Insertion in a BST is quite similar with search in a BST. Here we only provide recursive solution in Java and C++ for your reference.

|  |
| --- |
| /\*\*  \* Definition for a binary tree node.  \* public class TreeNode {  \* int val;  \* TreeNode left;  \* TreeNode right;  \* TreeNode(int x) { val = x; }  \* }  \*/  class Solution {  public TreeNode insertIntoBST(TreeNode root, int val) {  if (root == null) {  return new TreeNode(val); // return a new node if root is null  }  if (root.val < val) { // insert to the right subtree if val > root->val  root.right = insertIntoBST(root.right, val);  } else { // insert to the left subtree if val <= root->val  root.left = insertIntoBST(root.left, val);  }  return root;  }  } |

The time complexity of the insertion operation is the same with search operation which is O(h). Or O(N) in the worst case and O(logN) ideally if the tree is well organized.

The space complexity of the recursion soultion is O(h) as well. In other word, O(N) in the worst case and O(logN) ideally. If you implement the algorithm iteratively, the space complexity can be O(1).