#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct Edge {

int u, v, weight;

bool operator<(Edge const& other) {

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 find\_set(int v) {

if (v == parent[v])

return v;

return parent[v] = find\_set(parent[v]);

}

void union\_sets(int a, int b) {

a = find\_set(a);

b = find\_set(b);

if (a != b) {

if (rank[a] < rank[b])

swap(a, b);

parent[b] = a;

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

rank[a]++;

}

}

};

int main() {

int n, m;

cin >> n >> m;

vector<Edge> edges(m);

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

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

}

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

DSU dsu(n);

vector<Edge> result;

for (Edge e : edges) {

if (dsu.find\_set(e.u) != dsu.find\_set(e.v)) {

result.push\_back(e);

dsu.union\_sets(e.u, e.v);

}

}

for (Edge e : result) {

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

}

return 0;

}