Node: 5, Distance: 4
Node: 6, Distance: 2147483647
Node: 7, Distance: 2147483647
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 2147483647
Node: 11, Distance: 2147483647
Node: 12, Distance: 0
Node: 13, Distance: 2147483647
Dist Array:
A -> 2147483647
B -> 2147483647
C -> 2147483647
D -> 2147483647
E -> 4

F -> 2147483647
G -> 2147483647
H -> 2147483647
I -> 2147483647
J -> 2147483647
K -> 2147483647
L -> 0
M -> 2147483647
Parent Array:
A -> @
B -> @
C -> @
D -> @
E -> L
F -> @
G -> @
H -> @
I -> @
J -> @
K -> @
L -> @
M -> @

Inserted F
Heap Contents:
Node: 1, Distance: 2147483647
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 2147483647
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 2147483647
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 2147483647
Node: 11, Distance: 2147483647
Node: 12, Distance: 0
Node: 13, Distance: 2147483647
Dist Array:
A -> 2147483647
B -> 2147483647
C -> 2147483647
D -> 2147483647

E -> 4
F -> 2
G -> 2147483647
H -> 2147483647
I -> 2147483647
J -> 2147483647
K -> 2147483647
L -> 0
M -> 2147483647
Parent Array:
A -> @
B -> @
C -> @
D -> @
E -> L
F -> L
G -> @
H -> @
I -> @
J -> @
K -> @
L -> @
M -> @

Inserted G
Heap Contents:
Node: 1, Distance: 2147483647
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 2147483647
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 5
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 2147483647
Node: 11, Distance: 2147483647
Node: 12, Distance: 0
Node: 13, Distance: 2147483647
Dist Array:
A -> 2147483647
B -> 2147483647
C -> 2147483647

D -> 2147483647
E -> 4
F -> 2
G -> 5
H -> 2147483647
I -> 2147483647
J -> 2147483647
K -> 2147483647
L -> 0
M -> 2147483647
Parent Array:
A -> @
B -> @
C -> @
D -> @
E -> L
F -> L
G -> L
H -> @
I -> @
J -> @
K -> @
L -> @
M -> @

Inserted J
Heap Contents:
Node: 1, Distance: 2147483647
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 2147483647
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 5
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 3
Node: 11, Distance: 2147483647
Node: 12, Distance: 0
Node: 13, Distance: 2147483647
Dist Array:
A -> 2147483647
B -> 2147483647

C -> 2147483647
D -> 2147483647
E -> 4
F -> 2
G -> 5
H -> 2147483647
I -> 2147483647
J -> 3
K -> 2147483647
L -> 0
M -> 2147483647
Parent Array:
A -> @
B -> @
C -> @
D -> @
E -> L
F -> L
G -> L
H -> @
I -> @
J -> L
K -> @
L -> @
M -> @

Inserted M
Heap Contents:
Node: 1, Distance: 2147483647
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 2147483647
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 5
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 3
Node: 11, Distance: 2147483647
Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:
A -> 2147483647

B -> 2147483647
C -> 2147483647
D -> 2147483647
E -> 4
F -> 2
G -> 5
H -> 2147483647
I -> 2147483647
J -> 3
K -> 2147483647
L -> 0
M -> 1
Parent Array:
A -> @
B -> @
C -> @
D -> @
E -> L
F -> L
G -> L
H -> @
I -> @
J -> L
K -> @
L -> @
M -> L

Inserted A
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 2147483647
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 5
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 3
Node: 11, Distance: 2147483647
Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:

A -> 4
B -> 2147483647
C -> 2147483647
D -> 2147483647
E -> 4
F -> 2
G -> 5
H -> 2147483647
I -> 2147483647
J -> 3
K -> 2147483647
L -> 0
M -> 1
Parent Array:
A -> F
B -> @
C -> @
D -> @
E -> L
F -> L
G -> L
H -> @
I -> @
J -> L
K -> @
L -> @
M -> L

Inserted D
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 5
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 3
Node: 11, Distance: 2147483647
Node: 12, Distance: 0
Node: 13, Distance: 1

Dist Array:
A -> 4
B -> 2147483647
C -> 2147483647
D -> 3
E -> 4
F -> 2
G -> 5
H -> 2147483647
I -> 2147483647
J -> 3
K -> 2147483647
L -> 0
M -> 1
Parent Array:
A -> F
B -> @
C -> @
D -> F
E -> L
F -> L
G -> L
H -> @
I -> @
J -> L
K -> @
L -> @
M -> L

Inserted G
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 4
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 3
Node: 11, Distance: 2147483647
Node: 12, Distance: 0

Node: 13, Distance: 1
Dist Array:
A -> 4
B -> 2147483647
C -> 2147483647
D -> 3
E -> 4
F -> 2
G -> 4
H -> 2147483647
I -> 2147483647
J -> 3
K -> 2147483647
L -> 0
M -> 1
Parent Array:
A -> F
B -> @
C -> @
D -> F
E -> L
F -> L
G -> J
H -> @
I -> @
J -> L
K -> @
L -> @
M -> L

Inserted K
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 2147483647
Node: 3, Distance: 2147483647
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 4
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 3
Node: 11, Distance: 4

Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:
A -> 4
B -> 2147483647
C -> 2147483647
D -> 3
E -> 4
F -> 2
G -> 4
H -> 2147483647
I -> 2147483647
J -> 3
K -> 4
L -> 0
M -> 1
Parent Array:
A -> F
B -> @
C -> @
D -> F
E -> L
F -> L
G -> J
H -> @
I -> @
J -> L
K -> J
L -> @
M -> L

Inserted B
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 5
Node: 3, Distance: 2147483647
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 4
Node: 8, Distance: 2147483647
Node: 9, Distance: 2147483647
Node: 10, Distance: 3

Node: 11, Distance: 4
Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:
A -> 4
B -> 5
C -> 2147483647
D -> 3
E -> 4
F -> 2
G -> 4
H -> 2147483647
I -> 2147483647
J -> 3
K -> 4
L -> 0
M -> 1
Parent Array:
A -> F
B -> D
C -> @
D -> F
E -> L
F -> L
G -> J
H -> @
I -> @
J -> L
K -> J
L -> @
M -> L

Inserted I
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 5
Node: 3, Distance: 2147483647
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 4
Node: 8, Distance: 2147483647
Node: 9, Distance: 5

Node: 10, Distance: 3
Node: 11, Distance: 4
Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:
A -> 4
B -> 5
C -> 2147483647
D -> 3
E -> 4
F -> 2
G -> 4
H -> 2147483647
I -> 5
J -> 3
K -> 4
L -> 0
M -> 1
Parent Array:
A -> F
B -> D
C -> @
D -> F
E -> L
F -> L
G -> J
H -> @
I -> K
J -> L
K -> J
L -> @
M -> L

Inserted C
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 5
Node: 3, Distance: 8
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 4
Node: 8, Distance: 2147483647

Node: 9, Distance: 5
Node: 10, Distance: 3
Node: 11, Distance: 4
Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:
A -> 4
B -> 5
C -> 8
D -> 3
E -> 4
F -> 2
G -> 4
H -> 2147483647
I -> 5
J -> 3
K -> 4
L -> 0
M -> 1
Parent Array:
A -> F
B -> D
C -> E
D -> F
E -> L
F -> L
G -> J
H -> @
I -> K
J -> L
K -> J
L -> @
M -> L

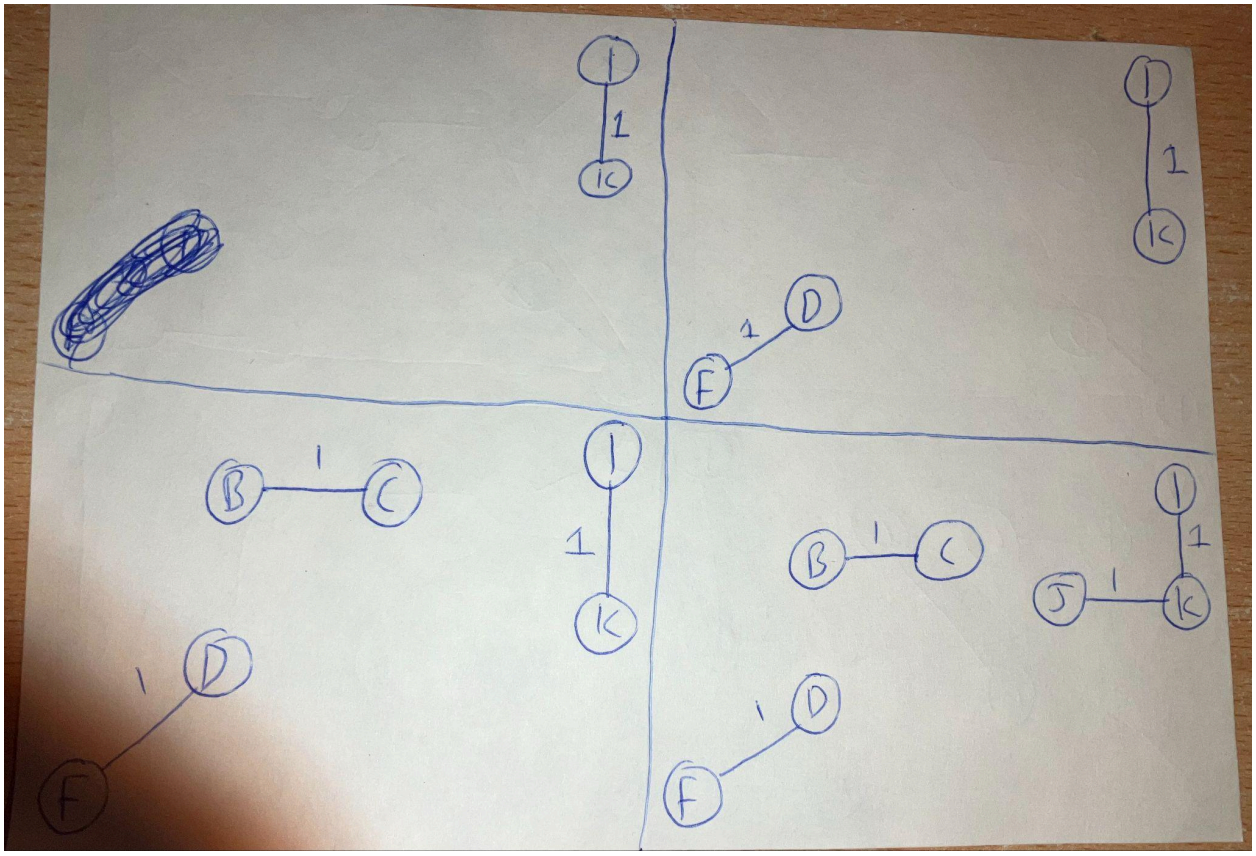
Inserted H
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 5
Node: 3, Distance: 8
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2
Node: 7, Distance: 4

