11.Given a graph represented by an edge list, implement Kruskal's Algorithm to find the Minimum Spanning Tree (MST) and its total weight.

Test Case 1:

Input:

n = 4

m = 5

edges = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]

Output:

Edges in MST: [(2, 3, 4), (0, 3, 5), (0, 1, 10)]

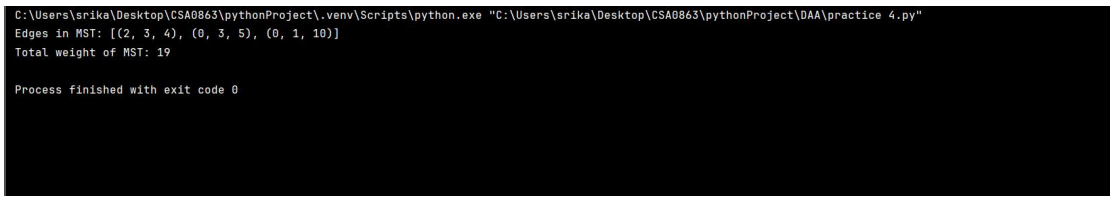
Total weight of MST: 19

Program:

```
class DisjointSet:
    def __init__(self, n):
        self.parent = list(range(n))
        self.rank = [0] * n
    def find(self, u):
        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:
            if self.rank[root_u] > self.rank[root_v]:
                self.parent[root_v] = root_u
            elif self.rank[root_u] < self.rank[root_v]:
                self.parent[root_u] = root_v
            else:
```

```
            self.parent[root_v] = root_u
            self.rank[root_u] += 1
def kruskal(n, edges):
    edges.sort(key=lambda x: x[2])
    disjoint_set = DisjointSet(n)
    mst = []
    total_weight = 0
    for u, v, weight in edges:
        if disjoint_set.find(u) != disjoint_set.find(v):
            disjoint_set.union(u, v)
            mst.append((u, v, weight))
            total_weight += weight
    return mst, total_weight
n = 4
m = 5
edges = [(0, 1, 10), (0, 2, 6), (0, 3, 5), (1, 3, 15), (2, 3, 4)]
mst, total_weight = kruskal(n, edges)
print("Edges in MST:", mst)
print("Total weight of MST:", total_weight)
```

Output:



Time complexity:

O(mlogm)