Objective: **Implementation and analysis of Kruskal's Algorithm**

Kruskal's Algorithm

Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph

Kruskal's algorithm

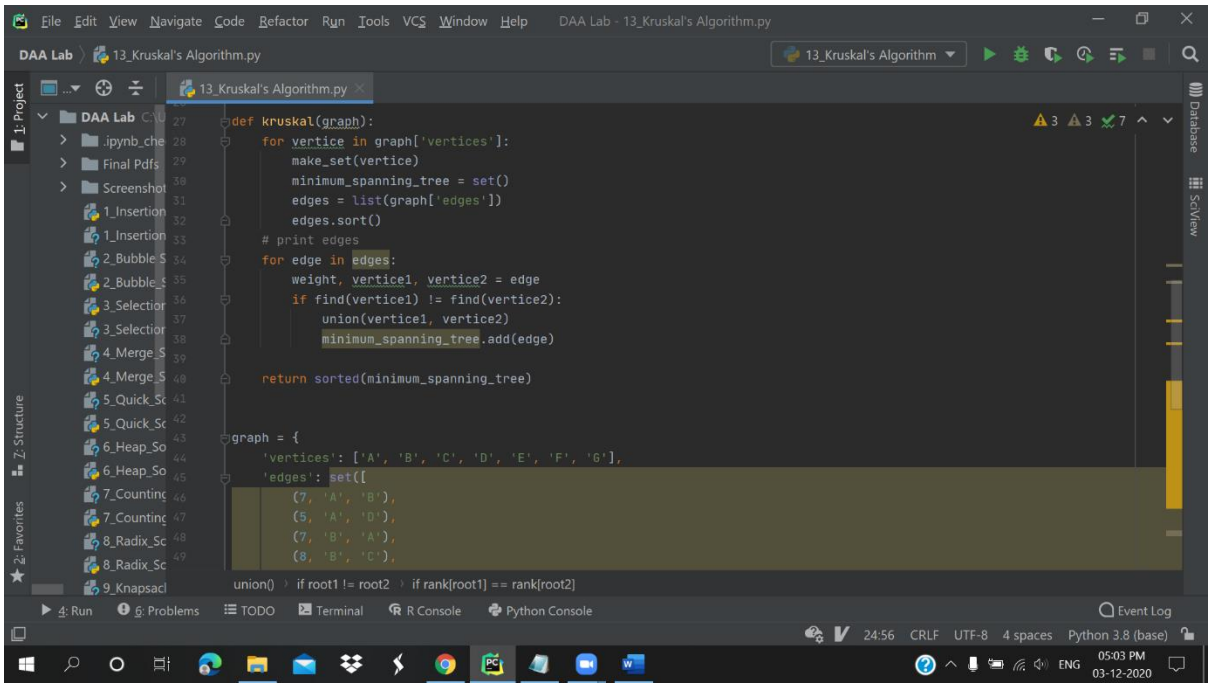
The steps for implementing Kruskal's algorithm are as follows:

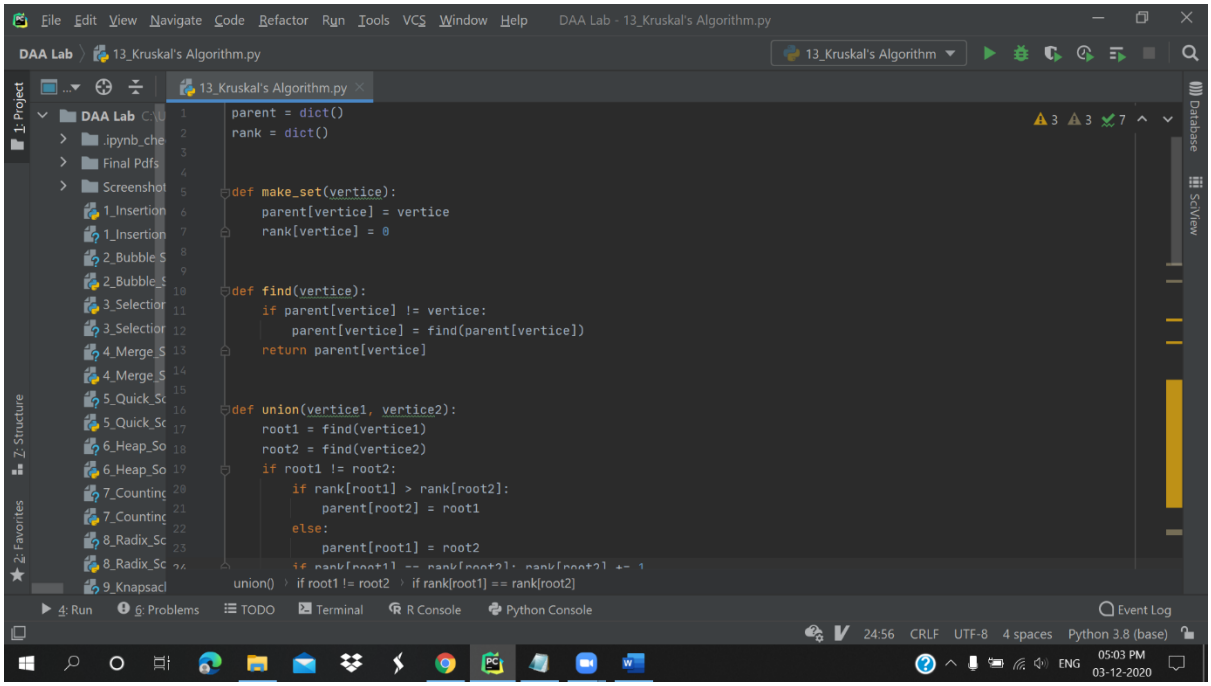
1. Sort all the edges from low weight to high
2. Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
3. Keep adding edges until we reach all vertices.

