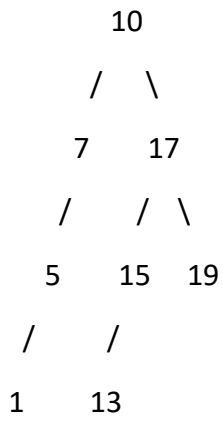
Find the Kth Largest Element in Binary Search Tree CodeStudio

You are given a binary search tree (BST), and your task is to create a program that finds the Kth largest element in the BST. The Kth largest element is the Kth largest value in the sorted (in-order) list of values in the tree.

Example:



K = 4

Output: The 4 Largest Element of BST: 13

Approach 1: Find the kth Largest Element in a Binary Search Tree using the Inorder Traversal

- **Function Purpose:** This approach finds the kth largest element in the BST by performing an inorder traversal of the tree.
- **Explanation:**
    1. The function performs an inorder traversal and stores the result in the 'inorderAns' vector.
    2. It returns the kth largest element from the vector (0-based index).
- **Time Complexity: The time complexity of this approach is O(N), where N is the number of nodes in the tree, as it needs to visit all nodes.**
- **Space Complexity: The space complexity is O(N) due to the vector used to store the inorder traversal values.**

Approach 2: Find the kth Largest Element in a Binary Search Tree (Optimized)

- **Function Purpose:** This approach finds the kth largest element in the BST using an optimized approach without storing the entire inorder traversal.
- **Explanation:**
    1. The function uses a helper function to perform a reverse inorder traversal of the tree, maintaining an index for counting the visited nodes.
    2. It returns the kth largest element without storing the entire traversal result.
- **Time Complexity: The time complexity of this approach is O(H), where H is the height of the tree. In the worst case, it's O(N) for a skewed tree.**
- **Space Complexity: The space complexity is O(H) due to the recursive call stack, where H is the height of the tree. In the worst case, it's O(N) for a skewed tree.**

Conclusion:

- Approach 1 uses an inorder traversal to find the Kth largest element and is straightforward to implement.
- Approach 2 provides an optimized solution in terms of space complexity, making it more efficient for large trees in terms of memory usage.
- Both approaches accurately find the Kth largest element in a BST.
- The choice between the two approaches depends on the specific requirements and characteristics of the tree being processed.