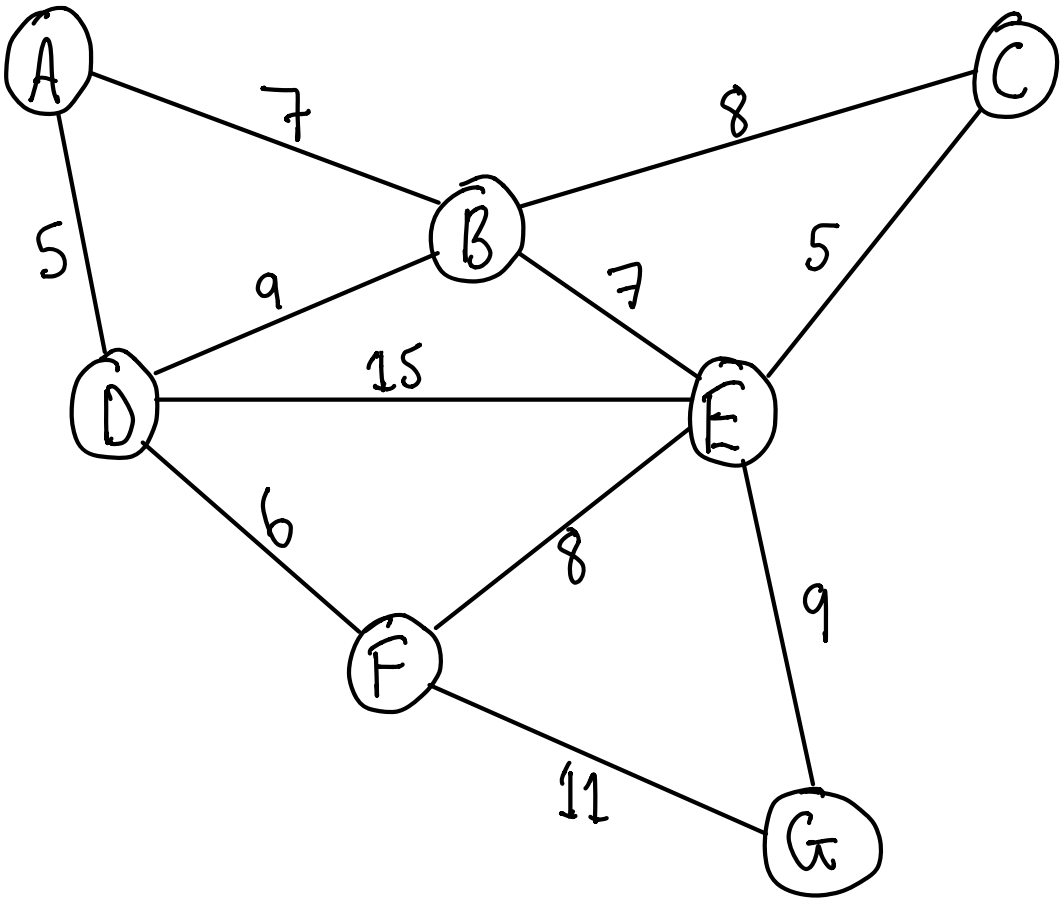
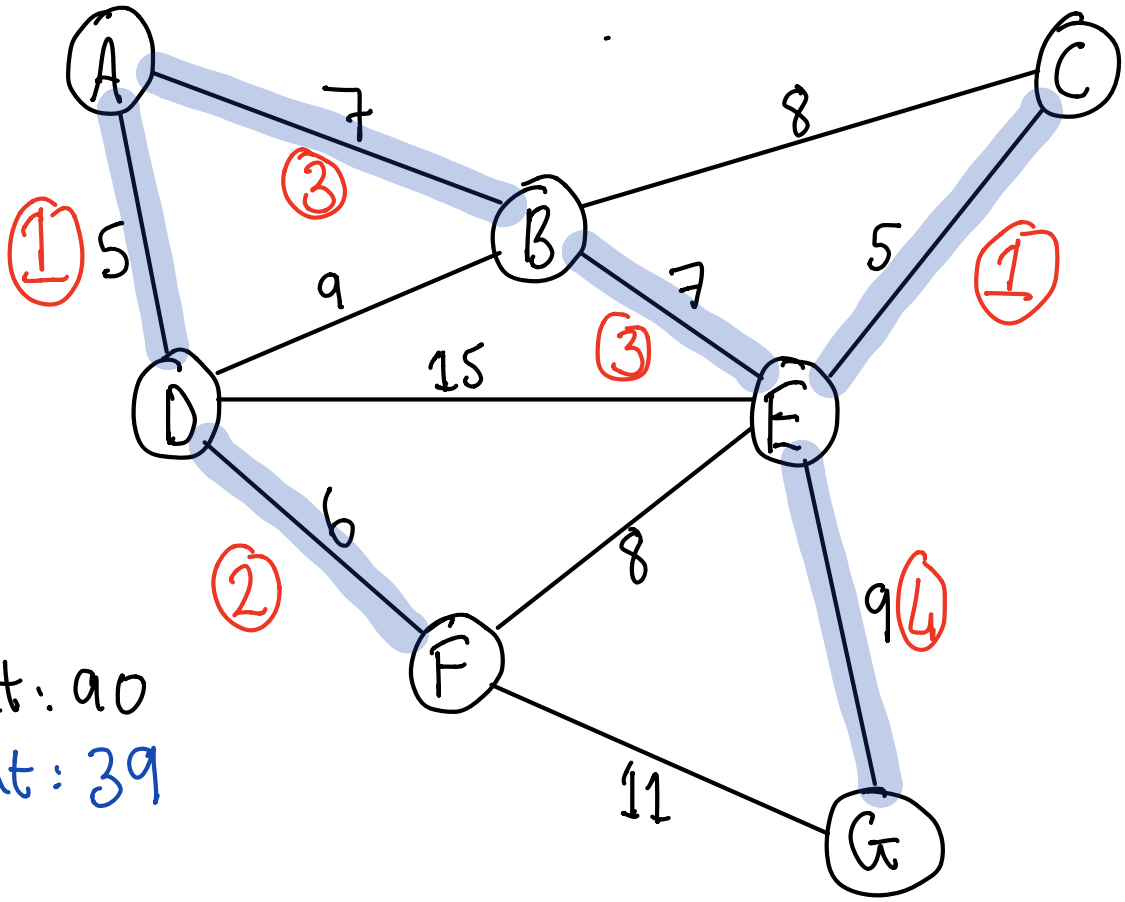
Given this following tree structure