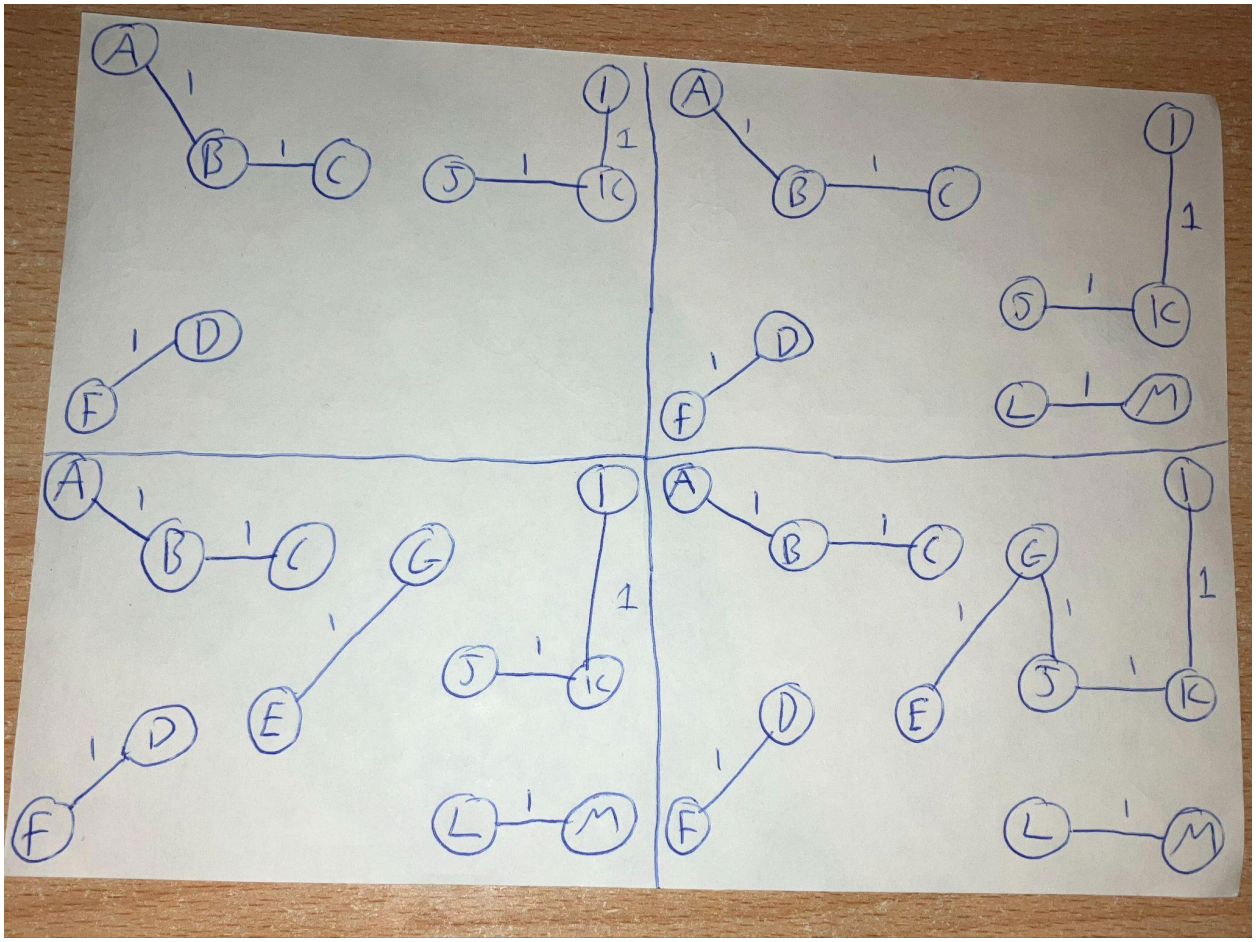
Node: 8, Distance: 7
Node: 9, Distance: 5
Node: 10, Distance: 3
Node: 11, Distance: 4
Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:
A -> 4
B -> 5
C -> 8
D -> 3
E -> 4
F -> 2
G -> 4
H -> 7
I -> 5
J -> 3
K -> 4
L -> 0
M -> 1
Parent Array:
A -> F
B -> D
C -> E
D -> F
E -> L
F -> L
G -> J
H -> G
I -> K
J -> L
K -> J
L -> @
M -> L

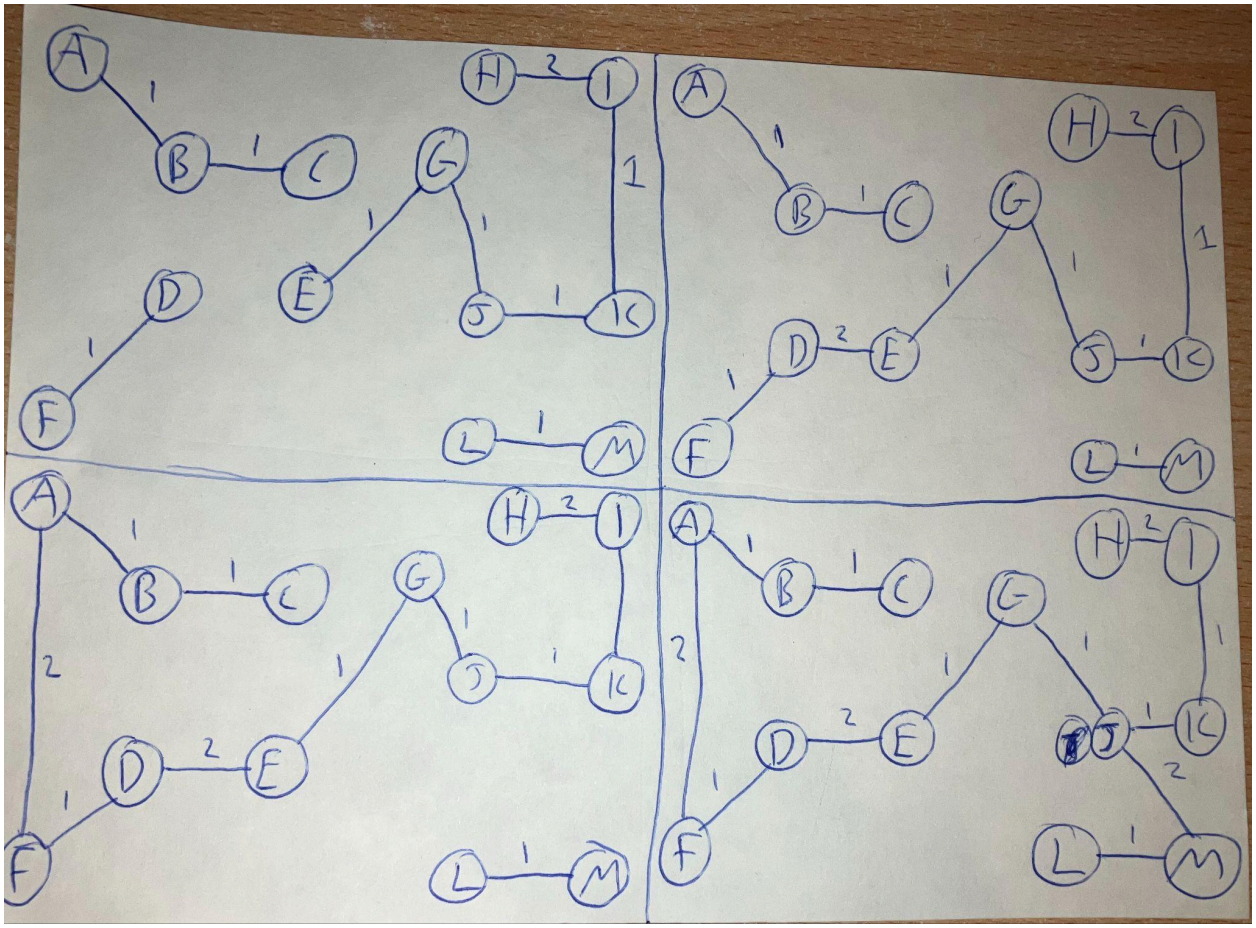
Inserted C
Heap Contents:
Node: 1, Distance: 4
Node: 2, Distance: 5
Node: 3, Distance: 6
Node: 4, Distance: 3
Node: 5, Distance: 4
Node: 6, Distance: 2

Node: 7, Distance: 4
Node: 8, Distance: 7
Node: 9, Distance: 5
Node: 10, Distance: 3
Node: 11, Distance: 4
Node: 12, Distance: 0
Node: 13, Distance: 1
Dist Array:
A -> 4
B -> 5
C -> 6
D -> 3
E -> 4
F -> 2
G -> 4
H -> 7
I -> 5
J -> 3
K -> 4
L -> 0
M -> 1
Parent Array:
A -> F
B -> D
C -> B
D -> F
E -> L
F -> L
G -> J
H -> G
I -> K
J -> L
K -> J
L -> @
M -> L

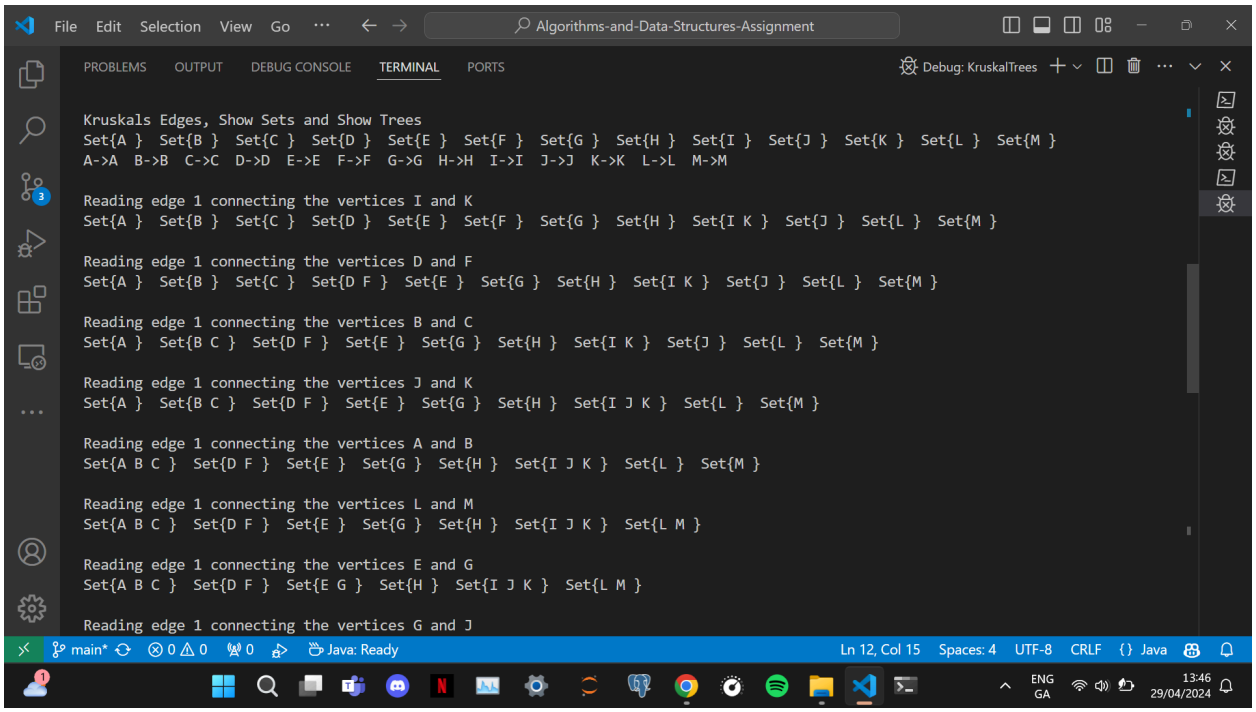
5. Step by step construction of the MST using Kruskal's algorithm




