

Question – Kruskal's algorithms

Code

```
1 def kruskal(graph):
2     edges = []
3     for v in graph:
4         for u, w in graph[v]:
5             edges.append((w, v, u))
6
7     edges.sort()
8
9     sets = {}
10    for v in graph:
11        sets[v] = v
12
13    mst = []
14
15    for w, v1, v2 in edges:
16        root1 = find(sets, v1)
17        root2 = find(sets, v2)
18
19        if root1 != root2:
20            mst.append((v1, v2, w))
21            sets[root2] = root1
22
23    total_weight = sum(w for v1, v2, w in mst)
24
25    return mst, total_weight
26
27 def find(sets, v):
28     if sets[v] != v:
29         sets[v] = find(sets, sets[v])
30     return sets[v]
```

```
31
32 # Example usage
33 graph = {
34     'A': [('B', 5), ('D', 4)],
35     'B': [('C', 3), ('D', 2)],
36     'C': [('D', 6)],
37     'D': [('B', 2), ('A', 4)]
38 }
39
40 mst, total_weight = kruskal(graph)
41 print("Minimum spanning tree (MST):")
42 for u, v, weight in mst:
43     print("{} - {} : {}".format(u, v, weight))
44 print("Total Weight:", total_weight)
```

Output –

```
PS C:\Users\aryan\OneDrive - st.niituniversity.in\DAA Assignment\Assignmet -1
.in/DAA Assignment/Assignmet -11/03.py"
Minimum Spanning Tree:
B - D : 2
B - C : 3
A - D : 4
Total Weight: 9
```

Analysis-

Kruskal's algorithm is a simple, efficient, and widely used algorithm for finding the minimum spanning tree of a graph. It is particularly useful when the graph is sparse (i.e., has relatively few edges compared to the number of vertices) or when the edges are already sorted. However, it may not be the best choice when the graph is very dense or when a faster algorithm is needed for very large graphs.

Algorithm works by iteratively adding the smallest edge in the graph that does not create a cycle until all vertices are in the same connected component. The implementation uses a disjoint-set data structure for cycle detection, and uses path compression to improve the efficiency of finding the root of a set.

Time Complexity – $O(E \log E)$

The time complexity of Kruskal's algorithm remains the same for the case

Space Complexity – $O(E+V)$