Complexity Analysis of CriticalPathAnalyzer

1. Tree Traversal (traverseAndAddToQueue)

The method traverses the entire tree recursively. Each node is visited once, and for every node:

- - The method of the priority queue is called to add the node with its depth.
- - The recursive call is made for each child.

Time Complexity

Let:

- - N: Total number of nodes in the tree.
- - k: Average branching factor of the tree (number of children per node).

Each node is visited once, resulting in O(N) time for traversal. Priority queue insertions take O(N log N), giving an overall time complexity of O(N log N).

Space Complexity

The recursive depth depends on the tree's height h:

- - For a balanced tree, h = O(log N).
- - For a skewed tree, h = O(N).

The priority queue stores up to N nodes, consuming O(N) space. Auxiliary objects also require O(N) space.

2. Priority Queue Operations

The priority queue stores all N nodes with their depths. When retrieving or displaying the critical path, poll operations extract elements in descending order of depth. Each poll operation takes O(log N).

Time Complexity

Extracting N elements takes O(N log N) in total.

Space Complexity

The priority queue itself requires O(N) space.

3. Auxiliary Storage and Result Retrieval

The getCriticalPath method creates a new list and sorts it. Sorting N elements takes O(N log N), and the list requires O(N) space.

4. Summary Table

Operation	Time Complexity	Space Complexity
Analyze Tree	O(N log N)	O(N)
Display Path	O(N log N)	O(N)
Get Critical Path	O(N log N)	O(N)
Total	O(N log N)	O(N)