Algorithms CA Graph

C22483302 – Sergei Larionov

1.

In this assignment, we explore three different algorithms for finding the (MST) (SPT) in a graph: Prim's algorithm, Dijkstra's algorithm, and Kruskal's algorithm (including BFS and DFS). These algorithms are fundamental in graph theory and have various real-world applications, such as network design, routing protocols, and optimization. We will implement and analyze each algorithm step by step, providing visual representations of the graphs and their corresponding trees or paths.

2.

```
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 1: Selected vertex L (Distance: 0)

Heap contents: []

Parent array: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

Distance array: [0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF]

Step 2: Selected vertex M (Distance: 1)

Heap contents: [6, 10, 7, 5]

Parent array: [0, 0, 0, 0, 0, 12, 12, 12, 0, 0, 12, 0, 0, 12]

Distance array: [0, INF, INF, INF, INF, 4, 2, 5, INF, INF, 3, INF, 0, 1]

Step 3: Selected vertex F (Distance: 2)

Heap contents: [10, 10, 7, 5]

Parent array: [0, 0, 0, 0, 0, 12, 12, 12, 0, 0, 13, 0, 0, 12]

Distance array: [0, INF, INF, INF, INF, 4, 2, 5, INF, INF, 2, INF, 0, 1]

Step 4: Selected vertex D (Distance: 1)

Heap contents: [1, 10, 10, 5, 5, 7]

Parent array: [0, 6, 0, 0, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]

Distance array: [0, 2, INF, INF, 1, 2, 2, 5, INF, INF, 2, INF, 0, 1]

Step 5: Selected vertex A (Distance: 2)

Heap contents: [2, 10, 10, 5, 5, 7]

Parent array: [0, 6, 4, 0, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]

Distance array: [0, 2, 2, INF, 1, 2, 2, 5, INF, INF, 2, INF, 0, 1]

Step 6: Selected vertex B (Distance: 1)

Heap contents: [2, 10, 10, 5, 5, 7]

Parent array: [0, 6, 1, 0, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]

Distance array: [0, 2, 1, INF, 1, 2, 2, 5, INF, INF, 2, INF, 0, 1]

Step 7: Selected vertex C (Distance: 1)

Heap contents: [10, 5, 10, 7, 5]

Parent array: [0, 6, 1, 2, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]

Distance array: [0, 2, 1, 1, 1, 2, 2, 5, INF, INF, 2, INF, 0, 1]

Step 8: Selected vertex J (Distance: 2)

Heap contents: [5, 5, 10, 7]

Parent array: [0, 6, 1, 2, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]

Distance array: [0, 2, 1, 1, 1, 2, 2, 5, INF, INF, 2, INF, 0, 1]

Step 9: Selected vertex K (Distance: 1)

Heap contents: [7, 5, 10, 7, 5]

Parent array: [0, 6, 1, 2, 6, 6, 12, 12, 0, 0, 13, 10, 0, 12]

Distance array: [0, 2, 1, 1, 1, 2, 2, 1, INF, INF, 2, 1, 0, 1]

Step 10: Selected vertex G (Distance: 1)

Heap contents: [9, 5, 10, 7, 5]

Parent array: [0, 6, 1, 2, 6, 6, 12, 12, 0, 11, 13, 10, 0, 12]

