1. **Thêm Các Cạnh từ *S***: Thêm tất cả các cạnh từ tập *S* vào cây bao trùm. Vì tập *S* không chứa chu trình, việc này không tạo ra chu trình trong cây bao trùm.
2. **Áp dụng Thuật toán Kruskal**: Áp dụng thuật toán Kruskal cho các cạnh còn lại của đồ thị *G* mà không nằm trong tập *S*. Chỉ thêm cạnh vào cây bao trùm nếu nó không tạo ra chu trình với các cạnh đã có trong cây.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct Edge {

int u, v, weight;

bool inS;

bool operator<(const Edge& other) const {

return weight < other.weight;

}

};

struct DSU {

vector<int> parent, rank;

DSU(int n) : parent(n), rank(n, 0) {

for (int i = 0; i < n; i++) parent[i] = i;

}

int findSet(int u) {

if (u != parent[u]) parent[u] = findSet(parent[u]);

return parent[u];

}

void unionSets(int a, int b) {

a = findSet(a);

b = findSet(b);

if (a != b) {

if (rank[a] < rank[b]) swap(a, b);

parent[b] = a;

if (rank[a] == rank[b]) rank[a]++;

}

}

};

vector<Edge> modifiedKruskal(int V, vector<Edge>& edges) {

sort(edges.begin(), edges.end());

DSU dsu(V);

vector<Edge> result;

for (Edge& e : edges) {

if (e.inS || dsu.findSet(e.u) != dsu.findSet(e.v)) {

result.push\_back(e);

dsu.unionSets(e.u, e.v);

}

}

return result;

}

int main() {

int V, E;

cin >> V >> E;

vector<Edge> edges(E);

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

cin >> edges[i].u >> edges[i].v >> edges[i].weight;

edges[i].inS = false;

}

vector<Edge> mst = modifiedKruskal(V, edges);

for (const Edge& e : mst) {

cout << e.u << " - " << e.v << " : " << e.weight << endl;

}

return 0;

}