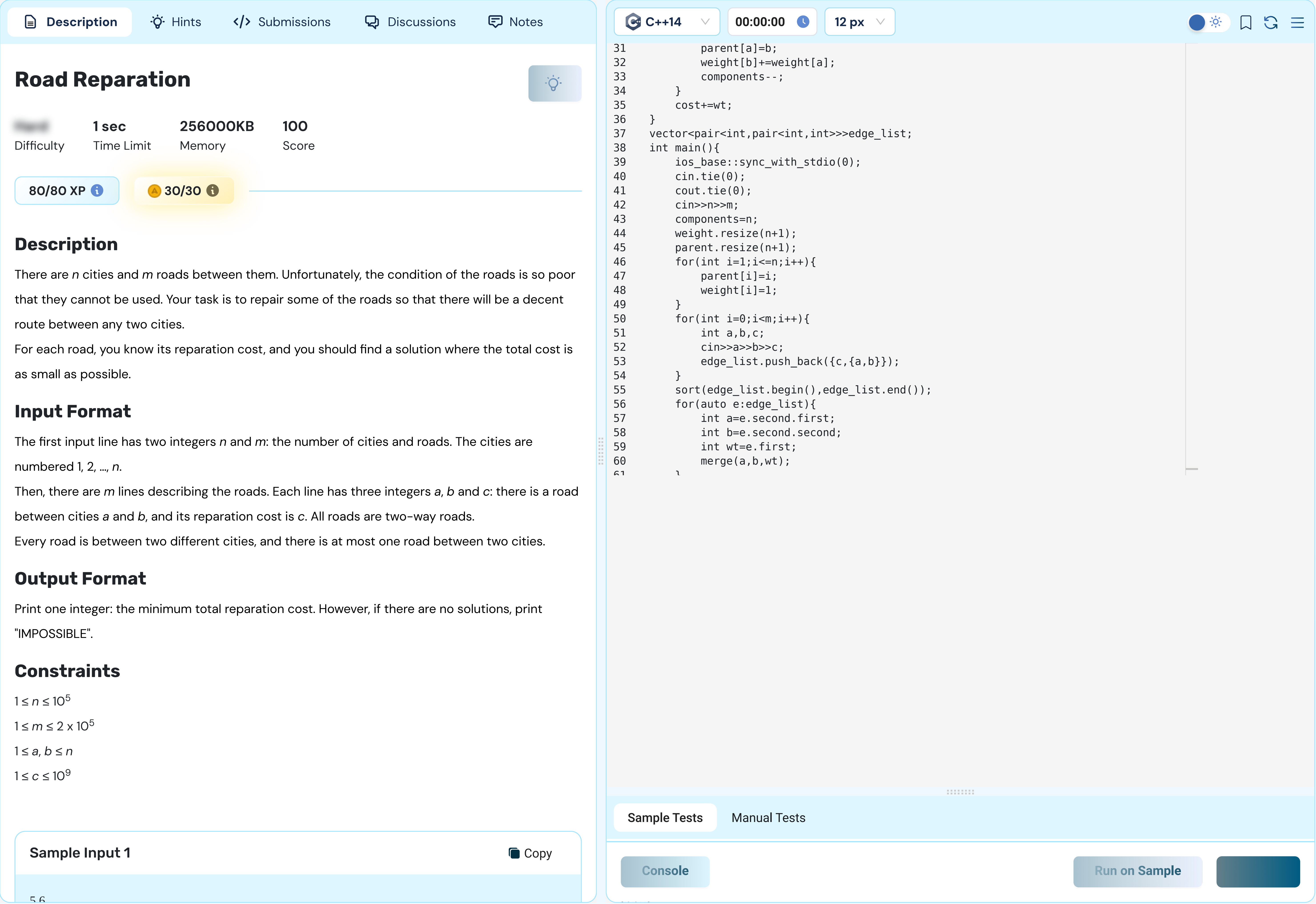

Road Reparation

[redacted]	1 sec	256000KB	100
Difficulty	Time Limit	Memory	Score

80/80 XP ⓘ 30/30 ⓘ

Description

There are *n* cities and *m* roads between them. Unfortunately, the condition of the roads is so poor that they cannot be used. Your task is to repair some of the roads so that there will be a decent route between any two cities.

For each road, you know its reparation cost, and you should find a solution where the total cost is as small as possible.

Input Format

The first input line has two integers *n* and *m*: the number of cities and roads. The cities are numbered 1, 2, ..., *n*.

Then, there are *m* lines describing the roads. Each line has three integers *a*, *b* and *c*: there is a road between cities *a* and *b*, and its reparation cost is *c*. All roads are two-way roads.

Every road is between two different cities, and there is at most one road between two cities.

Output Format

Print one integer: the minimum total reparation cost. However, if there are no solutions, print "IMPOSSIBLE".

Constraints

$1 \le n \le 10^5$

$1 \le m \le 2 \times 10^5$

$1 \le a, b \le n$

$1 \le c \le 10^9$

Sample Input 1

Copy

```
5 6
```

```
        parent[a]=b;
        weight[b]+=weight[a];
        components--;
    }
    cost+=wt;
}
vector<pair<int,pair<int,int>>>edge_list;
int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin>>n>>m;
    components=n;
    weight.resize(n+1);
    parent.resize(n+1);
    for(int i=1;i<=n;i++){
        parent[i]=i;
        weight[i]=1;
    }
    for(int i=0;i<m;i++){
        int a,b,c;
        cin>>a>>b>>c;
        edge_list.push_back({c,{a,b}});
    }
    sort(edge_list.begin(),edge_list.end());
    for(auto e:edge_list){
        int a=e.second.first;
        int b=e.second.second;
        int wt=e.first;
        merge(a,b,wt);
    }
```