A
B
C
I
J
K
D
F
L
M
G
E



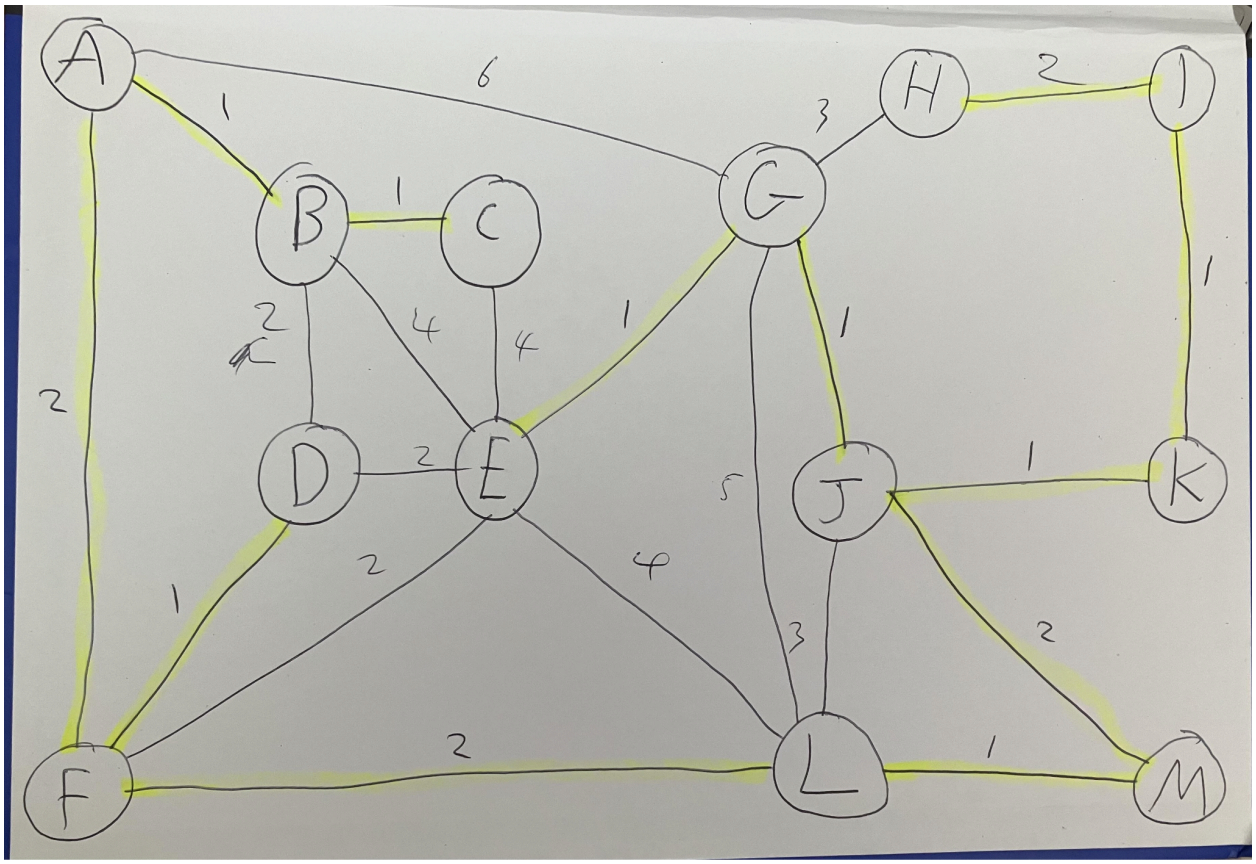
Union-find partition and set representations



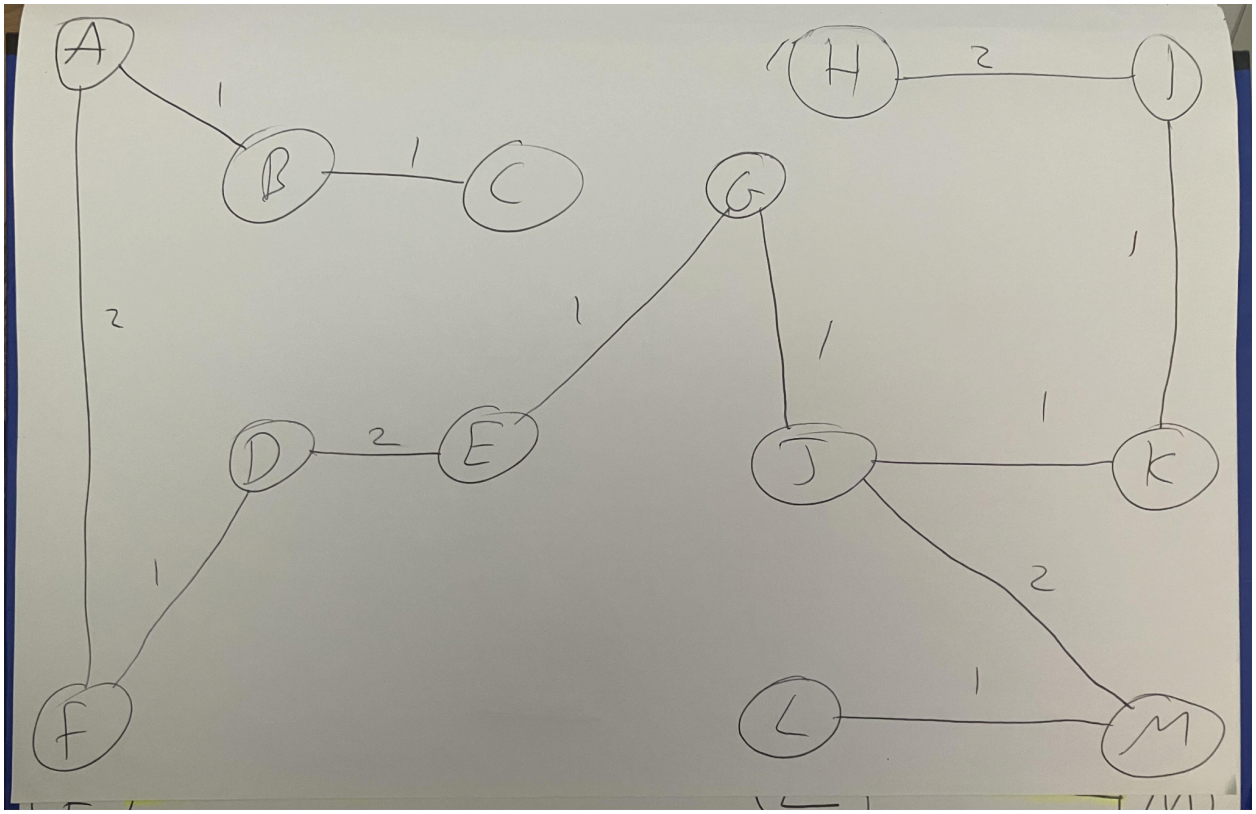
```
Reading edge 1 connecting the vertices G and J
Set{A B C }  Set{D F }  Set{E G I J K }  Set{H }  Set{L M }

Reading edge 2 connecting the vertices H and I
Set{A B C }  Set{D F }  Set{E G H I J K }  Set{L M }

Reading edge 2 connecting the vertices D and E
Set{A B C }  Set{D E F G H I J K }  Set{L M }

Reading edge 2 connecting the vertices A and F
Set{A B C D E F G H I J K }  Set{L M }

Reading edge 2 connecting the vertices J and M
Set{A B C D E F G H I J K L M }
```

