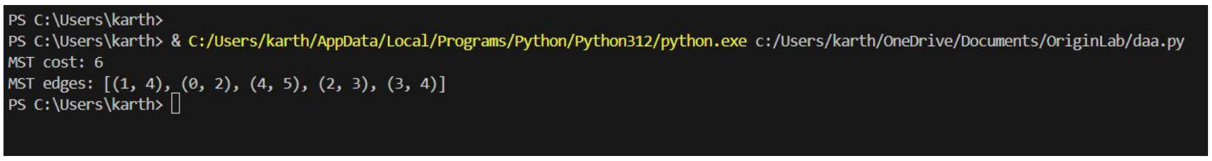
8. Kruskal's Algorithms,

Code:

```
class UnionFind:
    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])
        return self.parent[u]
    def union(self, u, v):
        root_u = self.find(u)
        root_v = self.find(v)
        if root_u != root_v:
            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
def kruskal_mst(graph):
    edges = []
    for u in graph:
        for v, weight in graph[u]:
            edges.append((weight, u, v))
    edges.sort()
    uf = UnionFind(len(graph))
    mst_cost = 0
    mst_edges = []
    for weight, u, v in edges:
        if uf.find(u) != uf.find(v):
            uf.union(u, v)
            mst_cost += weight
            mst_edges.append((u, v))
    return mst_cost, mst_edges
graph = {
    0: [(1, 10), (2, 1), (3, 4)],
    1: [(0, 10), (2, 3), (4, 0)],
    2: [(0, 1), (1, 3), (3, 2), (4, 8)],
    3: [(0, 4), (2, 2), (4, 2), (5, 7)],
    4: [(1, 0), (2, 8), (3, 2), (5, 1)],
    5: [(3, 7), (4, 1)]
}
mst_cost, mst_edges = kruskal_mst(graph)
print("MST cost:", mst_cost)
print("MST edges:", mst_edges)
```

output:



```
PS C:\Users\karth>
PS C:\Users\karth> & C:/Users/karth/AppData/Local/Programs/Python/Python312/python.exe c:/Users/karth/OneDrive/Documents/OriginLab/daa.py
MST cost: 6
MST edges: [(1, 4), (0, 2), (4, 5), (2, 3), (3, 4)]
PS C:\Users\karth>
```

Time complexity:

F(n)=o(eloge+elogv)