COA LAB – MINIMUM SPANNING TREE

SELVAKUMAR G (22MAI1004)

```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <algorithm>
#include <omp.h>

using namespace std;

struct edge {
    int u, v, w;
    edge(int u, int v, int w) : u(u), v(v), w(w) {}
};

bool operator<(const edge &a, const edge &b) {
    return a.w < b.w;
}

int find(int x, vector<int> &parent) {
    if (parent[x] == x) return x;
    return parent[x] = find(parent[x], parent);
}

void merge(int x, int y, vector<int> &parent) {
    x = find(x, parent);
    y = find(y, parent);
    if (x != y) parent[x] = y;
}

int main() {
    ifstream fin("input.txt");
    ofstream fout("output.txt");

    int n, m;
    fin >> n >> m;

    vector<edge> edges;
    for (int i = 0; i < m; i++) {
        int u, v, w;
        fin >> u >> v >> w;
        edges.push_back(edge(u, v, w));
    }

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

    vector<int> parent(n);
    for (int i = 0; i < n; i++) parent[i] = i;

    int ans = 0;
    #pragma omp parallel for reduction(+:ans)
    for (int i = 0; i < m; i++) {
        int u = edges[i].u;
        int v = edges[i].v;
        int w = edges[i].w;
        if (find(u, parent) != find(v, parent)) {
            merge(u, v, parent);
            ans += w;
        }
    }
```

```
    fout << ans << endl;

    return 0;
}
```