Kruskal's Algorithm

Finds a safe edge to add to a growing forest by finding all the edges that connect any two trees in the forest an edge (u,v) of least weight. It uses a disjoint-set data structure where each vertex is initally in its own set. This is known as UnionFind in my implementation.

Minimum Weight + Union Find



Weight: 90
Weight: 39

Path Compression

1. [A D], [B], [C E], [F] [G]
2. [A D F], [B], [C E], [G]
3. [A D F C E B], [G]
4. [A D F C E B G]

Kruskal Pseudo Code

```
MST-KRUSKAL(G, w)
 MST = ∅
 for each vertex v in G.V()
      MAKE-SET(v)
sort edges of G.E() ascendingly by weight
 for each edge (u,v) in G.E() order by weight
     if (!connected(u,v))
        UNION(u,v)
        MST.insert(Edge)
return MST
```

Expected Output

A	->	D	5
C	->	E	5
D	->	F	6
A	->	B	7
B	->	E	7
E	->	G	9

39 total weight

39 < 90 therefore is minimum tree.

How to compile my code

```
Eamonn Keogh@DESKTOP-0A91N31 MINGW64 ~/eclipse-workspace/Algorithms-DS/src (master)
$ javac kruskal/*.java
Note: kruskal\Graph.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

There is a warning because java doesn't like casting generic arrays

How to run my code (cmd line input)

