Binary Search Tree (Insertion) LeetCode

A Binary Search Tree (BST) is a binary tree data structure where each node has at most two children, typically referred to as the left child and the right child. The BST property, which makes it special, ensures that for every node:

1. All nodes in its left subtree have values less than or equal to the node's value.
2. All nodes in its right subtree have values greater than the node's value.

This property allows for efficient searching, insertion, and deletion operations, making BSTs suitable for a wide range of applications, including data storage and searching algorithms.

Approach 1: Function to insert a value into the binary search tree

1. It takes two parameters: **root**, a pointer to the current node in the BST, and **value**, the value you want to insert.
2. If the **root** is initially **nullptr**, it means the tree is empty or we've reached a leaf node. In this case, it creates a new node with the given **value** and makes it the root of the tree.
3. If the **value** already exists in the tree (equal to the **root->value**), no changes are needed, and the function returns.
4. If the **value** is less than the **root->value**, it recursively calls **insertBST** on the left subtree to insert the **value**.
5. If the **value** is greater than the **root->value**, it recursively calls **insertBST** on the right subtree to insert the **value**.
6. The function returns the **root** node after the insertion operation.

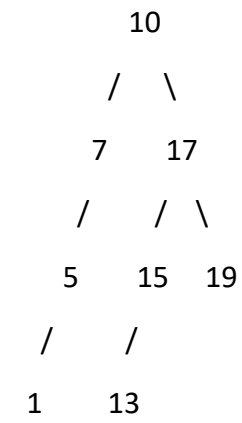
Time Complexity of BST Insertion:

The time complexity of the insertBST function depends on the height of the tree (h), where h is the longest path from the root to a leaf node. In the best case, when the tree is balanced, the time complexity is O(log n), where n is the number of nodes. In the worst case, when the tree is skewed, the time complexity is O(n).

Space Complexity of BST Insertion:

The space complexity is O(h) because of the recursive function calls, where h is the height of the tree. In the best case (balanced tree), it is O(log n), and in the worst case (skewed tree), it is O(n).

Example:



Level Order Traversal:

Level 0: 10

Level 1: 7, 17

Level 2: 5, 15, 19

Level 3: 1, 13

Inorder Traversal: 1, 5, 7, 10, 13, 15, 17, 19

Preorder Traversal: 10, 7, 5, 1, 17, 15, 13, 19

Postorder Traversal: 1, 5, 7, 13, 15, 19, 17, 10