6. Diagram showing the MST superimposed on the graph

Prim's MST

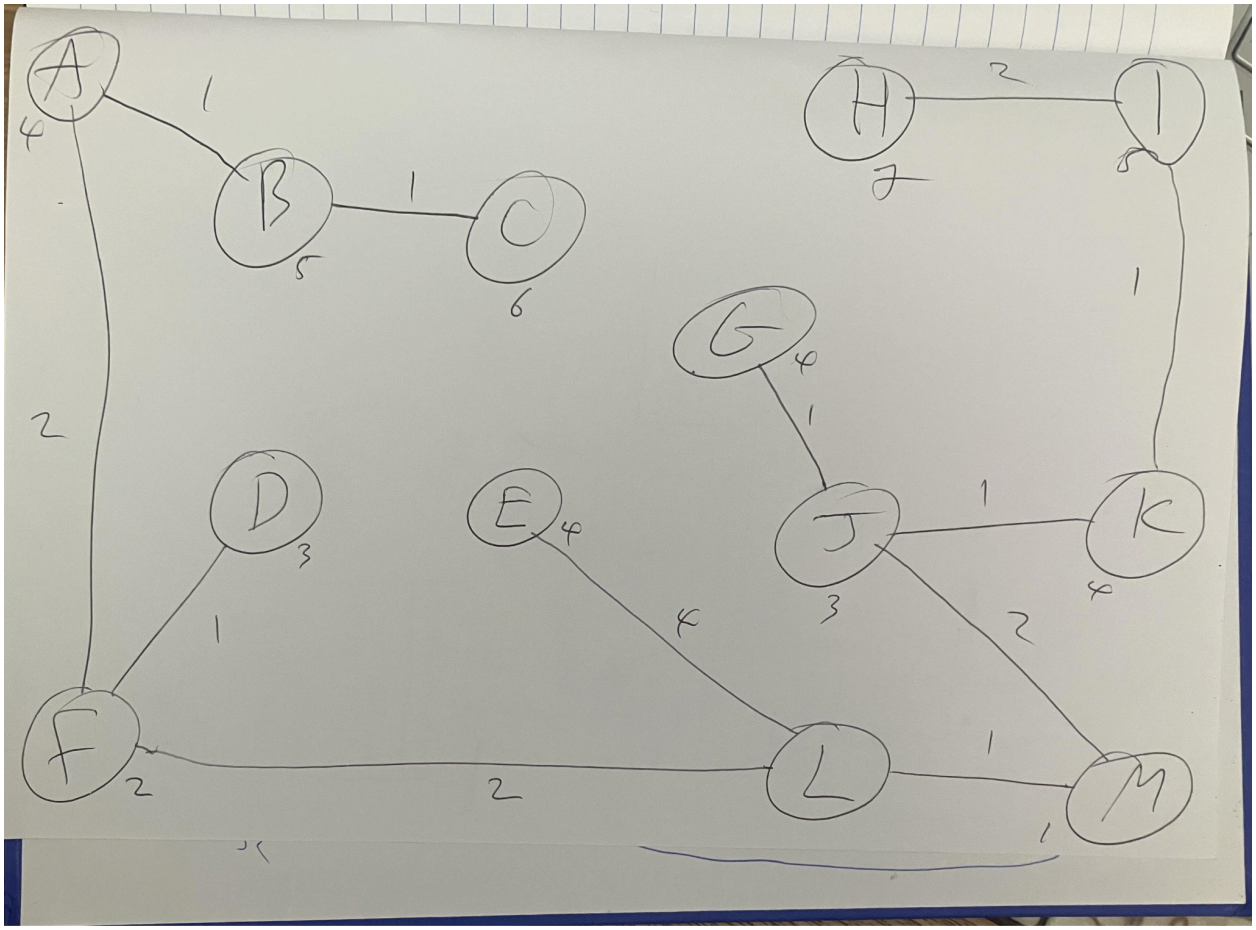


Kruskal's MST



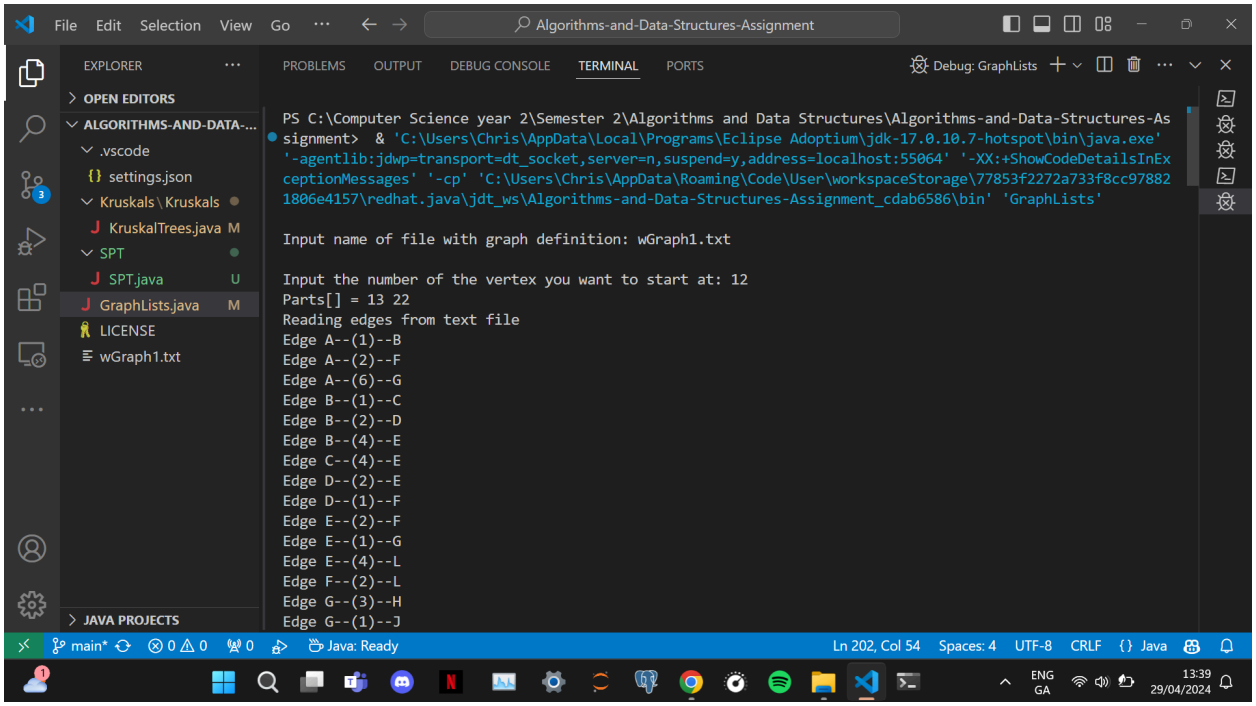
7. Diagram showing the SPT superimposed on the graph

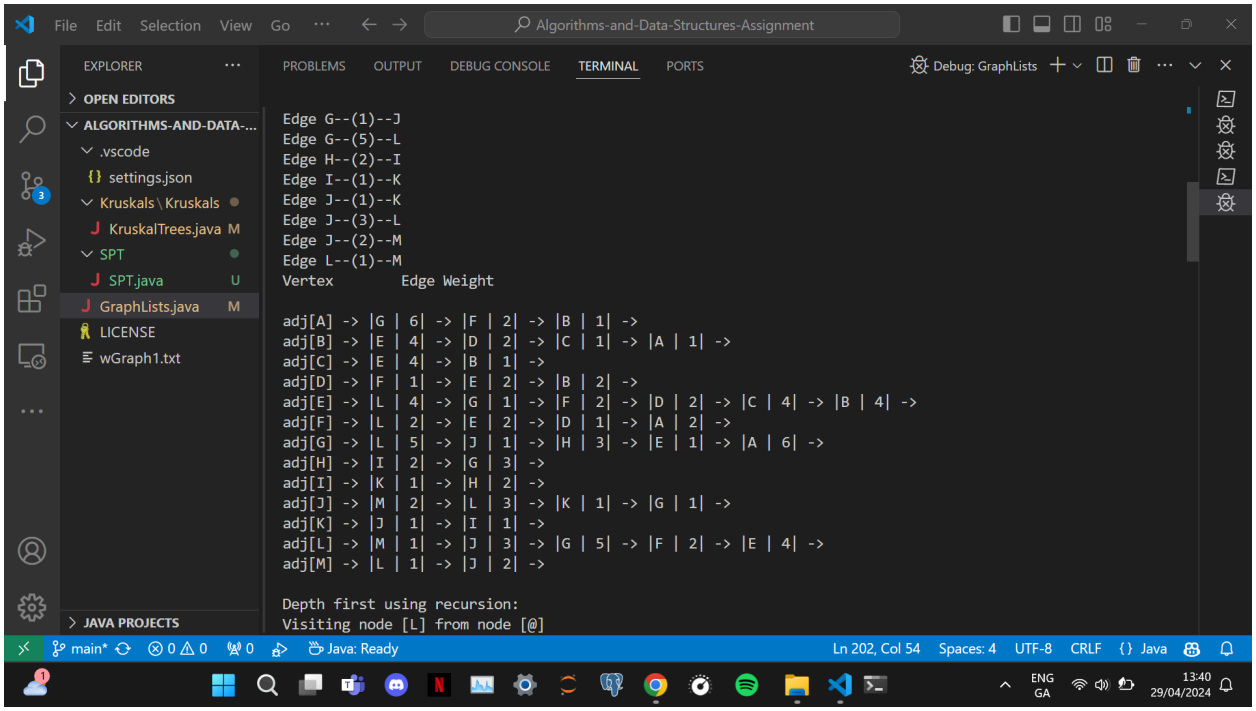
Djikstra's SPT

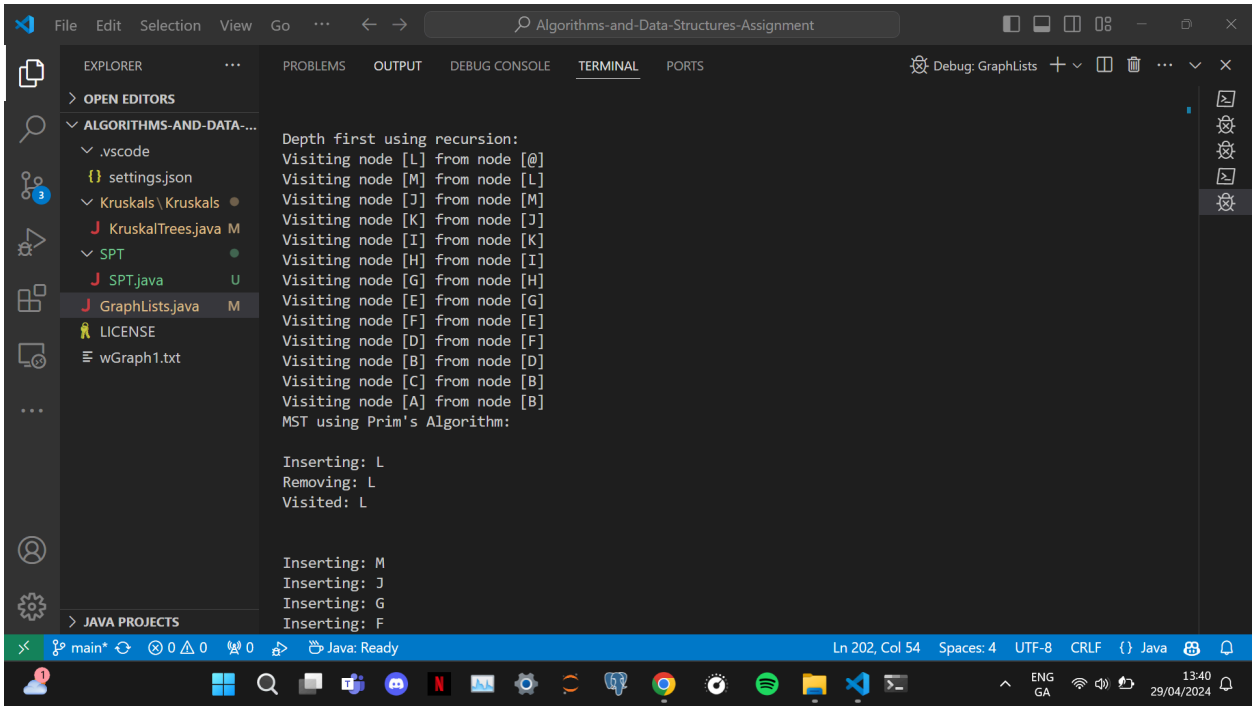


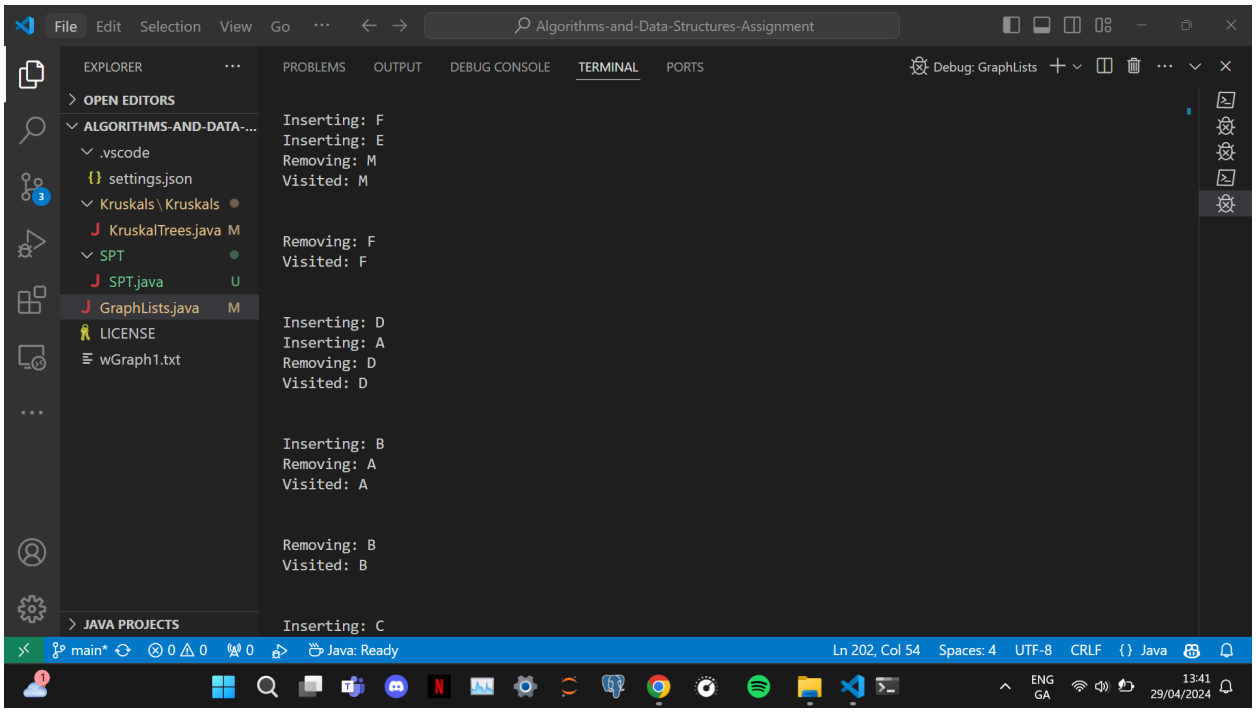
8. Screen captures showing all my programs executing

