V = Number of vertices in the graph.

E = Number of edges in the graph.

[2] Kruskal's Algorithm:

1. Initialize an empty list called "MST" to store the edges of the minimum spanning tree.

2. Sort all the edges of the graph in increasing order of their weights.

3. Initialize a disjoint set data structure to keep track of the connected components of the graph.

4. For each edge (u, v) in the sorted list of edges:

If it's safe edge or in other words, if adding the edge (u, v) to the MST does not create a cycle in the MST:

→ Then add the edge (u, v) to the MST and Merge the connected components of u and v in the disjoint set.

5. Repeat step 4 until MST contains (V - 1) edges.

Disjoint Set Data Structure Operations:

- **MakeSet(v):** Create a new set with a single element v in the first.

- **FindSet(v):** Find the representative of the set containing v.

- **UnionSets(u, v):** Merge the sets containing u and v into a single set.

Time Complexity:

- **Sorting the edges:** O (E log E).
- **Union-Find Operations:** O (E log V), Kruskal's algorithm processes edges in ascending order of weight and performs union-find operations on the vertices of each edge to check for cycles. The union-find operations take O (log V) time in the worst case. Since there are E edges, the total time complexity for union-find operations is O (E log V).

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

Space Complexity:

- The space complexity of the adjacency list representation of the graph is O (V + E), where V is the number of vertices and E is the number of edges.
- The space complexity of the disjoint-set data structure (union-find) is O(V) since it needs to store information for each vertex.

The overall space complexity of Kruskal's algorithm is O (V + E).