Construct Binary Tree from Inorder and Postorder Traversal

Key Observations:

1. Postorder traversal: [Left Subtree, Right Subtree, Root]
2. Inorder traversal: [Left Subtree, Root, Right Subtree]
3. Last element of postorder is always the root.
4. Locate this root in inorder to:
    - Identify left subtree elements (before root in inorder).
    - Identify right subtree elements (after root in inorder).

Approach:

- Use recursion with indices (avoid slicing arrays to keep O(1) extra space).
- Maintain: · postIndex (current root index in postorder, starting from end.)
    - hash map inMap to store value → index mapping for inorder.

Steps:

1. Build a map for inorder to quickly find root positions.
2. Recursive function buildTreeHelper (postorder, inStart, inEnd):
    - If inStart > inEnd, return nullptr.
    - Create root from postorder[postIndex], decrement postIndex.
    - Find root's index in inorder from inMap.
    - Build right subtree first (since postorder is Left-Right-Root, we process from right to left).
    - Build left subtree.

Complexity:

- Time: O(N) - each node is processed once, map lookup is O(1)
- Space: O(N) - hashmap + recursion stack (worst case skewed tree).