Practical – 11

AIM: Implementat Kruskal's algorithm.

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

class DSU
{
    int *parent;
    int *rank;

public:
    DSU(int n)
    {
        parent = new int[n];
        rank = new int[n];

        for (int i = 0; i < n; i++)
        {
            parent[i] = -1;
            rank[i] = 1;
        }
    }

    int find(int i)
    {
        if (parent[i] == -1)
            return i;

        return parent[i] = find(parent[i]);
    }

    void unite(int x, int y)
    {
        int s1 = find(x);
        int s2 = find(y);

        if (s1 != s2)
        {
            if (rank[s1] < rank[s2])
            {
                parent[s1] = s2;
            }
            else if (rank[s1] > rank[s2])
            {
                parent[s2] = s1;
            }
            else
            {
                parent[s2] = s1;
                rank[s1] += 1;
            }
        }
    }
};

class Graph
```

```
{
    vector<vector<int>> edgelist;
    int V;

public:
    Graph(int V) { this->V = V; }

    void addEdge(int x, int y, int w)
    {
        edgelist.push_back({w, x, y});
    }

    void kruskals_mst()
    {
        // Sort all edges
        sort(edgelist.begin(), edgelist.end());

        DSU s(V);
        int ans = 0;
        cout << "Following are the edges in the "
                "constructed MST"
             << endl;
        for (auto edge : edgelist)
        {
            int w = edge[0];
            int x = edge[1];
            int y = edge[2];

            if (s.find(x) != s.find(y))
            {
                s.unite(x, y);
                ans += w;
                cout << x << " -- " << y << " == " << w
                     << endl;
            }
        }
        cout << "Minimum Cost Spanning Tree: " << ans;
    }
};

int main()
{
    Graph g(4);
    g.addEdge(0, 1, 10);
    g.addEdge(1, 3, 15);
    g.addEdge(2, 3, 4);
    g.addEdge(2, 0, 6);
    g.addEdge(0, 3, 5);

    g.kruskals_mst();

    return 0;
}
```

OUTPUT

```
Following are the edges in the constructed MST
2 -- 3 == 4
0 -- 3 == 5
0 -- 1 == 10
Minimum Cost Spanning Tree: 19
```

Time analysis

Operation	Time Complexity (Typical Cases)
Sorting edges by weight	O(E log E)
Initializing disjoint-set forest	O(V)
Main Loop (V-1 iterations)	O(E log V) per iteration
- Finding next minimum-weight edge	O(1) from sorted array
- Checking for cycles using DSU	O(α(V)) amortized time per operation
Total Time Complexity	O(E log E + V α(V))

Export to Sheets

Applications

Network Design:

- Telecommunication Networks: Kruskal's algorithm can be used to design efficient and cost-effective telecommunication networks by connecting all locations with minimum total cable length.
- Computer Networks: It can be employed to design network topologies with minimum cost, ensuring that all computers are connected with optimal communication links.

Circuit Design:

- Printed Circuit Boards (PCBs): In electronics, Kruskal's algorithm can be applied to design the layout of connections on a printed circuit board to minimize the total length of connections.

Transportation Planning:

- Road Networks: In urban planning, Kruskal's algorithm can be used to plan road networks in a city, connecting important locations with the least cost in terms of road construction.
- Railway Networks: Similarly, it can be applied to design railway networks efficiently.

Resource Management:

- Water Supply Networks: Kruskal's algorithm can help in designing a water supply network to connect different areas with pipes of minimum total length, reducing construction costs.
- Power Grids: It can be used in the design of electrical power grids to connect different regions with the least amount of cabling.

Cluster Analysis:

- Data Clustering: Kruskal's algorithm, or variations of it, can be used in data analysis for clustering related data points, where the goal is to minimize the total dissimilarity or distance between clusters.

Molecular Biology:

- Phylogenetic Tree Construction: In bioinformatics, Kruskal's algorithm can be used to construct phylogenetic trees based on genetic data, representing evolutionary relationships among species.

Resource Allocation:

- Project Scheduling: In project management, Kruskal's algorithm can be used to optimize resource allocation and scheduling to complete a project in the least amount of time or cost.