NAME: - Darji Akshatkumar Hiteshbhai

RollNo: - 23MCD001

Branch: - M.tech-CSE**(Data Science)**

Subject: - Complexity Theory & Algorithms

Practical-5

Aim: Implement Kruskal's algorithm to find MST using Greedy approach.

Code for Kruskal's Algorithm –

```
#include <bits/stdc++.h>
using namespace std;

class DisjointSet
{
    vector<int> rank, parent, size;

public:
    DisjointSet(int n)
    {
        rank.resize(n + 1, 0);
        parent.resize(n + 1);
        size.resize(n + 1);
        for (int i = 0; i <= n; i++)
        {
            parent[i] = i;
            size[i] = 1;
        }
    }

    int findUPar(int node)
    {
        if (node == parent[node])
            return node;
        return parent[node] = findUPar(parent[node]);
    }

    void unionByRank(int u, int v)
    {
        int ulp_u = findUPar(u);
        int ulp_v = findUPar(v);
        if (ulp_u == ulp_v)
            return;
        if (rank[ulp_u] < rank[ulp_v])
        {
            parent[ulp_u] = ulp_v;
        }
        else if (rank[ulp_v] < rank[ulp_u])
```

```
        {
            parent[ulp_v] = ulp_u;
        }
        else
        {
            parent[ulp_v] = ulp_u;
            rank[ulp_u]++;
        }
    }

    void unionBySize(int u, int v)
    {
        int ulp_u = findUPar(u);
        int ulp_v = findUPar(v);
        if (ulp_u == ulp_v)
            return;
        if (size[ulp_u] < size[ulp_v])
        {
            parent[ulp_u] = ulp_v;
            size[ulp_v] += size[ulp_u];
        }
        else
        {
            parent[ulp_v] = ulp_u;
            size[ulp_u] += size[ulp_v];
        }
    }
};

class Solution
{
public:
    // Function to find sum of weights of edges of the Minimum Spanning Tree.
    int spanningTree(int V, vector<vector<int>> adj[])
    {
        vector<pair<int, pair<int, int>>> edges;
        for (int i = 0; i < V; i++)
        {
            for (auto it : adj[i])
            {
                int adjNode = it[0];
                int wt = it[1];
                int node = i;

                edges.push_back({wt, {node, adjNode}});
```

```
            }
        }
        DisjointSet ds(V);
        sort(edges.begin(), edges.end());
        int mstWt = 0;

        // Create a vector to store the selected edges in the MST
        vector<tuple<int, int, int>> selectedEdges;

        for (auto it : edges)
        {
            int wt = it.first;
            int u = it.second.first;
            int v = it.second.second;

            if (ds.findUPar(u) != ds.findUPar(v))
            {
                mstWt += wt;
                ds.unionBySize(u, v);

                selectedEdges.push_back(make_tuple(u, v, wt));
            }
        }

        // Print the selected edges with their weights
        cout << "Selected edges for the MST:" << endl;
        for (auto edge : selectedEdges)
        {
            int u, v, wt;
            // tuple reference i have to create to extract
            tie(u, v, wt) = edge;
            cout << "{" << u << "-" << v << ", Weight: " << wt << "}" << endl;
        }

        return mstWt;
    }
};

int main()
{

    int V = 9;
    vector<vector<int>> edges = {{1, 4, 1}, {1, 2, 2}, {2, 3, 3}, {2, 4, 3}, {1,
5, 4}, {3, 4, 5}, {2, 6, 7}, {3, 6, 8}, {4, 5, 9}};
    vector<vector<int>> adj[V];
```

```
    for (auto it : edges)
    {
        vector<int> tmp(2);
        tmp[0] = it[1];
        tmp[1] = it[2];
        adj[it[0]].push_back(tmp);

        tmp[0] = it[0];
        tmp[1] = it[2];
        adj[it[1]].push_back(tmp);
    }

    Solution obj;
    cout << "MST for : " << "{{1,4,1}, {1,2,2}, {2,3,3}, {2,4,3}, {1,5,4},
{3,4,5}, {2,6,7}, {3,6,8}, {4,5,9}}" << endl;
    int mstWt = obj.spanningTree(V, adj);
    cout << "The sum of all the edge weights: " << mstWt << endl;
    return 0;
}
```

Output –

Test Case – 1

```
PS H:\Nirma\CTA\Practical-5> g++ -o kru kruskal.cpp
PS H:\Nirma\CTA\Practical-5> ./kru
MST for : {{1,4,1}, {1,2,2}, {2,3,3}, {2,4,3}, {1,5,4}, {3,4,5}, {2,6,7}, {3,6,8}, {4,5,9}}
Selected edges for the MST:
{1-4, Weight: 1}
{1-2, Weight: 2}
{2-3, Weight: 3}
{1-5, Weight: 4}
{2-6, Weight: 7}
The sum of all the edge weights: 17
PS H:\Nirma\CTA\Practical-5>
```

Test Case – 2

```
PS H:\Nirma\CTA\Practical-5> g++ -o kru kruskal.cpp
PS H:\Nirma\CTA\Practical-5> ./kru
MST for : {{0, 1, 2}, {0, 3, 6}, {1, 2, 3}, {1, 3, 8}, {1, 4, 5}, {4, 2, 7}}
Selected edges for the MST:
{0-1, Weight: 2}
{1-2, Weight: 3}
{1-4, Weight: 5}
{0-3, Weight: 6}
The sum of all the edge weights: 16
PS H:\Nirma\CTA\Practical-5>
```

Test Case – 3

```
PS H:\Nirma\CTA\Practical-5> g++ -o kru kruskal.cpp
PS H:\Nirma\CTA\Practical-5> ./kru
MST for : {{0, 1, 2}, {0, 2, 1}, {1, 2, 1}, {2, 3, 2}, {3, 4, 1}, {4, 2, 2}}
Selected edges for the MST:
{0-2, Weight: 1}
{1-2, Weight: 1}
{3-4, Weight: 1}
{2-3, Weight: 2}
The sum of all the edge weights: 5
PS H:\Nirma\CTA\Practical-5>
```