**Name:** Siddhant Kumar Sahu

**Batch:** E3, 57

**PRN:** 202301070159

**Practical - 4**

**CODE**

#include<bits/stdc++.h>

using namespace std;

class DisjointSet {

    vector<int> rank, parent;

public:

    DisjointSet(int n) {

        rank.resize(n + 1, 0);

        parent.resize(n + 1);

        for (int i = 0; i <= n; i++) {

            parent[i] = i;

        }

// ShubhamSP

    }

    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]++;

        }

    }

};

int kruskal\_algo(int n, vector<vector<pair<int, int>>> adj) {

    vector<pair<int, pair<int, int>>> edges;

    for (int i = 0; i < n; i++) {

        for (auto it : adj[i]) {

            int adjNode = it.first;

            int wt = it.second;

           // ShubhamSP edges.push\_back({wt, {i, adjNode}});

        }

    }

    DisjointSet ds(n);

    sort(edges.begin(), edges.end());

    int mstwt = 0;

    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.unionByRank(u, v);

        }

    }

    return mstwt;

}

int prims\_algo(vector<vector<pair<int, int>>> adj, int n) {

    priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

    vector<int> vis(n, 0);

    pq.push({0, 0});

    int sum = 0;

    while (!pq.empty()) {

        auto it = pq.top(); pq.pop();

        int node = it.second;

        int wt = it.first;

        if (vis[node]) continue;

        vis[node] = 1;

        sum += wt;

// ShubhamSP

        for (auto it : adj[node]) {

            int adjNode = it.first;

            int edgewt = it.second;

            if (!vis[adjNode]) {

                pq.push({edgewt, adjNode});

            }

        }

    }

    return sum;

}

int main() {

    int n = 5;

    vector<vector<pair<int, int>>> adj(n);

    adj[0].push\_back({1, 2});

    adj[1].push\_back({0, 2});

    adj[0].push\_back({3, 6});

    adj[3].push\_back({0, 6});

    adj[1].push\_back({2, 3});

    adj[2].push\_back({1, 3});

    adj[1].push\_back({3, 8});

    adj[3].push\_back({1, 8});

    adj[1].push\_back({4, 5});

    adj[4].push\_back({1, 5});

    adj[2].push\_back({4, 7});

    adj[4].push\_back({2, 7});

    cout << "Kruskal MST: " << kruskal\_algo(n, adj) << endl;

    cout << "Prim MST: " << prims\_algo(adj, n) << endl;

}

**OUTPUT**