```
Eamonn Keogh@DESKTOP-0A91N31 MINGW64 ~/eclipse-workspace/Algorithms-DS/src (master)
$ java kruskal.Main myGraph.txt
```

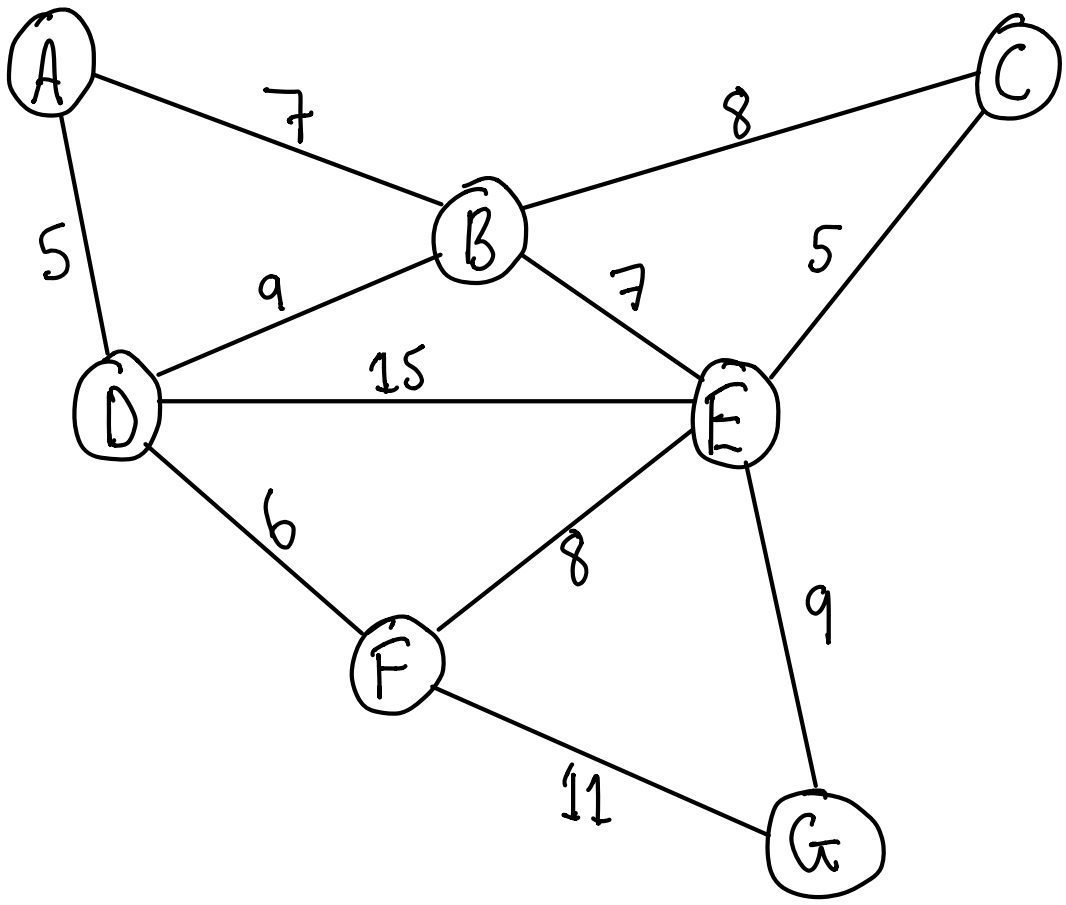
My code takes command line arguements myGraph.txt is the arguement

Program Output

```
Eamonn Keogh@DESKTOP-0A91N31 MINGW64 ~/eclipse-workspace/Algorithms-DS/src (master)
$ java kruskal.Main myGraph.txt
Vertices: 7 Edges: 11

Adjacency List: 90 weighting
A -> [B, 7] [D, 5]
B -> [C, 8] [D, 9] [E, 7]
C -> [E, 5]
D -> [E, 15] [F, 6]
E -> [F, 8] [G, 9]
F -> [G, 11]

Minimum Spanning Tree: 39 weighting
[D, 5]
[E, 5]
[F, 6]
[B, 7]
[E, 7]
[G, 9]