Distance array: [0, 2, 1, 1, 1, 2, 2, 1, INF, 1, 2, 1, 0, 1]

Step 11: Selected vertex I (Distance: 1)

Heap contents: [5, 7, 5, 10, 5, 8]

Parent array: [0, 6, 1, 2, 6, 7, 12, 12, 7, 11, 13, 10, 0, 12]

Distance array: [0, 2, 1, 1, 1, 1, 2, 1, 3, 1, 2, 1, 0, 1]

Step 12: Selected vertex E (Distance: 1)

Heap contents: [7, 5, 5, 10, 8, 8]

Parent array: [0, 6, 1, 2, 6, 7, 12, 12, 9, 11, 13, 10, 0, 12]

Distance array: [0, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 0, 1]

Step 13: Selected vertex H (Distance: 2)

Heap contents: [8]

Parent array: [0, 6, 1, 2, 6, 7, 12, 12, 9, 11, 13, 10, 0, 12]

Distance array: [0, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 0, 1]

4.

Step 1: Selected vertex L (Distance: 0)

Heap contents: []

Parent array: [0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]

Distance array: [0, INF, INF, INF, INF, INF, INF, INF, INF, INF, INF, 0, INF]

Step 2: Selected vertex M (Distance: 1)

Heap contents: [6, 10, 7, 5]

Parent array: [0, -1, -1, -1, -1, 12, 12, 12, -1, -1, 12, -1, -1, 12]

Distance array: [0, INF, INF, INF, INF, 4, 2, 5, INF, INF, 3, INF, 0, 1]

Step 3: Selected vertex F (Distance: 2)

Heap contents: [10, 5, 7]

Parent array: [0, -1, -1, -1, -1, 12, 12, 12, -1, -1, 12, -1, -1, 12]

Distance array: [0, INF, INF, INF, INF, 4, 2, 5, INF, INF, 2, INF, 0, 1]

Step 4: Selected vertex J (Distance: 3)

Heap contents: [4, 1, 7, 5]

Parent array: [0, 6, -1, -1, 6, 12, 12, 12, -1, -1, 12, -1, -1, 12]

Distance array: [0, 4, INF, INF, 3, 4, 2, 5, INF, INF, 3, INF, 0, 1]

Step 5: Selected vertex D (Distance: 3)

Heap contents: [7, 1, 7, 5, 11]

Parent array: [0, 6, -1, -1, 6, 12, 12, 10, -1, -1, 12, 10, -1, 12]

Distance array: [0, 4, INF, INF, 3, 4, 2, 4, INF, INF, 3, 4, 0, 1]

Step 6: Selected vertex G (Distance: 4)

Heap contents: [1, 5, 7, 2, 11]

Parent array: [0, 6, 4, -1, 6, 12, 12, 10, -1, -1, 12, 10, -1, 12]

Distance array: [0, 4, 5, INF, 3, 4, 2, 4, INF, INF, 3, 4, 0, 1]

Step 7: Selected vertex A (Distance: 4)

Heap contents: [5, 11, 7, 2, 8]

Parent array: [0, 6, 4, -1, 6, 12, 12, 10, 7, -1, 12, 10, -1, 12]

Distance array: [0, 4, 5, INF, 3, 4, 2, 4, 7, INF, 3, 4, 0, 1]

Step 8: Selected vertex E (Distance: 4)

Heap contents: [11, 2, 7, 8]

Parent array: [0, 6, 4, -1, 6, 12, 12, 10, 7, -1, 12, 10, -1, 12]

Distance array: [0, 4, 5, INF, 3, 4, 2, 4, 7, INF, 3, 4, 0, 1]

Step 9: Selected vertex K (Distance: 4)

Heap contents: [7, 2, 3, 8]

Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, -1, 12, 10, -1, 12]

Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, INF, 3, 4, 0, 1]

Step 10: Selected vertex G (Distance: 4)

Heap contents: [9, 2, 3, 8]

Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]

Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]

Step 11: Selected vertex I (Distance: 5)

Heap contents: [2, 8, 3]

Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]

Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]

Step 12: Selected vertex B (Distance: 5)

Heap contents: [8, 3]

Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]

Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]

Step 13: Selected vertex C (Distance: 6)

Heap contents: [3, 8]

Parent array: [0, 6, 4, 2, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]

Distance array: [0, 4, 5, 6, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]

Step 14: Selected vertex C (Distance: 6)

Heap contents: [8]

Parent array: [0, 6, 4, 2, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]

Distance array: [0, 4, 5, 6, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]

Step 15: Selected vertex H (Distance: 7)

Heap contents: []

Parent array: [0, 6, 4, 2, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]

Distance array: [0, 4, 5, 6, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]

5.

a. Adding edge 1-2:

MST: (1-2)

Union-Find: {1}, {2}

b. Adding edge 2-3:

MST: (1-2, 2-3)

Union-Find: {1}, {2, 3}

c. Adding edge 4-6:

MST: (1-2, 2-3, 4-6)

Union-Find: {1}, {2, 3}, {4, 6}

d. Adding edge 5-7:

MST: (1-2, 2-3, 4-6, 5-7)

Union-Find: {1}, {2, 3, 5, 7}, {4, 6}

e. Adding edge 7-10:

MST: (1-2, 2-3, 4-6, 5-7, 7-10)

Union-Find: {1}, {2, 3, 5, 7, 10}, {4, 6}

f. Adding edge 9-11:

MST: (1-2, 2-3, 4-6, 5-7, 7-10, 9-11)

Union-Find: {1}, {2, 3, 5, 7, 9, 10, 11}, {4, 6}

g. Adding edge 10-11:

MST: (1-2, 2-3, 4-6, 5-7, 7-10, 9-11, 10-11)

Union-Find: {1}, {2, 3, 5, 7, 9, 10, 11}, {4, 6}

h. Adding edge 12-13:

MST: (1-2, 2-3, 4-6, 5-7, 7-10, 9-11, 10-11, 12-13)

Union-Find: {1}, {2, 3, 5, 7, 9, 10, 11, 12, 13}, {4, 6}

i. Adding edge 1-6:

MST: (1-2, 2-3, 4-6, 5-7, 7-10, 9-11, 10-11, 12-13)

Union-Find: {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13}

j. Adding edge 4-5:

MST: (1-2, 2-3, 4-5, 4-6, 5-7, 7-10, 9-11, 10-11, 12-13)

Union-Find: {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13}

k. Adding edge 6-12:

MST: (1-2, 2-3, 4-5, 4-6, 5-7, 6-12, 7-10, 9-11, 10-11, 12-13)

Union-Find: {1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13}

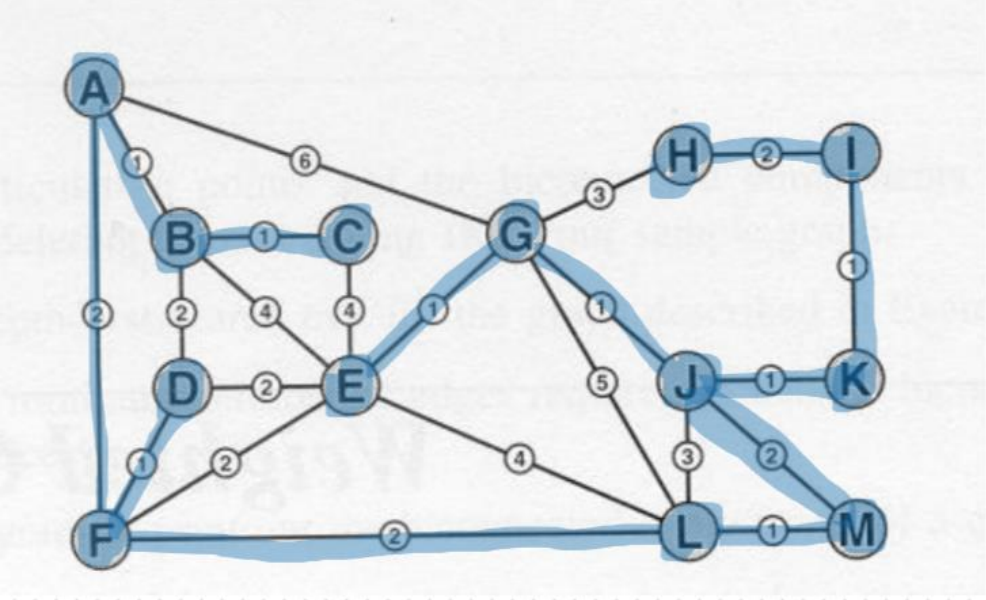
l. Adding edge 8-9:

MST: (1-2, 2-3, 4-5, 4-6, 5-7, 6-12, 7-10, 8-9, 9-11, 10-11, 12-13)

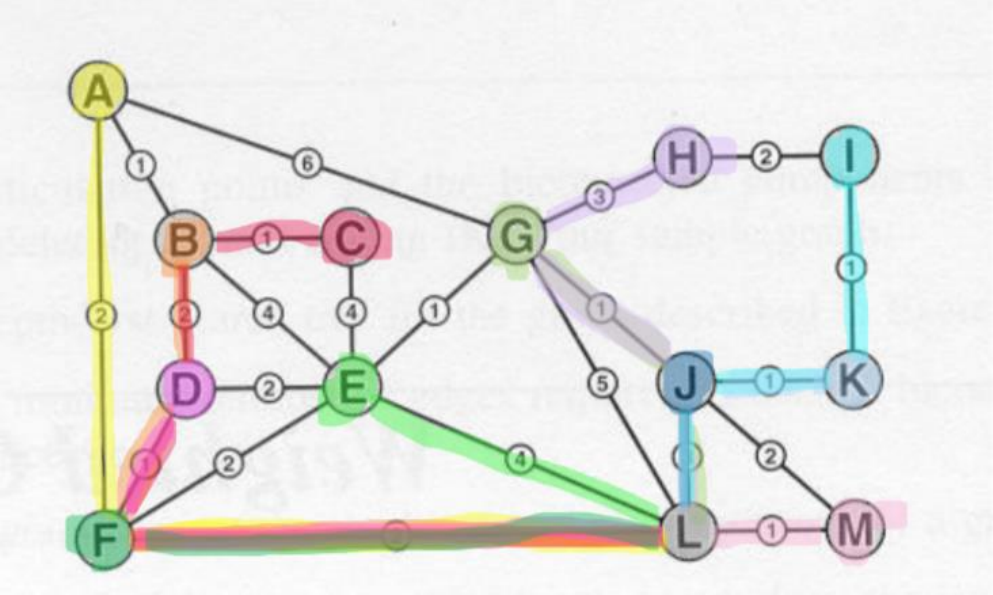
Union-Find: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}

Total weight of MST: 16

6.



7.



8.

C22483302GraphLists.java:

