LEETCODE 98

By

Priyansh

To validate A Given tree is Bst or Not :-

DEFINATION: -

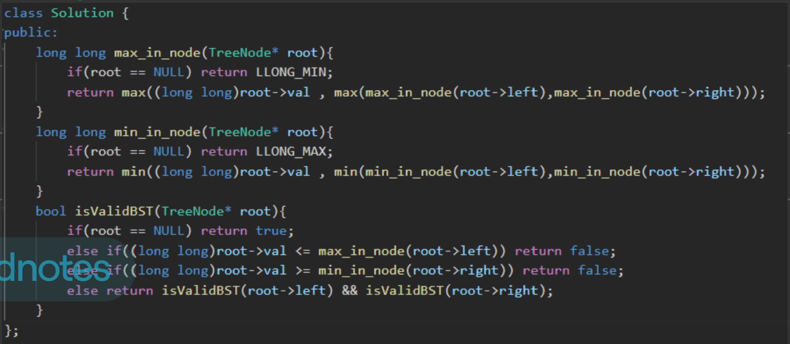
A BST is special form of Binary tree where for each node its left nodes(left subtree)→value is smaller then roots value and also right subtree or right subnodes are having a value greater then value of root...

Brute force Method :-

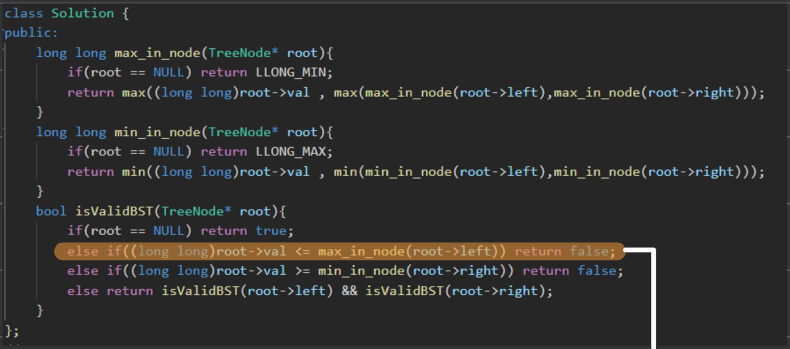
Algorithm:

To visit each node of a BST and validate its defination if it found voilating defination of BST at any node we would immediately return FALSE signifying its not a BST...

Code Snippet:

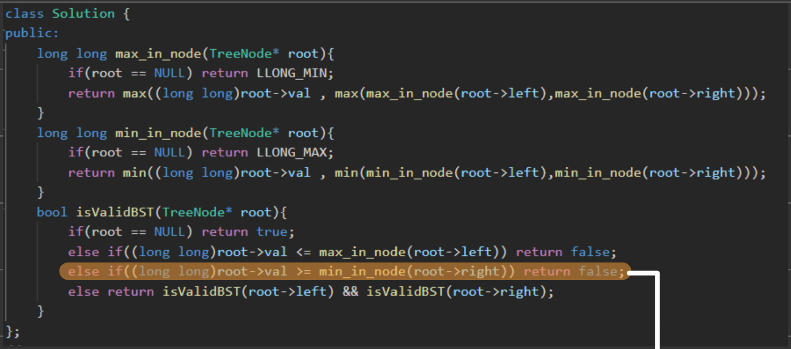


```cpp
class Solution {
public:
    long long max_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MIN;
        return max((long long)root->val , max(max_in_node(root->left),max_in_node(root->right)));
    }
    long long min_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MAX;
        return min((long long)root->val , min(min_in_node(root->left),min_in_node(root->right)));
    }
    bool isValidBST(TreeNode* root){
        if(root == NULL) return true;
        else if((long long)root->val <= max_in_node(root->left)) return false;
        else if((long long)root->val >= min_in_node(root->right)) return false;
        else return isValidBST(root->left) && isValidBST(root->right);
    }
};
```



```cpp
class Solution {
public:
    long long max_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MIN;
        return max((long long)root->val , max(max_in_node(root->left),max_in_node(root->right)));
    }
    long long min_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MAX;
        return min((long long)root->val , min(min_in_node(root->left),min_in_node(root->right)));
    }
    bool isValidBST(TreeNode* root){
        if(root == NULL) return true;
        else if((long long)root->val <= max_in_node(root->left)) return false;
        else if((long long)root->val >= min_in_node(root->right)) return false;
        else return isValidBST(root->left) && isValidBST(root->right);
    }
};
```

