**a.**

Kruskal's algorithm finds a Minimum Spanning Tree (MST) for a connected, weighted, and undirected graph. It works by:

1. Sorting all edges in non-decreasing order of their weights.
2. Adding edges to the MST, ensuring no cycles are formed, until all vertices are connected.

**Required Sub-algorithms**

1. **Disjoint Set (Union-Find) Data Structure**
   * MakeSet(n): Initializes n disjoint sets for each vertex.
   * Find(u): Finds the representative (root) of the set containing u.
   * Union(u, v): Merges the sets containing u and v.
2. **Kruskal’s Algorithm**
   * Sort edges by weight.
   * Iterate through sorted edges and use Find and Union to decide whether to include the edge in the MST.

**b.**

1. **Sorting the Edges**
   * Time Complexity: O(Elog⁡E)O(E \log E)O(ElogE), where EEE is the number of edges.
   * Reason: Sorting algorithms like Merge Sort or Quick Sort are used.
2. **Disjoint Set Operations**
   * MakeSet: O(V)O(V)O(V), where VVV is the number of vertices.
   * Find: Nearly constant time (O(α(V))O(\alpha(V))O(α(V))), where α\alphaα is the inverse Ackermann function.
   * Union: Nearly constant time (O(α(V))O(\alpha(V))O(α(V))).
3. **Kruskal’s Algorithm**
   * Overall Time Complexity: O(Elog⁡E+E⋅α(V))O(E \log E + E \cdot \alpha(V))O(ElogE+E⋅α(V)), dominated by edge sorting.

**Correctness of Kruskal’s Algorithm**

* The algorithm ensures a cycle-free MST due to the use of the disjoint set.
* It guarantees minimal cost by always choosing the smallest weight edge that doesn't form a cycle.

**Space Complexity**

* Space for storing edges: O(E)O(E)O(E).
* Space for disjoint set: O(V)O(V)O(V).