Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

Graph Traversal, MST & SPT Algorithm Assignment

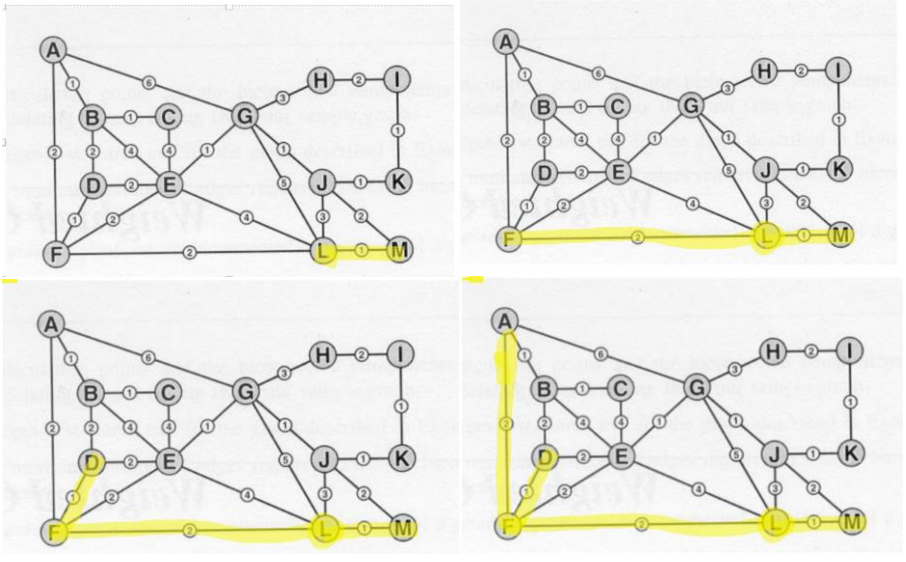
Introduction

In this assignment I have created two files: GraphLists.java and KruskalTrees.java. Both programs when executed at command line they prompt the user for the name of the file that should contain a graph, and also it prompts the user for a starting vertex. The graph is read by the program and the vertices/edges are added into a list to then apply the different algorithms. In GraphLists.java the program computes the graphs' SPT and MST and they are outputted on the terminal. It also outputs some of the workings so we can see it working step by step.
For Depth-first traversal I used recursion, and for Breadth-first traversal I used a queue; using the Cormens algorithm to implement it on the code. For Prim and Dijkstra, I represent the graph using an adjacency list data structure.
The KruskalTree.java uses an array of edges. Instead of using a queue I used a priority heap, for efficient edge selection, I also use siftdown() to maintain the heap property. To initialize the vertices as a separate set I use the unionfindsets class.