Eamonn Keogh@DESKTOP-0A91N31 MINGW64 ~/eclipse-workspace/Algorithms-DS/src (master)
$
```

Prim's Algorithm

Given the same graph as before find the mst.

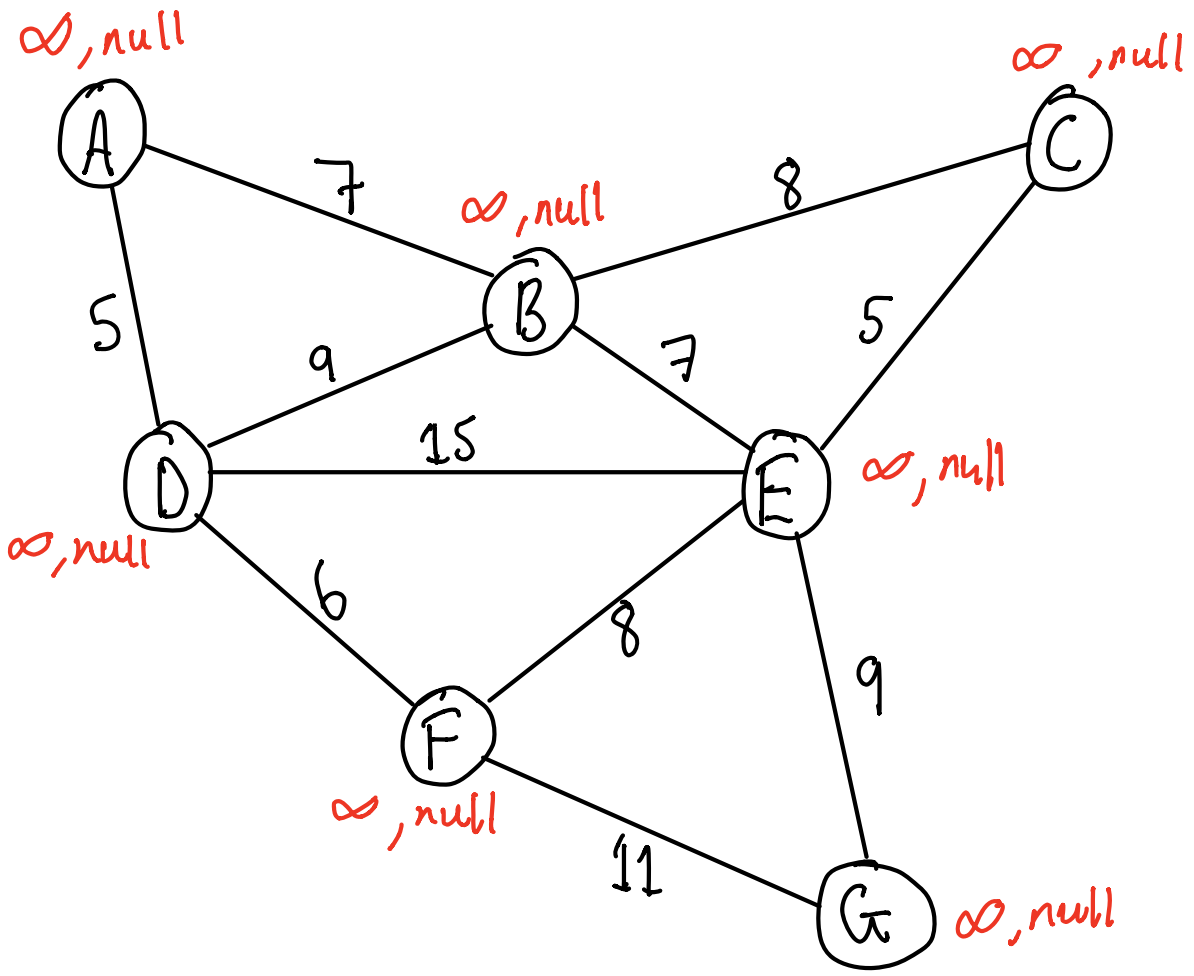


Prim's has the property that that the edges always form a single tree. This is the opposite of Kruskal where many trees are formed and merged together using find and union.

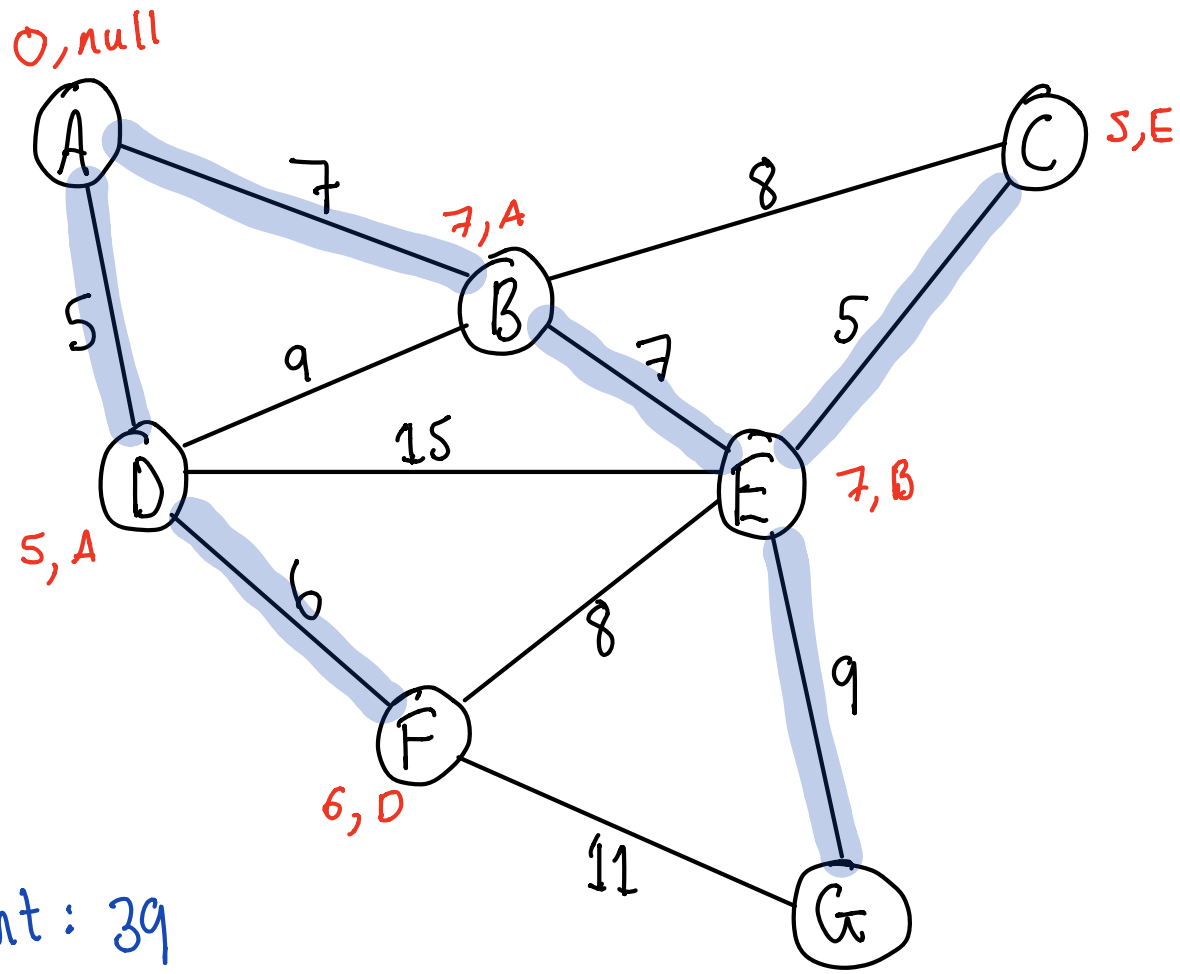
The algorithm starts from a random vertex and for each safe edge for the vertices. Differs from Kruskal's as sorting vertex keys not edges themselves.

The distance from each vertex is initially set to infinity. Then the algorithm must keep track of which vertices are adjacent and each parent vertex distance that is less.

Initial State (no parent key, inf distance)



After setting all keys and parents



Weight : 39

weight : 90

Code Output

```
Eamonn Keogh@DESKTOP-0A91N31 MINGW64 ~/eclipse-workspace/Algorithms-DS/src (master)
$ javac prim/PrimLists.java

Eamonn Keogh@DESKTOP-0A91N31 MINGW64 ~/eclipse-workspace/Algorithms-DS/src (master)
$ java prim.PrimLists
Vertices: 7 Edges: 11

Adjacency List:
A -> [D, 5] [B, 7]
B -> [E, 7] [D, 9] [C, 8] [A, 7]
C -> [E, 5] [B, 8]
D -> [F, 6] [E, 15] [B, 9] [A, 5]
E -> [G, 9] [F, 8] [D, 15] [C, 5] [B, 7]
F -> [G, 11] [E, 8] [D, 6]
G -> [F, 11] [E, 9]


Weight of MST = 39


Minimum Spanning tree is:
A -> @
B -> A
C -> E
D -> A
E -> B
F -> D
G -> E
```