Practical No. : 3

Name : Mohan Kadambande
Roll No. : 13212
Aim : Implement Krushkal's Aigorithm using Greedy Search Algorthm.
Code :

```
class DisjointSet:
    def __init__(self, n):
        self.parent = [i for i in 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):
        rootX = self.find(x)
        rootY = self.find(y)

        # Union by rank
        if rootX != rootY:
            if self.rank[rootX] > self.rank[rootY]:
                self.parent[rootY] = rootX
            elif self.rank[rootX] < self.rank[rootY]:
                self.parent[rootX] = rootY
            else:
                self.parent[rootY] = rootX
                self.rank[rootX] += 1

# Function to implement Kruskal's Algorithm
def kruskal_mst(V, edges):
    # Sort all edges in non-decreasing order of weight
    edges.sort(key=lambda edge: edge[2])

    disjoint_set = DisjointSet(V)
    mst = []

    # Process each edge in sorted order
    for u, v, weight in edges:
        if disjoint_set.find(u) != disjoint_set.find(v):
            disjoint_set.union(u, v)
            mst.append((u, v, weight))
    # Print the MST
    print("Edge \tWeight")
    for u, v, weight in mst:
        print(f"{u} - {v} \t{weight}")

# Main function to take user input
def main():
    V = int(input("Enter the number of vertices: "))
    E = int(input("Enter the number of edges: "))

    edges = []
    print("Enter the edges (u, v, weight):")
    for _ in range(E):
        u, v, weight = map(int, input().split())
        edges.append((u, v, weight))

    # Call Kruskal's algorithm
    kruskal_mst(V, edges)

if __name__ == "__main__":
    main()
```

Output :

```
Enter the number of vertices: 4
Enter the number of edges: 5
Enter the edges (u, v, weight):
0 1 10
0 2 6
0 3 5
1 3 15
2 3 4
Edge    Weight
2 - 3   4
0 - 3   5
0 - 1   10
```