K-th Smallest Element in a BST

This problem takes advantage of the BST property:
- The inorder traversal of a BST gives sorted node values.
- The k-th smallest element is the k-th value visited in inorder traversal.

Approach 1: Simple Inorder Traversal (O(n) time, O(h) space):

Steps:
1. Perform inorder traversal (left → root → right).
2. Keep a counter while traversing.
3. When the counter equals k, return that node's value.

Complexity:
- Time: O(k+h) worst-case (visiting k nodes; h is height)
- Space: O(h) due to recursion stack.

Follow-Up: Optimized for Frequent Modifications

If the BST is modified frequently (insert/delete) and we need k-th smallest queries often:

Optimization:
- Augment the BST nodes with an extra field: size = number of nodes in its subtree.
- While inserting/deleting, update size.
- To find k-th smallest:
  1. Use size(left subtree) to decide:
     - If k == size(left) + 1, return root.
     - If k <= size(left), search left subtree.
     - Else, search right subtree with k - size(left) - 1.

This reduces query time to O(h) and supports efficient updates.