**Kruskal’s Minimum Spanning Tree Algorithm**

#include <bits/stdc++.h>

class Dsu{

vector<int>parent;

vector<int>rank;

public:

Dsu(int n):parent(n), rank(n){

iota(parent.begin(),parent.end(),0);

}

int findParent(int x){

if(x==parent[x])return x;

return parent[x] = findParent(parent[x]);

}

bool Union(int x, int y){

int p1 = findParent(x);

int p2 = findParent(y);

if(p1==p2)return false;

if(rank[p1]<rank[p2]){

parent[p1]=p2;

}else if(rank[p1]>rank[p2]){

parent[p2] = p1;

}else{

parent[p2]=p1;

rank[p1]++;

}

return true;

}

};

typedef vector<int> vi;

int kruskalMST(int n, int m, vector<vector<int>> &graph) {

Dsu ds(n+1);

priority\_queue<vi,vector<vi>,greater<vi>>pq;

for(auto &x: graph){

pq.push({x[2],x[0],x[1]});

}

int ans=0;

int e=0;

while(e!=(n-1) && !pq.empty()){

int w = pq.top()[0];

int u = pq.top()[1];

int v = pq.top()[2];

pq.pop();

int p1 = ds.findParent(u);

int p2 = ds.findParent(v);

if(p1!=p2){

ans+=w;

ds.Union(u,v);

e++;

}

}

return ans;

}