KRUSKAL ALGORITHM

```
import java.util.*;
public class KruskalAlgorithm {
    static class Edge implements Comparable<Edge> {
        int src, dest, weight;
        Edge(int src, int dest, int weight) {
            this.src = src;
            this.dest = dest;
            this.weight = weight;
        }
        public int compareTo(Edge compareEdge) {
            return this.weight - compareEdge.weight;
        }
    }
    static class Subset {
        int parent, rank;
        Subset(int parent, int rank) {
            this.parent = parent;
            this.rank = rank;
        }
    }
    static int V;
    static ArrayList<Edge> edges = new ArrayList<>();
    static void addEdge(int src, int dest, int weight) {
        Edge = new Edge(src, dest, weight);
        edges.add(edge);
    }
```

```
static int find(Subset[] subsets, int i) {
    if (subsets[i].parent != i)
        subsets[i].parent = find(subsets, subsets[i].parent);
    return subsets[i].parent;
}
static void union(Subset[] subsets, int x, int y) {
    int xroot = find(subsets, x);
    int yroot = find(subsets, y);
    if (subsets[xroot].rank < subsets[yroot].rank)
        subsets[xroot].parent = yroot;
    else if (subsets[xroot].rank > subsets[yroot].rank)
        subsets[yroot].parent = xroot;
    else {
        subsets[yroot].parent = xroot;
        subsets[xroot].rank++;
    }
}
static void kruskalMST() {
    ArrayList<Edge> result = new ArrayList<>();
    int e = 0;
    int i = 0;
    Collections.sort(edges);
    Subset[] subsets = new Subset[V];
    for (int v = 0; v < V; v++)
        subsets[v] = new Subset(v, 0);
    while (e < V - 1 && i < edges.size()) {
        Edge next_edge = edges.get(i++);
        int x = find(subsets, next_edge.src);
```

```
            int y = find(subsets, next_edge.dest);
            if (x != y) {
                result.add(next_edge);
                union(subsets, x, y);
                e++;
            }
        }
        System.out.println("Minimum Spanning Tree:");
        for (Edge : result)
            System.out.println(edge.src + " - " + edge.dest + " : " + edge.weight);
    }
    public static void main(String[] args) {
        V = 4;
        addEdge(0, 1, 10);
        addEdge(0, 2, 6);
        addEdge(0, 3, 5);
        addEdge(1, 3, 15);
        addEdge(2, 3, 4);
        kruskalMST();
    }
}
```

OUTPUT:

```
Minimum Spanning Tree:
2 - 3 : 4
0 - 3 : 5
0 - 1 : 10
PS C:\Users\HP\Desktop\LP2>
```