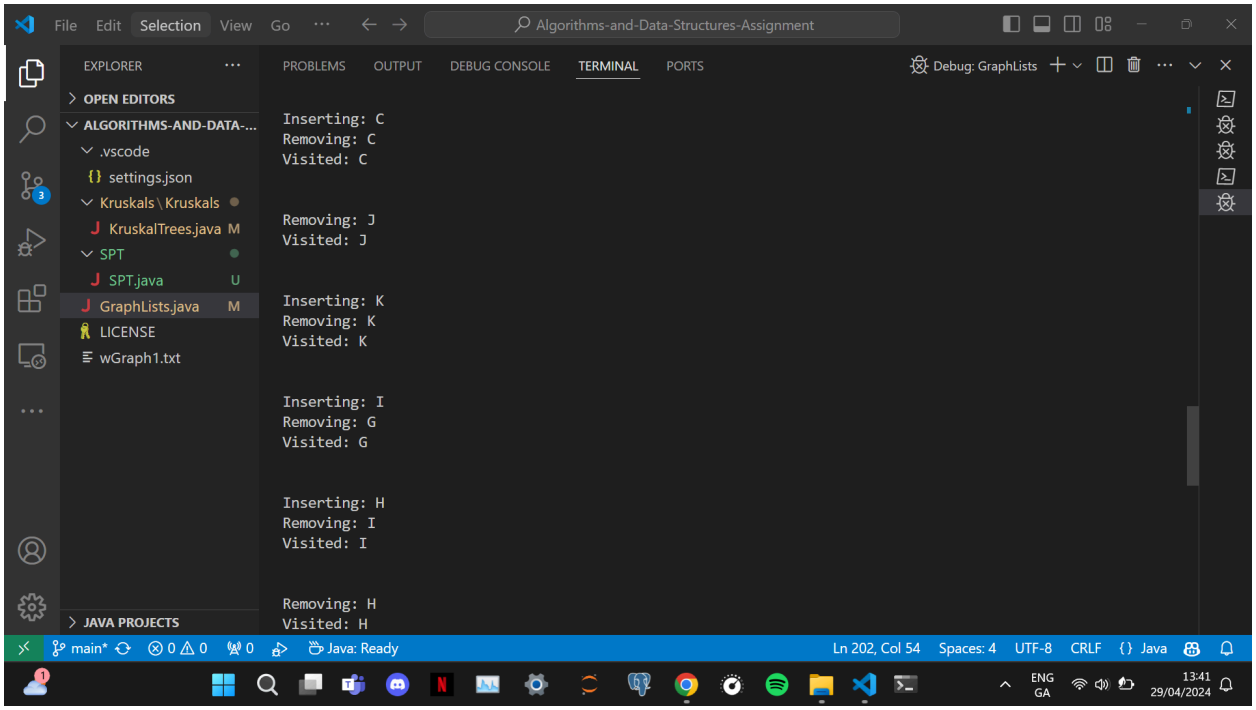
GraphLists.java

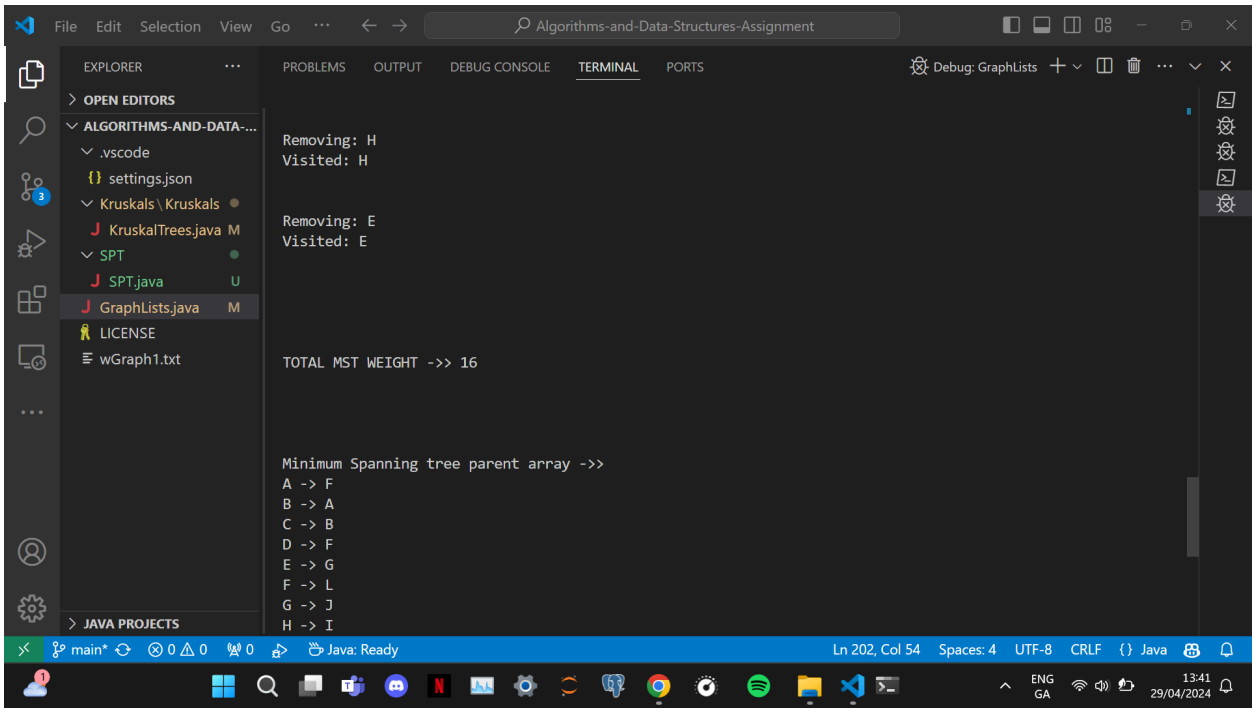

File Edit Selection View Go
Algorithms-and-Data-Structures-Assignment
EXPLORER
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
GraphLists.java M
LICENSE
wGraph1.txt
JAVA PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: GraphLists
PS C:\Computer Science year 2\Semester 2\Algorithms and Data Structures\Algorithms-and-Data-Structures-Assignment> & 'C:\Users\Chris\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.10.7-hotspot\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:55064' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Chris\AppData\Roaming\Code\User\workspaceStorage\77853f2272a733f8cc978821806e4157\redhat.java\jdt_ws\Algorithms-and-Data-Structures-Assignment_cdab6586\bin' 'GraphLists'
Input name of file with graph definition: wGraph1.txt
Input the number of the vertex you want to start at: 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--(1)--G
Edge E--(4)--L
Edge F--(2)--L
Edge G--(3)--H
Edge G--(1)--J
main*
Java: Ready
Ln 202, Col 54
Spaces: 4
UTF-8
CRLF
Java
ENG
GA
13:39
29/04/2024

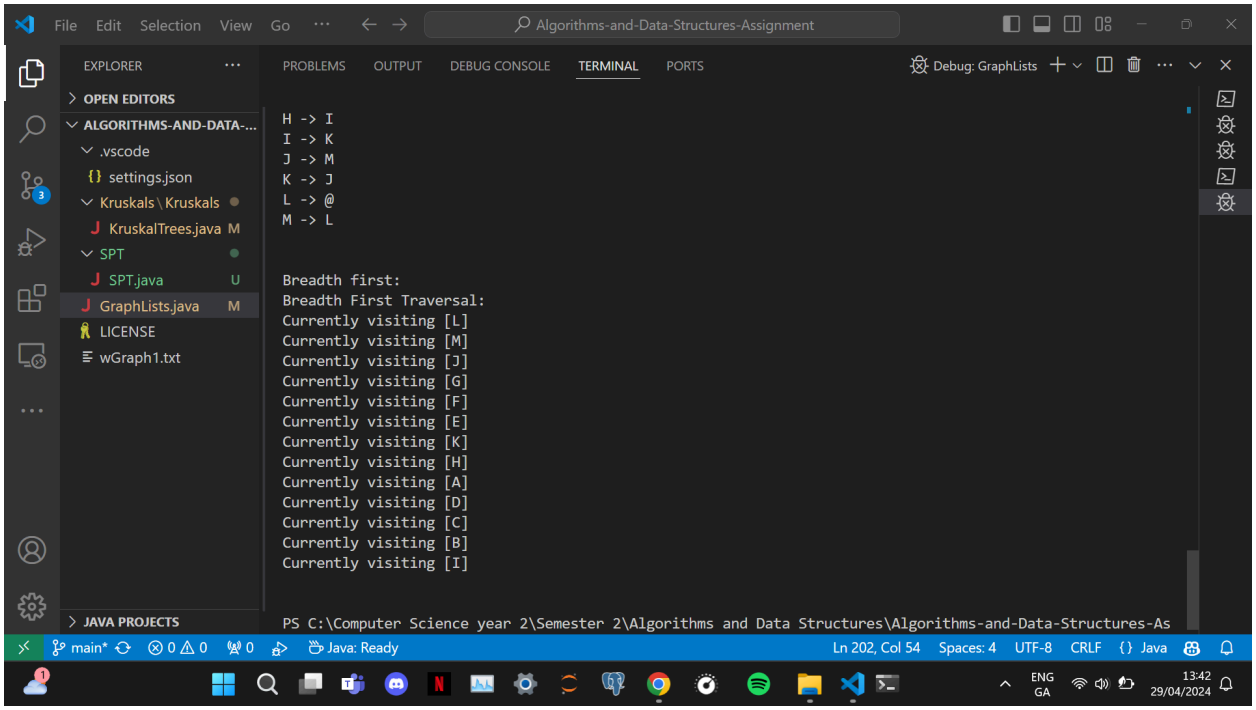

File Edit Selection View Go
Algorithms-and-Data-Structures-Assignment
EXPLORER
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
GraphLists.java M
LICENSE
wGraph1.txt
JAVA PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: GraphLists
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
Vertex Edge Weight
adj[A] -> |G | 6| -> |F | 2| -> |B | 1| ->
adj[B] -> |E | 4| -> |D | 2| -> |C | 1| -> |A | 1| ->
adj[C] -> |E | 4| -> |B | 1| ->
adj[D] -> |F | 1| -> |E | 2| -> |B | 2| ->
adj[E] -> |L | 4| -> |G | 1| -> |F | 2| -> |D | 2| -> |C | 4| -> |B | 4| ->
adj[F] -> |L | 2| -> |E | 2| -> |D | 1| -> |A | 2| ->
adj[G] -> |L | 5| -> |J | 1| -> |H | 3| -> |E | 1| -> |A | 6| ->
adj[H] -> |I | 2| -> |G | 3| ->
adj[I] -> |K | 1| -> |H | 2| ->
adj[J] -> |M | 2| -> |L | 3| -> |K | 1| -> |G | 1| ->
adj[K] -> |J | 1| -> |I | 1| ->
adj[L] -> |M | 1| -> |J | 3| -> |G | 5| -> |F | 2| -> |E | 4| ->
adj[M] -> |L | 1| -> |J | 2| ->
Depth first using recursion:
Visiting node [L] from node [@]
main*
Java: Ready
Ln 202, Col 54
Spaces: 4
UTF-8
CRLF
Java
ENG
GA
13:40
29/04/2024


