# 96. Kruskal's Algorithms

**Code:**

```
class UnionFind:
    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
```

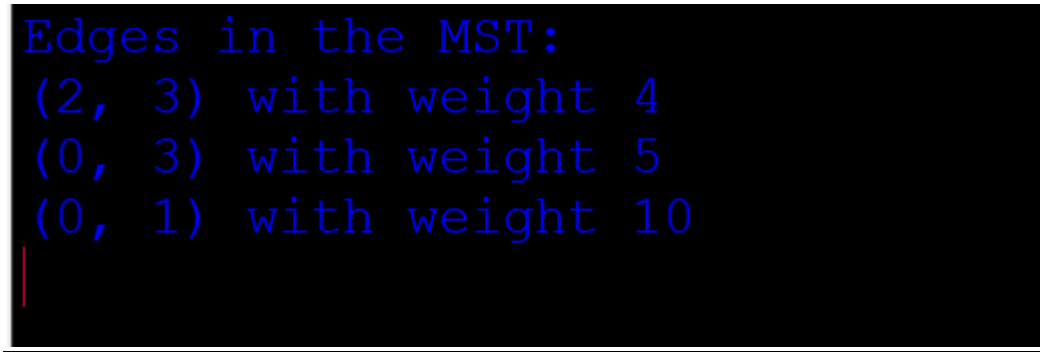
```
def kruskal(n, edges):
    uf = UnionFind(n)
    mst = []
    edges.sort(key=lambda x: x[2])

    for u, v, weight in edges:
        if uf.find(u) != uf.find(v):
            uf.union(u, v)
            mst.append((u, v, weight))

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

mst = kruskal(n, edges)
print("Edges in the MST:")
for u, v, weight in mst:
    print(f"({u}, {v}) with weight {weight}")
```

**Output:**




## Time Complexity:

- T(n)= O(logv)