```
#include <iostream>

#include <algorithm>

using namespace std;


const int MAX = 100;


struct Edge {

   int u, v, weight;

};


Edge edges[MAX];

int parent[MAX];


int find(int i) {

   if (parent[i] == i) return i;

   return parent[i] = find(parent[i]);

}


void unionSet(int a, int b) {

   parent[find(a)] = find(b);

}


bool compare(Edge a, Edge b) {

   return a.weight < b.weight;

}
```

```
int main() {
    int n, e;
    cout << "Enter number of vertices and edges: ";
    cin >> n >> e;

    cout << "Enter edges (u v weight):\n";
    for (int i = 0; i < e; i++)
        cin >> edges[i].u >> edges[i].v >> edges[i].weight;

    // Initialize disjoint set
    for (int i = 0; i < n; i++) parent[i] = i;

    sort(edges, edges + e, compare);

    int cost = 0;
    cout << "Edges in MST:\n";
    for (int i = 0; i < e; i++) {
        int u = edges[i].u;
        int v = edges[i].v;
        if (find(u) != find(v)) {
            cout << u << " - " << v << " : " << edges[i].weight << "\n";
            cost += edges[i].weight;
            unionSet(u, v);
        }
    }
    cout << "Total cost of MST: " << cost << endl;

    return 0;

}
```