File Edit Selection View Go
Algorithms-and-Data-Structures-Assignment
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: GraphLists
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
GraphLists.java M
LICENSE
wGraph1.txt
JAVA PROJECTS
Depth first using recursion:
Visiting node [L] from node [@]
Visiting node [M] from node [L]
Visiting node [J] from node [M]
Visiting node [K] from node [J]
Visiting node [I] from node [K]
Visiting node [H] from node [I]
Visiting node [G] from node [H]
Visiting node [E] from node [G]
Visiting node [F] from node [E]
Visiting node [D] from node [F]
Visiting node [B] from node [D]
Visiting node [C] from node [B]
Visiting node [A] from node [B]
MST using Prim's Algorithm:
Inserting: L
Removing: L
Visited: L
Inserting: M
Inserting: J
Inserting: G
Inserting: F
main*
Java: Ready
Ln 202, Col 54
Spaces: 4
UTF-8
CRLF
Java
ENG
GA
13:40
29/04/2024

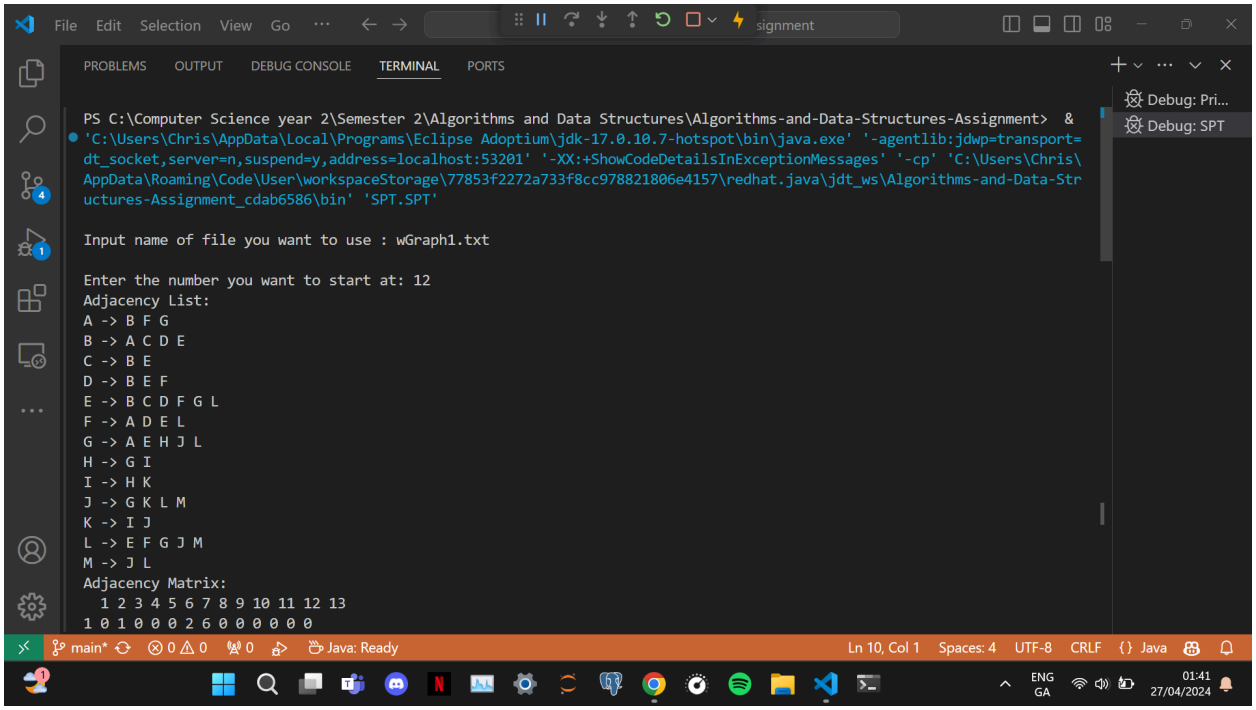

File Edit Selection View Go
Algorithms-and-Data-Structures-Assignment
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: GraphLists
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
GraphLists.java M
LICENSE
wGraph1.txt
JAVA PROJECTS
Inserting: F
Inserting: E
Removing: M
Visited: M
Removing: F
Visited: F
Inserting: D
Inserting: A
Removing: D
Visited: D
Inserting: B
Removing: A
Visited: A
Removing: B
Visited: B
Inserting: C
main*
Java: Ready
Ln 202, Col 54
Spaces: 4
UTF-8
CRLF
Java
ENG
GA
13:41
29/04/2024

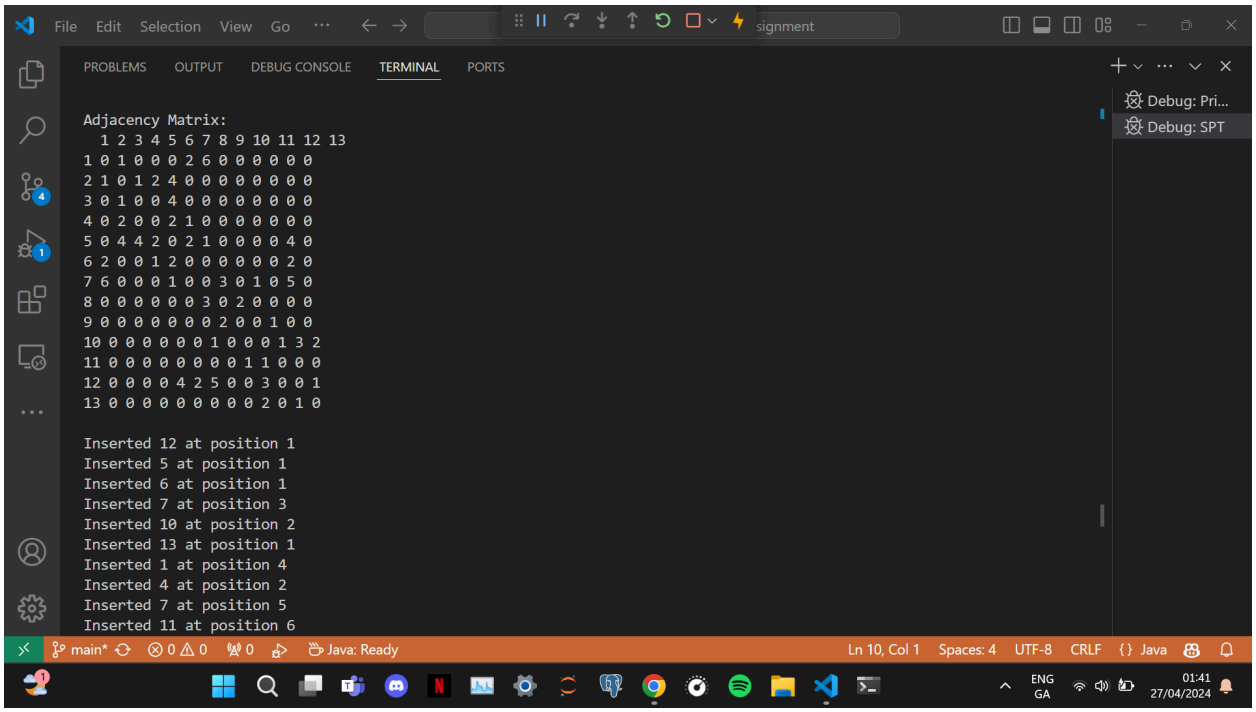

File Edit Selection View Go
Algorithms-and-Data-Structures-Assignment
EXPLORER
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
GraphLists.java M
LICENSE
wGraph1.txt
JAVA PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: GraphLists
Inserting: C
Removing: C
Visited: C
Removing: J
Visited: J
Inserting: K
Removing: K
Visited: K
Inserting: I
Removing: G
Visited: G
Inserting: H
Removing: I
Visited: I
Removing: H
Visited: H
main* 0 0 0
Java: Ready
Ln 202, Col 54 Spaces: 4 UTF-8 CRLF Java
ENG GA
13:41 29/04/2024

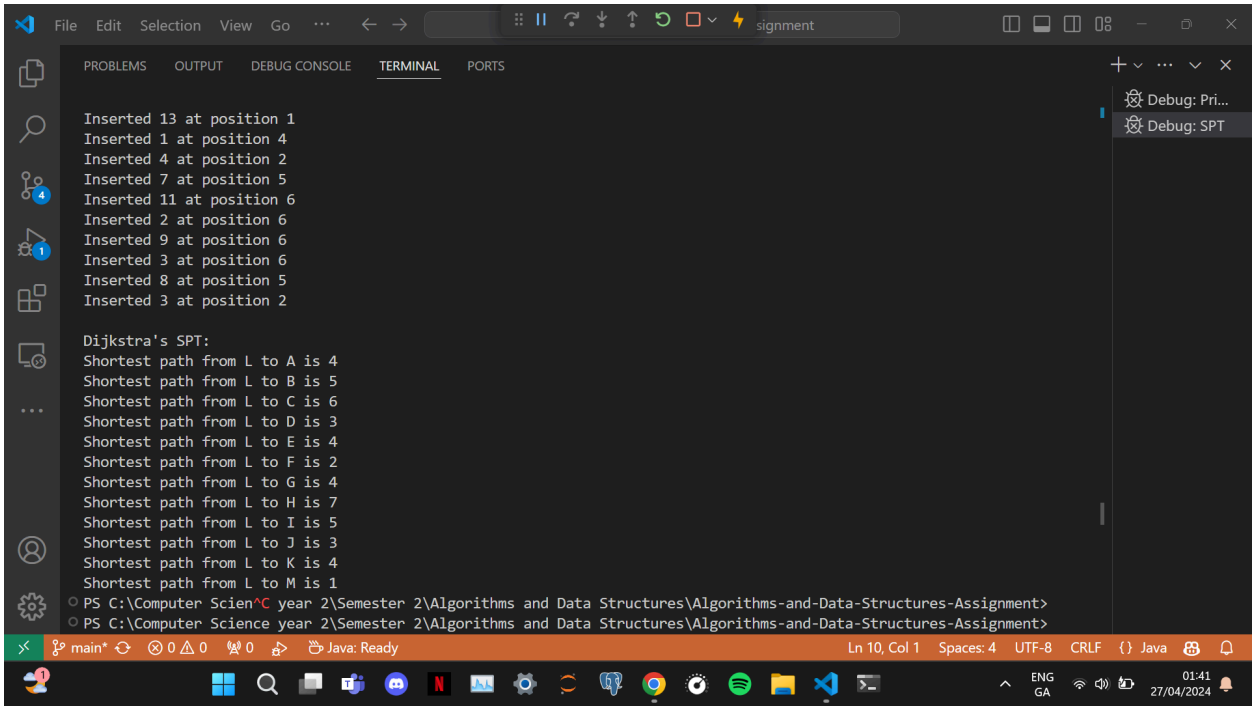

File Edit Selection View Go
Algorithms-and-Data-Structures-Assignment
EXPLORER
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
GraphLists.java M
LICENSE
wGraph1.txt
JAVA PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: GraphLists
Removing: H
Visited: H
Removing: E
Visited: E
TOTAL MST WEIGHT -->> 16
Minimum Spanning tree parent array -->>
A -> F
B -> A
C -> B
D -> F
E -> G
F -> L
G -> J
H -> I
main* 0 0 0
Java: Ready
Ln 202, Col 54 Spaces: 4 UTF-8 CRLF Java
ENG GA
13:41 29/04/2024


