Kruskal’s algorithm to find MST of a network

a)

Kruskal's algorithm is a greedy algorithm used to find the minimum spanning tree (MST) of a connected, undirected graph. It works by sorting the edges in non-decreasing order of their weights and then adding edges to the MST while ensuring no cycles are formed.
Here are the steps and algorithms needed:

1. **Disjoint Set (Union-Find) Data Structure:**

The Union-Find data structure is crucial for Kruskal’s algorithm to efficiently manage and merge disjoint sets of nodes while checking for cycles.

- **Find Operation**: This operation finds the root of the set containing the element. It uses path compression to optimize future queries.
- **Union Operation**: This operation merges two sets. It uses union by rank to keep the tree shallow, making future operations faster.

2. **Kruskal’s Algorithm:**

- **Sort all the edges** in the graph in increasing order of their weights.
- **Initialize a disjoint set** data structure to manage connected components.
- **Iterate through the sorted edges** and for each edge, check if it forms a cycle:
  - ✓ If it doesn’t form a cycle (i.e., the two vertices are in different components), include this edge in the MST and perform a union operation.
  - ✓ If it forms a cycle, discard the edge.
- **Repeat until you have included (n - 1) edges** (where n is the number of vertices in the graph). At this point, the MST is complete.

b)

Time Complexity Analysis:

1. **Sorting the edges:**
   Sorting the edges takes **O(ElogE),** where E is the number of edges.
2. **Union-Find Operations:**
   The time complexity of each union and find operation using path compression and union by rank is almost constant, specifically **$O(\alpha(n))$,** where **$\alpha(n)$** is the inverse Ackermann function, which grows extremely slowly and is nearly constant for all practical values of **n.**
   - ✓ For **E** edges, the total complexity of union and find operations is **$O(E\alpha(n))$.**
3. **Overall Complexity:**
   Therefore, the total time complexity of Kruskal’s algorithm is dominated by the sorting step, resulting in:
   $O(E\log E + E\alpha(n)) \approx O(E\log E)$
   <u>**This is efficient for sparse graphs (where E is much less than n^2).**</u>

Space Complexity Analysis:

1. **Disjoint Set:**
   The space complexity for the Union-Find data structure is **O(n),** where n is the number of nodes (vertices).
2. **Edges:**
   The space complexity for storing the edges is **O(E),** where **E** is the number of edges in the graph.
3. **Overall Complexity:**
   The space complexity **is O(n+E**), which is efficient.

Correctness:

- Kruskal's algorithm ensures that no cycles are formed because the algorithm only adds an edge if the two vertices of the edge are in different connected components (checked using the Union-Find data structure).
- The algorithm guarantees that the MST is optimal because it always adds the next smallest edge that doesn't form a cycle, following the greedy approach.

c)

Kruskal.cs