```
/*
    Time complexity: O(E * log(V))
    Space compleity: O(V^2)

    where E is the number of edges in the graph and
    V is the number of vertices in the graph
*/

#include <iostream>
#include <vector>
#include <climits>
using namespace std;

int findMinVertex(vector<int> &weights, vector<bool> &visited, int v) {
    int minVertex = -1;

    for (int i = 0; i < v; ++i) {
        if (!visited[i] &&
            (minVertex == -1 || weights[i] < weights[minVertex])) {
            minVertex = i;
        }
    }

    return minVertex;
}

void printMST(vector<vector<int>> &graph, int v) {
    vector<int> parent(v);
    vector<int> weights(v, INT_MAX);
    vector<bool> visited(v, false);

    parent[0] = -1;
    weights[0] = 0;

    for (int i = 0; i < v - 1; ++i) {
        // Find Min Vertex
        int minVertex = findMinVertex(weights, visited, v);
        visited[minVertex] = true;
        // Explore unvisted neighbours
        for (int j = 0; j < v; ++j) {
            if (graph[minVertex][j] != 0 && !visited[j]) {
                if (graph[minVertex][j] < weights[j]) {
                    weights[j] = graph[minVertex][j];
                    parent[j] = minVertex;
                }
            }
        }
    }
    for (int i = 1; i < v; ++i) {
        cout << min(parent[i], i) << " " << max(parent[i], i) << " " << weights[i] << "\n";
    }
}

int main() {
    int v, e;
    cin >> v >> e;

    vector<vector<int>> graph(v, vector<int>(v, 0));

    for (int i = 0, s, d, weight; i < e; ++i) {
        cin >> s >> d >> weight;
        graph[s][d] = weight;
        graph[d][s] = weight;
    }

    printMST(graph, v);
}
```