CIE 2

**Name: Soham Makarand Limaye**

**Roll No: 222038**

**PRN: 22111058**

**Batch: B2**

**Aim:**You are given an undirected weighted graph with nodes and edges. The nodes are numbered from and to. Find the total weight of the minimum spanning tree, as well as one specific minimum spanning tree using Kruskal’s algorithm. Note that there may be multiple different minimum spanning trees. You need to construct any one of them.

**Objective**:

The objective of this assignment is to use Kruskal's algorithm to find the total weight of the minimum spanning tree of an undirected weighted graph with nodes and edges. Additionally, the objective is to construct any one of the possible minimum spanning trees for the given graph.

**Theory :**

Kruskal's algorithm is a greedy algorithm used to find the minimum spanning tree (MST) of a given undirected weighted graph. A minimum spanning tree is a subset of the edges that connects all the vertices of the graph with the minimum possible total edge weight. The algorithm starts with an empty set of edges and repeatedly adds the next smallest edge to the set, as long as it does not create a cycle. A cycle is created if the two endpoints of the edge are already connected by the edges in the set. The algorithm terminates when all the vertices are connected.The steps to implement Kruskal's algorithm are as follows:Sort all the edges of the graph in non-decreasing order of their weights.Initialize an empty set of edges. Iterate through the sorted edges and add each edge to the set if it does not create a cycle. Repeat step 3 until all the vertices are connected or all the edges have been considered. The total weight of the minimum spanning tree is the sum of the weights of all the edges in the set. It is important to note that there may be multiple minimum spanning trees for a given graph. Kruskal's algorithm constructs any one of the possible minimum spanning trees. The time complexity of Kruskal's algorithm is O(E log E), where E is the number of edges in the graph. This makes it an efficient algorithm for finding minimum spanning trees in large graphs.

**Code :**

import java.util.\*; public class Main { static class Edge { int src, dest, weight;

public Edge(int src, int dest, int weight)

{

this.src = src; this.dest = dest; this.weight = weight;

}

}

static class Subset { int parent, rank;

public Subset(int parent, int rank)

{

this.parent = parent; this.rank = rank;

}

}

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)));

graphEdges.sort(new Comparator<Edge>() {

@Override public int compare(Edge o1, Edge o2)

{

return o1.weight - o2.weight;

}

});

kruskals(V, graphEdges);

}

private static void kruskals(int V, List<Edge> edges)

{

int j = 0; int

noOfEdges = 0;

Subset subsets[] = new Subset[V];

Edge results[] = new Edge[V]; for (int i = 0; i < V; i++) { subsets[i] = new Subset(i, 0);

}

while (noOfEdges < V - 1) { Edge nextEdge = edges.get(j); int x = findRoot(subsets, nextEdge.src); int y = findRoot(subsets, nextEdge.dest); if (x != y) { results[noOfEdges] = nextEdge; union(subsets, x, y);

noOfEdges++;

} j++;

}

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);

}

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++;

}

}

private static int findRoot(Subset[] subsets, int i)

{

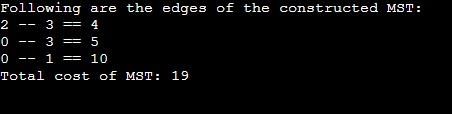
if (subsets[i].parent == i) return subsets[i].parent; subsets[i].parent

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

}

}

**Output :**



**Conclusion:**

In conclusion, Kruskal's algorithm is a simple and efficient greedy algorithm used to find the minimum spanning tree of an undirected weighted graph. It starts with an empty set of edges and adds the next smallest edge to the set, as long as it does not create a cycle. The algorithm terminates when all the vertices are connected.

The total weight of the minimum spanning tree is the sum of the weights of all the edges in the set.