```
Enter the file name: wGraph1.txt
Enter the starting vertex 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--(1)--G
Edge E--(4)--L
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] -> |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| ->

1. Depth First
2. Breadth First
3. MST Prim
4. SPT Dijkstra
5. Exit
```

```
1

Depth First Search starting from vertex L:
L M J K I H G E F D B C A
```

```
2

Breadth First Search starting from vertex L:
L M J G F E K H A D C B I
```

```
3
Step 1: Selected vertex L (Distance: 0)
Heap contents: []
Parent array: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Distance array: [0, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 214748364
7, 0, 2147483647]
Step 2: Selected vertex M (Distance: 1)
Heap contents: [6, 10, 7, 5]
Parent array: [0, 0, 0, 0, 0, 12, 12, 12, 0, 0, 12, 0, 0, 12]
Distance array: [0, 2147483647, 2147483647, 2147483647, 2147483647, 4, 2, 5, 2147483647, 2147483647, 3, 2147483647, 0, 1]
Step 3: Selected vertex F (Distance: 2)
Heap contents: [10, 10, 7, 5]
Parent array: [0, 0, 0, 0, 0, 12, 12, 12, 0, 0, 13, 0, 0, 12]
Distance array: [0, 2147483647, 2147483647, 2147483647, 2147483647, 4, 2, 5, 2147483647, 2147483647, 2, 2147483647, 0, 1]
Step 4: Selected vertex D (Distance: 1)
Heap contents: [1, 10, 10, 5, 5, 7]
Parent array: [0, 6, 0, 0, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]
Distance array: [0, 2, 2147483647, 2147483647, 1, 2, 2, 5, 2147483647, 2147483647, 2, 2147483647, 0, 1]
Step 5: Selected vertex A (Distance: 2)
Heap contents: [2, 10, 10, 5, 5, 7]
Parent array: [0, 6, 4, 0, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]
Distance array: [0, 2, 2, 2147483647, 1, 2, 2, 5, 2147483647, 2147483647, 2, 2147483647, 0, 1]
Step 6: Selected vertex B (Distance: 1)
Heap contents: [2, 10, 10, 5, 5, 7]
Parent array: [0, 6, 1, 0, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]
Distance array: [0, 2, 1, 2147483647, 1, 2, 2, 5, 2147483647, 2147483647, 2, 2147483647, 0, 1]
Step 7: Selected vertex C (Distance: 1)
Heap contents: [10, 5, 10, 7, 5]
Parent array: [0, 6, 1, 2, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]
Distance array: [0, 2, 1, 1, 1, 2, 2, 5, 2147483647, 2147483647, 2, 2147483647, 0, 1]
Step 8: Selected vertex J (Distance: 2)
Heap contents: [5, 5, 10, 7]
Parent array: [0, 6, 1, 2, 6, 6, 12, 12, 0, 0, 13, 0, 0, 12]
Distance array: [0, 2, 1, 1, 1, 2, 2, 5, 2147483647, 2147483647, 2, 2147483647, 0, 1]
Step 9: Selected vertex K (Distance: 1)
Heap contents: [7, 5, 10, 7, 5]
Parent array: [0, 6, 1, 2, 6, 6, 12, 10, 0, 0, 13, 10, 0, 12]
Distance array: [0, 2, 1, 1, 1, 2, 2, 1, 2147483647, 2147483647, 2, 1, 0, 1]
```

