Implement Greedy search algorithm for Kruskal’s Minimal Spanning Tree 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(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:

/tmp/ZYruWDIe11.o

Following are the edges in the constructed MST

2 -- 3 == 4

0 -- 3 == 5

0 -- 1 == 10

Minimum Cost Spanning Tree: 19