94. minimum spainning tree
program:

```
def find(parent, i):
    if parent[i] == i:
        return i
    parent[i] = find(parent, parent[i])  # Path compression
    return parent[i]
def union(parent, rank, x, y):
    x_root = find(parent, x)
    y_root = find(parent, y)

    if rank[x_root] < rank[y_root]:
        parent[x_root] = y_root
    elif rank[x_root] > rank[y_root]:
        parent[y_root] = x_root
    else:
        parent[y_root] = x_root
        rank[x_root] += 1
def kruskal_mst(vertices, graph):
    result = []
    i, e = 0, 0

    graph = sorted(graph, key=lambda item: item[2])
    parent = []
    rank = []
    for node in range(vertices):
        parent.append(node)
        rank.append(0)
    while e < vertices - 1:
        u, v, w = graph[i]
        i += 1
        x = find(parent, u)
        y = find(parent, v)
        if x != y:
            e += 1
            result.append([u, v, w])
            union(parent, rank, x, y)

    return result
vertices = 4
edges = [
    [0, 1, 10],
    [0, 2, 6],
    [0, 3, 5],
    [1, 3, 15],
    [2, 3, 4]
]
mst = kruskal_mst(vertices, edges)
print("Minimum Spanning Tree:")
for u, v, weight in mst:
    print(f"Edge: {u} - {v}, Weight: {weight}")
```

Output:

```
Minimum Spanning Tree:
Edge: 2 - 3, Weight: 4
Edge: 0 - 3, Weight: 5
Edge: 0 - 1, Weight: 10

=== Code Execution Successful ===
```

Time complexity:O(ElogV)