Assignment no 7

Problem statement: You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Edge {
    int src, dest, cost;
    bool operator<(const Edge& other) const {
        return cost < other.cost;
    }
};


class DSU {
    vector<int> parent, rank;

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

    }
    int find(int x) {
        if (parent[x] != x)
            parent[x] = find(parent[x]);
        return parent[x];
    }


    bool unionSet(int x, int y) {
        int rootX = find(x);
        int rootY = find(y);

        if (rootX == rootY) return false;

        if (rank[rootX] > rank[rootY]) {
            parent[rootY] = rootX;
        } else if (rank[rootX] < rank[rootY]) {
            parent[rootX] = rootY;
        } else {
            parent[rootY] = rootX;
            rank[rootX]++;
        }
        return true;
    }
};

class Graph {
    int V;
    vector<Edge> edges;
```

```
public:
    Graph(int vertices) : V(vertices) {}

    void addEdge(int src, int dest, int cost) {
        edges.push_back({src, dest, cost});
    }

    void kruskalMST() {
        sort(edges.begin(), edges.end());

        DSU dsu(V);
        vector<Edge> mst;
        int minCost = 0;

        for (const auto& edge : edges) {
            if (dsu.unionSet(edge.src, edge.dest)) {
                mst.push_back(edge);
                minCost += edge.cost;
            }
        }



        cout << "Minimum Spanning Tree (MST):\n";
        for (const auto& edge : mst) {
            cout << "Office " << edge.src << " - Office " << edge.dest << " : Cost = " <<
edge.cost << "\n";
        }
        cout << "Total Minimum Cost: " << minCost << endl;
    }
};
```

```
int main() {
    int V = 5;
    Graph g(V);



    g.addEdge(0, 1, 10);
    g.addEdge(0, 2, 20);
    g.addEdge(1, 2, 30);
    g.addEdge(1, 3, 5);
    g.addEdge(2, 3, 15);
    g.addEdge(3, 4, 8);



    g.kruskalMST();

    return 0;
}
```

Output:

```
Minimum Spanning Tree (MST):
Office 1 - Office 3 : Cost = 5
Office 3 - Office 4 : Cost = 8
Office 0 - Office 1 : Cost = 10
Office 2 - Office 3 : Cost = 15
Total Minimum Cost: 38
```