File Edit Selection View Go
Algorithms-and-Data-Structures-Assignment
EXPLORER
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
GraphLists.java M
LICENSE
wGraph1.txt
JAVA PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: GraphLists
H -> I
I -> K
J -> M
K -> J
L -> @
M -> L
Breadth first:
Breadth First Traversal:
Currently visiting [L]
Currently visiting [M]
Currently visiting [J]
Currently visiting [G]
Currently visiting [F]
Currently visiting [E]
Currently visiting [K]
Currently visiting [H]
Currently visiting [A]
Currently visiting [D]
Currently visiting [C]
Currently visiting [B]
Currently visiting [I]
PS C:\Computer Science year 2\Semester 2\Algorithms and Data Structures\Algorithms-and-Data-Structures-As
main*
Java: Ready
Ln 202, Col 54
Spaces: 4
UTF-8
CRLF
Java
ENG
GA
13:42
29/04/2024

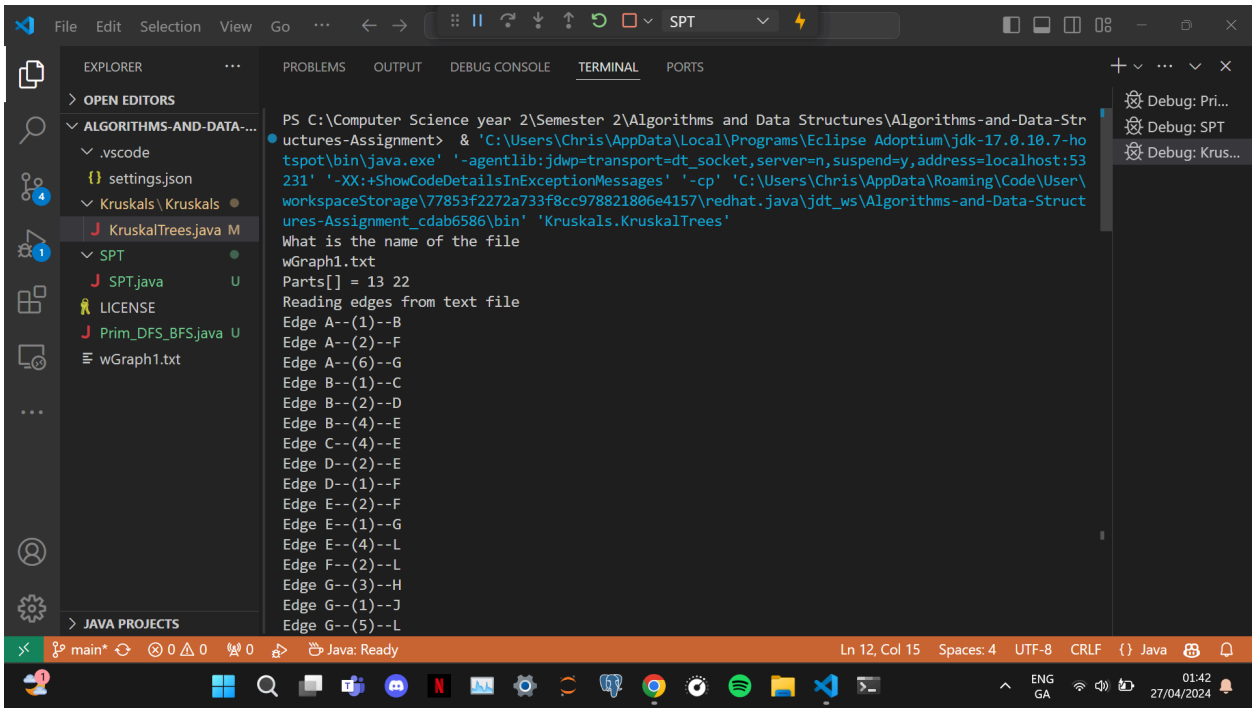
SPT.java

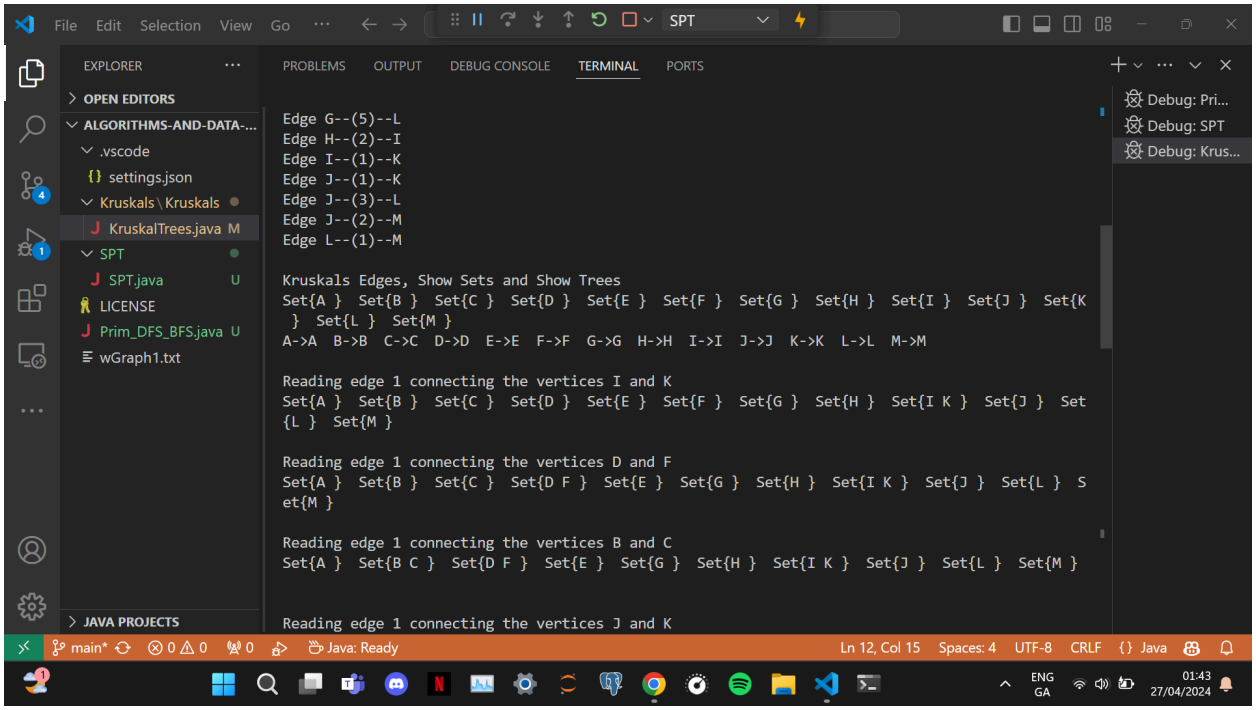

File Edit Selection View Go
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: Pri...
Debug: SPT
PS C:\Computer Science year 2\Semester 2\Algorithms and Data Structures\Algorithms-and-Data-Structures-Assignment> & 'C:\Users\Chris\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.10.7-hotspot\bin\java.exe' '-agentlib:jdwp=transport=dt_socket,server=n,suspend=y,address=localhost:53201' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Chris\AppData\Roaming\Code\User\workspaceStorage\77853f2272a733f8cc978821806e4157\redhat.java\jdt_ws\Algorithms-and-Data-Structures-Assignment_cdab6586\bin' 'SPT.SPT'
Input name of file you want to use : wGraph1.txt
Enter the number you want to start at: 12
Adjacency List:
A -> B F G
B -> A C D E
C -> B E
D -> B E F
E -> B C D F G L
F -> A D E L
G -> A E H J L
H -> G I
I -> H K
J -> G K L M
K -> I J
L -> E F G J M
M -> J L
Adjacency Matrix:
1 2 3 4 5 6 7 8 9 10 11 12 13
1 0 1 0 0 0 2 6 0 0 0 0 0 0
main* 0 0 0 Java: Ready Ln 10, Col 1 Spaces: 4 UTF-8 CRLF Java
ENG GA 01:41 27/04/2024

