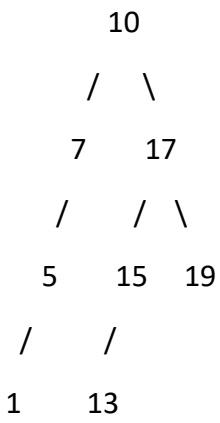
Lowest Common Ancestor (LCA) of BST LeetCode

Given a binary search tree (BST), find the lowest common ancestor (LCA) node of two given nodes in the BST.

"The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**)."

Example:



P = 5, Q = 13

Output: The Lowest Common Ancestor of 5 and 13 is: 10

Approach 1: Recursive function to find the Lowest Common Ancestor (LCA) of two nodes in a Binary Search Tree

- **Function Purpose:** This approach uses a recursive function to find the LCA of two nodes in a BST.
- **Explanation:**
    1. The function recursively traverses the tree and compares the values of the current node with the values of the two target nodes, p and q.
    2. Depending on the values, it continues the search in the left or right subtree.
    3. When it finds the LCA, it returns the LCA node.
- **Time Complexity: The time complexity of this approach is O(H), where H is the height of the tree.**
- **Space Complexity: The space complexity is O(H), primarily due to the recursive function call stack.**

Approach 2: Iterative function to find the Lowest Common Ancestor (LCA) of two nodes in a Binary Search Tree

- **Function Purpose:** This approach uses an iterative function to find the LCA of two nodes in a BST.
- **Explanation:**
    1. The function iteratively traverses the tree and compares the values of the current node with the values of the two target nodes, p and q.
    2. Depending on the values, it moves to the left or right subtree.
    3. When it finds the LCA, it returns the LCA node.
- **Time Complexity: The time complexity of this approach is O(H), where H is the height of the tree.**
- **Space Complexity: The space complexity is O(1) as it uses a constant amount of additional space for iterative traversal.**

Conclusion:

- Both approaches efficiently find the Lowest Common Ancestor (LCA) of two nodes in a Binary Search Tree (BST).
- The recursive approach is straightforward to implement, while the iterative approach is more space-efficient as it uses an iterative algorithm.
- The choice between the two approaches depends on the specific requirements and characteristics of the tree being processed.