Documentation

The problem "Binary Tree Paths" involves identifying all possible root-to-leaf paths in a binary tree and returning them in a specified format. A path is defined as a sequence of connected nodes that starts at the root and ends at a leaf node, where a leaf is any node with no children. To tackle this, we use a Depth-First Search (DFS) approach, which efficiently explores each path from the root to a leaf. This recursive DFS approach helps traverse the tree systematically, allowing us to build paths from the root to each leaf node without missing any paths.

The DFS approach is especially suitable here because it enables us to track each unique path as we traverse deeper into the tree. We start from the root node and move down each branch, appending each node's value to a path string. For each node, if it is a leaf, we consider the path complete and add it to our result list. If not, we continue by exploring its left and right children. This way, as we navigate each branch, we build paths dynamically, which lets us capture the structure of each unique route from the root to the leaf nodes.

To manage path construction, we update the path string by appending the node values separated by arrows (->). This ensures that each path is visually clear, following the specified format. When we reach a leaf node, this complete path is saved into our final result list. A key advantage of this approach is that each path is built only once as we move through the tree, making the solution efficient and concise. Additionally, we avoid extra processing by immediately identifying and storing paths once we reach a leaf, reducing redundant operations.

In terms of efficiency, this solution is optimal for the problem constraints. Each node in the tree is visited once, making the time complexity (O(N)), where (N) is the total number of nodes. This is efficient since we only visit nodes as necessary and don't revisit any node. Moreover, the space complexity is also kept within bounds at (O(N)), accounting for the recursion stack in the DFS, especially in cases where the tree is unbalanced. This ensures the solution is scalable to the maximum constraints specified by the problem.

Overall, this solution is an effective way to capture all root-to-leaf paths in a binary tree with minimal complexity and maximum readability. The recursive DFS with path-building captures the tree structure accurately, ensuring all paths are stored as strings in the required format. This approach is not only intuitive for binary trees but also adaptable, making it useful for similar problems where path tracking in a tree structure is required. The method provides a clear and concise means to convert each path in the tree into a string, providing an elegant solution to the "Binary Tree Paths" problem.