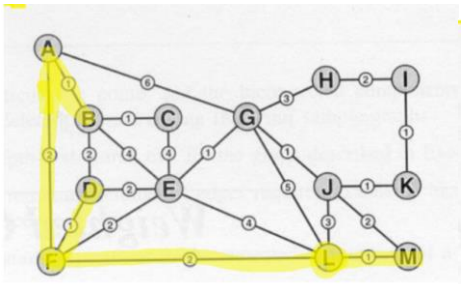
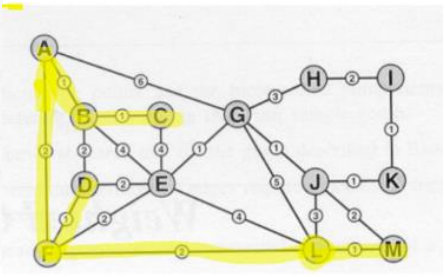
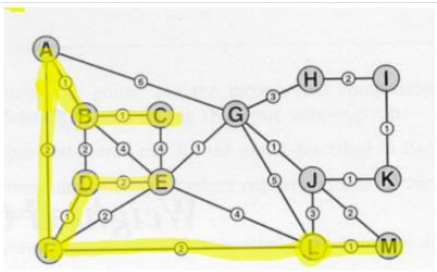
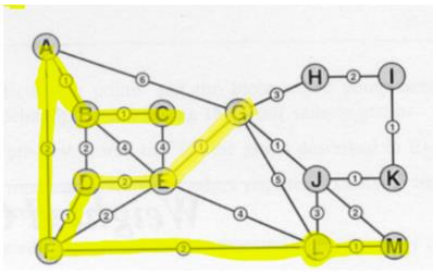
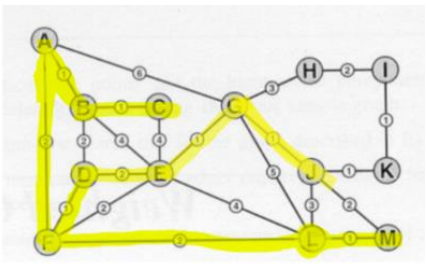
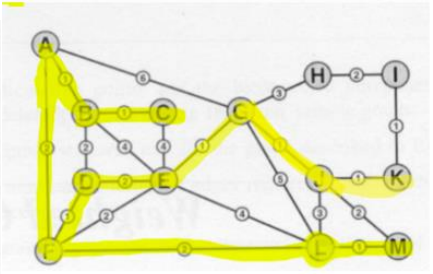
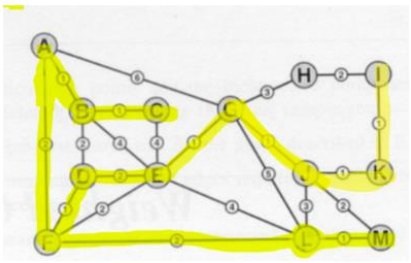
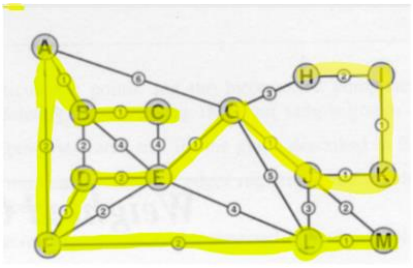
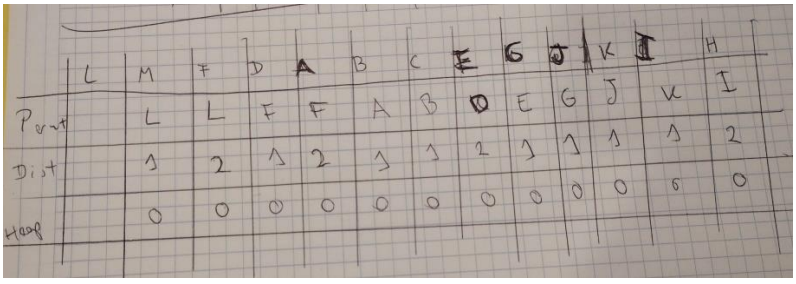
Adjacency list diagram of wGraph1.txt:

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

MST using prim:

