Course Name:	**Competitive Programming Laboratory (216U01L401)**	**Semester:**	**IV**
Date of Performance:	**20 / 02 / 2025**	**DIV/ Batch No:**	**A1**
Student Name:	**Aaryan Sharma**	**Roll No:**	**16010123012**

Experiment No: 2

Title: To implement competitive programming problems based on Minimum Spanning Trees.

Aim and Objective of the Experiment:

1. **Understand** the concepts of Tree
2. **Apply** the Tree concepts to solve the problem
3. **Implement** the solution to given problem statement
4. **Create** test cases for testing the solution
5. **Analyze** the result for efficiency

COs to be achieved:

CO2: Analyze and optimize algorithms using amortized analysis and bit manipulation, equipping them to tackle complex computational problems.

Books/ Journals/ Websites referred:

1. https://codeforces.com/

Theory:

Kruskal's Algorithm is a greedy approach to finding the Minimum Spanning Tree (MST) of a connected, undirected, and weighted graph. The MST is a subset of edges that connects all vertices while minimizing the total edge weight and ensuring no cycles are formed. The algorithm works by first sorting all edges in non-decreasing order of weight. It then initializes a Disjoint Set Union (DSU) data structure to manage connected components. As it iterates through the sorted edges, it adds an edge to the MST only if it connects two different components, thereby avoiding cycles. The process continues until the MST contains exactly (n-1) edges.

Problem statement

Connected undirected weighted graph without self-loops and multiple edges is given. Graph contains n vertices and m edges.

For each edge (u, v) find the minimal possible weight of the spanning tree that contains the edge (u, v).

The weight of the spanning tree is the sum of weights of all edges included in spanning tree.
Input -
First line contains two integers n and m ($1 \le n \le 2 \cdot 105$, $n - 1 \le m \le 2 \cdot 105$) — the number of vertices and edges in graph.
Each of the next m lines contains three integers ui, vi, wi ($1 \le ui, vi \le n$, $ui \ne vi$, $1 \le wi \le 109$) — the endpoints of the i-th edge and its weight.
Output -
Print m lines. i-th line should contain the minimal possible weight of the spanning tree that contains i-th edge.
The edges are numbered from 1 to m in order of their appearing in input.

Input
5 7
1 2 3
1 3 1
1 4 5
2 3 2
2 5 3
3 4 2
4 5 4

Output
9
8
11
8
8
8
9

Code:

```
#include <bits/stdc++.h>
using namespace std;

vector<int> parent, dsuRank;
int find(int x)
{
  if (parent[x] != x)
  {
```

```
    parent[x] = find(parent[x]);
  }
  return parent[x];
}

// Union by rank
void unite(int x, int y)
{
  int rootX = find(x);
  int rootY = find(y);
  if (rootX != rootY)
  {
    if (dsuRank[rootX] > dsuRank[rootY])
    {
      parent[rootY] = rootX;
    }
    else if (dsuRank[rootX] < dsuRank[rootY])
    {
      parent[rootX] = rootY;
    }
    else
    {
      parent[rootY] = rootX;
      dsuRank[rootX]++;
    }
  }
}

typedef tuple<int, int, int> Edge;
vector<Edge> kruskalMST(int n, vector<Edge> &edges)
{
  sort(edges.begin(), edges.end());
  parent.resize(n + 1);
  dsuRank.resize(n + 1, 0);

  for (int i = 0; i <= n; i++)
  {
    parent[i] = i;
  }

  vector<Edge> mst;
  for (const Edge &edge : edges)
  {
    int weight, u, v;
```

```
    tie(weight, u, v) = edge;
    if (find(u) != find(v))
    {
      mst.push_back(edge);
      unite(u, v);
    }
  }

  if (mst.size() != n - 1)
  {
    cout << "Graph is disconnected, MST is incomplete!" << endl;
  }

  return mst;
}

int main()
{
  int n, m;
  cin >> n >> m;

  vector<Edge> edges;
  for (int i = 0; i < m; i++)
  {
    int u, v, weight;
    cin >> u >> v >> weight;
    edges.push_back({weight, u, v});
  }

  vector<Edge> mst = kruskalMST(n, edges);

  for (const Edge &edge : mst)
  {
    int weight, u, v;
    tie(weight, u, v) = edge;
    cout << u << " - " << v << " : " << weight << endl;
  }

  cout << "No. of edges in MST: " << mst.size() << endl;
}
```

Output:

Sample Input

```
5 7
1 2 3
1 3 1
1 4 5
2 3 2
2 5 3
3 4 2
4 5 4
```

Your Output

```
1 - 3 : 1
2 - 3 : 2
3 - 4 : 2
2 - 5 : 3
No. of edges in MST: 4
```

Conclusion:

I have implemented Kruskal's Algorithm to find the Minimum Spanning Tree (MST) using the Disjoint Set Union (DSU) data structure.