1. Xây dựng MST bằng thuật toán Kruskal hoặc Prim.
2. Sử dụng DFS để tìm tất cả các cạnh cầu.
3. Đối với mỗi cạnh không phải cầu, xác định xem việc loại bỏ nó có làm tăng trọng số của MST không.

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct Edge {

int u, v, weight;

bool operator<(const Edge& other) const {

return weight < other.weight;

}

};

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

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

vector<int> parent(V);

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

auto findSet = [&](int u) {

while (u != parent[u]) u = parent[u];

return u;

};

vector<Edge> mst;

for (Edge& e : edges) {

int setU = findSet(e.u);

int setV = findSet(e.v);

if (setU != setV) {

mst.push\_back(e);

parent[setU] = setV;

}

}

return mst;

}

void findBridges(int u, int p, int& time, vector<int>& disc, vector<int>& low, const vector<vector<int>>& adj, vector<Edge>& bridges) {

disc[u] = low[u] = ++time;

for (int v : adj[u]) {

if (v == p) continue;

if (disc[v] == -1) {

findBridges(v, u, time, disc, low, adj, bridges);

low[u] = min(low[u], low[v]);

if (low[v] > disc[u]) bridges.push\_back({u, v, 0}); // Cạnh (u, v) là cầu

} else {

low[u] = min(low[u], disc[v]);

}

}

}

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;

}

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

vector<vector<int>> adj(V);

for (Edge& e : mst) {

adj[e.u].push\_back(e.v);

adj[e.v].push\_back(e.u);

}

vector<int> disc(V, -1), low(V, -1);

vector<Edge> bridges;

int time = 0;

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

if (disc[i] == -1) {

findBridges(i, -1, time, disc, low, adj, bridges);

}

}

for (Edge& e : bridges) {

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

}

return 0;

}