94. Minimum Spanning Tree (Krushkal's Algorithm)

AIM: To find the minimum (or) shortest path by using the Krushkal'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])  # Path compression
        return self.parent[u]

    def union(self, u, v):
        root_u = self.find(u)
        root_v = self.find(v)

        if root_u != root_v:
            # Union by rank
            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
            return True
        return False

def kruskal_mst(edges, n):
    edges.sort()  # Sort edges by weight
```

```
    ds = DisjointSet(n)
    mst = []

    for weight, u, v in edges:
        if ds.union(u, v):
            mst.append((u, v, weight))
            if len(mst) == n - 1:  # Found n-1 edges, which is enough for MST
                break

    return mst
edges = [
    (2, 0, 1), (6, 0, 3), (3, 1, 2), (5, 1, 4),
    (7, 2, 4), (8, 3, 4), (9, 3, 2)
]
num_vertices = 5  # Number of vertices in the graph

mst = kruskal_mst(edges, num_vertices)
print("Edges in the Minimum Spanning Tree (Kruskal's algorithm):")
for u, v, weight in mst:
    print(f"{u} - {v}: {weight}")
```

OUTPUT:

```
Edges in the Minimum Spanning Tree (Kruskal's
    algorithm):
0 - 1: 2
1 - 2: 3
1 - 4: 5
0 - 3: 6
```

TIME COMPLEXITY: O(E log E)