```
Step 10: Selected vertex G (Distance: 1)
Heap contents: [9, 5, 10, 7, 5]
Parent array: [0, 6, 1, 2, 6, 6, 12, 10, 0, 11, 13, 10, 0, 12]
Distance array: [0, 2, 1, 1, 1, 2, 2, 1, 2147483647, 1, 2, 1, 0, 1]
Step 11: Selected vertex I (Distance: 1)
Heap contents: [5, 7, 5, 10, 5, 8]
Parent array: [0, 6, 1, 2, 6, 7, 12, 10, 7, 11, 13, 10, 0, 12]
Distance array: [0, 2, 1, 1, 1, 1, 2, 1, 3, 1, 2, 1, 0, 1]
Step 12: Selected vertex E (Distance: 1)
Heap contents: [7, 5, 5, 10, 8, 8]
Parent array: [0, 6, 1, 2, 6, 7, 12, 10, 9, 11, 13, 10, 0, 12]
Distance array: [0, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 0, 1]
Step 13: Selected vertex H (Distance: 2)
Heap contents: [8]
Parent array: [0, 6, 1, 2, 6, 7, 12, 10, 9, 11, 13, 10, 0, 12]
Distance array: [0, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 1, 0, 1]
```

```
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
```

```
4
Steps:
Selected vertex L (Distance: 0)
Heap contents: []
Parent array: [0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]
Distance array: [0, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 2147483647, 214748364
7, 0, 2147483647]
  Updating distance to vertex M from 2147483647 to 1
  Updating distance to vertex J from 2147483647 to 3
  Updating distance to vertex G from 2147483647 to 5
  Updating distance to vertex F from 2147483647 to 2
  Updating distance to vertex E from 2147483647 to 4
Selected vertex M (Distance: 1)
Heap contents: [6, 10, 7, 5]
Parent array: [0, -1, -1, -1, -1, 12, 12, 12, -1, -1, 12, -1, -1, 12]
Distance array: [0, 2147483647, 2147483647, 2147483647, 2147483647, 4, 2, 5, 2147483647, 2147483647, 3, 2147483647, 0, 1]
Selected vertex F (Distance: 2)
Heap contents: [10, 5, 7]
Parent array: [0, -1, -1, -1, -1, 12, 12, 12, -1, -1, 12, -1, -1, 12]
Distance array: [0, 2147483647, 2147483647, 2147483647, 2147483647, 4, 2, 5, 2147483647, 2147483647, 3, 2147483647, 0, 1]
  Updating distance to vertex D from 2147483647 to 3
  Updating distance to vertex A from 2147483647 to 4
Selected vertex J (Distance: 3)
Heap contents: [4, 1, 7, 5]
Parent array: [0, 6, -1, -1, 6, 12, 12, 12, -1, -1, 12, -1, -1, 12]
Distance array: [0, 4, 2147483647, 2147483647, 3, 4, 2, 5, 2147483647, 2147483647, 3, 2147483647, 0, 1]
  Updating distance to vertex K from 2147483647 to 4
  Updating distance to vertex G from 5 to 4
Selected vertex D (Distance: 3)
Heap contents: [7, 1, 7, 5, 11]
Parent array: [0, 6, -1, -1, 6, 12, 12, 10, -1, -1, 12, 10, -1, 12]
Distance array: [0, 4, 2147483647, 2147483647, 3, 4, 2, 4, 2147483647, 2147483647, 3, 4, 0, 1]
  Updating distance to vertex B from 2147483647 to 5
Selected vertex G (Distance: 4)
Heap contents: [1, 5, 7, 2, 11]
Parent array: [0, 6, 4, -1, 6, 12, 12, 10, -1, -1, 12, 10, -1, 12]
Distance array: [0, 4, 5, 2147483647, 3, 4, 2, 4, 2147483647, 2147483647, 3, 4, 0, 1]
  Updating distance to vertex H from 2147483647 to 7
Selected vertex A (Distance: 4)
```

