a. Algorithms Needed for Kruskal's Algorithm

1. **Sort the edges**: First, sort all the edges of the graph in non-decreasing order of their weights.
   - Input: A list of edges with weights.
   - Output: Sorted list of edges based on their weights.
2. **Find-Union Data Structure**: Use a data structure like Union-Find (Disjoint Set Union) to manage the merging of sets during the execution of Kruskal's algorithm.
   - **Find**: Determine the set that an element belongs to.
   - **Union**: Merge two sets together.
3. **Kruskal's Algorithm**:
   - Initialize a forest (each node is its own set).
   - Traverse the sorted edge list and for each edge:
     - If the two nodes are in different sets, add the edge to the MST and union the sets.
     - If the two nodes are in the same set, skip the edge (to avoid cycles).

b. Analysis of the Algorithms

1. **Sorting the edges**: Sorting takes O(E log E), where EEE is the number of edges. Sorting is the most expensive part of Kruskal's algorithm.
2. **Find-Union Operations**:
   - **Find**: The time complexity for each find operation is almost constant if path compression is used, i.e., $O(\alpha(V))$, where α is the inverse Ackermann function, which grows very slowly.
   - **Union**: The union operation is also nearly constant, i.e., $O(\alpha(V))$, especially if union by rank/size is used.
3. **Overall Time Complexity**: The overall time complexity of Kruskal's algorithm is dominated by the sorting step, which is O(E log E). The union-find operations are very efficient and almost constant time.