```
/*
    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> &distance, vector<int> &visited, int v) {
    int minVertex = -1;

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

    return minVertex;
}

void printShortestDistance(vector<vector<int>> &edges, int v) {
    vector<int> distance(v, INT_MAX);
    vector<int> visited(v, false);
    distance[0] = 0;

    for (int i = 0; i < v - 1; ++i) {
        int minVertex = findMinVertex(distance, visited, v);
        visited[minVertex] = true;
        for (int j = 0; j < v; ++j) {
            if (edges[minVertex][j] != 0 && !visited[j]) {
                int dist = distance[minVertex] + edges[minVertex][j];
                if (dist < distance[j]) {
                    distance[j] = dist;
                }
            }
        }
    }

    for (int i = 0; i < v; ++i) {
        cout << i << " " << distance[i] << "\n";
    }
}
```

```
int main() {
    int v, e;
    cin >> v >> e;
    vector<vector<int>> edges(v, vector<int>(v, 0));

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

    printShortestDistance(edges, v);
}
```