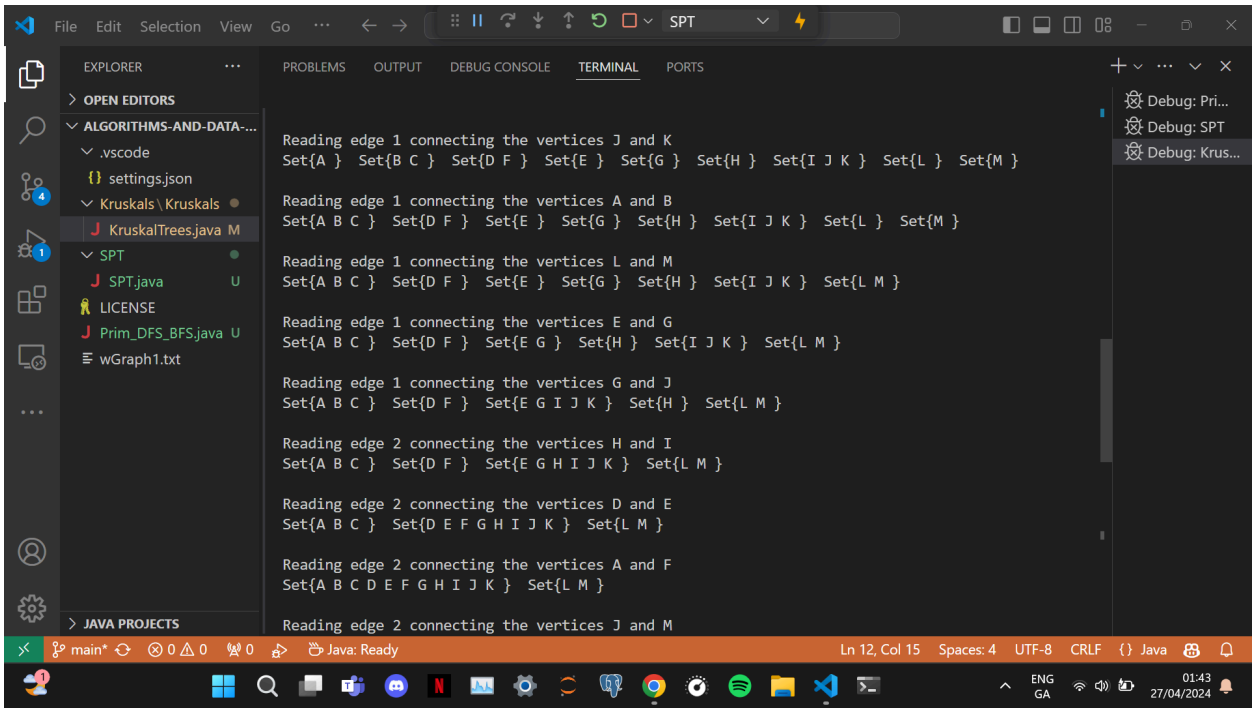

File Edit Selection View Go
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Debug: Pri...
Debug: SPT
Adjacency Matrix:
1 2 3 4 5 6 7 8 9 10 11 12 13
1 0 1 0 0 0 2 6 0 0 0 0 0 0
2 1 0 1 2 4 0 0 0 0 0 0 0 0
3 0 1 0 0 4 0 0 0 0 0 0 0 0
4 0 2 0 0 2 1 0 0 0 0 0 0 0
5 0 4 4 2 0 2 1 0 0 0 0 4 0
6 2 0 0 1 2 0 0 0 0 0 0 2 0
7 6 0 0 0 1 0 0 3 0 1 0 5 0
8 0 0 0 0 0 0 3 0 2 0 0 0 0
9 0 0 0 0 0 0 0 2 0 0 1 0 0
10 0 0 0 0 0 0 1 0 0 0 1 3 2
11 0 0 0 0 0 0 0 0 1 1 0 0 0
12 0 0 0 0 4 2 5 0 0 3 0 0 1
13 0 0 0 0 0 0 0 0 0 2 0 1 0
Inserted 12 at position 1
Inserted 5 at position 1
Inserted 6 at position 1
Inserted 7 at position 3
Inserted 10 at position 2
Inserted 13 at position 1
Inserted 1 at position 4
Inserted 4 at position 2
Inserted 7 at position 5
Inserted 11 at position 6
main* 0 0 0 Java: Ready Ln 10, Col 1 Spaces: 4 UTF-8 CRLF Java
ENG GA 01:41 27/04/2024

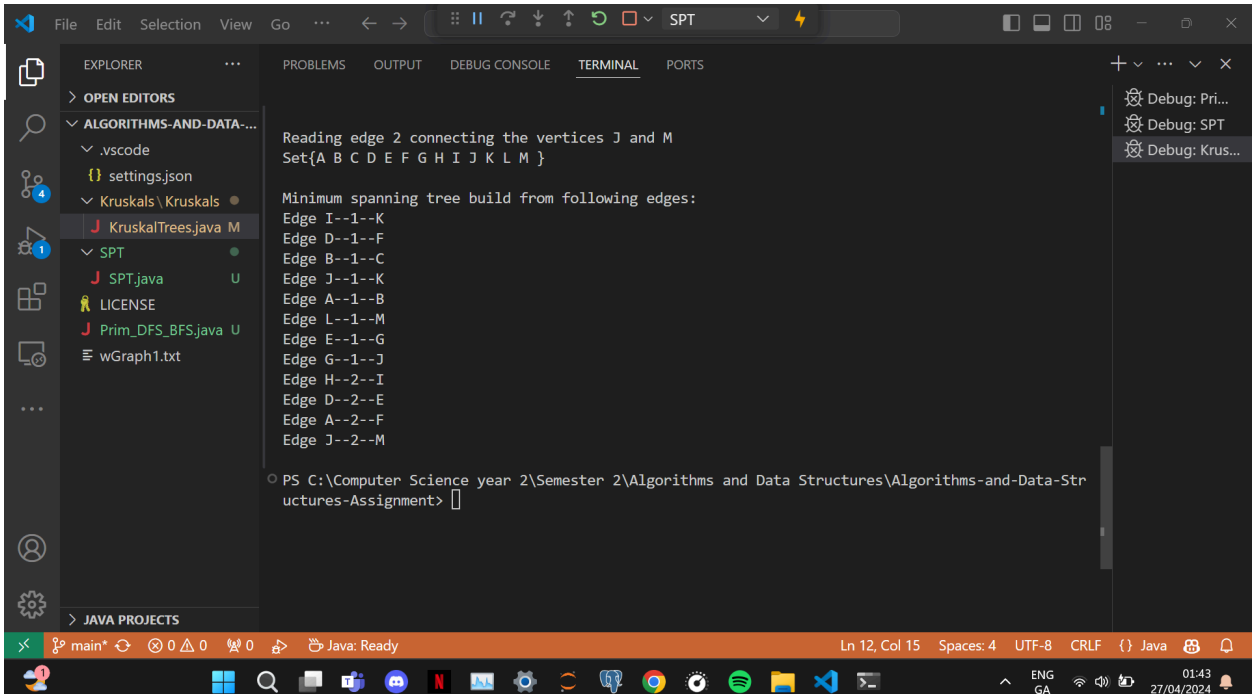


KruskalTrees.java




File Edit Selection View Go
SPT
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
LICENSE
Prim_DFS_BFS.java U
wGraph1.txt
JAVA PROJECTS
Debug: Pri...
Debug: SPT
Debug: Krus...
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
Kruskals Edges, Show Sets and Show Trees
Set{A } Set{B } Set{C } Set{D } Set{E } Set{F } Set{G } Set{H } Set{I } Set{J } Set{K } Set{L } Set{M }
A->A B->B C->C D->D E->E F->F G->G H->H I->I J->J K->K L->L M->M
Reading edge 1 connecting the vertices I and K
Set{A } Set{B } Set{C } Set{D } Set{E } Set{F } Set{G } Set{H } Set{I K } Set{J } Set{L } Set{M }
Reading edge 1 connecting the vertices D and F
Set{A } Set{B } Set{C } Set{D F } Set{E } Set{G } Set{H } Set{I K } Set{J } Set{L } Set{M }
Reading edge 1 connecting the vertices B and C
Set{A } Set{B C } Set{D F } Set{E } Set{G } Set{H } Set{I K } Set{J } Set{L } Set{M }
Reading edge 1 connecting the vertices J and K
main*
Java: Ready
Ln 12, Col 15 Spaces: 4 UTF-8 CRLF Java
ENG GA
01:43 27/04/2024


File Edit Selection View Go
SPT
EXPLORER
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
LICENSE
Prim_DFS_BFS.java U
wGraph1.txt
JAVA PROJECTS
Debug: Pri...
Debug: SPT
Debug: Krus...
Reading edge 1 connecting the vertices J and K
Set{A } Set{B C } Set{D F } Set{E } Set{G } Set{H } Set{I J K } Set{L } Set{M }
Reading edge 1 connecting the vertices A and B
Set{A B C } Set{D F } Set{E } Set{G } Set{H } Set{I J K } Set{L } Set{M }
Reading edge 1 connecting the vertices L and M
Set{A B C } Set{D F } Set{E } Set{G } Set{H } Set{I J K } Set{L M }
Reading edge 1 connecting the vertices E and G
Set{A B C } Set{D F } Set{E G } Set{H } Set{I J K } Set{L M }
Reading edge 1 connecting the vertices G and J
Set{A B C } Set{D F } Set{E G I J K } Set{H } Set{L M }
Reading edge 2 connecting the vertices H and I
Set{A B C } Set{D F } Set{E G H I J K } Set{L M }
Reading edge 2 connecting the vertices D and E
Set{A B C } Set{D E F G H I J K } Set{L M }
Reading edge 2 connecting the vertices A and F
Set{A B C D E F G H I J K } Set{L M }
Reading edge 2 connecting the vertices J and M
main*
Java: Ready
Ln 12, Col 15 Spaces: 4 UTF-8 CRLF Java
ENG GA
01:43 27/04/2024


File Edit Selection View Go
SPT
EXPLORER
OPEN EDITORS
ALGORITHMS-AND-DATA-...
.vscode
settings.json
Kruskals \ Kruskals
KruskalTrees.java M
SPT
SPT.java U
LICENSE
Prim_DFS_BFS.java U
wGraph1.txt
JAVA PROJECTS
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Reading edge 2 connecting the vertices J and M
Set{A B C D E F G H I J K L M }
Minimum spanning tree build from following edges:
Edge I--1--K
Edge D--1--F
Edge B--1--C
Edge J--1--K
Edge A--1--B
Edge L--1--M
Edge E--1--G
Edge G--1--J
Edge H--2--I
Edge D--2--E
Edge A--2--F
Edge J--2--M
PS C:\Computer Science year 2\Semester 2\Algorithms and Data Structures\Algorithms-and-Data-Structures-Assignment>
Debug: Pri...
Debug: SPT
Debug: Krus...
main*
Java: Ready
Ln 12, Col 15
Spaces: 4
UTF-8
CRLF
Java
ENG
GA
01:43
27/04/2024

9. Discussion/analysis/reflection

GraphLists.java

The output demonstrates the execution of various graph algorithms on a graph defined in the "wGraph1.txt" file. Prim's MST algorithm efficiently constructs a minimum spanning tree, showcasing the process of selecting edges and forming the tree incrementally. The total weight of the generated MST is reported as 16. Following this, both depth-first and breadth-first traversals are performed starting from vertex 12. These traversals provide insights into the graph's connectivity and structure, revealing the order in which vertices are visited from the starting vertex. Overall, this analysis demonstrates the effectiveness of the implemented algorithms in solving fundamental graph problems and understanding the properties of the given graph. Additionally, it highlights the importance of graph algorithms in various applications, including network optimization, pathfinding, and data analysis.

1. Prim's MST Algorithm: O((V + E)logV) with adjacency list.
2. Depth-First Search (DFS): O(V + E) with adjacency list.
3. Breadth-First Search (BFS): O(V + E) with adjacency list.

SPT.java

The provided output demonstrates the execution of Dijkstra's algorithm to find the shortest paths from vertex L to all other vertices in the graph defined in the file "wGraph1.txt". The algorithm utilizes a min-priority queue implemented using a heap to efficiently explore the graph's edges. The adjacency list and matrix representations of the graph are displayed before the algorithm runs, showcasing the graph's structure. After running Dijkstra's algorithm, the shortest paths from vertex L to all other vertices are printed, revealing the shortest path lengths.

This output provides valuable insights into the efficiency and effectiveness of Dijkstra's algorithm in finding shortest paths in a weighted graph. It highlights the importance of data structures like heaps and adjacency lists in optimizing graph traversal algorithms. Additionally, it showcases the practical application of such algorithms in solving real-world problems like finding the shortest route in transportation networks or optimizing network routing protocols.

The time complexity of Dijkstra's algorithm is O((V + E) log V), where V is the number of vertices and E is the number of edges in the graph.

KruskalTrees.java

The output showcases the execution of Kruskal's Minimum Spanning Tree Algorithm on a graph defined in the file "wGraph1.txt". Initially, it reads the graph's edges from the file and displays them. Then, it demonstrates the progression of Kruskal's algorithm, showing the sets and trees formed during the process and detailing the edges selected for the minimum spanning tree. Finally, it presents the minimum spanning tree constructed from the selected edges. Kruskal's algorithm efficiently identifies the minimum

spanning tree by iteratively selecting the smallest edge that doesn't form a cycle until all vertices are included, ensuring that the resulting tree spans all vertices with the minimum total weight.

The time complexity of Kruskal's algorithm is O(E log E), where E is the number of edges in the graph.