

Task 2: Kruskal's Algorithm for MST

find the Implement minimum spanning tree of a given connected, undirected graph with non-negative edge weights.

```
package Day9_10;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
public class Kruskal {

//Java program for Kruskal's algorithm
// Defines edge structure
static class Edge {
    int src, dest, weight;

    public Edge(int src, int dest, int weight)
    {
        this.src = src;
        this.dest = dest;
        this.weight = weight;
    }
}

// Defines subset element structure
static class Subset {
    int parent, rank;

    public Subset(int parent, int rank)
    {
        this.parent = parent;
        this.rank = rank;
    }
}

// Starting point of program execution
public static void main(String[] args)
{
    int V = 4;
    List<Edge> graphEdges = new ArrayList<Edge>(
        List.of(new Edge(0, 1, 10), new Edge(0, 2, 6),
            new Edge(0, 3, 5), new Edge(1, 3, 15),
            new Edge(2, 3, 4)));

    // Sort the edges in non-decreasing order
    // (increasing with repetition allowed)
    graphEdges.sort(new Comparator<Edge>() {
        @Override public int compare(Edge o1, Edge o2)
        {
            return o1.weight - o2.weight;
        }
    });

    kruskals(V, graphEdges);
}
```

```

}

// Function to find the MST
private static void kruskals(int V, List<Edge> edges)
{
    int j = 0;
    int noOfEdges = 0;

    // Allocate memory for creating V subsets
    Subset subsets[] = new Subset[V];

    // Allocate memory for results
    Edge results[] = new Edge[V];

    // Create V subsets with single elements
    for (int i = 0; i < V; i++) {
        subsets[i] = new Subset(i, 0);
    }

    // Number of edges to be taken is equal to V-1
    while (noOfEdges < V - 1) {

        // Pick the smallest edge. And increment
        // the index for next iteration
        Edge nextEdge = edges.get(j);
        int x = findRoot(subsets, nextEdge.src);
        int y = findRoot(subsets, nextEdge.dest);

        // If including this edge doesn't cause cycle,
        // include it in result and increment the index
        // of result for next edge
        if (x != y) {
            results[noOfEdges] = nextEdge;
            union(subsets, x, y);
            noOfEdges++;
        }

        j++;
    }

    // Print the contents of result[] to display the
    // built MST
    System.out.println(
        "Following are the edges of the constructed MST:");
    int minCost = 0;
    for (int i = 0; i < noOfEdges; i++) {
        System.out.println(results[i].src + " -- "
            + results[i].dest + " == "
            + results[i].weight);
        minCost += results[i].weight;
    }
    System.out.println("Total cost of MST: " + minCost);
}

// Function to unite two disjoint sets
private static void union(Subset[] subsets, int x,
    int y)
{
    int rootX = findRoot(subsets, x);

```

```

        int rootY = findRoot(subsets, y);

        if (subsets[rootY].rank < subsets[rootX].rank) {
            subsets[rootY].parent = rootX;
        }
        else if (subsets[rootX].rank
                 < subsets[rootY].rank) {
            subsets[rootX].parent = rootY;
        }
        else {
            subsets[rootY].parent = rootX;
            subsets[rootX].rank++;
        }
    }

    // Function to find parent of a set
    private static int findRoot(Subset[] subsets, int i)
    {
        if (subsets[i].parent == i)
            return subsets[i].parent;

        subsets[i].parent
            = findRoot(subsets, subsets[i].parent);
        return subsets[i].parent;
    }
}

```

```

1 package Day9_10;
2 import java.util.ArrayList;
3 import java.util.Comparator;
4 import java.util.List;
5 public class Kruskal {
6
7
8 //Java program for Kruskal's algorithm
9 // Defines edge structure
10 static class Edge {
11     int src, dest, weight;
12
13     public Edge(int src, int dest, int weight)
14     {
15         this.src = src;
16         this.dest = dest;
17         this.weight = weight;
18     }
19 }
20
21 // Defines subset element structure
22 static class Subset {
23     int parent, rank;
24
25     public Subset(int parent, int rank)
26     {
27         this.parent = parent;
28         this.rank = rank;
29     }
30 }
31
32 // Function to find parent of a set
33 private static int findRoot(Subset[] subsets, int i)
34 {
35     if (subsets[i].parent == i)
36         return subsets[i].parent;
37
38     subsets[i].parent
39         = findRoot(subsets, subsets[i].parent);
40     return subsets[i].parent;
41 }
42
43 // Function to find parent of a set
44 private static int findRoot(Subset[] subsets, int i)
45 {
46     if (subsets[i].parent == i)
47         return subsets[i].parent;
48
49     subsets[i].parent
50         = findRoot(subsets, subsets[i].parent);
51     return subsets[i].parent;
52 }
53 }

```

Console Output:
 <terminated> Kruskal [Java Application] C:\Users\Nikita\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.16.0.2.v20210721-1149\jre\bin\javaw.exe (Jun 4, 2024, 3:43:15 PM)
 Following are the edges of the constructed MST:
 2 -- 3 == 4
 0 -- 3 == 5
 0 -- 1 == 10
 Total cost of MST: 19