```
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.graph = []

    def add_edge(self, u, v, weight):
        self.graph.append([u, v, weight])

    def find(self, parent, i):
        if parent[i] == i:
            return i
        return self.find(parent, parent[i])

    def union(self, parent, rank, x, y):
        root_x = self.find(parent, x)
        root_y = self.find(parent, y)
        if rank[root_x] < rank[root_y]:
            parent[root_x] = root_y
        elif rank[root_x] > rank[root_y]:
            parent[root_y] = root_x
        else:
            parent[root_y] = root_x
            rank[root_x] += 1

    def kruskal_mst(self):
        result = []
        i = 0
        e = 0

        self.graph = sorted(self.graph, key=lambda item: item[2])

        parent = []
        rank = []

        for node in range(self.V):
            parent.append(node)
```

```
            rank.append(0)

        while e < self.V - 1:
            u, v, weight = self.graph[i]
            i += 1
            x = self.find(parent, u)
            y = self.find(parent, v)
            if x != y:
                e += 1
                result.append([u, v, weight])
                self.union(parent, rank, x, y)

        self.print_mst(result)

    def print_mst(self, result):
        print("Edge \tWeight")
        for u, v, weight in result:
            print(f"{u} - {v}\t{weight}")

# Example usage
g = Graph(4)
g.add_edge(0, 1, 10)
g.add_edge(0, 2, 6)
g.add_edge(0, 3, 5)
g.add_edge(1, 3, 15)
g.add_edge(2, 3, 4)

g.kruskal_mst()
```