97. Boruvka's Algorithm
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 boruvka(n, edges):
    disjoint_set = DisjointSet(n)
    mst = []
    mst_cost = 0

    num_components = n

    while num_components > 1:
        # Step 2: Find the smallest edge for each component
        cheapest = [-1] * n

        for u, v, weight in edges:
            set_u = disjoint_set.find(u)
            set_v = disjoint_set.find(v)

            if set_u != set_v:
                if cheapest[set_u] == -1 or cheapest[set_u][2] > weight:
                    cheapest[set_u] = (u, v, weight)
                if cheapest[set_v] == -1 or cheapest[set_v][2] > weight:
                    cheapest[set_v] = (u, v, weight)

        # Step 3: Add the smallest edges to the MST and merge components
        for edge in cheapest:
            if edge != -1:
                u, v, weight = edge
                set_u = disjoint_set.find(u)
```

```
            set_v = disjoint_set.find(v)

            if set_u != set_v:
                disjoint_set.union(set_u, set_v)
                mst.append((u, v, weight))
                mst_cost += weight
                num_components -= 1

    return mst, mst_cost

# Example usage:
n = 4
edges = [
    (0, 1, 10),
    (0, 2, 6),
    (0, 3, 5),
    (1, 3, 15),
    (2, 3, 4)
]

mst, mst_cost = boruvka(n, edges)
print("Edges in MST:", mst)
print("Total cost of MST:", mst_cost)
```

OUTPUT:-

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

=== Code Execution Successful ===
```

TIME COMPLEXITY:-O(E log V)