96.Boruvka’s Algorithm

AIM: To find the minim(or) shortest path by using the Boruvka’s Algorithm

PROGRAM:

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

    def add_edge(self, src, dest, weight):
        self.graph.append([src, dest, weight])

    def boruvka_mst(self):
        parent = [-1] * self.V
        cheapest = [[-1, -1, float('inf')]] * self.V
        result = []

        def find(subsets, i):
            if subsets[i] == -1:
                return i
            subsets[i] = find(subsets, subsets[i])  # Path compression
            return subsets[i]

        def union(subsets, x, y):
            xroot = find(subsets, x)
            yroot = find(subsets, y)

            if xroot != yroot:
                if subsets[xroot] < subsets[yroot]:
                    subsets[xroot] = yroot
                else:
                    subsets[yroot] = xroot
```

```
        num_trees = self.V
        while num_trees > 1:
            for i in range(len(self.graph)):
                src, dest, weight = self.graph[i]
                set1 = find(parent, src)
                set2 = find(parent, dest)

                if set1 != set2:
                    if weight < cheapest[set1][2]:
                        cheapest[set1] = [src, dest, weight]
                    if weight < cheapest[set2][2]:
                        cheapest[set2] = [src, dest, weight]

            for node in range(self.V):
                if cheapest[node][0] != -1:
                    set1 = find(parent, cheapest[node][0])
                    set2 = find(parent, cheapest[node][1])

                    if set1 != set2:
                        result.append([cheapest[node][0], cheapest[node][1], cheapest[node][2]])
                        union(parent, set1, set2)
                        num_trees -= 1

            parent = [-1] * self.V

        return result

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)


print("Edges in MST:")

print(g.boruvka_mst())
```

OUTPUT:

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

TIME COMPLEXITY: O(V^2 proportional to V)