Check if a Binary Tree is BST: Simple and Efficient Approach

Given a Binary Tree, the task is to check whether the given binary tree is Binary Search Tree or not. A binary search tree (BST) is a node-based binary tree data structure which has the following properties.

- The left subtree of a node contains only nodes with keys less than the node’s key.
- The right subtree of a node contains only nodes with keys greater than the node’s key.
- Both the left and right subtrees must also be binary search trees.

From the above properties it naturally follows that:

Each node (item in the tree) has a distinct key.

```
// C++ program to check if a given tree is BST.
#include <bits/stdc++.h>
using namespace std;

/* A binary tree node has data, pointer to left child and a pointer to right child */
struct Node {
    int data;
    struct Node *left, *right;

    Node(int data) {
        this->data = data;
        left = right = NULL;
    }
}

// Utility function to check if Binary Tree is BST
```
bool isBSTUtil(struct Node* root, int& prev) {
    // traverse the tree in inorder fashion and
    // keep track of prev node
    if (root) {
        if (!isBSTUtil(root->left, prev))
            return false;

        // Allows only distinct valued nodes
        if (root->data <= prev)
            return false;

        // Initialize prev to current
        prev = root->data;

        return isBSTUtil(root->right, prev);
    }

    return true;
}

// Function to check if Binary Tree is BST
bool isBST(Node* root) {
    int prev = INT_MIN;
    return isBSTUtil(root, prev);
}

/* Driver program to test above functions*/
int main() {
    struct Node* root = new Node(5);
    root->left = new Node(2);
    root->right = new Node(15);
    root->left->left = new Node(1);
    root->left->right = new Node(4);

    if (isBST(root))
        cout << "Is BST";
    else
        cout << "Not a BST";

    return 0;
}

Output:
Is BST