Worksheet 1.3

Name: Ayush Yadav
UID: 25IMC13004
Branch: MCA(AIML)
Section/Group: 25MAM(1)-A
Semester:1st
Date of Performance:10/09/2025
Subject Name: Design and Analysis of Algorithms Lab
Subject Code: 25CAP-612

1. Aim/Overview of the practical:

Experiment-3:

Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal's algorithm.

2. Coding :

```
class DisjointSet:
    def __init__(self, n):
        self.parent = [i for i in range(n)]
        self.rank = [0] * n

    def find(self, u):
        # Path compression
        if self.parent[u] != u:
            self.parent[u] = self.find(self.parent[u])
        return self.parent[u]

    def union(self, u, v):
        root_u = self.find(u)
        root_v = self.find(v)

        if root_u != root_v:
            # Union by rank
            if self.rank[root_u] < self.rank[root_v]:
                self.parent[root_u] = root_v
            elif self.rank[root_u] > self.rank[root_v]:
```

```
            self.parent[root_v] = root_u
        else:
            self.parent[root_v] = root_u
            self.rank[root_u] += 1


def kruskal(n, edges):
    """
    n     : number of vertices
    edges : list of edges in form (weight, u, v)
    """
    edges.sort()  # Step 1: sort edges by weight
    dsu = DisjointSet(n)


    mst_weight = 0
    mst_edges = []

    for w, u, v in edges:
        if dsu.find(u) != dsu.find(v):  # Step 2: check for cycle
            dsu.union(u, v)
            mst_weight += w
            mst_edges.append((u, v, w))

    return mst_weight, mst_edges


# Example usage
if __name__ == "__main__":
    edges = [
        (10, 0, 1),
        (6, 0, 2),
        (5, 0, 3),
        (15, 1, 3),
        (4, 2, 3)
    ]
```

```
mst_weight, mst_edges = kruskal(4, edges)

print("Edges in MST:", mst_edges)
print("Total weight of MST:", mst_weight)
```

```
Edges in MST: [(2, 3, 4), (0, 3, 5), (0, 1, 10)]
Total weight of MST: 19
```

3. Learning outcomes (What I have learnt):

1.Understand the concept of Minimum Spanning Tree (MST)

2.Learn how Kruskal's algorithm works

3.Implement Union-Find (Disjoint Set) data structure

4.Apply sorting techniques to edges based on weight

5.Enhance problem-solving and algorithmic thinking skills.