```
Heap contents: [5, 11, 7, 2, 8]
Parent array: [0, 6, 4, -1, 6, 12, 12, 10, 7, -1, 12, 10, -1, 12]
Distance array: [0, 4, 5, 2147483647, 3, 4, 2, 4, 7, 2147483647, 3, 4, 0, 1]
Selected vertex E (Distance: 4)
Heap contents: [11, 2, 7, 8]
Parent array: [0, 6, 4, -1, 6, 12, 12, 10, 7, -1, 12, 10, -1, 12]
Distance array: [0, 4, 5, 2147483647, 3, 4, 2, 4, 7, 2147483647, 3, 4, 0, 1]
  Updating distance to vertex C from 2147483647 to 8
Selected vertex K (Distance: 4)
Heap contents: [7, 2, 3, 8]
Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, -1, 12, 10, -1, 12]
Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, 2147483647, 3, 4, 0, 1]
  Updating distance to vertex I from 2147483647 to 5
Selected vertex G (Distance: 4)
Heap contents: [9, 2, 3, 8]
Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]
Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]
Selected vertex I (Distance: 5)
Heap contents: [2, 8, 3]
Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]
Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]
Selected vertex B (Distance: 5)
Heap contents: [8, 3]
Parent array: [0, 6, 4, 5, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]
Distance array: [0, 4, 5, 8, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]
  Updating distance to vertex C from 8 to 6
Selected vertex C (Distance: 6)
Heap contents: [3, 8]
Parent array: [0, 6, 4, 2, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]
Distance array: [0, 4, 5, 6, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]
Selected vertex C (Distance: 6)
Heap contents: [8]
Parent array: [0, 6, 4, 2, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]
Distance array: [0, 4, 5, 6, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]
Selected vertex H (Distance: 7)
Heap contents: []
Parent array: [0, 6, 4, 2, 6, 12, 12, 10, 7, 11, 12, 10, -1, 12]
Distance array: [0, 4, 5, 6, 3, 4, 2, 4, 7, 5, 3, 4, 0, 1]
```

```
Shortest Paths from vertex L using Dijkstra's Algorithm:
L -> A: 4  Path: L -> F -> A
L -> B: 5  Path: L -> F -> D -> B
L -> C: 6  Path: L -> F -> D -> B -> C
L -> D: 3  Path: L -> F -> D
L -> E: 4  Path: L -> E
L -> F: 2  Path: L -> F
L -> G: 4  Path: L -> J -> G
L -> H: 7  Path: L -> J -> G -> H
L -> I: 5  Path: L -> J -> K -> I
L -> J: 3  Path: L -> J
L -> K: 4  Path: L -> J -> K
L -> L: 0  Path: L
L -> M: 1  Path: L -> M
```

```
5
Exiting...
PS C:\Users\serge\OneDrive\Documents\Year2\Semester2\AlgorithmsAndData\AlgorithmsCA>
```

Kruskal.java:

```
Enter the file name: wGraph1.txt

Edges of MST using Kruskal's Algorithm:
A - B : 1
B - C : 1
D - F : 1
E - G : 1
G - J : 1
I - K : 1
J - K : 1
L - M : 1
A - F : 2
D - E : 2
F - L : 2
H - I : 2
Total weight of MST: 16
```

9.

Through this assignment, I gained a deeper understanding of graph theory and its practical applications. Implementing algorithms like Prim's, Dijkstra's, and Kruskal's allowed me to grasp the intricacies of graph traversal, shortest path determination, and MST construction. Visualizing the step-by-step execution of these algorithms helped me appreciate their efficiency and correctness. Analyzing the output and comparing the results provided insights into algorithmic complexity and performance. Overall, this assignment enhanced my problem-solving skills and underscored the importance of algorithmic techniques in solving real-world problems efficiently.