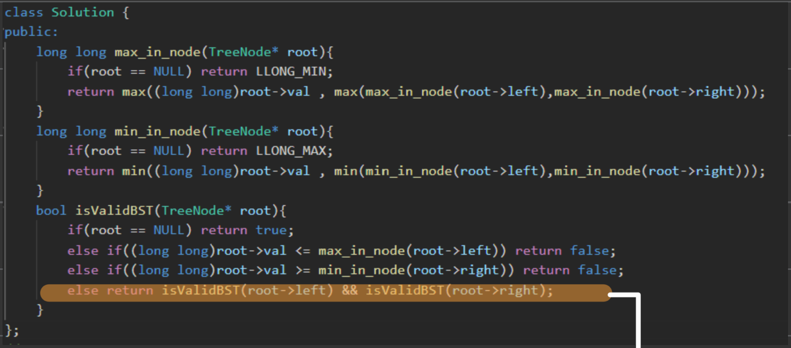
→ root → val Ko must be > max value of left Subtree node...
if not return false



```cpp
class Solution {
public:
    long long max_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MIN;
        return max((long long)root->val , max(max_in_node(root->left),max_in_node(root->right)));
    }
    long long min_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MAX;
        return min((long long)root->val , min(min_in_node(root->left),min_in_node(root->right)));
    }
    bool isValidBST(TreeNode* root){
        if(root == NULL) return true;
        else if((long long)root->val <= max_in_node(root->left)) return false;
        else if((long long)root->val >= min_in_node(root->right)) return false;
        else return isValidBST(root->left) && isValidBST(root->right);
    }
};
```

→ root → val Ko must be < min value of right Subtree node..

if not return false



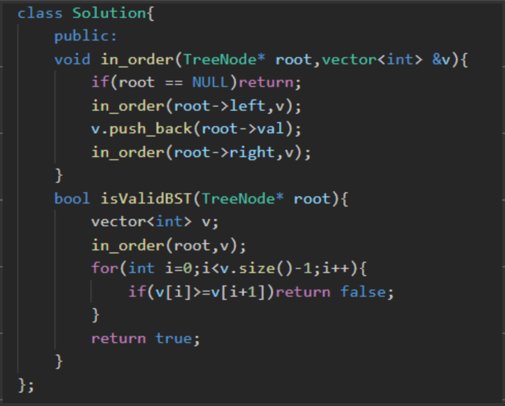
```
class Solution {
public:
    long long max_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MIN;
        return max((long long)root->val , max(max_in_node(root->left),max_in_node(root->right)));
    }
    long long min_in_node(TreeNode* root){
        if(root == NULL) return LLONG_MAX;
        return min((long long)root->val , min(min_in_node(root->left),min_in_node(root->right)));
    }
    bool isValidBST(TreeNode* root){
        if(root == NULL) return true;
        else if((long long)root->val <= max_in_node(root->left)) return false;
        else if((long long)root->val >= min_in_node(root->right)) return false;
        else return isValidBST(root->left) && isValidBST(root->right);
    }
};
```

→ Now this line Signify recursive calling of left subtree node and right Subtree node for further cheaking of left and right Subtree node... if any one is false return false...
And if no condition fulfilled and Null Acheived Now the given tree is A Valid BST...

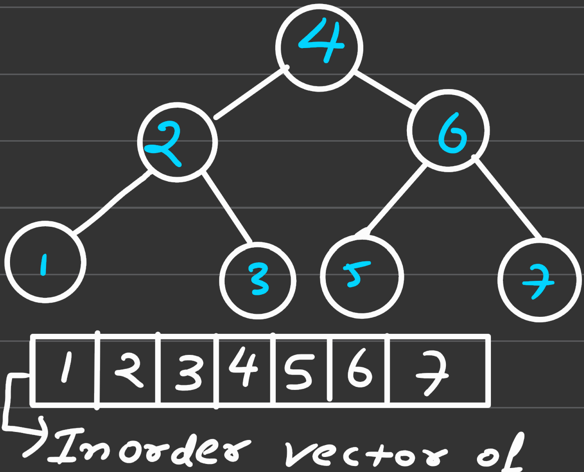
OPTIMISED SOLUTION:

We would use inorder traversal and store inorder traversal in a vector if that vector is sorted means the given Tree is a valid BST..
(INorder of BST is SORTED)

Code ImplEMENTATION:-



```cpp
class Solution{
    public:
    void in_order(TreeNode* root,vector<int> &v){
        if(root == NULL)return;
        in_order(root->left,v);
        v.push_back(root->val);
        in_order(root->right,v);
    }
    bool isValidBST(TreeNode* root){
        vector<int> v;
        in_order(root,v);
        for(int i=0;i<v.size()-1;i++){
            if(v[i]>=v[i+1])return false;
        }
        return true;
    }
};
```



Inorder vector of
<u>BST</u>