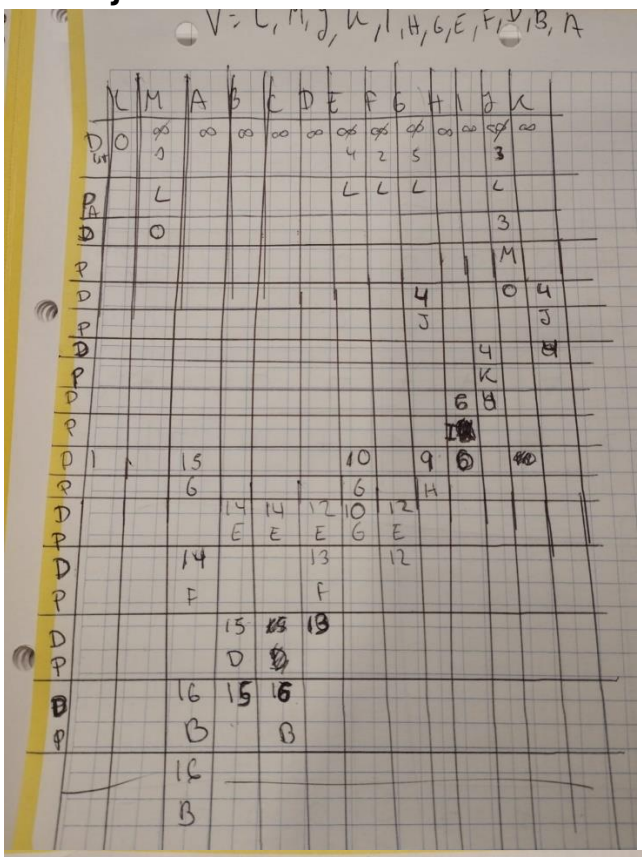


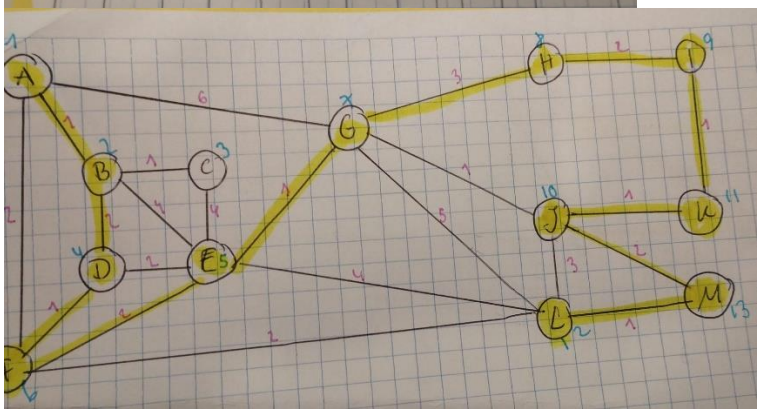
Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

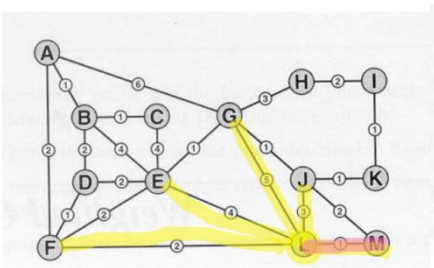










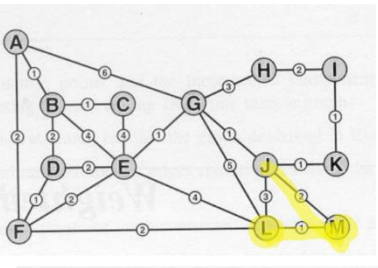
Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

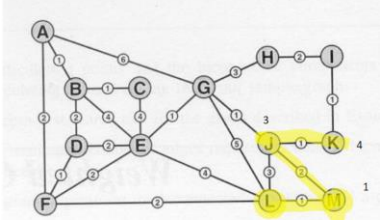
SPT Dijkstra:

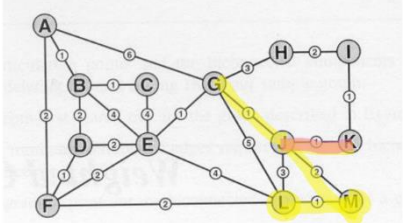




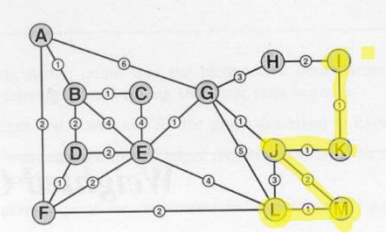
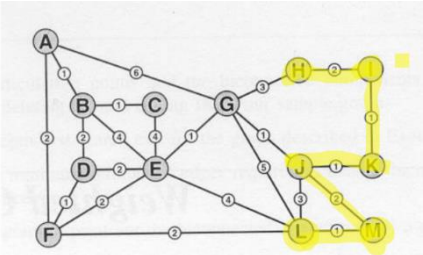
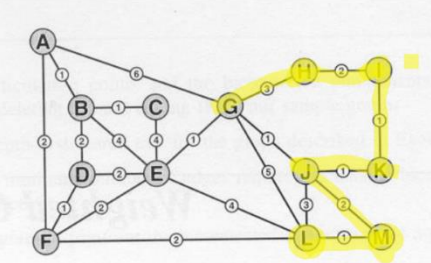
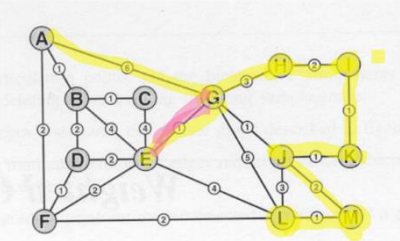
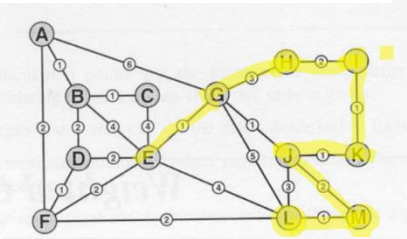
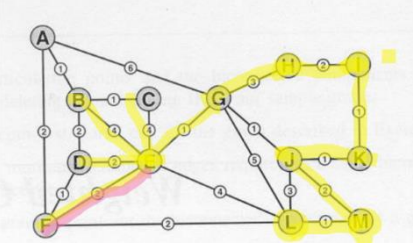
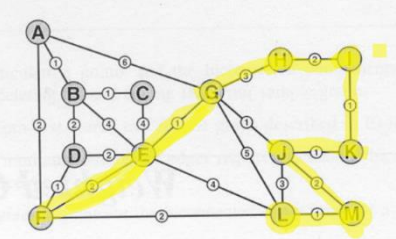
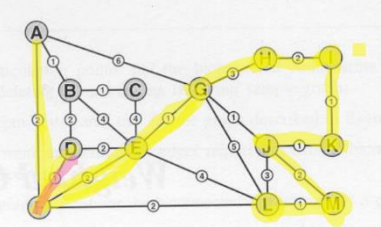
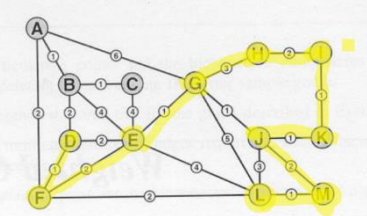
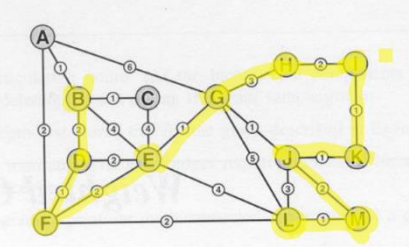
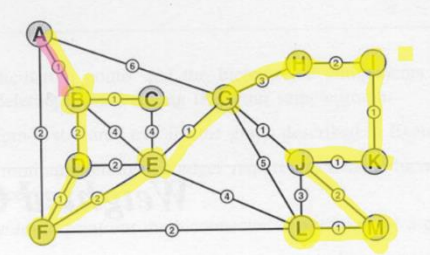
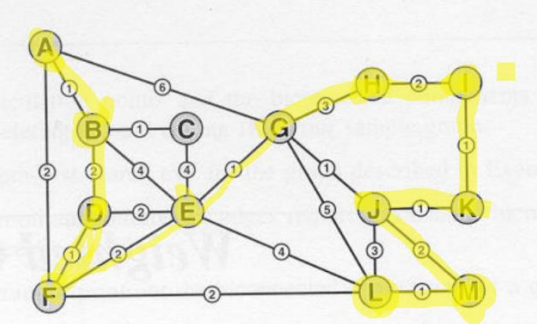






