Analysis of the Algorithm

Kruskal's algorithm is a greedy algorithm used to find the Minimum Spanning Tree (MST) of a connected, undirected graph with weighted edges. The algorithm follows these steps:

1. **Initialization**:
    - Create a graph with a specified number of vertices.
    - Store edges in a list, where each edge is represented as a tuple of two vertices and a weight.
2. **Sorting Edges**:
    - Sort all edges in non-decreasing order based on their weights. This is crucial because the algorithm processes the smallest edges first to ensure that the MST remains minimal.
3. **Union-Find Structure**:
    - Use a union-find (disjoint-set) data structure to keep track of which vertices are in which components. This helps in efficiently checking whether adding an edge would create a cycle.
    - The **find** function implements path compression to flatten the structure of the tree whenever **find** is called, which speeds up future operations.
    - The **union** function uses union by rank to attach smaller trees under larger trees, keeping the overall structure balanced.
4. **Building the MST**:
    - Iterate through the sorted edges and for each edge, check if the vertices of the edge belong to different components using the **find** function.
    - If they do, add the edge to the MST and unite the two components using the **union** function.
    - Stop when the number of edges in the MST equals **V - 1**, where **V** is the number of vertices.
5. **Output**:
    - Print the edges included in the MST and the total cost of the MST.

Complexity Analysis

- **Time Complexity**:
    - Sorting the edges takes (O(E \log E)), where (E) is the number of edges.
    - Each union and find operation takes nearly constant time, (O(\alpha(V))), where (\alpha) is the inverse Ackermann function, which grows very slowly.
    - Overall, the time complexity is (O(E \log E)).

- **Space Complexity**:
    - The space complexity is (O(V + E))