# **BINARY SEARCH TREE**

# 25/11/2024

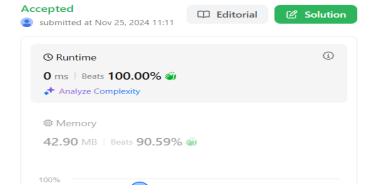
### 1. Search in BST:

```
class Solution {
   public TreeNode searchBST(TreeNode root, int val) {
       if(root== null) return null;
       if (root.val==val) return root;
       if(root.val<val){</pre>
           return searchBST(root.right,val);
       }else{
           return searchBST(root.left,val);
       }
   }
}
 Accepted
                                     ☐ Editorial
                                                       Solution
    submitted at Nov 25, 2024 10:51
     O Runtime
                                                               (i)
     0 ms | Beats 100.00% 🞳
      Analyze Complexity
     Memory
     44.86 MB | Beats 93.37% 🞳
      Analyze Complexity
```

#### 2. VALIDATE BINARY SEARCH TREE:

```
class Solution {
   public boolean isValidBST(TreeNode root) {
      return validate(root, Long.MIN_VALUE, Long.MAX_VALUE);
   }

   private boolean validate(TreeNode node, long min, long max) {
      if (node == null) return true;
      if (node.val <= min || node.val >= max) return false;
      return validate(node.left, min, node.val) && validate(node.right, node.val, max);
   }
}
```



# 3.BINARY SERACH TREE:

```
class Node {
  int value;
  Node left, right;
  Node(int value) {
     this.value = value;
     this.left = null;
     this.right = null;
class BST {
  Node root;
  BST() {
     root = null;
  }
  public void insert(int value) {
     root = insertRec(root, value);
  private Node insertRec(Node root, int value) {
     if (root == null) {
       root = new Node(value);
       return root;
```

```
if (value < root.value) {
        root.left = insertRec(root.left, value);
     } else if (value > root.value) {
        root.right = insertRec(root.right, value);
     return root;
  public boolean isValidBST() {
     return validate(root, Long.MIN VALUE, Long.MAX VALUE);
  private boolean validate(Node node, long min, long max) {
     if (node == null) {
        return true;
     if (node.value <= min || node.value >= max) {
        return false;
     return validate(node.left, min, node.value) && validate(node.right,
node.value, max);
public class Main {
  public static void main(String[] args) {
     BST tree = new BST();
     tree.insert(10);
     tree.insert(5);
     tree.insert(15);
     tree.insert(3);
     tree.insert(7);
     System.out.println(tree.isValidBST());
Output: true
```