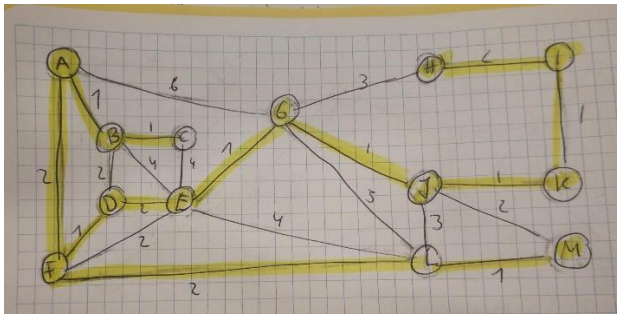


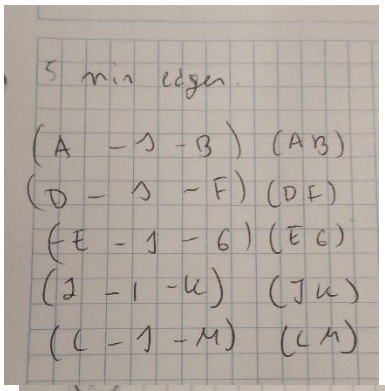
Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

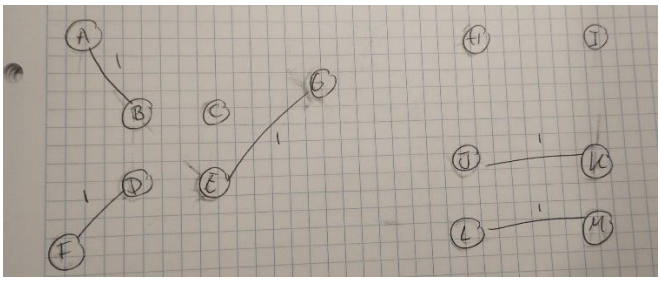













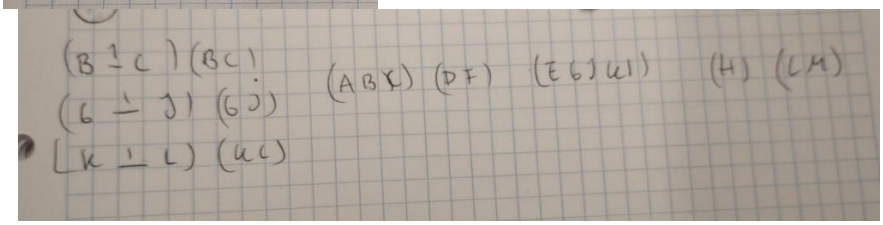
Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

