#include <iostream>

#include <vector>

#include <limits>

using namespace std;

struct Edge {

int u, v, weight;

};

struct Node {

int vertex;

Edge\* bestEdge = nullptr;

Node\* next = nullptr;

Node\* prev = nullptr;

Node(int v) : vertex(v), next(this), prev(this) {}

void addEdge(Edge\* e) {

if (!bestEdge || e->weight < bestEdge->weight) {

bestEdge = e;

}

}

void merge(Node\* other) {

if (!other) return;

Node\* tail1 = this->prev;

Node\* tail2 = other->prev;

tail1->next = other;

other->prev = tail1;

tail2->next = this;

this->prev = tail2;

}

};

void boruvkaMST(int V, const vector<Edge>& edges) {

vector<Node\*> forest(V);

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

forest[i] = new Node(i);

}

int numComponents = V;

while (numComponents > 1) {

for (auto& e : edges) {

forest[e.u]->addEdge(&e);

forest[e.v]->addEdge(&e);

}

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

if (forest[i] && forest[i]->bestEdge) {

int u = forest[i]->bestEdge->u;

int v = forest[i]->bestEdge->v;

if (forest[u] != forest[v]) {

forest[u]->merge(forest[v]);

forest[v] = nullptr;

numComponents--;

}

}

}

}

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

if (forest[i]) {

delete forest[i];

}

}

}

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;

}

boruvkaMST(V, edges);

return 0;

}