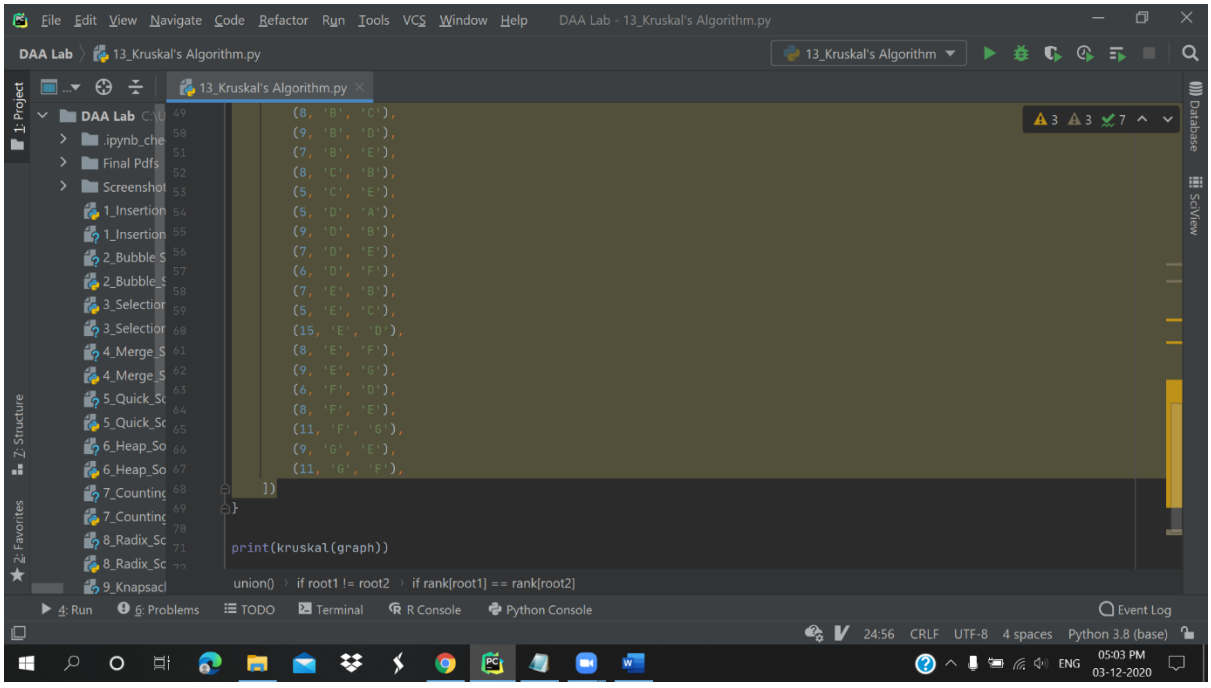
Code:

```
parent = dict()
rank = dict()
def make_set(vertice):
    parent[vertice] = vertice
    rank[vertice] = 0
def find(vertice):
    if parent[vertice] != vertice:
        parent[vertice] = find(parent[vertice])
    return parent[vertice]
def union(vertice1, vertice2):
    root1 = find(vertice1)
    root2 = find(vertice2)
    if root1 != root2:
        if rank[root1] > rank[root2]:
            parent[root2] = root1
        else:
            parent[root1] = root2
        if rank[root1] == rank[root2]: rank[root2] +=
1
def kruskal(graph):
    for vertice in graph['vertices']:
        make_set(vertice)
        minimum_spanning_tree = set()
        edges = list(graph['edges'])
        edges.sort()
    # print edges
    for edge in edges:
        weight, vertice1, vertice2 = edge
        if find(vertice1) != find(vertice2):
            union(vertice1, vertice2)
            minimum_spanning_tree.add(edge)
    return sorted(minimum_spanning_tree)

graph = {
    'vertices': ['A', 'B', 'C', 'D', 'E', 'F', 'G'],
    'edges': set([
        (7, 'A', 'B'),
        (5, 'A', 'D'),
        (7, 'B', 'A'),
        (8, 'B', 'C'),
        (9, 'B', 'D'),
        (7, 'B', 'E'),
        (8, 'C', 'B'),
        (5, 'C', 'E'),
        (5, 'D', 'A'),
        (9, 'D', 'B'),
        (7, 'D', 'E'),
        (6, 'D', 'F'),
        (7, 'E', 'B'),
```

```
            (5, 'E', 'C'),
            (15, 'E', 'D'),
            (8, 'E', 'F'),
            (9, 'E', 'G'),
            (6, 'F', 'D'),
            (8, 'F', 'E'),
            (11, 'F', 'G'),
            (9, 'G', 'E'),
            (11, 'G', 'F'),
        ])
}

print(kruskal(graph)
```