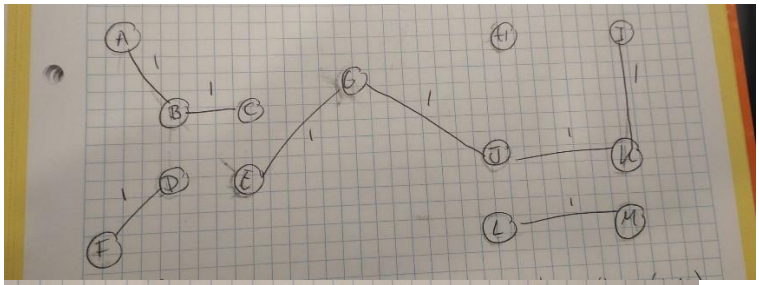
Kruskal:

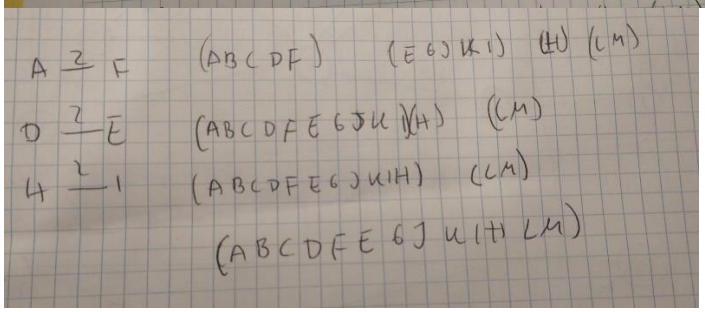












Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

Screen captures of output:

Graphlists.java:

```
Enter the name of the text file containing the graph: wGraph1.txt
Enter the starting vertex: 11
Parts[] = 13 22
Reading edges from text file
Edge A--(2)--F
Edge A--(1)--B
Edge B--(2)--D
Edge D--(1)--F
Edge B--(1)--C
Edge B--(4)--E
Edge D--(2)--E
Edge C--(4)--E
Edge E--(2)--F
Edge A--(6)--G
Edge E--(1)--G
Edge E--(4)--L
Edge F--(2)--L
Edge G--(5)--L
Edge G--(1)--J
Edge G--(3)--H
Edge H--(2)--I
Edge J--(1)--K
Edge J--(3)--L
Edge J--(2)--M
Edge L--(1)--M
Edge I--(1)--K
```

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

Depth-First Traversal:
K I H G J M L F E C B D A

Breadth-First Traversal:
K I J H M L G F E A D C B
```

```
Prim's Algorithm for MST:
Adding vertex K to MST
Adding vertex I to MST
Adding vertex J to MST
Adding vertex G to MST
Adding vertex E to MST
Adding vertex M to MST
Adding vertex L to MST
Adding vertex D to MST
Adding vertex F to MST
Adding vertex B to MST
Adding vertex C to MST
Adding vertex A to MST
Adding vertex H to MST


Weight of MST = 16
```

Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

```
Weight of MST = 16


Minimum Spanning tree parent array is:
A -> B, Distance: 1, Heap position: 0
B -> D, Distance: 2, Heap position: 0
C -> B, Distance: 1, Heap position: 0
D -> E, Distance: 2, Heap position: 0
E -> G, Distance: 1, Heap position: 0
F -> D, Distance: 1, Heap position: 0
G -> J, Distance: 1, Heap position: 0
H -> I, Distance: 2, Heap position: 1
I -> K, Distance: 1, Heap position: 0
J -> K, Distance: 1, Heap position: 0
L -> M, Distance: 1, Heap position: 0
M -> J, Distance: 2, Heap position: 0
```

