[**Validate Binary Search Tree**](https://leetcode.com/problems/validate-binary-search-tree/)

**public** **class** ValidateBST {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

TreeNode root = **new** TreeNode(10);

root.left = **new** TreeNode(12);

root.left.left = **new** TreeNode(3);

root.left.right = **new** TreeNode(7);

root.right = **new** TreeNode(15);

root.right.right = **new** TreeNode(18);

System.***out***.println(*isValidBST*(root));

}

**public** **static** **boolean** isValidBST(TreeNode root) {

**if**(root == **null**) {

**return** **true**;

}

**return** *helper*(root , Long.***MAX\_VALUE*** , Long.***MIN\_VALUE***);

}

**public** **static** **boolean** helper(TreeNode root , Long min , Long max) {

**if**(root == **null**) {

**return** **true**;

}

**if**(root.val >= min || root.val <= max) {

**return** **false**;

}

**return** *helper*(root.left , (**long**)root.val , max) && *helper*(root.right, min , (**long**) root.val);

}

}

Time complexity : O(n) n is Number of nodes in given tree

Space Complexity : O(1) , constant space