Problem – 6 : Kruskal's Algorithm – Minimum Spanning Tree

```
class UnionFind:
    def __init__(self, n):
        self.parent = list(range(n))
        self.rank = [0] * n

    def find(self, x):
        if self.parent[x] != x:
            self.parent[x] = self.find(self.parent[x])  # Path compression
        return self.parent[x]

    def union(self, x, y):
        root_x, root_y = self.find(x), self.find(y)

        if root_x == root_y:
            return False
        if self.rank[root_x] < self.rank[root_y]:
            self.parent[root_x] = root_y
        elif self.rank[root_x] > self.rank[root_y]:
            self.parent[root_y] = root_x
        else:
            self.parent[root_y] = root_x
            self.rank[root_x] += 1
        return True

def kruskal(graph):
    n = len(graph)
    edges = []
    for i in range(n):
        for j in range(i + 1, n):
            if graph[i][j] != 0:
```

```
                edges.append((i, j, graph[i][j]))

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

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

    return mst

graph = [
    [0, 2, 0, 6, 0],
    [2, 0, 3, 8, 5],
    [0, 3, 0, 0, 7],
    [6, 8, 0, 0, 9],
    [0, 5, 7, 9, 0],
]

minimum_spanning_tree = kruskal(graph)
print("Minimum Spanning Tree:")
for edge in minimum_spanning_tree:
    print(f"{edge[0]} -- {edge[1]}: weight {edge[2]}")
```