```
Dijkstra:
Adding vertex K to Shortest Path Tree
Parent of I is K, Distance: 1, Heap position: 0
Parent of J is K, Distance: 1, Heap position: 0
Adding vertex I to Shortest Path Tree
Parent of H is I, Distance: 3, Heap position: 0
Adding vertex J to Shortest Path Tree
Parent of M is J, Distance: 3, Heap position: 0
Parent of L is J, Distance: 4, Heap position: 0
Parent of G is J, Distance: 2, Heap position: 0
Adding vertex G to Shortest Path Tree
Parent of E is G, Distance: 3, Heap position: 0
Parent of A is G, Distance: 8, Heap position: 0
Adding vertex M to Shortest Path Tree
Adding vertex H to Shortest Path Tree
Adding vertex E to Shortest Path Tree
Parent of F is E, Distance: 5, Heap position: 0
Parent of C is E, Distance: 7, Heap position: 0
Parent of D is E, Distance: 5, Heap position: 0
Parent of B is E, Distance: 7, Heap position: 0
Adding vertex L to Shortest Path Tree
Adding vertex D to Shortest Path Tree
Adding vertex F to Shortest Path Tree
Parent of A is F, Distance: 7, Heap position: 2
Adding vertex B to Shortest Path Tree
Adding vertex C to Shortest Path Tree
Adding vertex A to Shortest Path Tree
```

```
Shortest Path Tree:
Parent of A is F, Distance: 7
Parent of B is E, Distance: 7
Parent of C is E, Distance: 7
Parent of D is E, Distance: 5
Parent of E is G, Distance: 3
Parent of F is E, Distance: 5
Parent of G is J, Distance: 2
Parent of H is I, Distance: 3
Parent of I is K, Distance: 1
Parent of J is K, Distance: 1
Parent of L is J, Distance: 4
Parent of M is J, Distance: 3
PS C:\Users\Iria\Desktop\ALGOassignment>
```

Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

KruskalTrees.java:

```
PS C:\Users\Iria\Desktop\ALGOassignment> javac KruskalTrees.java
PS C:\Users\Iria\Desktop\ALGOassignment> java KruskalTrees

Input name of file with graph definition: wGraph1.txt
Parts[] = 13 22
Reading edges from text file
Edge A--(2)--F
Added edge: 1 6 2
Edge A--(1)--B
Added edge: 1 2 1
Edge B--(2)--D
Added edge: 2 4 2
Edge D--(1)--F
Added edge: 4 6 1
Edge B--(1)--C
Added edge: 2 3 1
Edge B--(4)--E
Added edge: 2 5 4
Edge D--(2)--E
Added edge: 4 5 2
Edge C--(4)--E
Added edge: 3 5 4
Edge E--(2)--F
Added edge: 5 6 2
Edge A--(6)--G
Added edge: 1 7 6
```

```
Added edge: 1 7 6
Edge E--(1)--G
Added edge: 5 7 1
Edge E--(4)--L
Added edge: 5 12 4
Edge F--(2)--L
Added edge: 6 12 2
Edge G--(5)--L
Added edge: 7 12 5
Edge G--(1)--J
Added edge: 7 10 1
Edge G--(3)--H
Added edge: 7 8 3
Edge H--(2)--I
Added edge: 8 9 2
Edge J--(1)--K
Added edge: 10 11 1
Edge J--(3)--L
Added edge: 10 12 3
Edge J--(2)--M
Added edge: 10 13 2
Edge L--(1)--M
Added edge: 12 13 1
Edge I--(1)--K
Added edge: 9 11 1
```

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

PS C:\Users\Iria\Desktop\ALGOassignment>
```

Iria Parada Murciego → c22305863
TU 856/2
Algorithms and Data structure assignment
Lec→ Richard Lawlor

Analysis:

What I have learned:

In the making of this assignment I have learned so many things, from more knowledge over java to the understanding of these algorithms.
As java is not the programming language in which I learned to program, and last year we used c programming for algorithms, this felt extremely hard as I felt like I didn't have enough knowledge over the language. By implementing the algorithms from their pseudocode, forced me to learn about java, in some occasions I had to check some code from last year to compare and try to implementing it into java.
However with the help of the labs and all the notes I realised it wasn't that different from c.
I had a few issues when implementing the algorithms. One of the main problems I had was that I kept getting errors when trying to display the proper weight of the MST. The program kept displaying random numbers like 0 or -9 which obviously aren't the correct answer. After changing the method of showMST() multiple times I realised the mistake was earlier in the code when adding the edges into the list. The mistake was that I was only adding one node(vertex) into the list so the program seemed to act like it didn't have any edges.
When writing the Kruskal file I had a similar error at the start but I immediately realised the problem.
To properly learn the algorithm I printed the code and drew some diagrams to understand how the code worked, and if it worked as it should.
One thing I used to help me with the assignment was watching YouTube videos about the algorithms to make the diagrams and the step by step constructions.