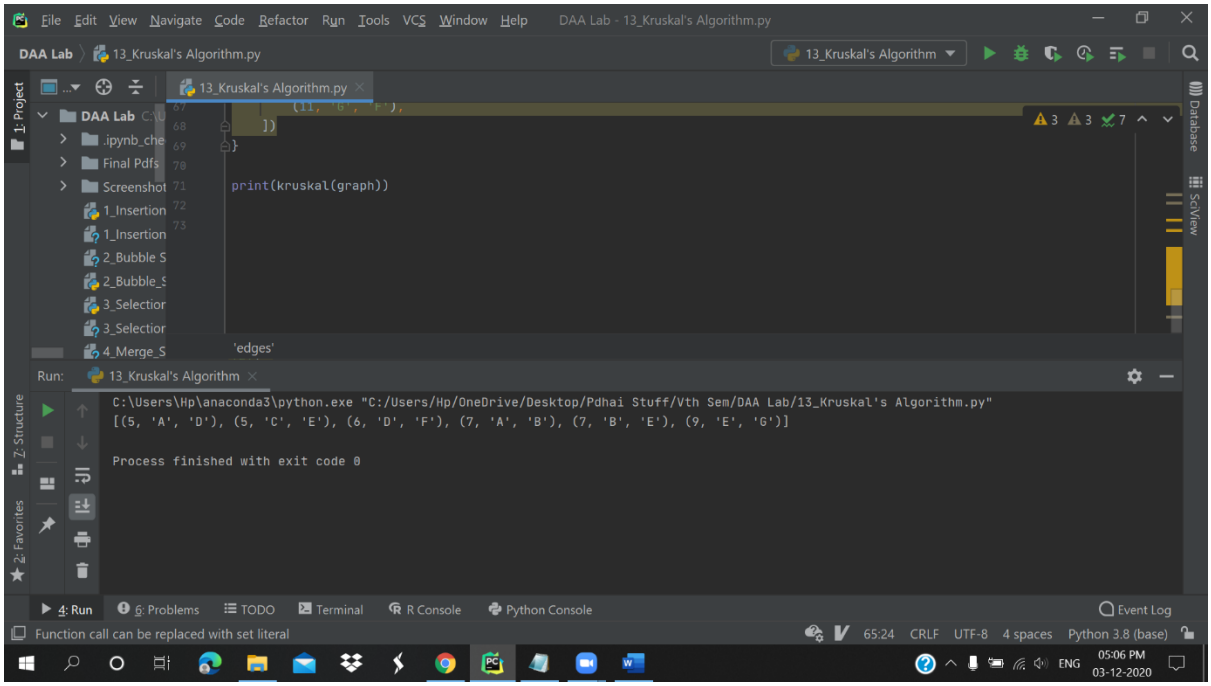






Output:

```
[(5, 'A', 'D'), (5, 'C', 'E'), (6, 'D', 'F'), (7, 'A', 'B'),
(7, 'B', 'E'), (9, 'E', 'G')]
```



Time Complexities:

The time complexity of Kruskal's algorithm is O (E log E).

Kruskal's Algorithm Applications:

- In order to layout electrical wiring
- In computer network (LAN connection)